



Ecole Polytechnique de l'Université de Tours

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

Projet Recherche & Développement

2021-2022

Prédiction de décrochage dans le milieu universitaire à l'aide de réseaux de neurones



POLYTECH[®]
TOURS

Entreprise

Polytech



Tuteur entreprise

Sabine BARRAT (Professeure)

Étudiant

Hugo LE BECHENNEC (DI5)

Tuteur académique

Sabine BARRAT

3 avril 2022

Liste des intervenants

Entreprise

Polytech
64 avenue Jean Portalis
37200 Tours, France
polytech.univ-tours.fr



Nom	Email	Qualité
Hugo LE BECHENNEC	hugo.lebechenneec@etu.univ-tours.fr	Étudiant DI5
Sabine BARRAT	sabine.barrat@univ-tours.fr	Tuteur académique, Département Informatique
Sabine BARRAT	sabine.barrat@univ-tours.fr	Tuteur entreprise, Professeure



Avertissement

Ce document a été rédigé par Hugo LE BECHENNEC susnommé l'auteur.

L'entreprise Polytech est représentée par Sabine Barrat susnommé le tuteur entreprise.

L'Ecole Polytechnique de l'Université de Tours est représentée par Sabine Barrat susnommé le tuteur académique.

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

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

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

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

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

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

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



Pour citer ce document

Hugo LE BECHENNEC, *Prédiction de décrochage dans le milieu universitaire à l'aide de réseaux de neurones*, Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2021-2022.

```
@mastersthesis{
  author={LE BECHENNEC, Hugo},
  title={Prédiction de décrochage dans le milieu universitaire à l'aide de réseaux de neurones:
},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université de Tours},
  address={Tours, France},
  year={2021-2022}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	vi
Liste des tableaux	viii
Liste des codes sources	ix
1 Introduction	1
1 Acteurs, enjeux et contexte	1
2 Objectifs	1
3 Hypothèses	2
4 Bases méthodologiques	2
2 Description générale	3
1 Environnement du projet	3
2 Caractéristiques des utilisateurs	3
3 Fonctionnalités du système	3
3.1 Entraînement du réseau de neurones	3
3.2 Utilisation de la plateforme numérique	4
4 Structure générale du système	5
4.1 Entraînement du réseau de neurones	6
4.2 Plateforme numérique	6

3	État de l'art / Veille technologique	8
1	État de l'art existant	8
	Le cycle de vie des Learning Analytics et les étapes du processus d'analyse.....	8
	Limitations liés aux recherches	10
	Méthodes et outils appliqués au Learning Analytics.....	10
2	Base de données et caractéristiques utilisées par la prédiction	13
3	Techniques de prédiction employées dans la littérature	16
4	Résultats obtenus dans la littérature	18
5	Techniques de visualisation employés dans la littérature	21
4	Analyse et conception	24
1	Analyse	24
1.1	Choix de la technique	24
1.2	Sélection de la base de données	25
1.3	Spécifications.....	27
1.3.1	Partie réseau de neurones	27
1.3.2	Partie plateforme numérique.....	28
2	Modélisation proposée.....	28
2.1	Traitement et transformation des données.....	28
	Conversion en données numériques	29
	Choix de la quantité maximale d'évaluation.....	30
5	Mise en oeuvre	31
1	Outils et librairie utilisés.....	31
2	Éléments d'implémentation, choix techniques	31
	Classe DataParser	31
	Classe DataConverter	32
	Classe NeuralNetwork	33
	Classe ConvertedData	33
	Exécution de l'algorithme.....	33
3	Analyse des résultats, évaluation, qualité	35
3.1	Premier résultat.....	36
3.2	Amélioration possible du modèle.....	37
	Autres améliorations possibles	39
6	Bilan et conclusion	40
1	Bilan du semestre 9	40
1.1	Travail réalisé	41
1.2	Retard pris	41

1.3	Travail restant	41
2	Planning pour le semestre 10	41
3	Bilan du semestre 10.....	41
3.1	Travail réalisé	42
4	Bilan sur la qualité	42
5	Bilan auto-critique.....	42
Annexes		43
A Planification, gestion de projet		44
1	Evolution du projet	44
1.1	Planning initial S9	45
1.2	Planning final S9	45
1.3	Planning initial S10	46
1.4	Planning final S10.....	46
2	Description des tâches.....	46
	Tâche 1 : Réalisation de l'état de l'art des Learning Analytics	46
	Tâche 2 : Rédaction du rapport : Acteurs, enjeux, contexte, objectif, hypothèses, bases méthodologiques.....	47
	Tâche 3 : Rédaction du rapport : Environnement du projet, caractéristiques des utilisateurs, fonctionnalités du système, structure générale du système	47
	Tâche 4 : Rédaction du rapport : État de l'art.....	47
	Tâche 5 : Recherche sur des méthodes spécifiques	47
	Tâche 6 : Rédaction du rapport : Éléments principaux de l'analyse..	47
	Tâche 7 : Rédaction du rapport : Annexes	47
	Tâche 8 : Rédaction des diapositives de la première soutenance	48
	Tâche 9 : Rédaction du rapport : Ce qui est et reste à faire, planning et organisation pour le semestre 10.....	48
	Tâche 10 : Dev : Importation de la base de données	48
	Tâche 10 : Test : Importation de la base de données	48
	Tâche 11 : Dev : Traitement et transformation des données	48
	Tâche 11 : Test : Traitement et transformation des données.....	48
	Tâche 12 : Dev : Importation de la base de données	48
	Tâche 12 : Dev : Importation de la base de données	49
B Description des interfaces		50
1	Interfaces matérielles/logicielles	50
2	Interfaces homme/machine.....	50

C	Cahier de Spécifications	51
1	Spécifications fonctionnelles	51
1.1	Partie réseau de neurones	51
1.1.1	Fonctionnalités à développer	51
1.1.2	Définition de la fonction 1 : Importation de la base de données...	51
	Présentation de la fonction 1 :	51
	Description de la fonction 1 :	51
1.1.3	Définition de la fonction 2 : Traitement et transformation des données	52
	Présentation de la fonction 2 :	52
	Description de la fonction 2 :	52
1.1.4	Définition de la fonction 3 : Entraînement et sauvegarde du réseau de neurones	52
	Présentation de la fonction 3 :	52
	Description de la fonction 3 :	53
1.2	Partie plateforme numérique	53
1.2.1	Fonctionnalités à développer	53
1.2.2	Définition de la fonction 4 : Authentification.....	53
	Présentation de la fonction 4 :	53
	Description de la fonction 4 :	53
1.2.3	Définition de la fonction 5 : Importation des données.....	54
	Présentation de la fonction 5 :	54
	Description de la fonction 5 :	54
1.2.4	Définition de la fonction 6 : Visualisation des prédictions.....	54
	Présentation de la fonction 6 :	54
	Description de la fonction 6 :	54
2	Spécifications non fonctionnelles	54
2.1	Contraintes de développement et conception	54
2.2	Contraintes de fonctionnement et d'exploitation.....	54
2.2.1	Performances	55
2.2.2	Sécurité	55
D	Cahier du développeur	56
1	Introduction	56
2	Diagrammes architecturaux et UML	56
3	Descriptions détaillées des données exploitées.....	56
4	Descriptions détaillées des classes, modules, réalisations	56
4.1	Importation et traitement des données	56
	Classe DataParser :	56
	Classe DataConverter :	59

Classe NeuralNetwork :	63
Classe ConvertedData :	63
E Document d'installation	65
Installation de Conda (Optionnel)	65
Installation de Python.....	65
Installation de PyTorch	65
Installation Matplotlib	65
Installation NumPy	65
Installation Pandas.....	66
Installation de scikit-learn	66
F Document d'utilisation	67
1 Récupération des données	67
2 Entraînement du réseau de neurones	67
G Cahier de test	69
1 Tests unitaires	69
1.1 DataParser	69
1.2 DataConverter	73
H Bibliographie	81
I Glossaire	83
J Acronymes	85

Table des figures

2 Description générale

2.1	Diagramme de cas d'utilisation du réseau de neurones lors de l'entraînement	4
2.2	Diagramme de séquence du réseau de neurones lors de l'entraînement	5
2.3	Diagramme de cas d'utilisation de la plateforme numérique.....	6
2.4	Diagramme de séquence de la plateforme numérique	7

3 État de l'art / Veille technologique

3.1	Cycle de vie des Learning Analytics	9
3.2	Étapes du processus d'analyses	10
3.3	Schéma d'un perceptron multicouche provenant de \edefhuang2013predicting11huang2013predicting__ - cite huang2013predicting.....	17
3.4	Résultat des moins bonnes comparaisons obtenues par \edefhuang2013predicting11huang2013predicting__ - cite huang2013predicting.....	18
3.5	Résultat des meilleures comparaisons obtenus par \edefhuang2013predicting11huang2013predicting__ - cite huang2013predicting.....	19
3.6	Résultat pour différentes architectures du réseau proposé par \edefimran2019predicting11imran2019predi cite imran2019predicting.....	20
3.7	Matrice de confusion et taux de précision des décrochages pour les différentes architectures.	21
3.8	Histogramme des taux de précision pour les différentes architectures.	21

4 Analyse et conception

4.1	Compte des mots-clés présents dans la littérature anglaise	24
-----	--	----

5 Mise en oeuvre

5.1	Graphique des pertes et de la précision selon les époques	38
-----	---	----

6 Bilan et conclusion

6.1 Bilan des tâches effectués au semestre 9	40
6.2 Diagramme de Gantt initial S10.....	42

A Planification, gestion de projet

A.1 Visualisation des tâches d'un tableau Trello	45
A.2 Diagramme de Gantt initial S9	45
A.3 Diagramme de Gantt final S9.....	45
A.4 Diagramme de Gantt initial S10.....	46
A.5 Diagramme de Gantt final S10.....	46

D Cahier du développeur

D.1 Diagramme des classes liés à l'entraînement du réseau de neurones	57
---	----



Liste des tableaux



Liste des codes sources

5.1	Méthode convertData	32
5.2	Chargement des données	33
5.3	Construction des données.....	34
5.4	Définition du modèle.....	34
5.5	Entraînement du modèle	34
5.6	Évaluation sur la base de test	35
F.1	Construction des données.....	67
F.2	Chargement des données	67
F.3	Répartition des bases	68
F.4	Taille d'un batch et nombre d'époque	68
F.5	Taux d'apprentissage	68
F.6	Nombre de couches et de neurones	68

1

Introduction

1 Acteurs, enjeux et contexte

En France et dans les pays européens les décrochages sont très élevés, entre 20% et 30% en moyenne [13]. Ces décrochages ont majoritairement lieu lors de la première année universitaire. Nous souhaitons de par ce projet prédire et prévenir les décrochages.

Beaucoup de recherches ont été menées sur ce sujet ces dernières années, toutefois une grande majorité des recherches portent sur les possibilités récentes de récupération de données apportées par les **Massive Open Online Courses (MOOCs)**. Les recherches portant sur des systèmes éducatifs classiques reposent généralement sur de la récupération de données non automatisée, notamment par le biais de sondages ou d'entretien par exemple. Toutefois, la normalisation des plateformes numériques dans les études supérieures peut permettre de récupérer des données similaires à celles des **MOOCs**, notamment les notes aux différents contrôles continus et aux examens, mais aussi potentiellement des supports de cours consultés, etc.

Les acteurs de ce projet sont les suivants :

- Client : Polytech Tours
- MOA : Sabine Barrat, Professeure au département informatique de Polytech Tours
- MOE : Hugo Le Bechennec, Étudiant en 5^{ème} année du département informatique de Polytech Tours

2 Objectifs

L'objectif final de ce Projet de Recherche & Développement est de prédire les étudiants en risque de décrochage, notamment de par l'utilisation de données issues de plateforme numériques, afin de réduire les décrochages, en apportant une aide aux étudiants considérés à risques.

Pour ce faire, on peut définir plusieurs objectifs.

Premièrement, il est nécessaire d'entraîner notre modèle sur un grand jeu de données. Ensuite, une implémentation de la visualisation de la prédiction est nécessaire, allant récupérer automatiquement les données nécessaires pour chaque étudiant.

3 Hypothèses

Afin de réaliser ce projet, il est nécessaire d'avoir accès à des données relatives aux étudiants afin de prédire de manière efficace leur risque de décrochage. Durant ce projet, nous utiliserons des bases de données publiques principalement issues de **MOOCs**. Toutefois dans le cas d'une utilisation réelle au sein d'établissements supérieurs français il est nécessaire de faire l'hypothèse que l'on dispose de données issues d'une plateforme numérique associée à l'établissement.

4 Bases méthodologiques

Pour ce projet, la méthode de gestion de projets utilisée est la méthode agile. Cette gestion de projet s'appuie sur l'utilisation de la Trello et d'un plugin permettant de réaliser un diagramme de Gantt à partir des tâches d'un tableau Trello. Plus de détails sur la gestion de projet sont disponible dans la section [A](#).

2

Description générale

1 Environnement du projet

L'environnement du projet est le suivant :

- Langage de programmation : Python 3.8.8
- Bibliothèque supplémentaire : PyTorch 1.10.1
- IDE : Spyder
- Matériel utilisé pour l'obtention des résultats : Intel® Core™ i5-4200H CPU @ 2.80GHz, 12Go de RAM

2 Caractéristiques des utilisateurs

Les utilisateurs de ce projet sont le MOE et la MOA afin de comparer les résultats par rapport à d'autres méthodes de la littérature. Puis dans un second temps les enseignants qui au travers d'une plateforme numérique telle que CELENE pourront consulter les étudiants à risque de décrochage.

3 Fonctionnalités du système

3.1 Entraînement du réseau de neurones

Le diagramme de cas d'utilisation du réseau de neurones lors de l'entraînement est décrit par la figure 2.1 :

Le MOE peut initier l'entraînement du réseau de neurones. Cet entraînement implique qu'il faille récupérer les données de la base publique. À la fin de chaque entraînement il est également nécessaire que le MOE et la MOA analyse les performances du réseau de neurones. Dans le cas où les performances du réseau sont bonnes, alors ce dernier est sauvegardé. Si les performances ne sont pas suffisantes, alors le MOE a la charge d'améliorer le réseau de neurones et d'entraîner à nouveau ce dernier.

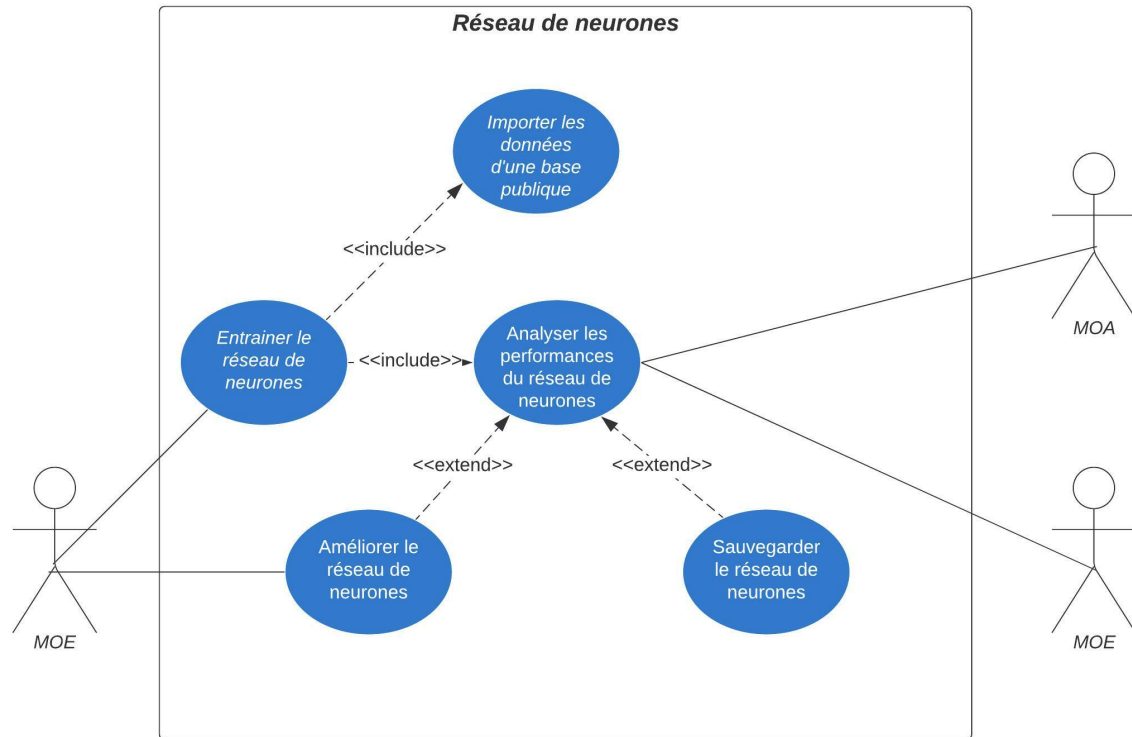


Figure 2.1 – Diagramme de cas d'utilisation du réseau de neurones lors de l'entraînement

Le diagramme de séquence de l'entraînement du réseau de neurones lors de l'entraînement est décrit par la figure 2.2 :

Le diagramme de séquence du réseau de neurones comporte 4 acteurs, le MOE et la MOA ainsi que le système et la base de données. Dans un premier temps le MOE demande au système d'importer les données, qui vont être lues dans la base de données et récupérées par le système. Le système va alors traiter ces données en ne gardant que les données d'intérêt et en créant de nouvelles données. Ces données traitées sont alors transformées en un tenseur qui est lui-même passé au réseau de neurones afin d'entraîner ce dernier. Lorsque l'entraînement du réseau de neurones est terminé, le MOE et la MOA, analysent ensemble les performances du réseau de neurones, et dans le cas où celles-ci sont acceptables, alors le réseau de neurones est sauvegardé, sinon le MOE doit améliorer le réseau de neurones et recommencer l'entraînement de ce dernier.

3.2 Utilisation de la plateforme numérique

Le diagramme de cas d'utilisation de la plateforme numérique est décrit par la figure 2.3 :

L'utilisateur peut se connecter à la plateforme numérique. Dans le cas où elle fournit de mauvaises informations de connexion alors un message d'erreur est affiché, sinon la visualisation des prédictions est chargée. L'utilisateur peut alors analyser les prédictions et mettre en place s'il le souhaite des actions pour aider les étudiants à risques.

Le diagramme de séquence de la plateforme numérique est décrit par la figure 2.4 :

Le diagramme de séquence de la plateforme numérique comporte 4 acteurs, l'utilisateur et le système, ainsi que deux bases de données, celle des utilisateurs, et celle des prédictions. L'utilisateur doit d'abord se connecter à la plateforme numérique, cette dernière va alors vérifier dans la base de données si les informations de connexion fournies sont correctes, si elles ne le sont pas alors un message d'erreur sera affiché sur la page de connexion. Une fois connecté, la

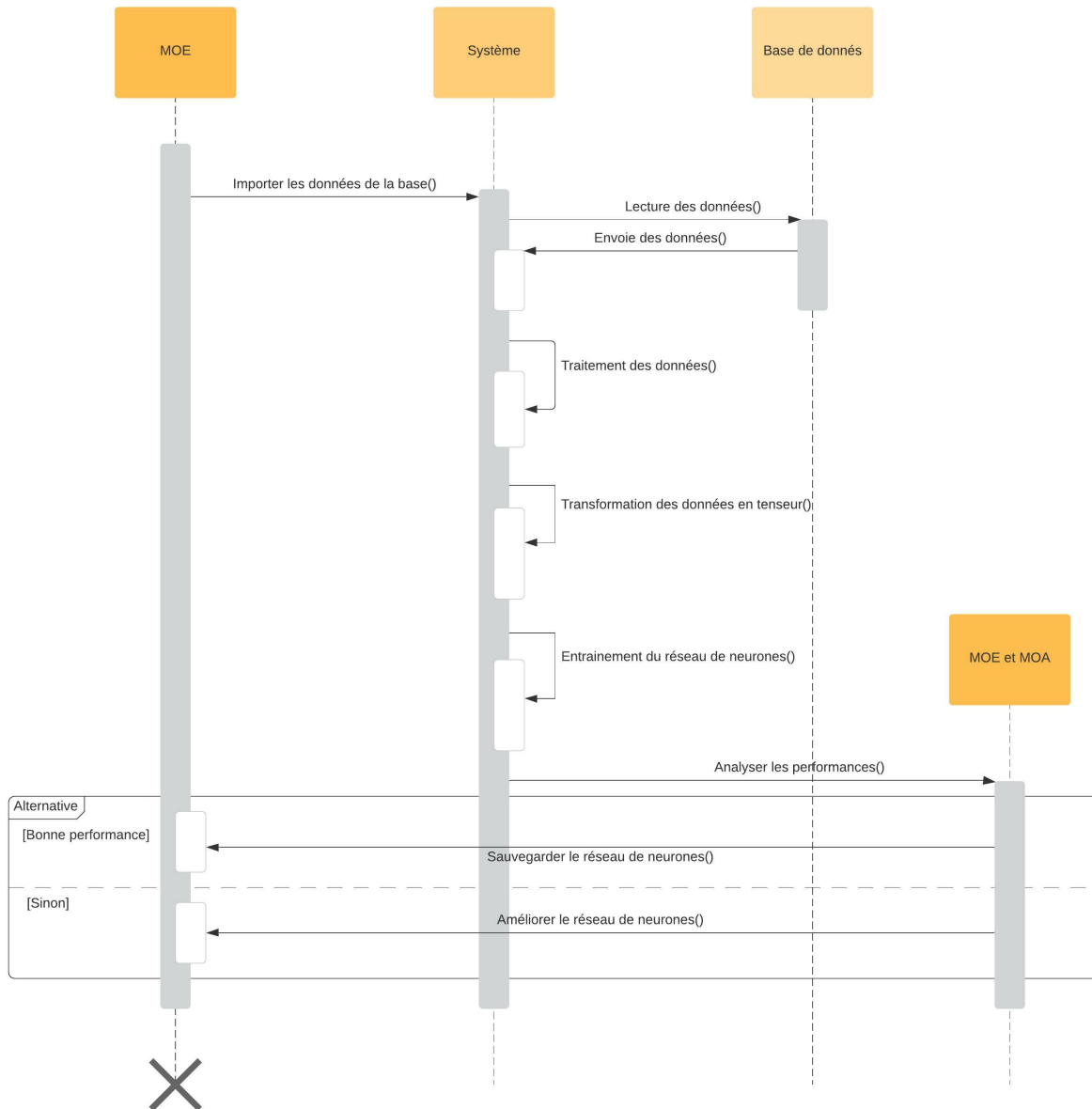


Figure 2.2 – Diagramme de séquence du réseau de neurones lors de l'entraînement

page de visualisation des prédictions sera chargée, les données de prédictions auxquels l'utilisateur peut accéder vont alors être récupérées dans la base de données des prédictions, et la page de visualisation sera affichée à l'utilisateur. Enfin l'utilisateur pourra analyser les prédictions et mettre en oeuvre des actions d'aides des étudiants à risques.

La description détaillée des fonctionnalités peut être retrouvée dans la partie 1 (Annexe C)

4 Structure générale du système

Comme l'on a pu le constater précédemment, on dispose ici de deux systèmes. Le réseau de neurones utilisé pour les prédictions et la plateforme numérique pour permettre la visualisation des prédictions.

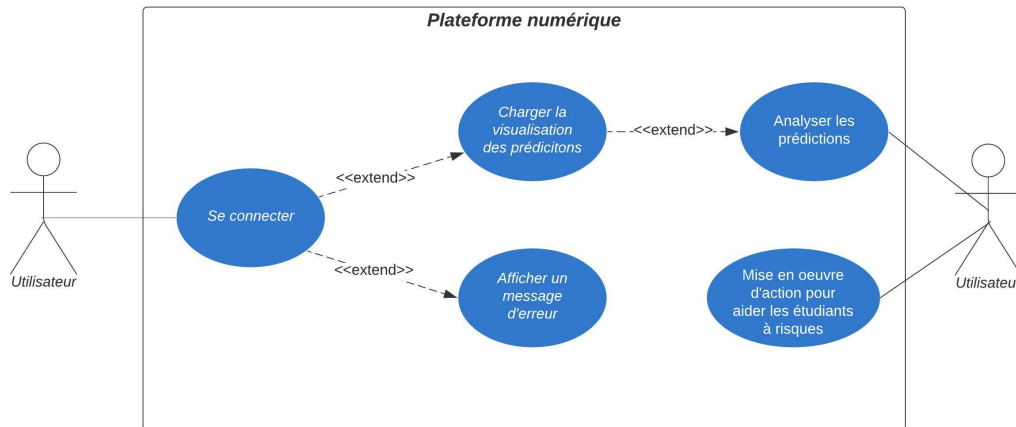


Figure 2.3 – Diagramme de cas d'utilisation de la plateforme numérique

4.1 Entraînement du réseau de neurones

L'entraînement du réseau de neurones est décomposé en plusieurs parties.

- L'importation de données d'une base publique.
- Le traitement et la transformation des données.
- L'entraînement du réseau de neurones.
- La sauvegarde du réseau de neurones.

4.2 Plateforme numérique

La plateforme numérique peut également être découpée en plusieurs parties.

- Un système d'authentification associé à une base de données d'utilisateur, impliquant l'existence d'une page de connexion.
- Une page de visualisation des prédictions, nécessitant un accès à l'ensemble des prédictions réalisées pour chaque élève.
- Une page d'ajout de notes. La prédiction des étudiants ayant reçu une nouvelle note est alors recalculée et visible lorsque l'utilisateur retourne sur la page de visualisation.

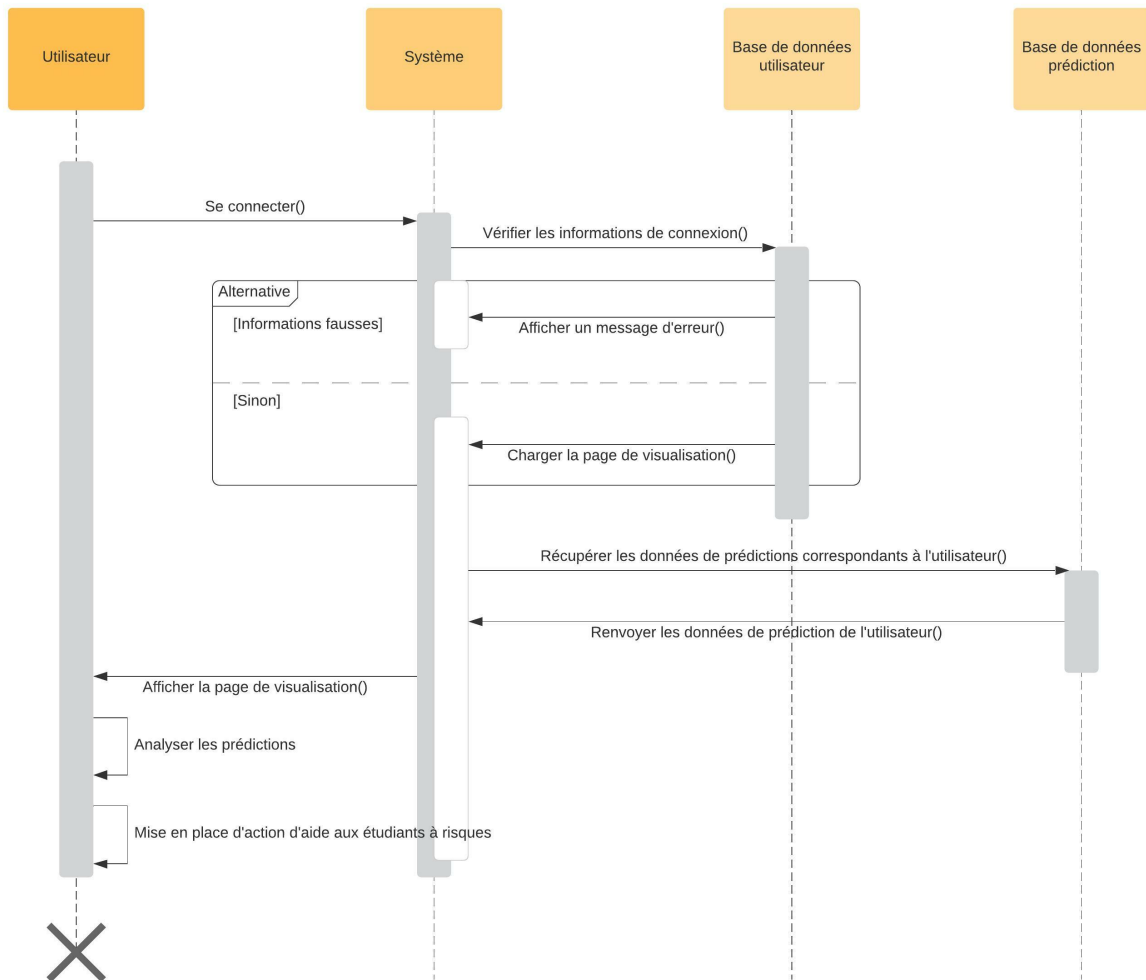


Figure 2.4 – *Diagramme de séquence de la plateforme numérique*

3

État de l'art / Veille technologique

Dans ce chapitre, on réalise dans un premier temps de l'état de l'art des **Learning Analytics (LA)** en général et l'on se focalisera au fur et à mesure sur des méthodes et des outils précis s'approchant de ce projet.

1 État de l'art existant

Plusieurs états de l'art ont déjà été réalisés sur le sujet des **LA**. Un état de l'art a notamment été réalisé sur les outils et méthodes issus de la recherche française [8]. Ce document se décompose de la manière suivante :

Dans un premier temps, l'étude et les caractéristiques sur lesquels celle-ci s'est appuyée sont présentés. Ensuite, un chapitre se focalise sur la construction du jeu de traces nécessaire à l'analyse et à la visualisation développées dans les chapitres suivants. Enfin une présentation des principaux outils est réalisée.

De nombreux états de l'art sont également disponibles dans la littérature anglaise. Notamment un état de l'art englobant l'ensemble des **LA** [9], mais aussi un état de l'art des méthodes de **Machine Learning (ML)** utilisées pour faire de la prédiction de notes [6].

Dans [9], les étapes de l'analyse des **LA** sont présentées ainsi que son cycle de vie. L'article traite ensuite des bases de données publiques existantes concernant l'enseignement supérieur, ainsi que des différentes techniques employées dans la littérature anglaise. Enfin le document conclut et imagine les futures tendances en matière de **LA**.

Enfin, [6] présente les principes de nombreux algorithmes de **ML** appliqués aux **LA**. Le document commence par décrire les critères de sélection pour leur état de l'art. Ils décrivent ensuite dans un second chapitre les méthodes de **ML** utilisées dans les articles de **LA** et classifient pour chaque article les méthodes qui y sont citées. Ensuite, l'article décrit techniquement certaines méthodes.

Le cycle de vie des Learning Analytics et les étapes du processus d'analyse

Le cycle de vie des Learning Analytics est décrit dans [9] par la figure 3.1. Celui-ci décompose les Learning Analytics en 4 parties distinctes :

- Learning environment : Ce sont les environnements numériques permettant un accès à différentes ressources numériques de travail, telles que les **ENT**, les **MOOCs**, etc.

- Big data : Ce sont les données stockées sous forme de **traces** ou de données personnelles par les étudiants, notamment par leur échange avec les différents environnements numériques ou au moment de leur inscription.
- Analytics : Ce sont les différentes méthodes employées pour déterminer des modèles cachés dans les données afin de les exploiter et en tirer des avantages. Ces méthodes sont nombreuses et ont différents buts, certaines seront détaillées par la suite.
- Action : Les résultats des différentes analyses sont interprétés et contribuent à l'amélioration de l'apprentissage et de l'enseignement, voir parfois des plateformes numériques formant ainsi ce cycle de vie des **LA**.

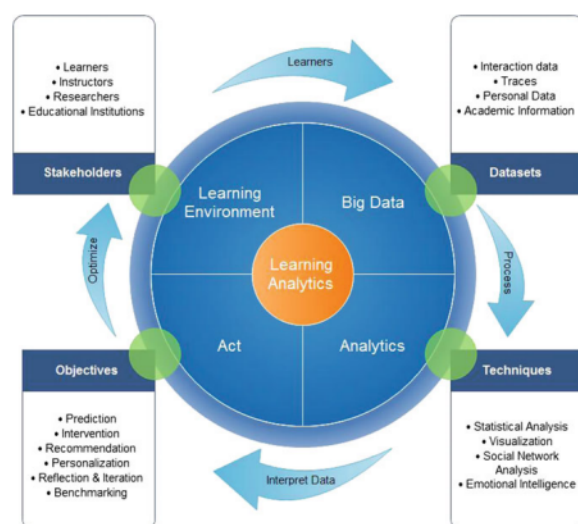


Fig. 1.2 Khalil and Ebner LA life cycle (Khalil and Ebner 2015)

Figure 3.1 – Cycle de vie des Learning Analytics

Le document décrit également plus en détail les étapes du processus d'analyse. On y distingue 5 étapes différentes décrites dans la figure 3.2 :

- Capturing : Les données doivent être capturées en temps réel au travers de différentes plateformes numériques tels que les **Espace Numérique de Travail (ENT)** combinés aux informations personnelles des étudiants qui ont été préalablement anonymisées.
- Reporting : Les données sont utilisées pour générer des modèles permettant de mesurer les progrès des étudiants. Le plus souvent cette mesure est réalisée par un utilisateur à l'aide de différentes techniques de visualisation, notamment dans le domaine des **LA** par l'utilisation de tableaux de bord.
- Predicting : Les données sont utilisées pour prédire les étudiants à risques et ceux avec de grandes chances de réussites. Ces prédictions sont également parfois utilisées pour gérer la répartition des ressources des différents enseignements.
- Acting : Les données de l'analyse sont utilisées afin d'intervenir dans les enseignements à risques pour les étudiants, ou bien pour apporter une aide aux étudiants risquant d'abandonner leurs études.
- Refining : Les informations sont utilisées afin d'améliorer continuellement les méthodes d'enseignement et d'apprentissage, formant le processus cyclique décrit dans la figure 3.2.

En France, [8] décompose également les étapes de l'analyse, cette fois en 3 étapes distinctes :

- La construction du jeu de traces : Afin de réaliser une analyse il est nécessaire d'avoir accès à des données. Celles-ci sont le plus souvent obtenues par la récupération automatique de traces sur les différents environnements numériques. Ces traces étant des données plus spécifiques que de simples log, il existe de nombreux modèles de traces différents appartenant à différents types décrits dans [8]. De même il existe différentes méthodes de stockage de



Figure 3.2 – Étapes du processus d'analyses

ces traces.

- L'analyse de ces jeux de traces : L'analyse peut être vue sous différentes approches selon [8], que ce soit par l'emploi d'outils généralistes de statistiques tel que R ou SAS, que par l'utilisation d'algorithmes automatiques, de langages dédiés ou d'outils graphiques.
- L'exploitation des analyses : Toute analyse doit être interprétée, dans ce document c'est la visualisation des analyses qui est mise en avant selon deux méthodes, soit l'utilisation d'outils de visualisation non dédiés aux LA soit par l'utilisation de tableaux de bord.

Bien que ce document soit récent, celui-ci cite majoritairement des documents publiés entre 2012 et 2016. De plus, l'utilisation des réseaux de neurones aux LA n'y est pas traitée, probablement par manque de recherche française dessus. Toutefois, le document décrit de nombreux autres outils et méthodes issus de la recherche française, notamment du projet Hubble.

Limitations liés aux recherches

Les recherches sur LA sont récentes, en effet comme le souligne [9], les recherches sur le sujet ont réellement commencé à partir de 2011, de ce fait un travail continu est nécessaire afin de prouver les différentes hypothèses. En France, on a également fait ce constat précédemment dans l'article [8]. De plus, les jeux de données étaient petits jusqu'à encore récemment avec la démocratisation des MOOCs. Cela implique des difficultés de mise en place de certaines méthodes. [8] soulève également le problème dans la littérature anglophone des recherches de cultures ayant des systèmes éducatifs totalement différents et dont les modèles pourraient marcher pour une culture mais pas une autre. Il soulève également la question éthique du traitement des données privées.

Méthodes et outils appliqués au Learning Analytics

Malgré la récence du sujet, il existe déjà de nombreuses méthodes et outils appliqués aux LA, les méthodes étaient déjà existantes et ont simplement été appliquées dans un contexte de LA, les outils sont pour certains d'entre eux également issus d'autres domaines tels que R par exemple, ou ont été modifiés tel que kTBS4LA qui est une surcouche de kTBS permettant le traitement des traces liées aux LA, d'autres outils ont été créés spécifiquement pour traiter les LA.

[9] résume l'ensemble des techniques utilisées dans les LA ainsi que leur application et les publications associées jusqu'en 2016. Les techniques décrites sont les suivantes :

- Prédiction : Technique appliquée à la prédiction des performances des étudiants et du comportement de ces derniers.
- Partitionnement des données (Clustering) : Méthode visant à regrouper les étudiants similaires en groupes selon leur apprentissage et leurs interactions.
- Détection d'anomalies (Outlier Detection) : Méthode visant à prédire les étudiants en difficultés ou avec des méthodes d'apprentissage inhabituelles.
- Fouille de données relationnelles (Relationship mining) : Cette méthode vise à identifier les relations entre les différents comportements des étudiants et de détecter les potentiels étudiants en difficultés.
- Analyse des réseaux sociaux : Exploite les données issues des différents moyens d'échanges disponibles sur les plateformes numériques pour interpréter les relations dans les activités de groupes et les interactions avec les moyens de communication.
- Exploration de processus (Process mining) : Cette technique essaie de refléter le comportement des étudiants par rapport aux données issues d'examens, en prenant en compte les notes, le niveau d'études, etc.
- Fouilles de texte (text mining) : Analyse de données issues de forums ou de chat sur les plateformes numériques.
- Réduction des données pour l'interprétation humaine (Distillation of data for human judgement) : Cette méthode a pour objectif d'aider les enseignants à visualiser et à analyser les activités des étudiants.
- Analyse avec des modèles existants (Discovery with models) : Cette méthode d'utilisation de modèle existant de fouille de données pour les appliquer à de nouvelles analyses de données vise selon [9] à identifier les relations entre le comportement des étudiants et leurs caractéristiques ou leur appartenance à un collectif d'individus par l'intégration de **modèle psychométrique** à des modèles de **ML**.
- Gamification : Méthode d'apprentissage visant à rendre l'enseignement amusant et motivant en utilisant des concepts issus des jeux tels que les succès ou l'obtention de points d'expérience.
- Apprentissage automatique (Machine Learning) : Le **ML** permet de trouver des informations cachées dans les données automatiquement grâce à des modèles prenant en compte de nouvelles données et s'adaptant d'eux-mêmes en fonction de ces données.
- Statistiques : Analyse et interprétation de données quantitatives pour la prise de décision.

Des méthodes de **ML** appliquées à la prédiction des performances sont aussi décrites dans [6], on y retrouve notamment les méthodes de **ML** suivantes :

- Factorisation matricielle : Cette méthode décompose une matrice en plusieurs autres matrices pour découvrir des variables cachées et prédire des valeurs manquantes dans la matrice initiale. Cette méthode n'est toutefois pas efficace avec de petits jeux de données.
- Régression linéaire multiple : Cette méthode permet de trouver des relations entre plusieurs caractéristiques, notamment pour permettre de prédire les futures notes des étudiants.
- Modèles de régression et de classification : Ces méthodes permettent de classer les étudiants en deux catégories, ceux qui ont de bons résultats et ceux qui ont des mauvais résultats.
- Perceptron multicouche : Modèle de réseaux de neurones utilisé selon [6] pour prédire le succès des étudiants selon les traces laissées par les étudiants.
- Machines de factorisation : Cette méthode généralise les modèles de factorisation telle que la factorisation matricielle qui ne prend normalement que des données entrées particulières, tandis que les machines de factorisation permettent des prédictions plus générales.
- Filtrage collaboratif : Cette méthode permet de construire des systèmes de recommandations qui dans le cadre des **LA** permet de prédire des notes en regroupant des étudiants similaires entre eux.
- Machine de Boltzmann restreinte : Cette méthode est utilisée pour comprendre la structure

d'une base de données. Dans le contexte des LA, [6] utilise cette méthode pour prédire les performances des étudiants dans différents cours.

Dans [8], de nombreux outils utilisés dans les LA sont cités, et notamment classifiés en 4 catégories :

- Les outils généralistes de statistiques tels que R, SAS, Stata, etc, utilisés dans d'autres domaines que les LA sont parfois également utilisés dans ce domaine.
- Les algorithmes automatiques tels que les réseaux de Petri avec Laalys, les automates à états finis avec T-store.
- L'utilisation de langages dédiés à la manipulation de traces avec des outils tels que UTL, kTBS ou SPARE-LNC.
- Les outils de manipulations graphiques tels que DDART, kTBS4LA, D3KODE, UNDER-TRACKS

D'autres outils sont également proposés dans [12], dans cet article, 3 catégories d'outils sont détaillées :

- Les outils de préparation des données, on y retrouve les outils suivants très communément utilisés :
 - EDM Workbench : Cet outil permet à l'utilisateur de définir des ensembles de caractéristiques dont les données sont ensuite regroupées en sous-ensembles.
 - Python : C'est le langage de programmation le plus utilisé pour la préparation des données, bien que pouvant faire face à des limitations lors de large jeu de données.
 - SQL : Ce langage permet de gérer les données présentes dans les bases de données. Sa force étant de permettre l'extraction de données spécifiques dans de larges jeux de données.
- Les outils d'analyses, ceux-ci sont nombreux et sont souvent spécialisés dans un ou plusieurs types d'analyses, l'auteur de [12] met en avant les outils suivants :
 - RapidMiner : Un logiciel gratuit et Open Source pour la préparation des données, le ML, le Deep Learning, le Text Mining, et les prédictions, Il présente un ensemble d'algorithmes de régression et de classification, mais aussi de clustering et d'extraction de règles d'association, ainsi qu'un langage de programmation graphique. De nombreuses mesures sont également présentes pour déterminer la validité d'un modèle.
 - WEKA : Un logiciel gratuit et Open Source également, proposant un ensemble d'algorithmes de ML, tels que la régression, la classification, l'extraction de règles d'association, le clustering et proposant également de la visualisation. Le tout étant disponible soit par une interface graphique, soit par des lignes de commandes, soit par une API Java.
 - SPSS : C'est un programme d'analyses statistiques incluant des régressions, des facteurs d'analyses et de corrélation, etc. Considéré comme un outil d'analyse statistique complet mais ayant des problèmes de modélisation comparé à d'autres outils, il est également moins personnalisable, flexible et documenté selon [12].
 - KNIME : Un logiciel gratuit et Open Source intégrant tous les algorithmes de WEKA, ainsi que de l'analyse des réseaux sociaux et de l'analyse des sentiments.
 - Orange : Un logiciel gratuit et Open Source permettant la visualisation des données, le ML, et des outils de Data Mining. L'outil est toutefois limité sur la taille des jeux de données qui peuvent être traités.
 - KEEL : Un logiciel gratuit et Open Source proposant un grand nombre d'algorithmes d'extraction de connaissances, de techniques de pré-traitement, de méthodes statistiques, etc.
 - Spark MLlib : Ce framework permet le traitement de large jeux de données à l'aide de nombreux processeurs. Le framework peut être utilisé avec de nombreux langages tels que le Java, le Python ou le SQL à l'aide d'une API."

- R analytical tool : Un logiciel gratuit et **Open Source** proposant des outils de Data Mining supervisés et non supervisés avec un environnement graphique.
- Les outils de visualisation, [12] décrit deux outils de visualisation :
 - Tableau : Un outil d'analyse et de visualisation de données interactives. L'outil ne nécessite aucune compétence en développement pour analyser des larges jeux de données. Il permet d'importer des données depuis différents formats de données standardisés, mais ne supporte pas l'exploration de données relationnelles ou l'analyse prédictive. De plus l'outil est commerciale.
 - D3.js : Une librairie JavaScript libre et **Open Source** permettant la manipulation de données issues de documents dynamiques aidant la recherche sur la visualisation de données interactives intégrées à des pages internet. Il ne nécessite pas d'installation, mais a des problèmes de compatibilité avec certains navigateurs internet et des problèmes de temps de calcul pour de larges jeux de données.

2 Base de données et caractéristiques utilisées par la prédiction

Afin de pouvoir mettre en place de quelconques méthodes, il est nécessaire d'avoir accès à des données concernant les élèves afin de prédire efficacement leurs risques de décrochage. Le type de données récoltées sont multiples, ceux-ci peuvent aussi bien venir d'**ENT**, que de sondages, ou encore de forums disponibles sur les environnements numériques. De plus, comme le souligne [11], les données récoltées par les environnements numériques sont très différentes les unes des autres, et compliquent la combinaison de ces données. De ce fait les caractéristiques utilisées par les différentes méthodes sont parfois très différentes. Pour s'en rendre compte nous allons dans un premier temps énumérer les caractéristiques utilisées dans les différentes bases de données afin de pouvoir les comparer et constater aussi bien les points communs que les différences entre ces bases. On constatera également que les grands jeux de données aujourd'hui disponibles sont dus à la démocratisation des **MOOCs**, de ce fait il faut prendre en compte que les caractéristiques de prédiction des décrochages peuvent différer par rapport à la prédiction de décrochage dans l'enseignement supérieur.

Premièrement, [3] décrit un classificateur utilisant une base de données de près de 4,5 millions d'utilisateurs issus d'une plateforme de **MOOCs**, ces données ont été récoltées pendant 4 ans sur un total de 256 cours différents. Les caractéristiques exploitées dans cet article pour chaque étudiant sont les suivants :

- La moyenne des notes du cours.
- Le temps moyen passé sur un travail comparé à la moyenne de temps passé par la classe.
- Le pourcentage de travail rendu.
- Le pourcentage de travail rendu en retard.
- Le pourcentage de travail rendu dans les temps.
- Le pourcentage de travail non rendu.
- Le pourcentage de travail rendu en retard ou non rendu.
- Le nombre de jours depuis le dernier travail rendu.
- L'écart entre le rendu et la date de rendu.
- Le nombre consécutif maximum de travail rendu en retard.

À noter que les moyennes ne sont pas forcément considérées comme bonnes pour les mêmes valeurs selon les cours. De ce fait un score est calculé, nommé Cote Z, qui consiste à soustraire à la note la moyenne des notes de tous les étudiants dans la ville, et diviser par l'écart-type du cours. Ainsi, on s'assure d'avoir pour le cours une valeur qui soit cohérente avec la moyenne des étudiants.

Ensuite, [1] propose également une méthode de prédiction des étudiants à risques. Là encore cette méthode s'applique à des **MOOCs**, toutefois le jeu de données ne contient les données de

seulement 1073 étudiants, ce qui est bien plus faible que l'article précédent mais bien plus proche de notre situation. Les caractéristiques exploitées dans cet article sont les suivantes :

- La durée du cours divisée en 3 catégories : 0-20, 20-30 et 30+
- L'âge de l'étudiant divisé en 4 catégories : 20-29, 30-39, 40-49, 50+
- Le sexe de l'étudiant
- Si l'étudiant est à plein temps ou non.
- Moyenne des moyennes à chaque cours.
- Le niveau académique de l'étudiant.
- Un score calculé à partir des coefficients des notes finales.
- Le nombre de fois qu'une section du cours a été lue par l'étudiant.
- Le nombre de postes publiés sur le forum du MOOCs par l'étudiant.
- Le nombre de fois que l'étudiant a lu les discussions du forum.
- Le nombre de fois que le cours a été consulté sur l'outil de consultation.
- Le nombre de fois que l'étudiant s'est connecté à l'outil de consultation du cours.

En 2017, [7] décrit un jeu de données publiques couramment utilisé dans les LA. Ce jeu de données est issu d'un programme de MOOCs comptabilisant 32 593 étudiants. Il est important de noter que chaque module propose différentes présentations par an, de ce fait, chaque présentation est identifiée par un identifiant dont la nomenclature est "Année+Lettre indicatrice du mois" où Janvier est représenté par A, Février par B, etc. La base de données est séparée en plusieurs tables.

- Les informations concernant les étudiants et leurs résultats dans chaque module suivi :
 - L'identifiant du module auquel l'étudiant est inscrit.
 - L'identifiant de la présentation auquel l'étudiant est inscrit.
 - L'identifiant de l'étudiant.
 - Le sexe de l'étudiant.
 - La région géographique de l'étudiant au moment où il participait au module.
 - Le plus haut niveau d'éducation au moment du module.
 - L'Indices of Multiple Deprivation (IMD) band de la localité de l'étudiant au moment du module. Cette mesure permet de mesurer la pauvreté de la localité.
 - La fenêtre d'âge de l'étudiant.
 - Le nombre de fois où l'étudiant a tenté le module.
 - Le nombre total de crédit pour les modules que l'étudiant est actuellement en train d'étudier.
 - Une variable indiquant si l'étudiant est sujet à un handicap ou non.
 - Les résultats finaux de l'étudiant dans les modules.
- Les informations concernant un enseignement :
 - L'identifiant du module.
 - L'identifiant de la présentation.
 - La durée en jour du module.
- Les informations concernant l'inscription d'un étudiant à un module :
 - L'identifiant du module.
 - L'identifiant de la présentation.
 - L'identifiant de l'étudiant.
 - Le jour de l'inscription de l'étudiant au module.
 - Le jour de désinscription de l'étudiant au module. Ce champ est vide lorsque l'étudiant à compléter le module, et vaut "Withdrawal" lorsque l'étudiant s'est désinscrit.
- Les informations relatives aux examens du module :
 - L'identifiant du module.
 - L'identifiant de la présentation.
 - L'identifiant de l'évaluation.
 - Le type d'évaluation du module. Il en existe trois : Tutor Marked Assessment (TMA),

- Computer Marked Assessment (CMA), et Final Exam (Exam).
- La date limite de rendu de l'évaluation. Si aucune date n'est précisée, c'est que l'évaluation a lieu lors de la dernière semaine du module.
- Le coefficient de l'évaluation.
- Les informations relatives aux résultats des étudiants aux modules :
 - L'identifiant de l'évaluation.
 - L'identifiant de l'étudiant.
 - Le jour de rendu de l'évaluation.
 - Une variable indiquant si le résultat de l'évaluation a été transféré d'une précédente évaluation.
 - Le résultat de l'étudiant à cette évaluation. Le résultat est compris entre 0 et 100. En dessous de 40 l'étudiant a raté l'évaluation.
- Les informations relatives aux matériels accordés pour un module :
 - L'identifiant du matériel.
 - L'identifiant du module.
 - L'identifiant de la présentation.
 - Le type d'activité associé au matériel du module.
 - La semaine à partir de laquelle la matériel a été utilisé.
 - La semaine jusqu'à laquelle le matériel a été utilisé.

En Corée du Sud, un large jeu de données a été créé par [2] de 784 309 étudiants d'une application et un site WEB de MOOCs regroupant 1021 cours avec une moyenne de 411,20 interactions par étudiants offrant un total de 131 441 538 d'interactions disponibles dans la base de données. C'est l'une des plus grandes bases de données ouvertes des LA. Cette base de données est divisée en 4 niveaux de données, KT1, KT2, KT3 et KT4. KT2 contenant les données de KT1 plus des données supplémentaires, etc. Le détail de ces données étant disponibles sur le GitHub <https://github.com/riiid/ednet> Les données des différents niveaux sont les suivantes :

- KT1 : Ce jeu de données se concentre sur les réponses des étudiants aux questions et leurs temps de réponse. Chaque question fait partie d'un lot de questions qui traite d'une même lecture, image ou audio. Le contenu détaillé est le suivant :
 - Le moment où la question a été donnée suivant le formalisme de l'heure UNIX en millisecondes.
 - L'identifiant de la question.
 - L'identifiant du lot de questions.
 - La réponse de l'étudiant représentant sous la forme d'un caractère compris entre "a" et "d" correspondant à la réponse au QCM.
 - Le temps de réponse à la question en millisecondes.
- KT2 : Ce jeu de données se concentre sur ce qui a amené un étudiant à suivre un enseignement. Qu'est-ce-qui les a poussés à suivre cet enseignement plutôt qu'un autre, ainsi que la plateforme depuis laquelle ils ont suivi l'enseignement. Le contenu détaillé est le suivant :
 - Le type d'action réalisé, 3 types sont disponibles :
 - "enter" : Cette action est enregistrée à chaque fois qu'un étudiant accède à un lot de questions.
 - "respond" : Cette action est enregistrée à chaque fois qu'un étudiant sélectionne une réponse à une question. L'étudiant pouvant sélectionner plusieurs fois des réponses et la dernière sélection étant la seule considérée au moment de la soumission de la question.
 - "submit" : Cette action est enregistrée lorsque les étudiants soumettent les réponses à un lot de questions.
 - L'identifiant de l'objet impliqué dans l'action. Cet objet pouvant être une question ou un lot de questions.

- Une variable indiquant de quelle partie de l'interface de la plateforme est-ce que l'étudiant répond aux questions d'un lot ou regarde un cours. Plusieurs sources sont disponibles :
 - "sprint" : Cette source permet aux étudiants de choisir ce qu'il souhaite étudier. Ils ne peuvent alors répondre qu'aux questions des parties qu'ils choisissent jusqu'à ce qu'ils changent de parties ou qu'ils changent de source.
 - "todays_recommendation : :sprint" et "todays_recommendation : :review_quiz" : Cette source propose chaque jour des recommandations de cours et de questions en fonction des connaissances actuelles de l'étudiant.
 - "adaptive_offer" : Lorsqu'un seuil de mauvaises réponses à certains mots clés est dépassé, la plateforme suggère des questions et des cours correspondants aux mots clés.
 - "tutor" : Cette source correspond aux questions proposées aux étudiants suivant un algorithme de recommandation.
 - "in_review" : Cette source permet aux étudiants de répondre à nouveau à des questions qu'ils ont déjà eu précédemment.
- La réponse à la question lorsque le type d'action réalisé est "respond".
- La plateforme utilisée par l'étudiant pour répondre à la question, soit par mobile soit par le site WEB.
- KT3 : Ce jeu de données se focalise sur les enseignements et les explications des réponses aux questions données après que l'étudiant ait répondu. On peut ainsi extraire de ce jeu de données combien de temps un étudiant a étudié un enseignement. Le contenu détaillé est le suivant :
 - Les valeurs "enter" ou "quit" sont rajoutées aux types d'actions de KT2, indiquant si les étudiants entrent ou quittent la vue des explications aux questions, ou s'ils commencent ou arrêtent une vidéo de cours.
 - L'identifiant de l'objet impliqué dans l'action comprend également les identifiants des explications et des vidéos dans ce niveau.
 - De nouvelles valeurs pour les sources sont également disponibles pour les vidéos de cours :
 - "archive" : Cette source regroupe l'ensemble des vidéos de cours.
 - "todays_recommendation : :lecture" : Cette source propose des vidéos de cours par recommandation.
- KT4 : Ce jeu de données ajoute toutes les autres actions récoltées, notamment si l'étudiant a pris un forfait payant à la plateforme, ou bien s'il a mis en pause une vidéo ou un audio. Le contenu détaillé est le suivant :
 - Les étudiants peuvent cacher les réponses aux questions et les afficher à nouveau. Ce choix est représenté par les valeurs "erase_choice" et "undo_erase_choice".
 - Les valeurs prises par le type d'actions présent dans KT2 peuvent prendre 4 nouvelles valeurs, correspondant à si les étudiants ont mis en pause une vidéo ou un audio ou non.
 - Le moment auquel la vidéo ou l'audio a été mis en pause ou repris.
 - Les valeurs "pay", "refund", "enroll_coupon" sont ajoutées aux types d'actions. Ces valeurs correspondent aux offres payantes de la plateforme.

3 Techniques de prédiction employées dans la littérature

Comme l'on a déjà pu le voir dans la première partie de cet état de l'art, il existe de nombreuses techniques employées pour réaliser des prédictions dans les LA. Nous allons dans cette partie détailler quelques unes des techniques employées.

Nous allons dans un premier temps détailler 4 techniques proposées et comparer par [4]. Ces techniques sont les suivantes :

- **Multiple Linear Regression (MLR)** (Régression linéaire multiple) : Cette méthode vise à expliquer la dépendance d'une variable à d'autres variables indépendantes. Elle s'écrit mathématiquement comme suit :

$$Y_i = a_0 + a_1X_{i1} + a_2X_{i2} + \dots + a_pX_{ip} + \epsilon_i \quad \forall i \in \{1, n\}$$

où $Y_i, \forall i \in \{1, n\}$ sont les variables dépendantes à expliquer. $X_{i1}, \dots, X_{ip}, \forall i \in \{1, n\}$ sont les variables indépendantes explicatives et $\epsilon_i, \forall i \in \{1, n\}$ sont les erreurs du modèles. Enfin a_0, \dots, a_p sont les variables à estimer décrivant le lien entre les variables indépendantes et la variable dépendante.

- **Multilayer Perceptron (MLP)** (Perceptron multicouche) : Le perceptron multicouche est composé de 3 types de couches comme décrit dans la figure 3.3, la couche d'entrée, les couches cachées et la couche de sortie.

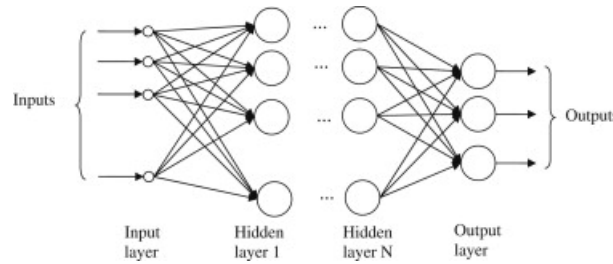


Figure 3.3 – Schéma d'un perceptron multicouche provenant de [4]

Chacune de ses couches possède un nombre variable de neurones, qui sont tous interconnectés d'une couche à l'autre. C'est-à-dire que chaque neurone d'une couche est connecté à tous les neurones de la couche précédente et de la couche suivante. Les données sont passées sous forme de nombres à la couche d'entrée, les neurones de la couche suivante sont alors calculés à l'aide d'une fonction d'activation, telle que les fonctions ReLu ou sigmoïd, jusqu'à atteindre la couche de sortie. Une erreur est calculée en fonction des valeurs obtenues dans la couche de sortie comparées aux valeurs attendues à l'aide d'une distance. La rétropropagation met alors à jour les poids de chaque couche en partant de la couche de sortie pour minimiser les erreurs.

- **Radial Basis Function (RBF)** network (Réseaux de fonctions de bases radiales) : Ce type de réseaux de neurones diffère des **MLP** par le fait qu'il possède systématiquement 3 couches, soit la couche d'entrée, la couche de sortie et une couche cachée et dont la fonction d'activation de la couche cachée est une fonction de base radiale, tandis que la fonction d'activation de la couche de sortie est une fonction linéaire. Il est également important de noter qu'il n'y a pas de rétropropagation dans ce type de réseaux de neurones.
- **Support Vector Machine (SVM)** (Machine à vecteurs de support) : Cette méthode de classification et de régression se base sur la théorie de Vapnik-Chervonenkis. Elle permet de séparer des données non linéaires en passant d'un ensemble de faibles dimensions à un ensemble avec beaucoup de dimensions où l'on pourra trouver une séparation linéaire des classes. Pour cela on utilise une fonction noyau, le plus souvent la fonction noyau utilisé est le noyau Gaussien :

$$K(x, y) = e^{-\frac{|x-y|^2}{2\sigma^2}}$$

où (x, y) sont les données du jeu d'entrée, et σ est un paramètre du modèle. La fonction de régression est alors donnée par :

$$f(x) = \sum_{i=1}^l (\alpha_i - \alpha_i^*) K(x_i \cdot x) + b$$

où les α sont des multiplicateurs de Lagrange et b est un paramètre du modèle.

4 Résultats obtenus dans la littérature

Nous allons dans cette partie traiter des résultats et des comparaisons de différentes techniques traitées dans la littérature. Il est important de rappeler que l'ensemble de ces résultats proviennent de base de données différentes, dans des pays et cultures différentes, avec des paramètres et des implémentations différentes, de ce fait, les résultats peuvent différer pour une même technique, et rien ne permet d'affirmer que si l'on met en place une technique efficace dans la littérature, elle le sera également avec un jeu de données de notre choix.

Nous allons dans un premier temps traiter de la comparaison des résultats réalisée par [4]. Ensuite nous traiterons d'un modèle de réseaux de neurones profond proposé par [5] en 2019, détaillant la structure de son modèle, ainsi que les jeux de données utilisés et les résultats obtenus.

Les **MLR**, **MLP**, **RBF**, **SVM** ont été comparés par [4]. Afin de comparer la précision des prédictions pour chaque modèle [4] a utilisé deux indicateurs :

- **Average Prediction Accuracy (APA)** (Précision moyenne des prédictions) : Il permet d'indiquer les performances moyennes de chaque modèle par la formule suivante :

$$APA = 1 - \frac{1}{n} \cdot \sum_{i=1}^n \left| \frac{P_i - A_i}{A_i} \right| \times 100$$

où n est la taille de l'échantillon, c'est-à-dire le nombre d'étudiants, P_i est le score prédit à l'examen du i ème étudiant et A_i est le score réel obtenu à l'examen par l' i ème étudiant.

- **Percentage of Accurate Predictions (PAP)** (Pourcentage de bonnes prédictions) : Cet indicateur est donné par le nombre total de bonnes prédictions divisé par le nombre total de prédictions réalisées. Dans cet article, une prédiction est considérée comme bonne avec une erreur de plus ou moins 10%.

Comme l'article présente les résultats pour différentes variables de prédictions, nous ne présentons ici que les pires et meilleurs résultats obtenus. Premièrement, on peut constater dans la figure 3.4 que les **APA** des 4 méthodes sont proches, la méthode des **SVM** étant la seule légèrement inférieure aux autres. L'indicateur bien que l'on ait ici les moins bons résultats sont relativement élevés, entre 87 et 90%. Toutefois, on peut constater que les **PAP** sont eux bien plus faibles, en étant compris entre 39,6 et 59,4%. On a donc de forts écarts d'une semaine à l'autre pour la même technique. On peut également constater qu'ici les **SVM** ont le meilleur pourcentage de réussite. Cela peut s'expliquer par le fait que les **SVM** lors de bonnes prédictions sont plus précis et qu'ils le sont plus régulièrement, mais lorsqu'ils font de mauvaises prédictions, ils sont plus éloignés de la note réelle que les autres techniques.

Model type	Average prediction accuracy (%)					Percentage of accurate predictions (%)				
	Sem. #1	Sem. #2	Sem. #3	Sem. #4	Four-sem. average	Sem. #1	Sem. #2	Sem. #3	Sem. #4	Four-sem. average
MLR	88.2	89.9	86.7	88.5	88.3	54.7	53.4	40.4	47.6	49.0
MLP	88.2	90.0	86.8	87.8	88.2	51.6	56.9	39.6	47.6	48.9
RBF	88.0	90.0	87.0	88.3	88.3	51.6	58.6	43.4	52.4	51.5
SVM	85.7	88.8	87.2	88.2	87.5	59.4	56.9	42.5	51.2	52.5

Figure 3.4 – Résultat des moins bonnes comparaisons obtenues par [4]

Dans le cas des meilleurs prédictions obtenues par [4] décrit dans la figure 3.5, on constate que

l'écart des **APA** pour les différentes techniques sont toujours faibles, avec 3% d'écart également, mais cette fois-ci avec une moyenne plus élevée aux alentours de 89%. Et l'on constate également cette augmentation des moyennes dans les **PAP**, qui vont cette fois-ci de 52,8 à 72,4%. L'écart de la moyenne de **PAP** de chaque technique est également ici plus élevé. En effet, le **MLP** semble moins bien performer avec 59,5% pour 64% pour les **SVM**.

Model type	Average prediction accuracy (%)					Percentage of accurate predictions (%)				
	Sem.	Sem.	Sem.	Sem.	Four-sem.	Sem.	Sem.	Sem.	Sem.	Four-sem.
	#1	#2	#3	#4	average	#1	#2	#3	#4	average
MLR	90.3	90.5	88.2	89.8	89.7	65.6	56.9	58.5	64.3	61.3
MLP	90.3	90.6	88.0	89.6	89.6	66.4	56.9	52.8	61.9	59.5
RBF	89.1	91.0	89.0	90.4	89.9	57.0	72.4	56.6	65.5	62.9
SVM	90.2	90.9	89.0	90.3	90.1	64.1	69.0	62.3	60.7	64.0

Figure 3.5 – Résultat des meilleures comparaisons obtenus par [4]

Globalement, sur l'ensemble des résultats fournis par [4], les **SVM** semblent être la meilleure méthode, et le **MLP** la moins bonne. Ce résultat n'est en soit pas surprenant puisque le jeu de données utilisé dans cet article ne comporte que 323 étudiants ce qui est faible pour entraîner un **MLP**.

Une prédiction de décrochage à l'aide d'un réseau de neurones profond a été proposé par [5] en 2019. Différentes architectures ont été proposées pour ce réseau, ainsi le nombre de couche cachées du réseau et la quantité de neurones à chaque couche sont variables.

Il y a ainsi soit 3, soit 5, soit 7 couches cachées selon les architectures. Et il y a soit 64, soit 128, soit 256, soit 512, soit 1024 neurones par couche.

Toutefois, certaines parties de l'architecture proposées par [5] restent fixes. Ainsi, le réseau prend 7 entrées et 2 sorties. La fonction de perte utilisée est l'entropie croisée et la fonction d'activation est ReLu, en ce qui concerne l'optimiseur, l'article a opté pour Adam. Enfin, le taux d'apprentissage est de 0.01 et la taille de chaque lot (ou batch) est de 1024.

Pour entraîner leur réseau, [5] a utilisé les bases de données HarvardX et MITx, qui comportent un total de 641 138 données d'individus anonymisés, provenant de cours ayant lieu d'automne 2012 à l'été 2013 sur la plateforme edX.

Cette base comporte 20 attributs, dont l'identifiant du cours et de l'étudiant, ainsi qu'un booléen permettant d'indiquer si un attribut est manquant ou non afin d'aider au prétraitement. Les données récoltées proviennent soit de données fournies par l'étudiant, soit par récupération automatique. Parmi ces attributs, 7 ont été choisis pour être utilisés par le réseau de neurones profond.

- Viewed
- Explored
- nEvent
- nDays_act
- nPlay_video
- nChapters
- nForum_posts

Malheureusement, le détail de ces attributs n'est pas fourni.

Enfin, l'attribut "Certified" agit comme le label à prédire, puisque cet attribut est un booléen indiquant si l'étudiant a complété ou non le cours. il vaut donc 1 lorsque l'étudiant a complété avec succès l'enseignement, et 0 lors d'un décrochage de l'étudiant.

Enfin le réseau a été entraîné, avec 60% des données ayant été réservées pour la base d'apprentissage, 25% pour la base de validation et 15% pour la base de test. Ainsi comme le montre la figure 3.6, [5] a ainsi obtenu une précision comprise entre 98,50% et 98,65% sur la base de test pour 3 couches cachées. Entre 97,46% et 98,63% pour 5 couches cachées. Et entre 98,28% et 98,55% pour 7 couches. Ces résultats sont très bons, toutefois il faut noter qu'ici lorsque l'on parle de décrochage, il s'agit de décrochage pour un enseignement, de plus les données d'entrées sont spécifiques aux MOOCs, ces données ne correspondent pas forcément à des données que l'on pourrait récupérer via les plateformes numériques utilisées dans l'enseignement supérieur français.

Table 3. Train, validate, and test accuracy results on the MOOC dataset [21] for DNN

Layers	Neurons	Train (%)	Validate (%)	Test (%)
3	64	98.73	98.44	98.50
3	128	98.54	98.39	98.61
3	256	98.59	98.41	98.65
3	512	98.69	98.41	98.56
3	1024	96.61	98.41	98.62
5	64	98.64	98.45	98.63
5	128	98.63	98.41	98.58
5	256	98.62	98.40	98.59
5	512	98.64	98.46	98.59
5	1024	97.53	96.26	97.46
7	64	98.50	98.00	98.28
7	128	98.55	98.27	98.49
7	256	98.55	98.20	98.43
7	512	98.48	98.38	98.49
7	1024	98.63	98.47	98.55

Figure 3.6 – Résultat pour différentes architectures du réseau proposé par [5]

Enfin, la matrice de confusion a été calculée pour chaque méthode, cette matrice permet de mesurer la qualité de la prédiction. Chaque ligne de la matrice correspond à une classe réelle et chaque colonne correspond à la classe prédite. Cette matrice permet ainsi d'avoir le nombre de vrais positifs, faux positifs, vrais négatifs et faux négatifs. Le taux de précision des décrochages et des étudiants ayant complété l'enseignement sont également calculés respectivement comme suit :

$$P(d)_{acc} = \frac{TP}{TP + FP} * 100$$

$$P(c)_{acc} = \frac{TN}{TN + FP} * 100$$

Avec :

- TP : Le nombre de vrai positif.
- FP : Le nombre de faux positif.
- FN : Le nombre de faux négatif.
- TN : Le nombre de vrai négatif.

Les résultats de la matrice de confusion et du taux de précision des décrochages sont disponibles et visibles dans la figure 3.7. Comme on peut le constater les taux de prédictions des décrochages sont très élevés, il y a très peu de faux positifs comparé aux vrais positifs, toutefois comme l'on peut également constater dans la figure 3.8 que le taux de prédictions des complétions des enseignements est faible. Le nombre de faux négatifs est en général 2 fois plus élevé que le nombre de vrais négatifs, le réseau a donc du mal à déterminer correctement les étudiants qui réussissent à terminer l'enseignement. Ce constat peut être problématique dans le cas de l'enseignement supérieur ou les décrochages bien que très élevés restent minoritaires comparés aux réussites, tandis que dans le cas de cet article, une écrasante majorité des étudiants abandonne

l'enseignement. Il y a donc un risque que ce qui se passe dans cet article pour les étudiants réussissant l'enseignement se passe pour les étudiants en décrochage dans notre cas.

Table 7. Dropout prediction accuracy on the MOOC dataset [21] for DNN

Hidden Layers	No. of Neurons	TP	FP	TN	FN	P(d)acc
3	64	93142	846	596	1588	99.10
3	128	92915	505	823	1929	99.46
3	256	92887	445	851	1989	99.52
3	512	93086	727	652	1707	99.23
3	1024	92895	484	843	1950	99.48
5	64	93011	589	727	1845	99.37
5	128	92695	314	1043	2120	98.66
5	256	92925	534	813	1900	99.43
5	512	92979	595	759	1839	99.36
5	1024	93738	2434	0	0	97.47
7	64	92277	185	1461	2249	99.80
7	128	92540	246	1198	2188	99.73
7	256	92494	261	1244	2173	99.72
7	512	92557	264	1181	2170	99.72
7	1024	93040	690	698	1744	99.26

Figure 3.7 – Matrice de confusion et taux de précision des décrochages pour les différentes architectures.

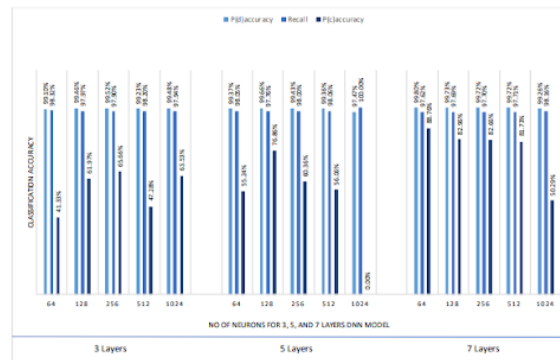


Figure 3.8 – Histogramme des taux de précision pour les différentes architectures.

5 Techniques de visualisation employés dans la littérature

L'utilisation de visualisation des données est très important dans le domaine des LA, celle-ci permette lors d'un bon traitement des données d'obtenir des informations complémentaires par rapport aux analyses réalisées mais peut également permettre de visualiser les performances d'une technique d'analyse, le tout en très peu de temps, grâce à l'aspect graphique rendant la compréhension très accessible.

Dans [11], l'intérêt de l'utilisation des tableaux de bord est discuté, ainsi que les phases du processus de fonctionnement de ces derniers. L'article fini également par décrire les résultats entre deux classes dont l'une utilise des tableaux de bord et l'autre non. A noter qu'ici encore, l'enseignement se fait à distance.

Pour présenter l'intérêt des tableaux de bords, 4 aspects sont décrits.

- L'aspect descriptif : Les tableaux de bords permettent la consultation des activités lors du suivi d'un cours en ligne.

- L'aspect prédictif : Les tableaux de bords peuvent permettre de représenter visuellement des étudiants ayant des chances de réussir l'enseignement ou non.
- L'aspect du diagnostic : Les tableaux de bords peuvent permettre dans certaines mesures d'approcher les raisons du décrochage d'un étudiant.
- L'aspect d'action : Les tableaux de bords peuvent permettre de récupérer des informations aidant à la décision de la mise en place d'action réalisables pour améliorer les progrès des étudiants dans un enseignement.

Afin d'expliquer le processus de construction des tableaux de bords, [11] décrit le processus en 5 phases. Ces phases sont très proches de celles vu précédemment dans la figure 3.2.

- Phase de récupération des données : Ces données sont récupérées par différentes bases de données, ou par des logs, ou les deux combinés.
- Phase d'analyse : Afin d'analyser les traces des activités des étudiants, 6 catégories d'indicateurs ont été créées :
 - Enseignement : Cette catégorie donne les informations générales sur les enseignements. 3 indicateurs ont été choisis dans [11] :
 - Nombre d'étudiants engagés dans l'enseignement.
 - Nombre de sections de cours planifiés.
 - Nombre d'activités créées.
 - Participation : Cette catégorie se focalise sur les actions qui permettent de considérer un étudiant comme actif. 2 indicateurs ont ici été choisis :
 - Les actions de consultation.
 - Les actions de contribution.
 - Section : Deux indicateurs sont utilisés pour calculer le niveau de progrès des étudiants dans chaque section de l'enseignement :
 - Les activités/ressources consultées par les étudiants dans chaque section.
 - Le nombre d'activités/ressources disponibles dans chaque section.
 - Progression : Trois indicateurs permettent d'expliquer le progrès d'un étudiant dans un enseignement :
 - Le nombre d'activités déjà complétées par l'étudiant.
 - Le nombre d'activités non complétées passé une date butoir.
 - Le nombre d'activités définies par l'enseignant au début de l'année scolaire.
 - Succès : Cette catégorie propose une estimation des performances d'un étudiant dans l'enseignement à distance suivant le modèle e-LSAM décrit dans [10]
- Phase de préparation des données : Cette phase permet la transformation et la préparation des données essentielles pour l'outil afin de générer des données au format JSON. Ces données préparées peuvent être utilisées par d'autres plateformes puisque les données sont standardisées.
- Phase de rapport : Ces rapports récupèrent les données nécessaires des fichiers JSON afin de proposer deux catégories de visualisation. La visualisation de rapports destinée aux étudiants, et celle destinée aux enseignants :
 - Trois interfaces sont proposées aux étudiants. Une première interface positionne les étudiants sur chaque section du cours ainsi que deux autres niveaux : Le niveau de progression du meilleur étudiant et le niveau moyen des étudiants de la classe. Un classement des étudiants de classe est également affiché. Une seconde interface permet aux étudiants de regarder le détail des progrès réalisés sur un cours. Enfin une troisième interface permet aux étudiants de consulter les notifications ayant été envoyées par la plateforme.
 - Le rapport des enseignants présente de son côté 4 interfaces. Premièrement une interface présente des données statistiques issues des indicateurs de la catégorie "Enseignement", ainsi que des graphiques résumant la consultation de chaque section par les étudiants de la classe. Une seconde interface propose un tableau montrant les questions-réponses

ayant été répondues et validées ou non par les élèves. Ensuite, une autre interface résume les devoirs que les étudiants ont la possibilité de rendre à l'instant donné. Enfin, la dernière interface affiche la liste des étudiants avec une estimation du temps passé par les étudiants sur les enseignements, ainsi qu'un indicateur du niveau de réussite et un statut de prédiction de décrochage ou de réussite.

- Phase d'action : Cette phase permet à l'enseignant de mettre en place de lui-même des actions d'aides aux étudiants, ou bien de paramétrer des notifications automatiques visant à alerter les étudiants sur différentes actions qu'ils peuvent être amenés à réaliser pour s'améliorer.

Comme énoncé précédemment, des résultats ont été présentés dans cet article. Une classe a été divisée en deux groupes de 13 personnes, un groupe ayant accès à l'outil, l'autre non. Les étudiants ayant eu accès à l'outil ont ainsi passé 3 fois plus de temps sur le cours en ligne (46h) que le second groupe (16h). Le taux moyen de réussite est également plus de deux fois plus élevé (85% contre 37%). La quantité de travail rendue ou rendue en retard est également plus importante dans le groupe disposant de l'outil (44% contre 11% de travail rendu à l'heure pour ceux disposant de l'outil, et 27% contre 15% de travail rendu en retard). Nécessairement, au vu des pourcentages précédents, la quantité de travail non rendue est quant à elle bien plus importante dans le groupe ne disposant pas de l'outil, avec 28% pour ceux disposant de l'outil contre 37% pour ceux n'en disposant pas.

Ces résultats montrent donc un fort intérêt de l'outil, toutefois, il est bon de noter, que premièrement l'outil peut être assez intrusif dans la vie de l'étudiant, notamment par l'utilisation de notifications. Mais également que ces résultats ont été réalisés sur un petit groupe facilement manipulable en mettant les étudiants les plus forts dans un groupe et les moins forts dans un autre.

4

Analyse et conception

1 Analyse

L'état de l'art précédent a permis de constater qu'il existe de nombreuses techniques utilisables pour prédire des décrochages. Dans cette section nous allons déterminer quelle technique serait la plus intéressante à utiliser. Il est également nécessaire de sélectionner une base de données adaptée à notre problème, ce choix sera détaillé dans une seconde partie.

1.1 Choix de la technique

Un compte des mots-clés présents dans les publications de la littérature anglaise jusqu'en 2017 a été réalisé par [9]. Ce compte est visible sur la figure 4.1

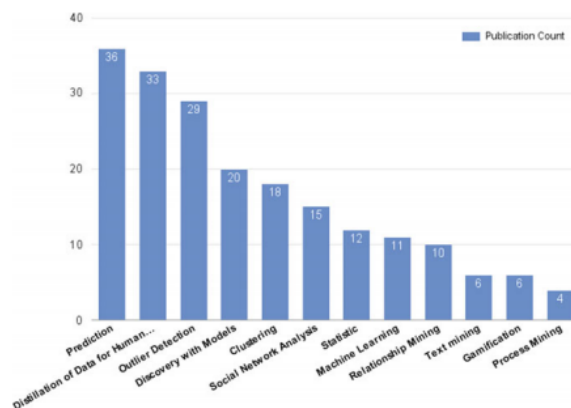


Figure 4.1 – Compte des mots-clés présents dans la littérature anglaise

Comme l'on peut le voir, malgré l'utilisation croissante du ML ces dernières années, le mot-clé n'apparaît que 11 fois comparé à d'autres domaines comme le clustering ou l'analyse de réseaux sociaux qui apparaissent 18 et 15 fois chacun. D'autres domaines sont encore moins présents dans la littérature, comme la fouille de texte, toutefois ce genre de méthodes n'est pas d'une grande utilité dans notre cas, car elles s'appliquent principalement aux forums, ou à des questions-réponses sur des plateformes numériques. Or l'utilisation principale des plateformes numériques

dans les universités françaises est le dépôt de notes, les QCMs, et les dépôts de supports de cours. Cela fait du **ML** l'un des domaines le moins utilisé pouvant être utilisé dans notre cas. Toutefois, les méthodes de **ML** demandent d'avoir accès à de grande quantité de données. Heureusement, depuis ces 5 dernières années, de nombreux jeux de données ont été rendus publics, notamment grâce à l'utilisation grandissante des **MOOCs**. Ces derniers proposent comme l'ont déjà pu le voir dans l'état de l'art des données qui ne sont pas forcément compatibles avec l'utilisation des plateformes numériques dans l'enseignement supérieur français. Toutefois on peut retrouver des points communs, tels que les notes aux examens, les actions réalisées à des QCMs, le téléchargement de supports de cours, etc. Il est donc envisageable d'extraire des bases de données de ces **MOOCs**, des données qui peuvent être communes avec l'enseignement supérieur afin d'entraîner un modèle de réseau de neurones sur un large jeu de données, et réaliser ensuite un second entraînement sur des données plus petites de l'enseignement supérieur français en mettant en place du **Transfer Learning**.

Cette méthode permet d'entraîner un modèle sur un petit jeu de données en initialisant le modèle avec des valeurs obtenues sur un modèle déjà fonctionnel le plus souvent ayant été entraîné sur un très grand jeu de données.

De cette manière il est envisageable de proposer un modèle malgré les assez petites données disponibles dans l'enseignement supérieur comparé aux MOOCs.

De plus, il est bon de noter qu'un modèle peut très bien fonctionner sur une filière mais être totalement inefficace sur une autre, du fait que les notes, les interactions avec les plateformes peuvent être totalement différentes selon les filières, notamment selon comment les étudiants et les enseignants vont avoir tendance à utiliser ces plateformes numériques. Des étudiants en informatique seront probablement plus enclin à les utiliser que des étudiants d'autres filières. De ce fait l'utilisation de **Transfer Learning** pourrait permettre la création de différents modèles selon les filières. On pu voir d'autres méthodes de **ML** précédemment. Notamment la factorisation matricielle, la régression linéaire multiple ou les machines à vecteurs de support.

La factorisation matricielle a deux inconvénients majeurs. Premièrement, cette technique peut rapidement prendre beaucoup de temps selon la taille des matrices, il faut donc éviter d'avoir de trop nombreuses caractéristiques. De plus, de nombreux optimum locaux sont accessibles avec cette méthode, ce qui complique la tâche de recherche d'un optimum global.

La régression linéaire multiple a besoin d'avoir des données complètes, les données doivent être représentatives de la réalité. Toutefois, cette méthode consiste à représenter un modèle multilinéaire avec une droite. De ce fait des points aberrants, tels que des notes très basses d'une personne en décrochage, ou au contraire des notes très hautes d'un individu comparé aux autres, pourraient fausser le modèle.

Les machines à vecteurs de support ont prouvé qu'elles pouvaient être très efficaces avec de grands jeux de données, et avec de très bons résultats. Toutefois comme ces dernières années les réseaux de neurones ont tendance à surpasser les performances des SVM, il paraît plus intéressant de se pencher sur les réseaux de neurones, malgré le temps d'apprentissage qui peut être assez conséquent. Toutefois, comme l'on ne nécessite pas d'avoir un apprentissage en temps réel, on peut se permettre d'avoir un apprentissage de quelques heures.

Enfin, un avantage du réseau de neurones que possèdent aussi d'autres techniques est qu'il est aisé de charger le modèle sauvegardé et de rajouter de nouvelles données d'entraînement. Ce nouvel entraînement pourrait se réaliser dès que l'on a obtenu un certain nombre de données. Et l'entraînement pourrait se faire automatiquement en validant le nouveau modèle s'il performe mieux que le précédent.

1.2 Sélection de la base de données

Pour choisir quelle base de données l'on va utiliser, il est important de définir ce que l'on recherche dans cette base de données. Premièrement, puisque l'on va ici entraîner un réseau de neurones, il

est nécessaire d'avoir accès à un jeu de données assez important. Deuxièmement, il est nécessaire de pouvoir obtenir directement ou par différents calculs si un étudiant a décroché ou non. Ensuite, les données qui vont être utilisées doivent être sélectionnées et la base de données doit le plus s'en rapprocher, en tenant compte du fait que les données peuvent être représentées de différentes manières et que l'une de ces manières peut être privilégiée.

On a pu voir dans la partie État de l'art (3) plusieurs bases de données publiques. La base LAK issue de [3], la base de données OAAI de [1], la base Open University Learning Analytics de [7], et enfin la base EdNet de [2].

Premièrement, la base de données OAAI ne prenant en compte que 1073 étudiants est probablement à écarter. De plus, cette base de données ne comporte aucune donnée permettant de déterminer si un étudiant est en situation de décrochage ou non. Il n'y a aucune variable directe, ni de variable de temps qui permettent de déduire un décrochage.

Ensuite, la base LAK se focalise sur les travaux rendus et les notes obtenues, on peut également déduire un potentiel décrochage grâce au pourcentage de travail rendu et grâce au nombre de jour depuis le dernier travail rendu. Mais il est nécessaire dans ce cas de faire des hypothèses qui ne seraient pas forcément représentatives de la réalité.

La base EdNet n'apporte pas non plus de données permettant de déterminer quels ont été les étudiants qui ont décroché, ni même de notations.

Il ne reste donc plus que la base Open University Learning Analytics de [7], celle-ci permet bien d'obtenir l'information de si un étudiant a raté son examen, en considérant l'examen raté en dessous d'un score de 40. Il existe également une variable qui indique si l'étudiant s'est désinscrit du module ou s'il l'a complété. On peut donc envisager deux prédictions différentes. Une qui va prédire si l'étudiant va réussir l'examen final d'un module en particulier. Et l'autre qui va prédire si l'étudiant va compléter le module ou non. Il est important de noter que cette base de données comporte de très nombreuses caractéristiques différentes, une sélection des bonnes caractéristiques à utiliser est donc ici très importante.

Les caractéristiques suivantes ont donc été gardées :

- L'identifiant de l'étudiant : Cet identifiant n'est pas passé en entrée au réseau de neurones, mais il est nécessaire pour associer le résultat du réseau à l'étudiant.
- L'identifiant du module : Cet identifiant sert principalement pour la visualisation afin d'associer le module au résultat et pour le pré-traitement des données pour ne passer au réseau de neurones que les informations associées à un seul étudiant et un seul module. On peut ici associer un module à une matière dans l'enseignement supérieur.
- L'identifiant de l'évaluation : Cet identifiant sert pour la visualisation et pour le pré-traitement des données.
- Les résultats finaux de l'étudiant dans les modules : Ces caractéristiques indiquant si un étudiant a raté, abandonné ou réussi un module sera le label, c'est-à-dire le résultat réel que l'on va comparer au résultat prédit afin d'entraîner notre réseau de neurones, dans le cas où l'on veut prédire les décrochages. Il ne sera pas utilisé sinon dans le cas où l'on souhaite prédire si l'étudiant va réussir ou non une évaluation.
- La durée en jour du module : Cette caractéristique peut être associée au nombre d'heures alloué à un enseignement dans les universités et peut potentiellement avoir un impact sur l'entraînement du réseau de neurones.
- Le type d'évaluation : Cette caractéristique est importante puisque le type d'évaluation peut avoir une incidence sur la difficulté de celle-ci.
- Le coefficient de l'évaluation : Ce coefficient pouvant influencer la quantité de travail réalisé en amont de l'évaluation par l'étudiant, il peut influencer la note qui sera obtenue.
- Le résultat de l'étudiant à une évaluation : Cette note comprise entre 0 et 100 sert de label dans le cas où l'on souhaite prédire le résultat à une évaluation. Elle peut également servir dans le cas de précédentes évaluations pour mieux déterminer si l'étudiant va réussir l'évaluation suivante ou non.

- La présentation associée à un module suivi par l'étudiant : Cette caractéristique est l'équivalent des semestres suivis par une promotion. Comme les résultats d'une promotion à une autre peuvent différer, cette caractéristique doit être prise en compte.

Certaines variables doivent également être créées à partir des caractéristiques précédentes :

- La moyenne des étudiants à une évaluation d'une présentation.
- La moyenne de l'étudiant sur l'ensemble des évaluations de la présentation.
- La moyenne des étudiants sur l'ensemble des évaluations d'une présentation.

Il est important que la quantité de décrochages ne soit pas déséquilibrée par rapport à la réalité, entraîner un modèle qui a 90% de décrochage n'aurait pas de sens lorsque l'enseignement supérieur a entre 20% et 30% de décrochage. J'ai donc calculé les statistiques des résultats aux examens finaux des étudiants de la base OULA, en prenant en compte les étudiants ayant réussi leur examen final et ceux ayant eu une mention, ainsi que ceux n'ayant pas passé l'examen et ceux qui l'ont raté.

	Réussi	Mention	Non passé	Raté
Quantité	12361	3024	10156	7052
Pourcent	37.9	9,3	31.2	21.6

Comme l'information de si les étudiants ayant raté, ont redoublé ou non, n'a pas été communiquée, on considère également ces étudiants en décrochage. On concatène donc ces données pour obtenir le nombre d'étudiant en décrochage et le nombre d'étudiant passant avec succès.

	Succès	Décrochage
Quantité	15385	17208
Pourcent	47,2	52,8

Comme on peut le constater le nombre de décrochage est plus élevé que dans l'enseignement supérieur avec 52.8% de décrochage. Les MOOCs étant plus sujets aux décrochages, ce chiffre n'est pas surprenant. Il n'est toutefois pas si élevé, et dans le cas où on considère que les étudiants ayant non passé l'examen final redoublent, alors on tombe même à 31.2% de décrochage ce qui se rapproche de la borne haute des décrochages dans l'enseignement supérieur. Toutefois, on tentera dans un premier temps de réaliser le modèle en comptant les étudiants ayant raté comme étant en décrochage puisqu'on veut minimiser les situations qui pourraient conduire à un décrochage.

1.3 Spécifications

On a déjà précisé précédemment que ce projet se décompose en 2 parties : la partie entraînement du réseau de neurones et la partie visualisation via une plateforme numérique. On découpe donc en conséquence les spécifications en deux parties également. Une description plus détaillée des spécifications est disponible dans la partie 1 (Annexe C)

1.3.1 Partie réseau de neurones

On distingue 3 fonctionnalités principales dans cette partie.

- L'importation de la base de données : Cette fonction importe dans le système l'ensemble des fichiers .csv de la base OULA utile au projet.
- Le traitement et la transformation des données : Cette fonction convertit les données importées en données numériques qui sont ensuite transformées sous la forme de tenseur afin d'être passées en entrée du réseau de neurones.
- L'entraînement et la sauvegarde du réseau de neurones : Le réseau de neurones est entraîné sur les données traitées et transformées de la base OULA. Les données de la base sont divisées en 3 bases : la base d'entraînement, la base de validation et la base de test. Le réseau de neurones n'est sauvegardé que lorsque le pourcentage de bonne prédiction sur la base de validation dépasse le pourcentage du précédent modèle.

1.3.2 Partie plateforme numérique

On distingue également 3 fonctionnalités principales dans cette partie.

- L'authentification : Cette fonction permet aux utilisateurs de se connecter. Ces derniers ne peuvent avoir accès à la vue qui leur est associée que s'ils sont connectés. Les étudiants peuvent voir les prédictions pour chacun des modules auxquels ils sont inscrits. Comme aucun enseignant n'est associé aux modules dans la base, on considérera qu'un module représente un enseignant, et lorsque l'on se connecte avec ce numéro de module, on a donc accès à l'ensemble des visualisations des étudiants de ce module. Dans le cadre de projet un mot de passe unique sera défini pour l'ensemble des utilisateurs.
- L'importation des données : Cette fonction permet de récupérer les données des prédictions, ainsi que les données annexes nécessaires à la visualisation.
- La visualisation des prédictions : Cette fonction permet d'afficher les visualisations associées à un utilisateur.

2 Modélisation proposée

Dans cette partie, nous allons décrire la modélisation proposée pour ce projet. Nous allons dans un premier temps traiter des données, notamment du traitement de ces dernières et leurs transformations en tenseur. Ensuite nous traiterons de l'architecture du réseau de neurones.

2.1 Traitement et transformation des données

Lors de l'importation des données, on récupère 4 listes contenant les données issues des 4 fichiers .csv utilisés. Ces 4 listes sont les suivantes :

- La liste des résultats finaux des étudiants [code_module, code_presentation, id_student, final_result]
- La liste des notes des étudiants aux évaluations [id_assessment, id_student, score]
- La liste des évaluations des présentations des modules avec leurs types d'évaluation [code_module, code_presentation, id_assessment, assessment_type]
- La liste des présentations des modules avec leurs durées [code_module, code_presentation, module_presentation_length]

Pour rappel, on souhaite dans un premier temps déterminer si un étudiant réussira l'examen d'une présentation d'un module, ce qui correspond à la donnée "final_result" disponible dans la liste des résultats finaux des étudiants. Dans un second temps on aimerait pouvoir déterminer si un étudiant est en décrochage ou non. Pour ce faire, avec les données que l'on possède, on peut considérer qu'un étudiant est en décrochage lorsqu'il a la moitié ou plus des examens finaux des

modules auxquels il est inscrit qui sont ratés.

Une donnée d'entrée correspond donc à une présentation d'un module suivi par un étudiant. Le triplet (`code_module`, `code_presentation`, `id_student`) détermine donc une entrée. De cette association il est possible de récupérer à partir de la liste des évaluations des présentations des modules avec leurs types d'évaluation la liste de double suivante [(`id_assessment`, `assessment_type`)]. L'identifiant de l'évaluation "`id_assessment`" permet également de récupérer les notes de l'étudiant aux évaluations à l'aide de "`id_student`". On obtient alors la liste de triplet suivant : [(`id_assessment`, `assessment_type`, `score`)]. Enfin la dernière variable utilisée dans une donnée "`module_presentation_length`" est obtainable depuis la liste des présentations des modules avec leurs durées.

On a alors la liste suivante représentant une donnée, [`code_module`, `code_presentation`, `id_student`, `module_presentation_length`, [`id_assessment`, `assessment_type`, `score`]]. Il faut également prendre en compte des variables à créer à partir des caractéristiques importées de la base de données comme on a pu le spécifier partie 1.2. Ces variables sont les suivantes :

- La moyenne des étudiants à une évaluation d'une présentation, que l'on identifiera par "`mean_eval`".
- La moyenne de l'étudiant sur l'ensemble des évaluations d'une présentation, que l'on identifiera par "`mean_student`".
- La moyenne des étudiants sur l'ensemble des évaluations d'une présentation, que l'on identifiera par "`global_mean`".

On obtient ainsi cette nouvelle représentation des données, [`code_module`, `code_presentation`, `id_student`, `module_presentation_length`, `mean_student`, `global_mean`, [`id_assessment`, `assessment_type`, `score`, `mean_eval`]].

On fait désormais face à deux points complexes. Premièrement, comment convertir l'ensemble des données en données numériques. Et deuxièmement, le réseau de neurones prenant des tenseurs de taille fixe en entrée, quelle doit être la taille maximale de la liste [`id_assessment`, `assessment_type`, `score`, `mean_eval`]. C'est à dire qu'il faut choisir arbitrairement le nombre maximal d'évaluation par présentation. Il faut choisir un nombre suffisamment grand pour que toutes les données puissent être exploitées mais pas trop pour ne pas ralentir l'apprentissage du réseau de neurones.

Conversion en données numériques

Les données importées respectent toutes un format :

- `code_module` : "XXX" où "X" est une lettre.
- `code_presentation` : "0000X", où "0000" est l'année de début de la présentation et "X" est une lettre représentant le mois de début de la présentation. "A" pour Janvier, "B" pour Février, etc ...
- `id_student` : "000000", nombre à 6 chiffres. Pas de conversion nécessaire.
- `module_presentation_length` : "000", durée en jours d'une présentation. Pas de conversion nécessaire.
- `id_assessment` : "00000", Nombre à 4 ou 5 chiffres. Pas de conversion nécessaire.
- `assessment_type` : "XXXX", Mot indiquant le type d'examen.
- `score` : "00", nombre compris entre 0 et 100. Une note de 0 ne serait pas pris en compte par le réseau de neurone, on rajoute donc 1 à la note pour s'assurer qu'un 0 soit pris en compte par le réseau.

Les variables créées quant à elles sont des moyennes, elles n'ont pas besoin d'être converties. Toutefois par soucis d'homogénéité, les moyennes seront réhaussées d'un point pour correspondre au réhaussement de la note "`score`".

Les données ayant besoin d'être converties sont donc "`code_module`", "`code_presentation`" et "`assessment_type`". "`code_module`" peut être converti aisément en convertissant chaque caractère

en son nombre équivalent dans la table ASCII et en concaténant ces nombres, ainsi "AAA" devient 656565 car "A" vaut 65 dans la table ASCII, "BBB" devient 666666, etc ...

"assessment_type" peut également être converti aisément en assignant aux différentes valeurs possibles un chiffre, "TMA" prenant la valeur 1, "CMA" la valeur 2, etc...

Enfin pour "code_presentation" il faut prendre en compte le fait que l'on ait déjà une année dans la variable en plus d'une lettre. Une possibilité est de multiplier par 100 l'année, puis de rajouter la valeur correspondant à la lettre, 1 pour A, 2 pour B, etc ... Ce qui donne par exemple pour "2013A" : "201301" ou pour "2013J" : 201310. On a de cette manière converti l'ensemble des données en données numériques.

Choix de la quantité maximale d'évaluation

On a souligné précédemment que la taille des données en entrée du réseau de neurones est fixe. Cela implique qu'il faut choisir un nombre maximal d'évaluations acceptées en entrée. Dans tous les cas, si l'on a moins d'évaluations que le nombre défini, cela n'impacte pas le réseau de neurones puisqu'il suffit d'initialiser arbitrairement à 0 l'ensemble des valeurs associées aux évaluations pour compléter le réseau de neurones. Afin de pouvoir déterminer cette quantité maximale, la donnée concernant le nombre d'évaluations ayant eu lieu par présentation a été récupérée. La quantité maximale trouvée a été de 14. On obtient ainsi $6 + 3 \times 14$ données soit 48 données en entrées du réseau de neurones, ce qui n'est pas trop élevé. On choisira donc cette quantité pour le réseau de neurones.

5

Mise en oeuvre

1 Outils et librairie utilisés

L'entièreté des développements liés à l'entraînement du réseau de neurones ont été réalisés en Python.

Le réseau de neurones a été réalisé avec Pytorch.

2 Éléments d'implémentation, choix techniques

Pour la partie entraînement du réseau de neurones, le choix a été fait de séparer chaque fonctionnalité en un fichier composé d'une ou plusieurs classes.

On a ainsi :

- La classe `DataParser` dans `DataParser.py` pour la fonctionnalité "Importation des données".
- La classe `DataConverter` dans `DataConverter.py` pour la fonctionnalité "Traitement et transformation des données".
- Les classes `NeuralNetwork` et `ConvertedData` dans `NeuralNetwork.py` pour la fonctionnalité "Entraînement et sauvegarde du réseau de neurones", ainsi que le main d'exécution du réseau de neurones.

Pour chaque fichier, un fichier de test a également été créé, excepté pour le fichier `NeuralNetwork.py` par faute de temps.

Classe `DataParser`

Cette classe récupère les données issues de la base de données OULA. Pour ce faire, il faut initialiser la classe avec comme paramètre le chemin vers le dossier contenant l'ensemble des données CSV.

L'appel à la fonction "parse" permet de récupérer les données. Cette fonction vérifie que le chemin spécifié est bien un dossier et que l'ensemble des fichiers CSV nécessaires sont bien dans le dossier. Chaque fichier est alors ouvert et lu ligne par ligne. Chaque champ du fichier que l'on souhaite conserver est alors renvoyé dans une liste, et cette dernière est ajoutée à l'attribut correspondant au fichier. On obtient ainsi 4 listes de listes pour les 4 fichiers d'où sont récupérées les données.

Classe DataConverter

Cette classe traite les données d'une instance de DataParser. Cette instance doit être passé comme paramètre lors de l'initialisation de la classe.

Trois fonctions sont notables :

- La fonction "convertData" qui permet de traiter et transformer les données.
- La fonction "saveConvertedData" qui permet de sauvegarder les données transformées dans un fichier csv.
- La fonction "loadConvertedData" qui permet de charger les données transformées depuis un fichier csv.

La méthode convertData, vérifie dans un premier temps que l'attribut "parsedData" est bien du type DataParser, sinon une exception est levée. Dans le cas où l'attribut est du bon type, on parcourt l'ensemble de l'attribut "final_results" qui contient le résultat final d'un étudiant à une présentation d'un module. Ces données permettent de récupérer dans un premier temps la durée de présentation d'une présentation, et les évaluations liées à cette évaluation. La liste des évaluations est retournée sous la forme [id_assessment, type_assessment], ces informations vont être utilisées pour récupérer les scores de chaque évaluation ainsi que la moyenne de l'évaluation à la condition que la liste des évaluations ne soit pas vide. La moyenne de l'étudiant sur cette présentation et la moyenne des étudiants sur l'ensemble des évaluations est également récupérée puis l'ensemble est converti au format numérique. Dans le cas où la liste des évaluations est vide, alors on initialise directement les variables converties avec 0 ou la liste vide. Ensuite on convertit le code du module, de la présentation et du label, c'est à dire du résultat de l'étudiant, au format numérique et l'on ajoute l'ensemble des données exceptés les évaluations à la liste des données converties. Pour terminer, on ajoute les données liées aux évaluations à la liste des données converties et on complète avec des 0 pour obtenir 14 évaluations qui est le maximum d'évaluations pour une présentation. Cette liste des données converties est ensuite ajoutée à la liste contenant l'ensemble des données. Cette fonction est donnée en détail dans le code source suivant :

```

1 def convertData(self):
2     if isinstance(self.parsedData, DataParser):
3         for final_result in self.parsedData.final_results:
4             code_module, code_presentation, id_student = final_result
5             [:3]
6             module_presentation_length = self.
7             getModulePresentationLength(final_result)
8             assessments = self.getAssessments(final_result)
9             if assessments != []:
10                 assessments = self.addScoresAndMeanEval(final_result,
11                 assessments)
12                 mean_student = self.getMeanStudent(assessments)
13                 global_mean = self.getGlobalMean(assessments)
14                 converted_mean_student = self.convertMeanAndScore(
15                 mean_student)
16                 converted_global_mean = self.convertMeanAndScore(
17                 global_mean)
18                 converted_assessments = self.convertAssessments(
19                 assessments)
20             else:
21                 converted_mean_student = 0
22                 converted_global_mean = 0
23                 converted_assessments = []
24             converted_code_module = self.convertCodeModule(

```

```

19         code_module)
20     converted_code_presentation = self.
21         convertCodePresentation(code_presentation)
22     converted_label = self.convertLabel(final_result[3])
23     converted_data = [[converted_code_module,
24                         converted_code_presentation,
25                         int(id_student), int(
26                             module_presentation_length)
27                         ,
28                         converted_mean_student,
29                         converted_global_mean],
30                     converted_label]
31     for index in range(14):
32         for indexj in range(4):
33             if converted_assessments != [] and index < len(
34                 converted_assessments):
35                 converted_data[0].append(
36                     converted_assessments[index][indexj])
37             else:
38                 converted_data[0].append(0)
39     self.convertedData.append(converted_data)
40 else:
41     raise TypeError("Wrong type for instance 'parsedData'.")

```

Code source 5.1 – Méthode *convertData*

Classe **NeuralNetwork**

Cette classe hérite de `torch.nn.Module`. Elle permet de définir la structure du réseau de neurone. Elle est uniquement composée de son constructeur et d'une méthode. Son constructeur permet de définir le nombre de couches cachées et le nombre de neurones par couches. La méthode "forward" permet la propagation des données à travers les différentes couches et fonctions d'activations du réseau.

Classe **ConvertedData**

Cette classe hérite de `torch.utils.data.Dataset`. Elle permet de définir notre base de données dans un format acceptable par un objet `DataLoader`. Les méthodes "`__len__`" et "`__getitem__`" sont redéfinies dans cette classe. Elles permettent respectivement de retourner la quantité de données, et la donnée et le label à un indice donné sous la forme d'un tenseur.

Exécution de l'algorithme

Dans le main du projet, on définit dans un premier temps la répartition des bases. Les données sont ensuite récupérées, soit en chargeant ces dernières avec le code suivant :

```

1 dataParser = DataParser("Data/")
2 dataParser.parse()
3 dataConverter = DataConverter(dataParser)
4 dataConverter.loadConvertedData("savedConvertedData.csv")

```

Code source 5.2 – Chargement des données

Soit en construisant ces données avec le code suivant :

```

1 dataParser = DataParser( "Data/" )
2 dataParser.parse()
3 dataConverter = DataConverter( dataParser )
4 dataConverter.convertData()
5 dataConverter.saveConvertedData( "savedConvertedData.csv" )

```

Code source 5.3 – Construction des données

La première option est toutefois conseillé puisque la construction des données peut prendre quelques heures.

Une fois les données récupérées on calcule la quantité de données dans la base d'entraînement, de validation et de test et on construit cette base avec la classe `ConvertedData` et un objet `DatalLoader` de `torch.utils.data`

On définit alors le modèle, la fonction de pertes et l'optimiseur comme suit :

```

1 model = NeuralNetwork()
2 CELoss = torch.nn.CrossEntropyLoss( weight=train_dataset.class_weights
   , reduction='mean' )
3 optimizer = torch.optim.Adam( model.parameters() , lr=0.001 , betas
   =(0.9 , 0.999) )

```

Code source 5.4 – Définition du modèle

On initialise alors les variables d'entraînement et l'on réalise l'entraînement du modèle, en conservant le modèle, uniquement si la précision obtenu sur la base de validation est meilleure que la précision du modèle précédemment conservé. Comme l'on peut le constater sur le code source ci-dessous, on entraîne le modèle sur l'ensemble des données de la base d'apprentissage, puis on évalue ce modèle sur l'ensemble de la base de validation. À noter également que l'on conserve la moyenne des pertes pour les deux bases. Afin de pouvoir construire un graphique à la fin de l'entraînement.

```

1 for epoch in range(epochs):
2     model.train()
3
4     train_losses = []
5     valid_losses = []
6
7     for batch, (data, label) in enumerate(train_dataloader):
8         optimizer.zero_grad()
9
10        output = model(data)
11        loss = CELoss(output, label)
12        loss.backward()
13        optimizer.step()
14
15        train_losses.append(loss.item())
16        if (batch * batch_size) % (batch_size * 100) == 0:
17            print( f' {batch * batch_size} / {nb_train_data} ' )
18
19    model.eval()
20    correct = 0
21    total = 0
22    with torch.no_grad():

```

```

23     for batch, (data, label) in enumerate(valid_dataloader):
24         output = model(data)
25         loss = CELoss(output, label)
26
27         valid_losses.append(loss.item())
28
29         for index in range(len(output)):
30             if torch.argmax(output[index]) == torch.argmax(label[
31                 index]):
32                 correct += 1
33             total += label.size(0)
34
35     mean_train_losses.append(np.mean(train_losses))
36     mean_valid_losses.append(np.mean(valid_losses))
37
38     accuracy = 100*correct/total
39     valid_acc_list.append(accuracy)
40     print('epoch : {}, train loss : {:.4f}, valid loss : {:.4f},
41         valid acc : {:.2f}%'\
42         .format(epoch+1, np.mean(train_losses), np.mean(
43             valid_losses), accuracy))
44
45     if accuracy > best_acc:
46         best_acc = accuracy
47         index_best_acc = epoch
48         torch.save(model, "best_model.pt")

```

Code source 5.5 – Entraînement du modèle

Enfin le modèle est évalué sur la base de test comme suit, et les graphiques et indicateurs de performances sont affichés.

```

1  model = torch.load("best_model.pt")
2  model.eval()
3  test_preds = torch.LongTensor()
4
5  for batch, (data, label) in enumerate(test_dataloader):
6      output = model(data)
7
8      pred = output.max(1, keepdim=True)[1]
9      test_preds = torch.cat((test_preds, pred), dim=0)
10
11  pred_list = test_preds.squeeze().tolist()

```

Code source 5.6 – Évaluation sur la base de test

3 Analyse des résultats, évaluation, qualité

Comme il a été précisé dans la partie 1.2 (Chapitre 4), on a dans un premier temps considéré un étudiant en décrochage lorsqu'il n'a pas passé l'examen terminal ce qui représente 31.2% des étudiants. Lors de l'entraînement du réseau de neurones, il s'est avéré que pour toutes les architectures et taux d'apprentissage testées non pas permis d'avoir un apprentissage du réseau de

neurones. En effet, à toute les époques, le réseau de neurones classifiait l'ensemble des étudiants dans une seule classe. Cela peut s'expliquer par la répartition des classes bien que des poids dépendant de cette répartition ait été mise en place au niveau de la fonction de perte. De ce fait, il a été rapidement décidé que les étudiants considérés en décrochage sont ceux n'ayant pas passé l'examen ou l'ayant raté, on monte ainsi à 52,8% de décrochage, ce qui est élevé mais permet ainsi d'avoir une meilleure répartition. Dans le cas où l'on aurait de bon résultat sur cette répartition, il serait envisageable de repasser sur la répartition précédente en utilisant les poids obtenus avec la répartition équilibré.

3.1 Premier résultat

Les premiers résultats sont intéressants, après plusieurs architectures et taux d'apprentissage testées, trois résultats ressortent :

Un premier résultat où la précision est de 47,05% sur la base d'entraînement et reste à cette précision, où l'ensemble des prédictions sont classés sur les étudiants ayant réussi l'examen final. Un second résultat similaire, avec une précision de 52,95% sur la base d'entraînement, où l'ensemble des prédictions sont classés cette fois ci sur les décrochages. On ne parlera donc pas de ces deux résultats qui ne sont pas intéressants.

Enfin, un autre résultat montre une précision aux environs de 69% sur la base d'entraînement mais de 84% sur la base de validation et de test. Ce résultat a été obtenu avec l'architecture suivante :

- 3 couches cachées, de tailles respectives : 25,25 et 10.
- Un batch de 1000 échantillons.
- Un taux d'apprentissage de 0.001.
- 150 époques.
- 50% des données pour la base d'entraînement, 15% pour la base de validation et 35% pour la base de test.

Bien que les différences de précisions obtenus soient assez étonnantes, ce premier résultat est très intéressant. En effet comme l'on peut le constater sur la matrice de confusion suivante, la quasi-intégralité des prédictions ayant été classer dans Non passé/Raté, le sont correctement. Là où l'on perd en précision pour les prédictions de Réussi/Mention, où 1792 de ces prédictions aurait dû être classé comme Non passé/Raté.

	Prédiction Non passé/Raté	Prédiction Réussi/Mention
Non passé/Raté réel	3819	1792
Réussi/Mention réel	33	5765

Pour traiter ces données correctement on peut également s'aider des données du tableau suivant :

	Prediction	Réalité
Non passé/Raté	3852	5611
Réussi/Mention	7557	5798

Premièrement, on peut calculer le taux de vrai positif :

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} = \frac{5765}{5765 + 33} = \frac{5765}{5798} = 99,4\%$$

où :

- TP est le nombre de vrai positif
- P est le nombre de label positif, avec $P = TP + FN$

— FN est le nombre de faux négatif

On a donc 99% des étudiants ayant réussi ou une mention qui ont été correctement classé.

On peut également calculer le taux de faux négatif :

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP} = \frac{3819}{3819 + 1792} = \frac{3819}{5611} = 68,0\%$$

où :

— TN est le nombre de vrai négatif

— N est le nombre de label négatif, avec $N = TN + FP$

— FP est le nombre de faux positif

On a donc 68% des étudiants n'ayant pas passé ou ayant raté l'examen qui ont été correctement classé.

On peut également calculer le taux de fausse omission :

$$FOR = \frac{FN}{PN} = \frac{FN}{FN + TN} = \frac{33}{33 + 3819} = \frac{33}{3852} = 0,8\%$$

où :

— FN est le nombre de faux négatif

— PN est le nombre de prédiction négative, avec $PN = FN + TN$

— TN est le nombre de vrai négatif

On a donc seulement 1% des étudiants qui sont classifiés comme en risque de décrochage, sans pour autant avoir raté leur examen.

Enfin, on peut calculer le taux de fausse découverte :

$$FDR = \frac{FP}{PP} = \frac{FP}{FP + TP} = \frac{1792}{1792 + 5765} = \frac{1792}{7557} = 23,7\%$$

où :

— FP est le nombre de faux positif

— PP est le nombre de prédiction positive, avec $PP = FP + TP$

— TP est le nombre de vrai positif

Ainsi, sur ce jeux de données, nous savons que parmi l'ensemble des élèves dont la prédiction est la réussite de l'examen, il y a environ 24% d'entre eux qui sont en réalité à risque de décrochage. Dans l'ensemble ces résultats sont corrects, notamment en ce qui concerne la prédiction des décrochages, mais il y a des améliorations nécessaires sur la prédiction des étudiants qui réussisse l'examen.

3.2 Amélioration possible du modèle

On a vu précédemment, que les premiers résultats étaient intéressants, toutefois des améliorations du modèle sont nécessaires que nous allons détailler par la suite. Dans un premier temps, nous détaillerons les améliorations possibles du modèle actuel, puis nous discuterons d'autres modèles possibles.

Des résultats précédents, on peut constater une statistique assez surprenante qui est que l'on a un taux de précision de 69% sur la base d'entraînement, tandis que l'on a un taux de précision de 84% sur la base de validation et de test. On peut d'autant plus constater cet écart de précision avec le graphique 5.1.

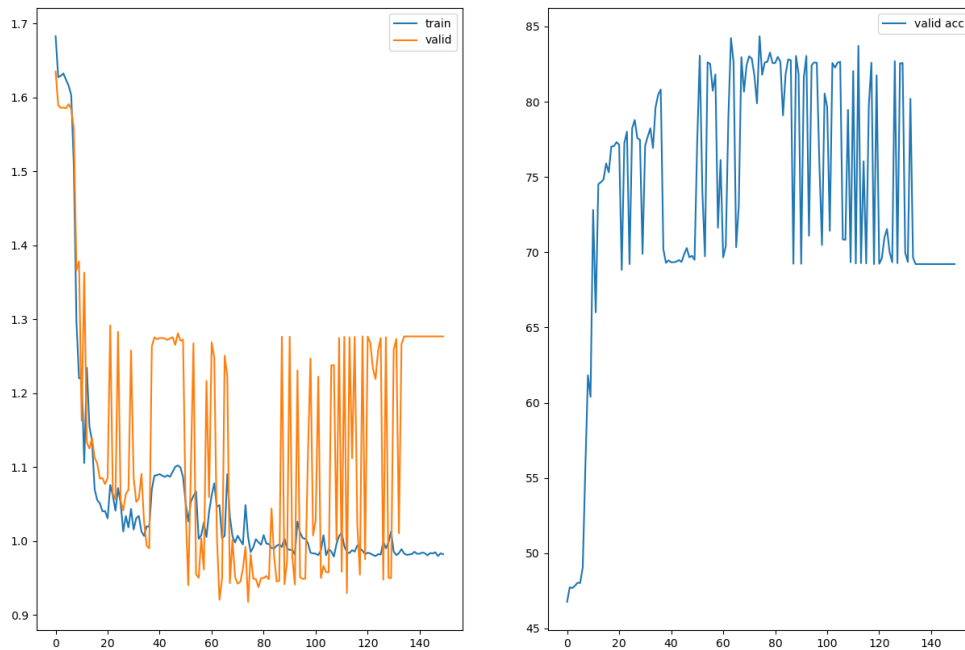


Figure 5.1 – Graphique des pertes et de la précision selon les époques

Cet écart est assez élevé, et a potentiellement un impact sur l'entraînement. Ainsi, pour palier à ce problème, le premier réflexe est de mettre en place de la validation croisée. Ainsi, pour de la validation croisée à 3 blocs par exemple, nous aurons le tableau de répartition suivant :

K	Bloc 1	Bloc 2	Bloc 3
1	Validation	Entraînement	Entraînement
2	Entraînement	Validation	Entraînement
3	Entraînement	Entraînement	Validation

Dans ce cas de figure, on définit donc dans un premier temps la base de test, et le reste des données est consacré à la base de validation et celle d'entraînement, toutefois ici on va réaliser l'entraînement sur 3 itérations différentes, ou la répartition entre la base de validation et d'entraînement est la précédente. À chaque itération, on va calculer un score de performances sur la base de validation et l'on ainsi obtenir 3 scores de performances sur lesquels on pourra faire des statistiques pour juger de l'impact de la validation sur le modèle. On peut ainsi améliorer le modèle en modifiant le nombre de données faisant partie de la validation, jusqu'à ce que l'impact de la validation sur le modèle soit faible, tout en gardant une précision importante sur la base de test.

D'autres améliorations sont possibles, notamment en modifiant l'architecture du modèle, aussi bien en augmentant ou diminuant le nombre de couches cachées du réseau de neurones, qu'en modifiant le nombre de neurones par couches, ou encore en modifiant le taux d'apprentissage. Enfin, il est possible que l'on obtienne de meilleurs résultats si l'on découpe le modèle en 4 classes et non plus 2 comme on a pu le faire jusqu'ici. Toutefois on peut tout de même constater un grand écart de répartition entre les étudiants qui ont réussi l'examen, qui compte pour 37,9% des étudiants, et ceux qui ont eu une mention qui compte pour 9,3% des étudiants. Cet écart de répartition ne peut permettre de garantir des résultats à minima aussi bon que ceux obtenus précédemment.

Autres améliorations possibles

Outre les améliorations du modèle, certains points de ce projet pourrait être améliorés. Durant mon entretien de qualité de mise en oeuvre, l'utilisation de Pandas pour importer les données CSV a été abordé. Mon manque de connaissance sur cette librairie fait que j'aurai pu gagner beaucoup de temps à importer avec la librairie les données plutôt qu'en réalisant moi même un parseur de données. En ce qui concerne le traitement de ces données, celui-ci est très long, l'utilisation d'une base de données aurait probablement été préférable pour obtenir les données souhaités plus efficacement. Toutefois, lorsque je m'en suis rendu compte, il ne me restait pas suffisamment de temps pour apporter ces modifications, ce qui aurait potentiellement pu m'empêcher d'avoir ces premiers résultats.

6

Bilan et conclusion

1 Bilan du semestre 9

Le tableau Trello récapitulatif du semestre 9 se trouve figure 6.1.

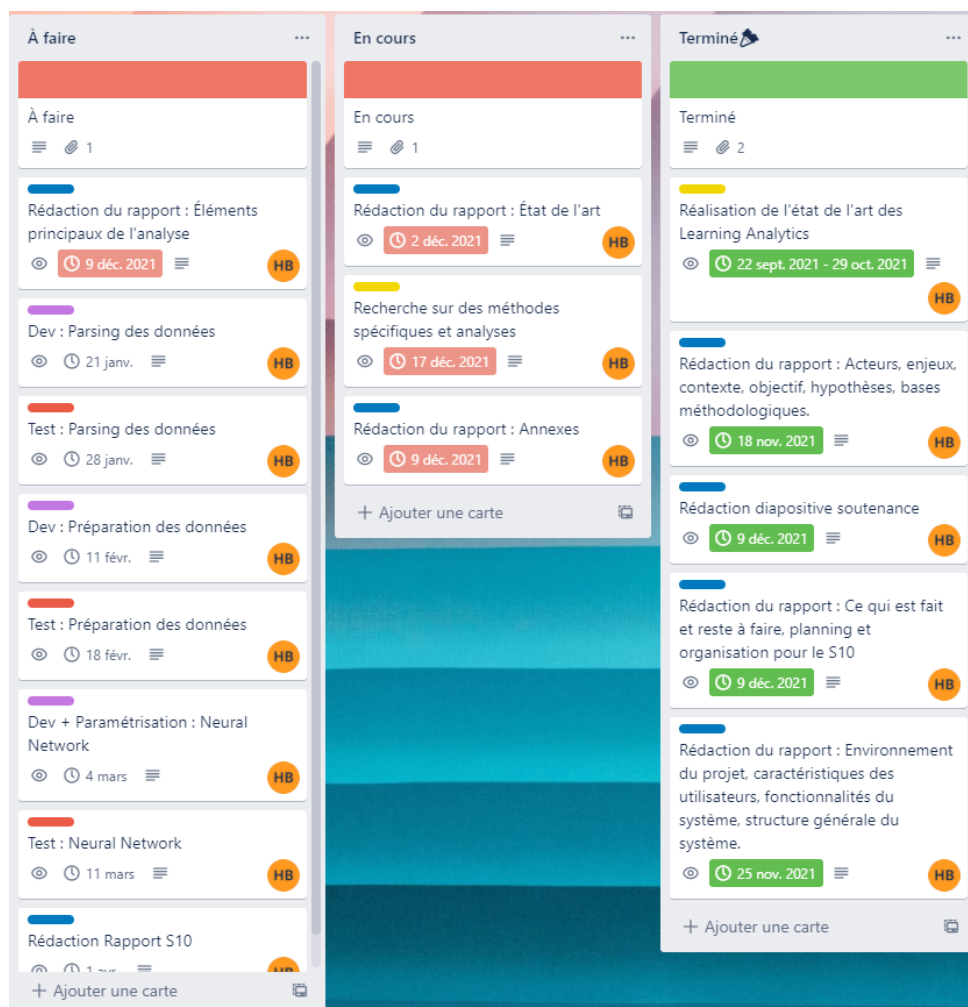


Figure 6.1 – Bilan des tâches effectués au semestre 9

1.1 Travail réalisé

Durant le semestre 9, le principal travail réalisé a été l'état de l'art. Les recherches de ce dernier sont terminées mais sa rédaction est toujours en cours bien qu'avancée. La rédaction des parties suivantes demandées pour le semestre 9 du rapport ont également été réalisées :

- L'introduction.
- La description générale du projet.
- L'état de l'art. Celui-ci n'étant pas totalement terminé.
- L'annexe planification et gestion de projet.

1.2 Retard pris

Du fait de la place importante de l'état de l'art de ce projet recherche et développement, plusieurs parties demandées pour le semestre 9 ont pris du retard.

- Les recherches pour l'analyse ne sont pas entièrement terminées.
- Les éléments principaux de l'analyse n'ont pas été rédigés.
- L'annexe description des interfaces n'a pas été réalisée.
- L'annexe cahier de spécifications n'a pas été réalisé.

1.3 Travail restant

En plus du retard pris, il est nécessaire au semestre 10 de mettre en œuvre le développement du projet, ainsi que poursuivre la rédaction du rapport.

2 Planning pour le semestre 10

Étant donné le retard pris sur la rédaction du rapport et l'analyse, les 3 premières semaines du semestre 10 seront consacrées à la poursuite de la rédaction du rapport et aux dernières recherches nécessaires à l'analyse. Le Gantt initial pour le semestre 10 sera alors réalisé une fois ces tâches faites. Ensuite, la mise en œuvre du projet commencera, avec en parallèle la rédaction des parties du rapport exigées pour le semestre 10.

3 Bilan du semestre 10

Comme prévu, la partie du semestre 10 a commencé 3 semaines plus tard. Étant donné ce fait, il a été décidé de ne pas implémenter la partie plateforme numérique pour se concentrer sur l'entraînement du réseau de neurones.

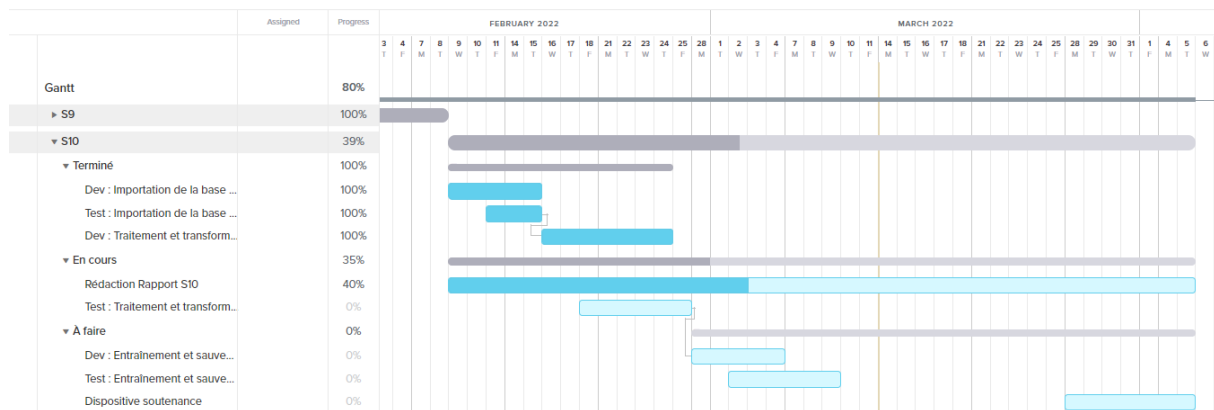


Figure 6.2 – Diagramme de Gantt initial S10

3.1 Travail réalisé

Durant le semestre, l'entièreté des développements de la partie entraînement du réseau de neurones a été réalisée, ainsi que les tests associés. Toutefois, des améliorations peuvent être apportées comme décrit dans **ERROR??**

4 Bilan sur la qualité

5 Bilan auto-critique

Annexes

A

Planification, gestion de projet

1 Evolution du projet

La gestion de projet suivant la méthode agile, le projet est amené à beaucoup d'évolution tout du long. En effet une réunion est organisée toutes les semaines avec l'encadrante Madame Barrat. De plus, les recherches ayant lieu jusqu'à fin décembre, celles-ci apportent de nouvelles découvertes et apportent de nouvelles tâches ou des modifications de certaines tâches précédemment planifiées.

Toutefois à l'aide de la plateforme Trello, il est aisé de créer et classer des tâches dans leur tableau comme dans la figure A.2. Plusieurs "power-up" ont été utilisé, "Planiway", "Gantt by Placker" et "TeamGantt" qui permet de visualiser sous forme d'un diagramme de Gantt les tâches. J'ai dû arrêter d'utiliser les deux premiers "power-up" puisqu'il fallait disposer d'un abonnement au bout d'un certain temps.

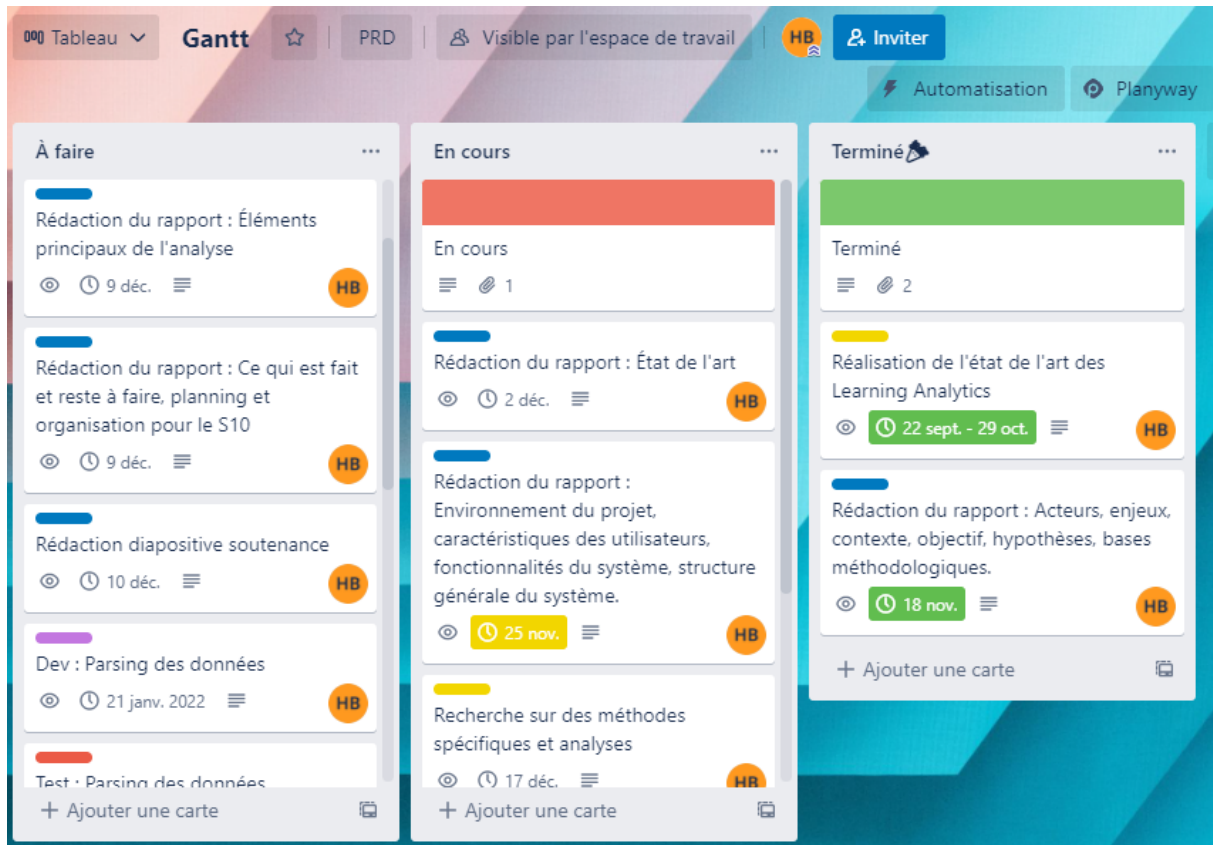


Figure A.1 – Visualisation des tâches d'un tableau Trello

1.1 Planning initial S9

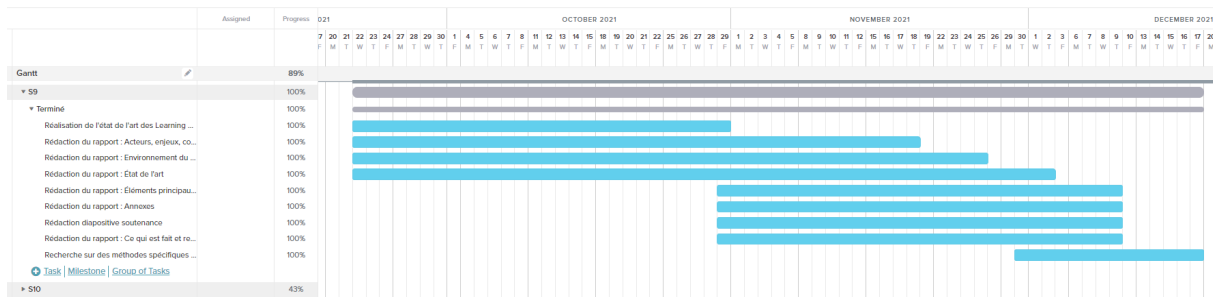


Figure A.2 – Diagramme de Gantt initial S9

1.2 Planning final S9

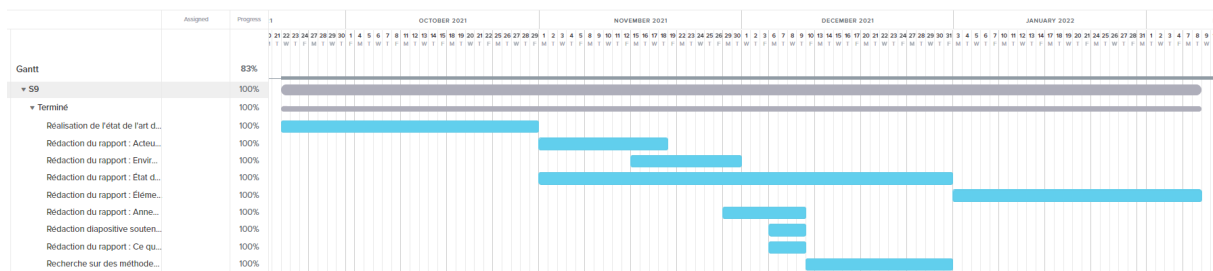


Figure A.3 – Diagramme de Gantt final S9

Comme on peut le constater sur ces deux diagrammes de Gantt, le projet a pris du retard au semestre 9. Cela vient principalement d'une erreur de ma part, où contrairement à ce que j'avais prévu, je n'ai commencé à rédiger l'état de l'art qu'après avoir terminé. Ce fait m'a fortement mis en retard puisque bien que j'avais pris des notes détaillées, il a fallu se souvenir des articles, ce qui a donc mis en retard le projet. De plus j'ai également sous-estimer la quantité de rapport nécessaire durant ce semestre, conduisant à encore plus de retard. Le planning du semestre 10 a donc été fait uniquement 3 semaines après le début du semestre 10. Il a été prévu de ne pas développer la partie plateforme numérique pour ce concentrer sur la partie entraînement du réseau de neurones.

1.3 Planning initial S10

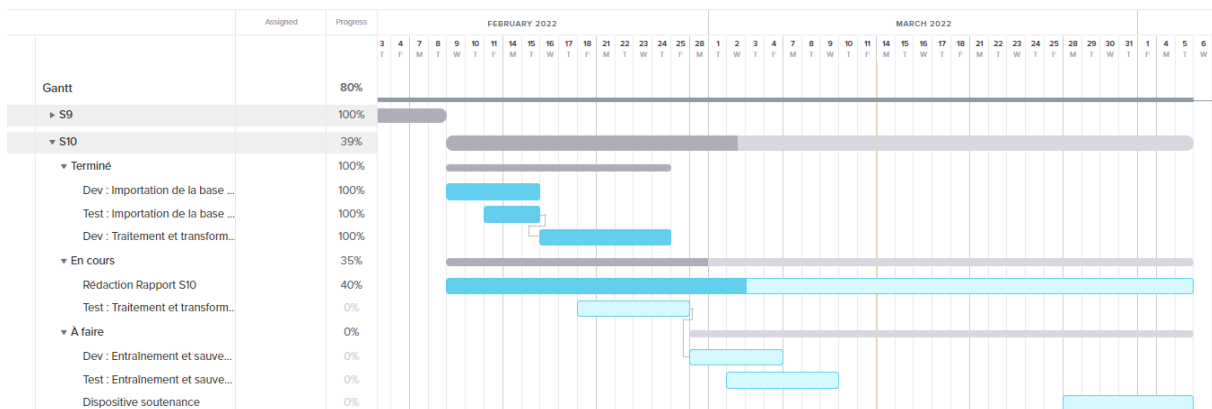


Figure A.4 – Diagramme de Gantt initial S10

1.4 Planning final S10

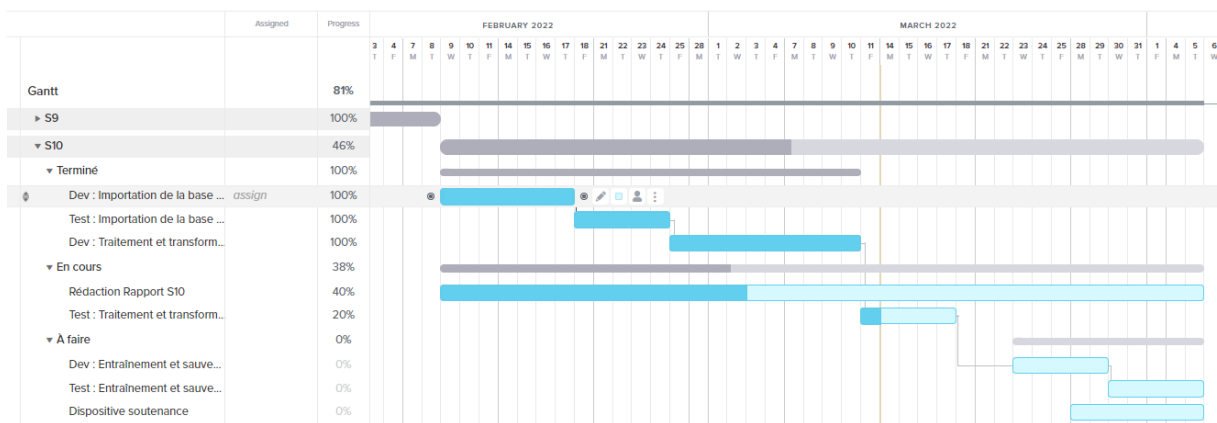


Figure A.5 – Diagramme de Gantt final S10

Le planning lors du semestre 10 a été mieux géré. Les prévisions pour les parties de développement ont été respecté toutefois, les tests ont pris plus de temps que je ne le pensais. J'ai toutefois quand même eu le temps de finir les développements puisque j'ai cette fois-ci laisser du temps en cas de retard.

2 Description des tâches

Tâche 1 : Réalisation de l'état de l'art des Learning Analytics

— Date de début : 22/09/2021

- Date de fin : 29/10/2021
- Durée : 38 jours
- Description : Recherche d'état de l'art existant sur les Learning Analytics, recherche plus spécifique sur les différentes méthodes et outils appliqués au Learning Analytics. Rédaction de notre propre état de l'art à partir des documents recherchés.

Tâche 2 : Rédaction du rapport : Acteurs, enjeux, contexte, objectif, hypothèses, bases méthodologiques

- Date de début : 22/09/2021
- Date de fin : 18/11/2021
- Durée : 58 jours
- Description : Rédaction de l'introduction du rapport. Nécessite d'avoir une compréhension du sujet et d'un objectif fixe.

Tâche 3 : Rédaction du rapport : Environnement du projet, caractéristiques des utilisateurs, fonctionnalités du système, structure générale du système

- Date de début : 22/09/2021
- Date de fin : 25/11/2021
- Durée : 65 jours
- Description : Rédaction de la description générale du rapport.

Tâche 4 : Rédaction du rapport : État de l'art

- Date de début : 22/09/2021
- Date de fin : 02/12/2021
- Durée : 72 jours
- Description : Rédaction de l'état de l'art des Learning Analytics, prenant en compte les états de l'art existant et les outils et méthodes proches du sujet.

Tâche 5 : Recherche sur des méthodes spécifiques

- Date de début : 30/10/2021
- Date de fin : 17/12/2021
- Durée : 49 jours
- Description : Recherches précises sur des méthodes de prédiction de décrochage. Ses recherches ont été déterminantes et on permit de décider d'étudier particulièrement les réseaux de neurones.

Tâche 6 : Rédaction du rapport : Éléments principaux de l'analyse

- Date de début : 30/10/2021
- Date de fin : 9/12/2021
- Durée : 41 jours
- Description : Début de rédaction de la partie 4 : Analyse et conception.

Tâche 7 : Rédaction du rapport : Annexes

- Date de début : 30/10/2021
- Date de fin : 9/12/2021
- Durée : 41 jours
- Description : Rédaction des spécifications fonctionnelles et non fonctionnelles détaillées dont les interfaces. Rédaction du cahier du développeur avec les détails et compléments de la partie 4 : analyse et conception. Rédaction de la planification et gestion de projet.

Tâche 8 : Rédaction des diapositives de la première soutenance

- Date de début : 30/10/2021
- Date de fin : 9/12/2021
- Durée : 41 jours
- Description : Rédaction des diapositives de la soutenance ayant lieu le 15/10/2021.

Tâche 9 : Rédaction du rapport : Ce qui est et reste à faire, planning et organisation pour le semestre 10

- Date de début : 30/10/2021
- Date de fin : 9/12/2021
- Durée : 41 jours
- Description : Rédaction des éléments qui ont pris du retard et reste à faire, création du planning initial pour le semestre 10.

Tâche 10 : Dev : Importation de la base de données

- Date de début : 9/02/2022
- Date de fin : 15/02/2022
- Durée : 6 jours
- Description : Développement de la fonctionnalité "Importation de la base de données".

Tâche 10 : Test : Importation de la base de données

- Date de début : 11/02/2022
- Date de fin : 15/02/2022
- Durée : 4 jours
- Description : Développement des tests de la fonctionnalité "Importation de la base de données".

Tâche 11 : Dev : Traitement et transformation des données

- Date de début : 16/02/2022
- Date de fin : 24/02/2022
- Durée : 8 jours
- Description : Développement de la fonctionnalité "Traitement et transformation des données".

Tâche 11 : Test : Traitement et transformation des données

- Date de début : 18/02/2022
- Date de fin : 25/02/2022
- Durée : 7 jours
- Description : Développement des tests de la fonctionnalité "Traitement et transformation des données".

Tâche 12 : Dev : Importation de la base de données

- Date de début : 26/02/2022
- Date de fin : 04/03/2022
- Durée : 7 jours
- Description : Développement de la fonctionnalité "Entraînement et sauvegarde du réseau de neurones".

Tâche 12 : Dev : Importation de la base de données

- Date de début : 2/03/2022
- Date de fin : 9/03/2022
- Durée : 7 jours
- Description : Développement des tests de la fonctionnalité "Entraînement et sauvegarde du réseau de neurones".

B

Description des interfaces

1 Interfaces matérielles/logicielles

2 Interfaces homme/machine

C

Cahier de Spécifications

Ce chapitre décrit les spécifications fonctionnelles et non fonctionnelles de ce projet.

1 Spécifications fonctionnelles

Comme l'on a pu le voir précédemment, ce projet est découpé en 2 parties. L'entraînement du réseau de neurones et le développement de la visualisation des prédictions sur une plateforme numérique.

1.1 Partie réseau de neurones

1.1.1 Fonctionnalités à développer

L'entraînement du réseau de neurones se réalise en plusieurs étapes.

Premièrement, il est nécessaire d'importer et de traiter les données de la base utilisé.

Ensuite, ces données doivent être transformées dans un format acceptable par le réseau de neurones.

Enfin, le réseau de neurones doit être entraîné et sauvegardé.

1.1.2 Définition de la fonction 1 : Importation de la base de données

Présentation de la fonction 1 :

- Nom de la fonction : Importation de la base de données.
- Les données de la base OULA doivent être importées pour être traitées.
- Primordiale.

Description de la fonction 1 :

Cette fonction importe dans le système l'ensemble des fichiers .csv de la base OULA utile à ce projet.

Importation de la base de données

Entrée : studentInfo.csv, studentAssessment.csv, assessment.csv, courses.csv

Sortie :

- La liste des résultats finaux des étudiants [code_module, code_presentation, id_student, final_result]
- La liste des notes des étudiants aux évaluations [id_assessment, id_student, score]
- La liste des évaluations des présentations des modules avec leurs types d'évaluation [code_module, code_presentation, id_assessment, assessment_type]
- La liste des présentations des modules avec leurs durées [code_module, code_presentation, module_presentation_length]

1.1.3 Définition de la fonction 2 : Traitement et transformation des données**Présentation de la fonction 2 :**

- Nom de la fonction : Traitement et transformation des données
- Les données doivent être traitées puis transformées pour quelles correspondent au format de données attendus par le réseau de neurones.
- Primordiale.

Description de la fonction 2 :

Cette fonction convertit les données importées en données numériques qui sont ensuite transformées sous la forme d'une liste de liste.

Traitement et transformation des données

Entrée : Les données traitées :

- La liste des résultats finaux des étudiants [code_module, code_presentation, id_student, final_result]
- La liste des notes des étudiants aux évaluations [id_assessment, id_student, score]
- La liste des évaluations des présentations des modules avec leurs types d'évaluation [code_module, code_presentation, id_assessment, assessment_type]
- La liste des présentations des modules avec leurs durées [code_module, code_presentation, module_presentation_length]

Sortie :

- La liste des données converties : [converted_code_module, converted_code_presentation, converted_id_student, converted_module_presentation_length, converted_mean_student, converted_global_mean, converted_id_assessment_1, converted_type_assessment_1, converted_score_1, converted_mean_eval_1, ... , converted_id_assessment_14, converted_type_assessment_14, converted_score_14, converted_mean_eval_14].

1.1.4 Définition de la fonction 3 : Entraînement et sauvegarde du réseau de neurones**Présentation de la fonction 3 :**

- Nom de la fonction : Entraînement et sauvegarde du réseau de neurones.
- Cette fonction entraîne le réseau de neurones sur la base de données OULA.
- Primordiale.

Description de la fonction 3 :

Le réseau de neurones est entraîné sur les données traitées et transformées de la base OULA. Les données de la base sont divisés en 3 bases. La base d'entraînement, la base de validation et la base de test. Le réseau de neurones n'est sauvegardé que lorsque le pourcentage de bonne prédiction sur la base de validation dépasse le pourcentage du précédent modèle.

Entraînement du modèle

Entrée : La liste des données converties : [converted_code_module, converted_code_presentation, converted_id_student, converted_module_presentation_length, converted_mean_student, converted_global_mean, converted_id_assessment_1, converted_type_assessment_1, converted_score_1, converted_mean_eval_1, ... , converted_id_assessment_14, converted_type_assessment_14, converted_score_14, converted_mean_eval_14].

Sortie : Le modèle ayant la meilleure précision sur la base de validation, et sa précision sur la base de test.

1.2 Partie plateforme numérique

Cette partie vient dans un second temps, il est primordiale que la première partie soit entièrement terminée pour pouvoir commencer celle-ci.

1.2.1 Fonctionnalités à développer

Le développement de la plateforme numérique se réalise également en plusieurs étapes.

Premièrement, il est nécessaire de pouvoir se connecter, pour avoir accès soit à la vue étudiant, soit à la vue enseignant.

Deuxièmement, il est nécessaire de pouvoir importer les données des prédictions.

Enfin, il faut pouvoir afficher la visualisation des prédictions.

1.2.2 Définition de la fonction 4 : Authentification**Présentation de la fonction 4 :**

- Nom de la fonction : Authentification.
- Les utilisateurs doivent pouvoir se connecter afin d'accéder à leur vue des prédictions.
- Primordiale

Description de la fonction 4 :

Cette fonction permet aux utilisateurs de se connecter. Ces derniers ne peuvent avoir accès à la vue qui leur est associé que si ils sont connectés. Les étudiants peuvent voir les prédictions pour chacun des modules auxquels ils sont inscrits. Comme aucun enseignants n'est associés aux modules dans la base, on considérera qu'un module représente un enseignant, et lorsque l'on se connecte avec ce numéro de module, on a donc accès à l'ensemble des visualisations des étudiants de ce module. Dans le cadre de ce projet un mot de passe unique sera défini pour l'ensemble des utilisateurs.

Authentification

Entrée : L'identifiant de l'étudiant ou du module, et son mot de passe.

Sortie : Redirection vers la page de visualisation associé à cet identifiant.

Pré-condition : Le système a accès aux données cryptés des identifiants et de leurs mots de passe associés.

1.2.3 Définition de la fonction 5 : Importation des données

Présentation de la fonction 5 :

- Nom de la fonction : Importation des données
- Les données issues des prédictions doivent être importées afin de réaliser la visualisation.
- Primordiale

Description de la fonction 5 :

Cette fonction permet de récupérer les données des prédictions, ainsi que les données annexes nécessaire à la visualisation.

Importation des données

Entrée : Les données issues des prédictions. Les données annexes, notamment `code_module`, `id_student`, `id_assessment`, `score`.

Sortie : Rien

1.2.4 Définition de la fonction 6 : Visualisation des prédictions

Présentation de la fonction 6 :

- Nom de la fonction : Visualisation des prédictions
- Les données sont utilisées pour réaliser les visualisations.
- Primordiale

Description de la fonction 6 :

Cette fonction permet d'afficher les visualisations associés à un utilisateur.

Visualisation des prédictions

Entrée : Les données issues des prédictions et les données annexes nécessaire à la visualisation.

Sortie : L'affichage des visualisations.

2 Spécifications non fonctionnelles

2.1 Contraintes de développement et conception

- Matériel : Aucun matériel particulier n'est nécessaire.
- Langages de programmation : Python pour l'entraînement du réseau de neurones. Angular pour la plateforme numérique.
- Librairies et bibliothèques nécessaires au développement : PyTorch

2.2 Contraintes de fonctionnement et d'exploitation

Les deux parties de ce projet sont soumises à des contraintes de fonctionnement. On distingue 4 types de contraintes, les performances, les capacités, la contrôlabilité et la sécurité.

2.2.1 Performances

L'entraînement du réseau de neurones ne demande pas de grosse contraintes de performances, on va principalement chercher à ce que l'entraînement prenne une à deux heures au maximum étant donné que l'on aura probablement besoin d'entraîner le réseau avec différents paramètres jusqu'à obtenir des performances suffisantes.

Une fois l'entraînement terminé, et le modèle généré, il faut que les prédictions lorsque l'on obtient de nouvelles données via la plateforme numérique ce fasse idéalement aux alentours d'une seconde. Afin que lorsque l'utilisateur retourne sur la page de visualisations, celle-ci soit visible directement.

Les visualisations doivent également être chargées de manière à rendre l'expérience utilisateur agréable, c'est à dire en moins d'une seconde également.

2.2.2 Sécurité

L'utilisateur a besoin de se connecter pour accéder aux visualisations de la plateforme numérique. Étant donné que l'on a deux types d'utilisateurs différents on doit s'assurer aux travers de différents tests que les utilisateurs ont uniquement accès à la visualisation qui leur est associée. Les étudiants ne doivent pas avoir accès aux visualisations des autres étudiants et les enseignants ne doivent pas avoir accès aux visualisations des modules qui ne leur sont pas associés. Ici comme l'on n'a pas d'enseignant associés aux modules, lorsque l'on se connecte en renseignant l'identifiant d'un module, il ne faut pas que l'on puisse accéder aux visualisations des autres modules en étant connecté avec cet identifiant de module.

D

Cahier du développeur

1 Introduction

2 Diagrammes architecturaux et UML

Le diagramme des classes liés à l'entraînement du réseau de neurones est comme figure [D.1](#)

3 Descriptions détaillées des données exploitées

4 Descriptions détaillées des classes, modules, réalisations

4.1 Importation et traitement des données

Classe `DataParser` :

- Description : Parseur de données de la base OULA.
- Attributs :
 - `rep_path` : (string) Le chemin vers le dossier contenant les données.
 - `csv_names` : (string[]) La liste des fichiers ".csv" à récupérer.
 - `final_results` : (string[][]) La liste des résultats finaux de chaque étudiant aux différents cours.
Respecte le format [code_module, code_presentation, id_student, final_result].
 - `scores` : (string[][]) La liste des notes de chaque étudiant aux différentes évaluations.
Respecte le format [id_assessment, id_student, score]
 - `assessments_types` : (string[][]) La liste des types d'évaluations des différentes présentations de modules.
Respecte le format [code_module, code_presentation, id_assessment, assessment_type].

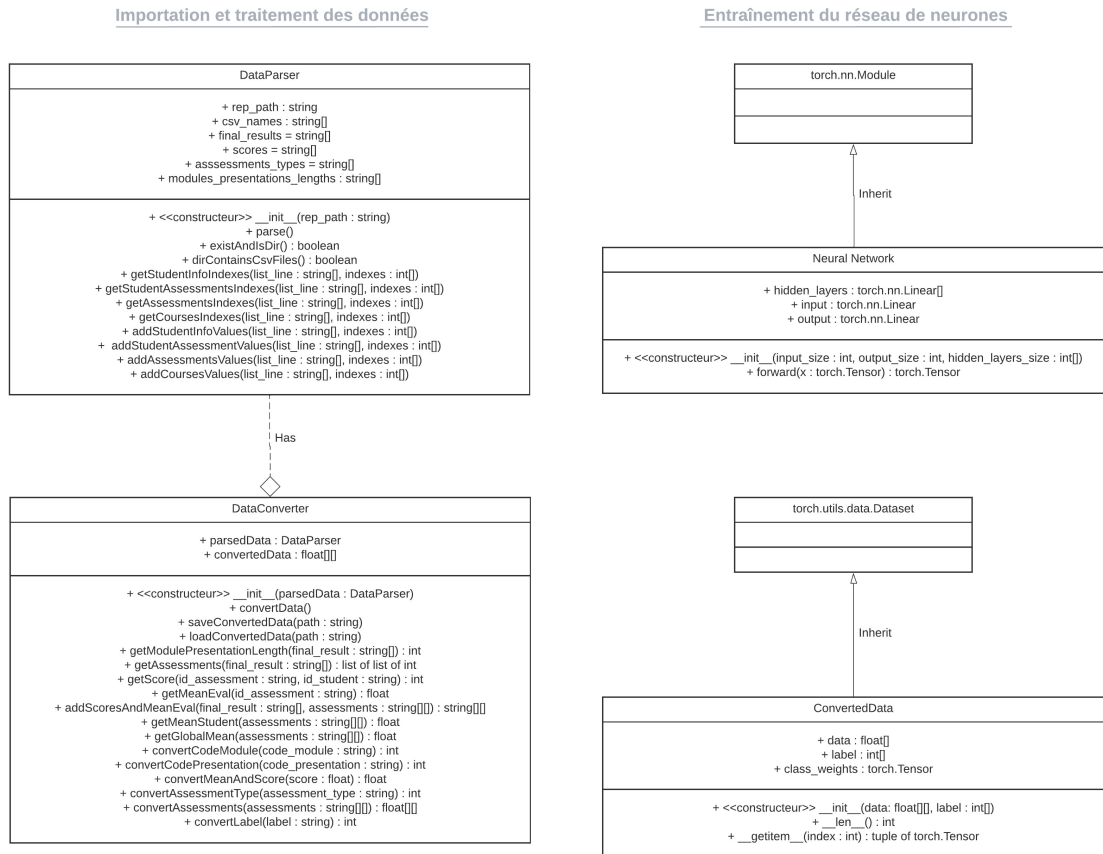


Figure D.1 – Diagramme des classes liés à l'entraînement du réseau de neurones

- `modules_presentations_lengths` : (`string[]`) La liste des durées des présentations de chaque module.
- `parse` :
 - Description : Les données sont collectées à condition que le dossier au chemin "`rep_path`" contient bien les fichiers ".csv" de la base OULA.
 - Exceptions levées :
 - `IOError` : Si l'un des fichiers ".csv" ne peut pas être ouvert.
- `existAndIsDir` :
 - Description : Vérifie que le chemin de l'attribut "`rep_path`" existe bien et est un dossier.
 - Retourne :
 - `res` : (`boolean`) Vrai si le chemin existe et est un dossier, faux sinon.
- `dirContainsCsvFiles` :
 - Description : Vérifie que le dossier de l'attribut "`rep_path`" contient bien les fichiers ".csv" présents dans la liste "`csv_names`".
 - Retourne :
 - `res` : (`boolean`) Vrai si tous les fichiers sont dans le dossier, faux sinon.

- `getStudentInfoIndexes` :
 - Description : Récupère les indices des champs que l'on souhaite récupérer dans le fichier "studentInfo.csv".
Les indices sont stockés dans la variable "indexes".
 - Paramètres :
 - `list_line` : (string[]) Liste contenant les intitulés de tous les champs de "studentInfo.csv"
 - `indexes` : (int[]) List des indices de tous les champs que l'on souhaite récupérer dans "studentInfo.csv"
Respecte le format [code_module_index, code_presentation_index, id_student_index, final_result_index]
- `getStudentAssessmentIndexes` :
 - Description : Récupère les indices des champs que l'on souhaite récupérer dans le fichier "studentAssessment.csv".
Les indices sont stockés dans la variable "indexes".
 - Paramètres :
 - `list_line` : (string[]) Liste contenant les intitulés de tous les champs de "studentAssessment.csv"
 - `indexes` : (int[]) List des indices de tous les champs que l'on souhaite récupérer dans "studentAssessment.csv"
Respecte le format [id_assessment_index, id_student_index, score_index]
- `getAssessmentsIndexes` :
 - Description : Récupère les indices des champs que l'on souhaite récupérer dans le fichier "assessments.csv".
Les indices sont stockés dans la variable "indexes".
 - Paramètres :
 - `list_line` : (string[]) Liste contenant les intitulés de tous les champs de "assessments.csv"
 - `indexes` : (int[]) List des indices de tous les champs que l'on souhaite récupérer dans "assessments.csv"
Respecte le format [code_module_index, code_presentation_index, id_assessment_index, assessment_type_index]
- `getCoursesIndexes` :
 - Description : Récupère les indices des champs que l'on souhaite récupérer dans le fichier "courses.csv".
Les indices sont stockés dans la variable "indexes".
 - Paramètres :
 - `list_line` : (string[]) Liste contenant les intitulés de tous les champs de "courses.csv"
 - `indexes` : (int[]) List des indices de tous les champs que l'on souhaite récupérer dans "courses.csv"
Respecte le format [code_module_index, code_presentation_index, module_presentation_length_index]
- `addStudentInfoValues` :
 - Description : Ajoute les champs désirés de "studentInfo.csv" a la liste "final_results". Vérifie d'abord que les indices sont non nuls, sinon les indices manquant sont affichés dans la console.
 - Paramètres :
 - `list_line` : (string[]) Liste contenant les intitulés de tous les champs de "studentInfo.csv"
 - `indexes` : (int[]) List des indices de tous les champs que l'on souhaite récupérer dans "studentInfo.csv"

- Respecte le format [code_module_index, code_presentation_index, id_student_index, final_result_index]
- addStudentAssessmentValues :
 - Description : Ajoute les champs désirés de "studentAssessment.csv" a la liste "scores". Vérifie d'abord que les indices sont non nuls, sinon les indices manquant sont affichés dans la console.
 - Paramètres :
 - list_line : (string[]) Liste contenant les intitulés de tous les champs de "studentAssessment.csv"
 - indexes : (int[]) List des indices de tous les champs que l'on souhaite récupérer dans "studentAssessment.csv"
 - Respecte le format [id_assessment_index, id_student_index, score_index]
- addAssessmentsValues :
 - Description : Ajoute les champs désirés de "assessments.csv" a la liste "assessments_types". Vérifie d'abord que les indices sont non nuls, sinon les indices manquant sont affichés dans la console.
 - Paramètres :
 - list_line : (string[]) Liste contenant les intitulés de tous les champs de "assessments.csv"
 - indexes : (int[]) List des indices de tous les champs que l'on souhaite récupérer dans "assessments.csv"
 - Respecte le format [code_module_index, code_presentation_index, id_assessment_index, assessment_type_index]
- addCoursesValues :
 - Description : Ajoute les champs désirés de "courses.csv" a la liste "modules_presentations_lengths". Vérifie d'abord que les indices sont non nuls, sinon les indices manquant sont affichés dans la console.
 - Paramètres :
 - list_line : (string[]) Liste contenant les intitulés de tous les champs de "courses.csv"
 - indexes : (int[]) List des indices de tous les champs que l'on souhaite récupérer dans "courses.csv"
 - Respecte le format [code_module_index, code_presentation_index, module_presentation_length_index]

Classe DataConverter :

- Description : Convertis les données en valeurs numériques et dans un format acceptable par le réseau de neurones.
- Attributs :
 - parsedData : (DataParser) Une instance de DataParser.
 - convertedData : (float[][]) La liste des données convertis.
 - Respecte le format [converted_code_module, converted_code_presentation, converted_id_student, converted_module_presentation_length, converted_mean_student, converted_global_mean, converted_id_assessment_1, converted_type_assessment_1, converted_score_1, converted_mean_eval_1, ..., converted_id_assessment_14, converted_type_assessment_14, converted_score_14, converted_mean_eval_14]
- Methods :
 - __init__ :
 - Description : Constructeur de la classe DataConverter. Prend en paramètre "parsedData", une instance d'un objet DataParser.
 - Paramètres :

- `parsedData` : (`DataParser`) Une instance de `DataParser`.
- `convertedData` ; (`float[][]`) La liste des données convertis.
Respecte le format [`converted_code_module`, `converted_code_presentation`, `converted_id_student`, `converted_module_presentation_length`, `converted_mean_student`, `converted_global_mean`, `converted_id_assessment_1`, `converted_type_assessment_1`, `converted_score_1`, `converted_mean_eval_1`, ..., `converted_id_assessment_14`, `converted_type_assessment_14`, `converted_score_14`, `converted_mean_eval_14`]
- Exceptions levées :
 - `TypeError` : Si le paramètre "`parsedData`" n'est pas une instance de `DataParser`.
- `convertData` :
 - Description : Convertis les données récupérées en valeurs numériques suivant le format [`converted_code_module`, `converted_code_presentation`, `converted_id_student`, `converted_module_presentation_length`, `converted_mean_student`, `converted_global_mean`, `converted_id_assessment_1`, `converted_type_assessment_1`, `converted_score_1`, `converted_mean_eval_1`, ..., `converted_id_assessment_14`, `converted_type_assessment_14`, `converted_score_14`, `converted_mean_eval_14`]
 - Exceptions levées :
 - `TypeError` : Si l'attribut "`parsedData`" n'est pas une instance de `DataParser`.
- `saveConvertedData` :
 - Description : Sauvegarde les données de l'attribut "`convertedData`" dans un fichier ".csv" dont le chemin est donné par "`path`".
 - Paramètres :
 - `path` : (`string`) Le chemin vers le fichier où écrire.
 - Exceptions levés :
 - `IOError` : Si le fichier ne peut pas être ouvert.
- `loadConvertData` :
 - Description : Charge les données stockées dans le fichier ".csv" dont le chemin est donné par "`path`" dans la liste "`convertedData`".
 - Exceptions levés :
 - `IOError` : Si le fichier ne peut pas être ouvert.
- `getModulePresentationLength` :
 - Description : Récupère la durée d'une présentation d'un module suivi par un étudiant selon les données contenus dans le paramètre "`final_result`".
 - Paramètres :
 - `final_result` : (`string[]`) Liste contenant le résultat final de l'étudiant à une présentation. Respecte le format [`code_module`, `code_presentation`, `id_student`, `final_result`]
 - Retourne :
 - (`int`) La durée de la présentation correspondant aux valeurs de "`final_result`".
- `getAssessments` :
 - Description : Récupère la liste des types d'évaluations et leur indice correspondant aux valeurs de "`code_module`" et "`code_presentation`" contenus dans le paramètre "`final_result`".
 - Paramètres :
 - `final_result` : (`string[]`) Liste contenant le résultat final de l'étudiant à une présentation. Respecte le format [`code_module`, `code_presentation`, `id_student`, `final_result`]
 - Retourne :
 - `assessments` : (`str[][]`) Liste contenant chaque indice et types d'évaluations d'une présentation d'un module. Respecte le format [[`id_assessment`, `assessment_`

- type]
- `getScore` :
 - Description : Récupère la note d'un étudiant a une évaluation.
 - Paramètres :
 - `id_assessment` : (string) L'identifiant de l'évaluation.
 - `id_student` : (string) L'identifiant de l'étudiant.
 - Retourne :
 - (int) La note correspondante à l'évaluation de l'étudiant.
- `getMeanEval` :
 - Description : Récupère la moyenne des étudiants à une évaluation.
 - Paramètres :
 - `id_assessment` : (string) L'identifiant de l'évaluation.
 - Retourne :
 - (int) La moyenne de l'évaluation.
- `addScoresAndMeanEval` :
 - Description : Ajoute la note d'un étudiant à une évaluation et la moyenne de cette évaluation a la liste "assessments"
 - Paramètres :
 - `final_result` : (string[]) La liste contenant le résultat final d'un étudiant à une présentation. Respecte le format [code_module, code_presentation, id_student, final_result].
 - `assessments` : (string[][]) La liste contenant les identifiants et les types d'évaluations d'une présentation suivi par un étudiant.
 - Exceptions levés :
 - `ValueError` : Si la taille d'une des sous listes de "assessments" est différent de 2.
 - Retourne :
 - (string [][]) La liste contenant les identifiants et les types d'évaluations d'une présentation suivi par un étudiant, ainsi que les notes de l'étudiant et la moyenne des évaluations.
- `getMeanStudent` :
 - Description : Récupère la moyenne d'un étudiant sur l'ensemble des évaluations.
 - Paramètres :
 - `assessments` : (string[][]) La liste contenant les identifiants et les types d'évaluations d'une présentation suivi par un étudiant, ainsi que les notes de l'étudiant et la moyenne des évaluations.
 - Exceptions levés :
 - `Value Error` : Si la taille d'une des sous listes de "assessments" est différent de 4.
 - Retourne :
 - (float) La moyenne de l'étudiant à une présentation.
- `getGlobalMean` :
 - Description : Récupère la moyenne des moyennes de chaque évaluation.
 - Paramètres :
 - `assessments` : (string[][]) La liste contenant les identifiants et les types d'évaluations d'une présentation suivi par un étudiant, ainsi que les notes de l'étudiant et la moyenne des évaluations.
 - Exceptions levés :
 - `ValueError` : Si la taille d'une des sous listes de "assessments" est différent de 4.
 - Retourne :
 - (float) La moyenne des moyennes de chaque évaluation.

- `convertCodeModule` :
 - Description : Convertis le paramètre "code_module" en une valeur numérique. Chaque caractère est convertis en son équivalent numérique dans la table ASCII. Exemple : "AAA" devient 656565.
 - Paramètres :
 - `code_module` : (string) L'identifiant du module.
 - Retourne :
 - L'identifiant du module convertis en une valeur numérique.
- `convertCodePresentation` :
 - Description : Convertis le paramètre "code_presentation" en une valeur numérique. "code_presentation" suit le format "0000A", la partie chiffre est multiplié par 100 et le caractère est convertis en son équivalent numérique moins 64. Exemple : "2022A" devient 202201
 - Paramètres :
 - `code_presentation` : (string) L'identifiant de la présentation.
 - Retourne :
 - (int) L'identifiant de la présentation convertis en une valeur numérique.
- `convertMeanAndScore` :
 - Description : Ajoute 1 à la note ou la moyenne passé en argument afin d'éviter qu'une note de 0 soit noté comme tel dans le réseau de neurones.
 - Paramètres :
 - `score` : (float) La moyenne ou la note.
 - Retourne :
 - (float) La moyenne ou la note plus 1.
- `convertAssessmentType` :
 - Description : Convertis le type d'évaluation en une valeur numérique. "TMA" devient 1. "CMA" devient 2. "Exam" devient 3.
 - Paramètres :
 - `assessment_type` : (string) Le type d'évaluation.
 - Exceptions levés :
 - `ValueError` : Si la valeur de "assessment_type" n'est pas "TMA", "CMA" ou "Exam"
 - Retourne :
 - (int) Le type d'évaluation convertis en valeur numérique.
- `convertAssessments` :
 - Description : Convertis l'ensemble des valeurs de la liste "assessments" en valeurs numériques.
 - Paramètres :
 - `assessments` : (string[]) La liste contenant les identifiants et les types d'évaluations d'une présentation suivi par un étudiant, ainsi que les notes de l'étudiant et la moyenne des évaluations.
 - Retourne :
 - `assessments` : (float[]) La liste "assessments" dont le contenu a été en valeurs numériques.
- `convertLabel` :
 - Description : Convertis le résultat final en une valeur numérique. Met à 0 le label quand l'étudiant a "Withdrawn" ou "Fail" et 1 sinon.
 - Paramètres :
 - `final_result` : (string) Le résultat final de l'étudiant à une présentation.
 - Exceptions levés :

- ValueError : Si le résultat final n'est pas une chaîne de caractère connu.
- Retourne :
 - (int) Le résultat final convertis en une valeur numérique.

Classe **NeuralNetwork** :

- Description : Architecture du réseau de neurones.
- Héritage : Hérite de torch.nn.Module.
- Attributs :
 - hidden_layers : (torch.nn.Linear[]) Liste des couches cachés du réseau de neurones.
 - input : (torch.nn.Linear) La couche d'entrée du réseau de neurones.
 - output : (torch.nn.Linear) La couche de sortie du réseau de neurones.
- Méthodes :
 - `__init__` :
 - Description : Constructeur de la classe NeuralNetwork. Prend en paramètre optionnel, "input_size", "output_size" et "hidden_layers_sizes".
 - Paramètres :
 - input_size : (int) (optionnel) La taille de la couche d'entrée. Vaut 62 par défaut.
 - output_size : (int) (optionnel) La taille de la couche de sortie. Vaut 2 par défaut.
 - hidden_layers_sizes : (int[]) (optionnel) La liste des tailles de chaque couche caché. Vaut [25,25,10] par défaut.
 - Assertion :
 - "hidden_layers_sizes" doit être une liste.
 - forward :
 - Description : Propage les données d'entrées à travers le réseau de neurones et retourne les données de sorties.
 - Paramètres :
 - x : (torch.Tensor) Les données d'entrées.
 - Retourne :
 - x : (torch.Tensor) Les données de sorties.

Classe **ConvertedData** :

- Description : Création d'un objet torch.utils.data.Dataset utilisable par un DataLoader.
- Héritage : Hérite de torch.utils.data.Dataset
- Attributs :
 - data : (float[]) Les données convertis sous la forme d'une liste.
 - label : (int[]) Les labels convertis sous la forme d'une liste.
 - class_weights : (torch.Tensor) Les poids de chaque classe présents dans "label".
- Méthodes :
 - `__init__` :
 - Description : Constructeur de la classe ConvertedData. Prend en paramètres "data" et "label".
 - Paramètres :
 - data : (float[]) Les données convertis sous la forme d'une liste.
 - label : (int[]) Les labels convertis sous la forme d'une liste.
 - `__len__` :
 - Description : Récupère la taille de la liste "data".
 - Retourne :
 - La taille de la liste "data".
 - `__getitem__` :

- Description :Récupère les données et le label à l'indice "index".
- Paramètres :
 - index : (int) L'indice des données et du label à récupérer.
- Retourne :
 - tensor_data : (torch.Tensor) Tenseur correspondant aux données à l'indice "index".
 - tensor_label : (torch.Tensor) Tenseur correspondant au label à l'indice "index".

E

Document d'installation

Au moment de l'écriture de ce rapport, il est nécessaire d'avoir les versions suivantes ou plus récentes de Python et de ces librairies :

- Python 3.8.8
- PyTorch 1.10.1
- Matplotlib 3.3.4
- NumPy 1.20.1
- Pandas 1.2.4
- scikit-learn 0.24.1

Pour faciliter l'installation de ces librairies il est recommandé d'utiliser Conda.

Installation de Conda (Optionnel)

Le détail des procédures d'installations est disponibles à l'adresse suivante :
<https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html>

Installation de Python

Le détail des procédures d'installations est disponibles à l'adresse suivante :
<https://www.python.org/downloads/release/python-388/>

Installation de PyTorch

Le détail des procédures d'installations est disponibles à l'adresse suivante :
<https://pytorch.org/get-started/previous-versions/>

Installation Matplotlib

Le détail des procédures d'installations est disponibles à l'adresse suivante :
<https://matplotlib.org/stable/users/installing/index.html>

Installation NumPy

Le détail des procédures d'installations est disponibles à l'adresse suivante :
<https://numpy.org/install/>

Installation Pandas

Le détail des procédures d'installations est disponibles à l'adresse suivante :

https://pandas.pydata.org/docs/getting_started/install.html

Installation de scikit-learn

Le détail des procédures d'installations est disponibles à l'adresse suivante :

<https://scikit-learn.org/stable/install.html>

F

Document d'utilisation

1 Récupération des données

Afin de récupérer les données il y a deux possibilités :

- La construction des données : Cette méthode est longue et est déconseillée. Lorsqu'elle est utilisée, il est fortement conseillé de sauvegarder les données dans un fichier CSV.
- Le chargement des données : Cette méthode permet de charger les données depuis un fichier CSV

Le construction et sauvegarde des données ce fait avec le code suivant :

```
1 dataParser = DataParser( "Data/" )
2 dataParser.parse()
3 dataConverter = DataConverter( dataParser )
4 dataConverter.convertData()
5 dataConverter.saveConvertedData( "savedConvertedData.csv" )
```

Code source F.1 – Construction des données

Le chargement des données se fait de la manière suivante :

```
1 dataParser = DataParser( "Data/" )
2 dataParser.parse()
3 dataConverter = DataConverter( dataParser )
4 dataConverter.loadConvertedData( "savedConvertedData.csv" )
```

Code source F.2 – Chargement des données

2 Entraînement du réseau de neurones

Pour entraîner le réseau de neurones il suffit d'exécuter le main présent dans le fichier NeuralNetwork.py.

Il est possible d'apporter des modifications au réseau de neurones. Les modifications possibles sont les suivantes :

- Modification des répartitions des bases.
- Modification de la taille d'un batch.
- Modification du nombre d'époque.
- Modification du taux d'apprentissage.
- Modification du nombre de neurones et de couches.

La répartition des bases est définis par les 3 variables suivantes :

```
1 train_ratio = 0.5
2 valid_ratio = 0.15
3 test_ratio = 0.35
```

Code source F.3 – Répartition des bases

La modification de la taille d'un batch et du nombre d'époque sont possibles en modifiant les variables :

```
1 batch_size = 1000
2 epochs = 150
```

Code source F.4 – Taille d'un batch et nombre d'époque

La modification du taux d'apprentissage est possible en modifiant l'optimiseur du réseau, il suffit de modifier le paramètre "lr" :

```
1 optimizer = torch.optim.Adam(model.parameters(), lr=0.001, betas
    =(0.9, 0.999))
```

Code source F.5 – Taux d'apprentissage

La modification du nombre de neurones et de couches est possible en modifiant l'initialisation de la classe NeuralNetwork. Le nombre d'éléments de la liste "hidden_layers_sizes" définit le nombre de couche cachés. Les valeurs définissent le nombre de neurones d'une couche.

```
1 model = NeuralNetwork(input_size = 62, output_size = 2,
    hidden_layers_sizes = [25,25,10])
```

Code source F.6 – Nombre de couches et de neurones



Cahier de test

1 Tests unitaires

1.1 DataParser

Nom de la méthode à tester	existAndIsDir
Nom de la fonction de test	test_path_exist_and_is_dir
Description du test	Test si l'attribut "rep_path" existe et est un répertoire.
Données d'entrées	path : "Test/Data"
Résultats attendus	True
Résultats obtenus	True
Nom de la fonction de test	test_path_exist_and_is_not_dir
Description du test	Test si l'attribut "rep_path" existe mais n'est pas un répertoire.
Données d'entrées	path : "Test/test.txt"
Résultats attendus	False
Résultats obtenus	False
Nom de la fonction de test	test_path_not_exist
Description du test	Test si l'attribut "rep_path" n'existe pas.
Données d'entrées	path : "Test/wrongPath"
Résultats attendus	False
Résultats obtenus	False
Nom de la méthode à tester	dirContainsCsvFiles
Nom de la fonction de test	test_dir_has_csv_files
Description du test	Test si l'attribut "rep_path" contient tout les fichiers CSV issues de la base OULA.
Données d'entrées	path : "Test/Data/AllFiles"

Résultats attendus	True
Résultats obtenus	True
Nom de la fonction de test	test_dir_has_no_csv_files
Description du test	Test si l'attribut "rep_path" ne contient pas tout les fichiers CSV issues de la base OULA.
Données d'entrées	path : "Test/Data/NoFiles"
Résultats attendus	False
Résultats obtenus	False
Nom de la méthode à tester	getStudentInfoIndexes
Nom de la fonction de test	test_all_student_info_indexes
Description du test	Test si lorsque les bons champs sont passés en paramètres, les indices retournés sont corrects.
Données d'entrées	indexes : [None] * 4 list_line : ["code_module", "code_presentation", "id_student", "random_field", "final_result"]
Résultats attendus	[0,1,2,4]
Résultats obtenus	[0,1,2,4]
Nom de la fonction de test	test_miss_a_student_info_index
Description du test	Test si lorsque des champs sont manquants, les indices retournés sont corrects.
Données d'entrées	indexes : [None] * 4 list_line : ["code_module", "code_presentation", "id_student", "random_field"]
Résultats attendus	[0,1,2,None]
Résultats obtenus	[0,1,2,None]
Nom de la méthode à tester	getStudentAssessmentIndexes
Nom de la fonction de test	test_all_student_assessment_indexes
Description du test	Test si lorsque les bons champs sont passés en paramètres, les indices retournés sont corrects.
Données d'entrées	indexes : [None] * 3 list_line : ["id_assessment", "id_student", "random_field", "score"]
Résultats attendus	[0,1,3]
Résultats obtenus	[0,1,3]
Nom de la fonction de test	test_miss_a_student_assessment_index
Description du test	Test si lorsque des champs sont manquants, les indices retournés sont corrects.
Données d'entrées	indexes : [None] * 3 list_line : ["id_assessment", "id_student", "random_field"]
Résultats attendus	[0,1,None]
Résultats obtenus	[0,1,None]
Nom de la méthode à tester	getAssessmentsIndexes
Nom de la fonction de test	test_all_assessments_indexes

Description du test	Test si lorsque les bons champs sont passés en paramètres, les indices retournés sont corrects.
Données d'entrées	indexes : [None] * 4 list_line : ["code_module", "code_presentation", "id_assessment", "random_field", "assessment_type"]
Résultats attendus	[0,1,2,4]
Résultats obtenus	[0,1,2,4]
Nom de la fonction de test	test_miss_an_assessments_index
Description du test	Test si lorsque des champs sont manquants, les indices retournés sont corrects.
Données d'entrées	indexes : [None] * 4 list_line : ["code_module", "code_presentation", "id_assessment", "random_field"]
Résultats attendus	[0,1,2,None]
Résultats obtenus	[0,1,2,None]
Nom de la méthode à tester	getCoursesIndexes
Nom de la fonction de test	test_all_courses_indexes
Description du test	Test si lorsque les bons champs sont passés en paramètres, les indices retournés sont corrects.
Données d'entrées	indexes : [None] * 3 list_line : ["code_module", "code_presentation", "random_field", "module_presentation_length"]
Résultats attendus	[0,1,3]
Résultats obtenus	[0,1,3]
Nom de la fonction de test	test_miss_a_courses_index
Description du test	Test si lorsque des champs sont manquants, les indices retournés sont corrects.
Données d'entrées	indexes : [None] * 3 list_line : ["code_module", "code_presentation", "random_field"]
Résultats attendus	[0,1,None]
Résultats obtenus	[0,1,None]
Nom de la méthode à tester	addStudentInfoValues
Nom de la fonction de test	test_all_courses_indexes
Description du test	Test si toutes les données des étudiants sont bien ajoutés à l'attribut "final_results"
Données d'entrées	indexes : [0,1,2,4] list_line : ["AAA", "2013J", "11391", "Random Value", "Pass"]
Résultats attendus	[["AAA", "2013J", "11391", "Pass"]]
Résultats obtenus	[["AAA", "2013J", "11391", "Pass"]]
Nom de la fonction de test	test_student_info_error_when_index_none
Description du test	Test si lorsqu'un indice est manquant, une exception est levée avec le bon texte.

Données d'entrées	indexes : [0,1,2,None] list_line : ["AAA", "2013J", "11391", "Random Value", "Pass"]
Résultats attendus	"Missing final_result index."
Résultats obtenus	"Missing final_result index."
Nom de la méthode à tester	addStudentAssessmentValues
Nom de la fonction de test	test_student_assessment_added_when_indexes_ok
Description du test	Test si toutes les données des étudiants sont bien ajoutés à l'attribut "scores".
Données d'entrées	indexes : [0,1,3] list_line : ["1752", "11391", "Random Value", "78"]
Résultats attendus	[["1752", "11391", "78"]]
Résultats obtenus	[["1752", "11391", "78"]]
Nom de la fonction de test	test_student_assessment_error_when_index_none
Description du test	Test si lorsqu'un indice est manquant, une exception est levée avec le bon texte.
Données d'entrées	indexes : [0,1,None] list_line : ["1752", "11391", "Random Value", "78"]
Résultats attendus	"Missing score index."
Résultats obtenus	"Missing score index."
Nom de la méthode à tester	addAssessmentsValues
Nom de la fonction de test	test_assessments_added_when_indexes_ok
Description du test	Test si toutes les données des étudiants sont bien ajoutés à l'attribut "assessments_types".
Données d'entrées	indexes : [0,1,2,4] list_line : ["AAA", "2013J", "1752", "Random Value", "TMA"]
Résultats attendus	[["AAA", "2013J", "1752", "TMA"]]
Résultats obtenus	[["AAA", "2013J", "1752", "TMA"]]
Nom de la fonction de test	test_assessments_error_when_index_none
Description du test	Test si lorsqu'un indice est manquant, une exception est levée avec le bon texte.
Données d'entrées	indexes : [0,1,2,None] list_line : ["AAA", "2013J", "1752", "Random Value", "TMA"]
Résultats attendus	"Missing assessment_type index."
Résultats obtenus	"Missing assessment_type index."
Nom de la méthode à tester	addCoursesValues
Nom de la fonction de test	test_all_courses_added_when_indexes_ok
Description du test	Test si toutes les données des étudiants sont bien ajoutés à l'attribut "modules_presentations_lengths".
Données d'entrées	indexes : [0,1,3] list_line : ["AAA", "2013J", "Random Value", "268"]
Résultats attendus	[["AAA", "2013J", "268"]]
Résultats obtenus	[["AAA", "2013J", "268"]]
Nom de la fonction de test	test_courses_error_when_index_none

Description du test	Test si lorsqu'un indice est manquant, une exception est levée avec le bon texte.
Données d'entrées	indexes : [0,1,None] list_line : ["AAA", "2013J", "Random Value", "268"]
Résultats attendus	"Missing module_presentation_length index."
Résultats obtenus	"Missing module_presentation_length index."
Nom de la méthode à tester	parse
Nom de la fonction de test	test_correct_dir_parse_successfully
Description du test	Test si les données sont bien récupérées lorsque l'on a le bon chemin vers le dossier contenant les données.
Données d'entrées	path : "Test/Data/AllFiles"
Résultats attendus	final_results[0] : ["AAA", "2013J", "11391", "Pass"] scores[0] : ["1752", "11391", "78"] assessments_types[0] : ["AAA", "2013J", "1752", "TMA"] modules_presentations_lengths[0] : ["AAA", "2013J", "268"]
Résultats obtenus	final_results[0] : ["AAA", "2013J", "11391", "Pass"] scores[0] : ["1752", "11391", "78"] assessments_types[0] : ["AAA", "2013J", "1752", "TMA"] modules_presentations_lengths[0] : ["AAA", "2013J", "268"]
Nom de la fonction de test	test_wrong_dir_does_nothing
Description du test	Test si les attributs sont vides lorsque le chemin vers le dossier contenant les données est incorrect.
Données d'entrées	path : "Test/Data/NoFiles"
Résultats attendus	final_results : [] scores : [] assessments_types : [] modules_presentations_lengths : []
Résultats obtenus	final_results : [] scores : [] assessments_types : [] modules_presentations_lengths : []

1.2 DataConverter

Nom de la méthode à tester	__init__
Nom de la fonction de test	test_init_ok_if_is_instance
Description du test	Test si l'initialisation est réussie lorsqu'un DataParser est passé en argument.
Données d'entrées	path : "Test/Data/AllFiles" dataParser : DataParser(path)
Résultats attendus	parsedData : dataParser convertedData : []
Résultats obtenus	parsedData : dataParser convertedData : []

Nom de la fonction de test	test_init_fail_if_is_not_instance
Description du test	Test si l'initialisation lève une exception "TypeError" lorsque le paramètre n'est pas une instance du type DataParser.
Données d'entrées	wrongType : "Fail"
Résultats attendus	TypeError
Résultats obtenus	TypeError
Nom de la méthode à tester	getModulePresentationLength
Nom de la fonction de test	test_should_get_module_presentation_length
Description du test	Test si getModulePresentationLength retourne la bonne valeur.
Données d'entrées	path : "Test/Data/AllFiles" final_result : ["AAA", "2013J", "11391", "Pass"]
Résultats attendus	"268"
Résultats obtenus	"268"
Nom de la méthode à tester	getAssessments
Nom de la fonction de test	test_should_get_module_presentation_length
Description du test	Test si getAssessments retourne la bonne valeur.
Données d'entrées	path : "Test/Data/AllFiles" final_result : ["AAA", "2013J", "11391", "Pass"]
Résultats attendus	[["1752", "TMA"], ["1753", "TMA"], ["1754", "TMA"], ["1755", "TMA"], ["1756", "TMA"]]
Résultats obtenus	[["1752", "TMA"], ["1753", "TMA"], ["1754", "TMA"], ["1755", "TMA"], ["1756", "TMA"]]
Nom de la méthode à tester	getScore
Nom de la fonction de test	test_should_get_score
Description du test	Test si getScore retourne la bonne valeur.
Données d'entrées	path : "Test/Data/AllFiles" id_assessment : "1752" id_student : "11391"
Résultats attendus	"78"
Résultats obtenus	"78"
Nom de la méthode à tester	getMeanEval
Nom de la fonction de test	test_should_get_mean_eval
Description du test	Test si getMeanEval retourne la bonne valeur.
Données d'entrées	path : "Test/Data/AllFiles" id_assessment : "1752"
Résultats attendus	70.11142061281338
Résultats obtenus	70.11142061281338
Nom de la méthode à tester	addScoresAndMeanEval
Nom de la fonction de test	test_should_add_scores_and_mean_eval_when_len_2
Description du test	Test si addScoresAndMeanEval ajoute correctement les notes et les moyennes à la liste des évaluations lorsque les sous-listes de "assessments" est de taille 2.

Données d'entrées	path : "Test/Data/AllFiles" final_result : ["AAA", "2013J", "11391", "Pass"] assessments : [[["1752", "TMA"], ["1753", "TMA"], ["1754", "TMA"], ["1755", "TMA"], ["1756", "TMA"]]]
Résultats attendus	[[["1752", "TMA", "78", 70.11142061281338], ["1753", "TMA", "85", 66.80116959064327], ["1754", "TMA", "80", 70.22658610271904], ["1755", "TMA", "85", 70.56765676567657], ["1756", "TMA", "82", 69.12751677852349]]
Résultats obtenus	[[["1752", "TMA", "78", 70.11142061281338], ["1753", "TMA", "85", 66.80116959064327], ["1754", "TMA", "80", 70.22658610271904], ["1755", "TMA", "85", 70.56765676567657], ["1756", "TMA", "82", 69.12751677852349]]
Nom de la fonction de test	test_should_raise_exception_when_assessments_len_not_2
Description du test	Test si addScoresAndMeanEval lève une exception lorsque les sous-listes de "assessments" ne sont pas de taille 2.
Données d'entrées	path : "Test/Data/AllFiles" final_result : ["AAA", "2013J", "11391", "Pass"] wrong_assessments : [[["1752", "TMA", "78", 70.11142061281338], ["1753", "TMA", "85", 66.80116959064327], ["1754", "TMA", "80", 70.22658610271904], ["1755", "TMA", "85", 70.56765676567657], ["1756", "TMA", "82", 69.12751677852349]]
Résultats attendus	ValueError
Résultats obtenus	ValueError
Nom de la méthode à tester	getMeanStudent
Nom de la fonction de test	test_should_add_scores_and_mean_eval_when_len_2
Description du test	Test si getMeanStudent retourne correctement la moyenne de l'étudiant lorsque les sous-listes de "assessments" sont de taille 4.
Données d'entrées	path : "Test/Data/AllFiles" assessments : [[["1752", "TMA", "78", 70.11142061281338], ["1753", "TMA", "85", 66.80116959064327], ["1754", "TMA", "80", 70.22658610271904], ["1755", "TMA", "85", 70.56765676567657], ["1756", "TMA", "82", 69.12751677852349]]
Résultats attendus	82
Résultats obtenus	82
Nom de la fonction de test	test_should_raise_exception_when_assessments_len_4
Description du test	Test si getMeanStudent et getGlobalMean lève une exception lorsque les sous-listes de "assessments" ne sont pas de taille 4.

Données d'entrées	path : "Test/Data/AllFiles" final_result : ["AAA", "2013J", "11391", "Pass"] wrong_assessments : [["1752", "TMA"], ["1753", "TMA"], ["1754", "TMA"], ["1755", "TMA"], ["1756", "TMA"]]
Résultats attendus	ValueError
Résultats obtenus	ValueError
Nom de la méthode à tester	getGlobalMean
Nom de la fonction de test	test_should_get_global_mean_when_assessments_len_4
Description du test	Test si getMeanStudent retourne correctement la moyenne de l'étudiant lorsque les sous-listes de "assessments" sont de taille 4.
Données d'entrées	path : "Test/Data/AllFiles" assessments : [["1752", "TMA", "78", 70.11142061281338], ["1753", "TMA", "85", 66.80116959064327], ["1754", "TMA", "80", 70.22658610271904], ["1755", "TMA", "85", 70.56765676567657], ["1756", "TMA", "82", 69.12751677852349]]
Résultats attendus	69.36686997007514
Résultats obtenus	69.36686997007514
Nom de la fonction de test	test_should_raise_exception_when_assessments_len_4
Description du test	Test si getMeanStudent et getGlobalMean lève une exception lorsque les sous-listes de "assessments" ne sont pas de taille 4.
Données d'entrées	path : "Test/Data/AllFiles" final_result : ["AAA", "2013J", "11391", "Pass"] wrong_assessments : [["1752", "TMA"], ["1753", "TMA"], ["1754", "TMA"], ["1755", "TMA"], ["1756", "TMA"]]
Résultats attendus	ValueError
Résultats obtenus	ValueError
Nom de la méthode à tester	convertCodeModule
Nom de la fonction de test	test_should_convert_code_module
Description du test	Test si "code_module" est bien convertis en une donnée numérique.
Données d'entrées	path : "Test/Data/AllFiles" code_module : "AAA"
Résultats attendus	656565
Résultats obtenus	656565
Nom de la méthode à tester	convertCodePresentation
Nom de la fonction de test	test_should_convert_code_presentation
Description du test	Test si "code_presentation" est bien convertis en une donnée numérique.
Données d'entrées	path : "Test/Data/AllFiles" code_presentation : "2013J"
Résultats attendus	201310
Résultats obtenus	201310

Nom de la méthode à tester	convertAssessmentType
Nom de la fonction de test	test_should_convert_code_presentation
Description du test	Test si "assessment_type" est bien convertis en une donnée numérique.
Données d'entrées	path : "Test/Data/AllFiles" type_tma : "TMA" type_cma : "CMA" type_exam : "Exam"
Résultats attendus	converted_tma : 1 converted_cma : 2 converted_exam : 3
Résultats obtenus	converted_tma : 1 converted_cma : 2 converted_exam : 3
Nom de la fonction de test	test_should_raise_exception_when_assessment_type_unknown
Description du test	Test si convertAssessmentType lève bien une exception lorsque "assessment_type" est inconnu.
Données d'entrées	path : "Test/Data/AllFiles" wrong_type : "examen"
Résultats attendus	ValueError
Résultats obtenus	ValueError
Nom de la méthode à tester	convertAssessments
Nom de la fonction de test	test_should_convert_assessments
Description du test	Test si "assessments" est bien convertis en une donnée numérique.
Données d'entrées	path : "Test/Data/AllFiles" assessments : [["1752","TMA","78",70.11142061281338], ["1753","TMA","85",66.80116959064327], ["1754","TMA","80",70.22658610271904], ["1755","TMA","85",70.56765676567657], ["1756","TMA","82",69.12751677852349]]
Résultats attendus	[[1752,1,79.0,71.11142061281338], [1753,1,86.0,67.80116959064327], [1754,1,81.0,71.22658610271904], [1755,1,86.0,71.56765676567657], [1756,1,83.0,70.12751677852349]]
Résultats obtenus	[[1752,1,79.0,71.11142061281338], [1753,1,86.0,67.80116959064327], [1754,1,81.0,71.22658610271904], [1755,1,86.0,71.56765676567657], [1756,1,83.0,70.12751677852349]]
Nom de la méthode à tester	convertLabel
Nom de la fonction de test	test_should_convert_label
Description du test	Test si "label" est bien convertis en une donnée numérique.

[illegible]

[illegible]

[illegible]

- [1] Jay BAINBRIDGE, James MELITSKI, Anne ZAHRADNIK, Eitel JM LAURIA, Sandeep JAYAPRAKASH et Josh BARON. « Using learning analytics to predict at-risk students in online graduate public affairs and administration education ». In : *Journal of Public Affairs Education* 21.2 (2015), p. 247-262.
- [2] Youngduck CHOI, Youngnam LEE, Dongmin SHIN, Junghyun CHO, Seoyon PARK, Seewoo LEE, Jineon BAEK, Chan BAE, Byungsoo KIM et Jaewe HEO. « Ednet : A large-scale hierarchical dataset in education ». In : *International Conference on Artificial Intelligence in Education*. Springer. 2020, p. 69-73.
- [3] Jacqueline FEILD, Nicholas LEWKOW, Sean BURNS et Karen GEBHARDT. « A generalized classifier to identify online learning tool disengagement at scale ». In : *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*. 2018, p. 61-70.
- [4] Shaobo HUANG et Ning FANG. « Predicting student academic performance in an engineering dynamics course : A comparison of four types of predictive mathematical models ». In : *Computers & Education* 61 (2013), p. 133-145.
- [5] Ali Shariq IMRAN, Fisnik DALIPI et Zenum KASTRATI. « Predicting student dropout in a MOOC : An evaluation of a deep neural network model ». In : *Proceedings of the 2019 5th International Conference on Computing and Artificial Intelligence*. 2019, p. 190-195.
- [6] Zafar IQBAL, Junaid QADIR, Adnan Noor MIAN et Faisal KAMIRAN. « Machine learning based student grade prediction : A case study ». In : *arXiv preprint arXiv :1708.08744* (2017).
- [7] Jakub KUZILEK, Martin HLOSTA et Zdenek ZDRAHAL. « Open university learning analytics dataset ». In : *Scientific data* 4.1 (2017), p. 1-8.
- [8] Marie LEFEVRE, Sébastien IKSAL, Julien BROISIN, Olivier CHAMPALLE, Valérie FONTANIEU, Christine MICHEL et Amel YESSAD. « DNE-GTnum2 Learning Analytics- Etat de l'art sur les outils et méthodes issus de la recherche française ». Thèse de doct. Ministère de l'éducation nationale, 2018.
- [9] Philipp LEITNER, Mohammad KHALIL et Martin EBNER. « Learning analytics in higher education—a literature review ». In : *Learning analytics : Fundamentals, applications, and trends* (2017), p. 1-23.

- [10] Yassine SAFSOUF, Khalifa MANSOURI et Franck POIRIER. « AN ANALYSIS TO UNDERSTAND THE ONLINE LEARNERS'SUCCESS IN PUBLIC HIGHER EDUCATION IN MOROCCO. » In : *Journal of Information Technology Education* 19 (2020).
- [11] Yassine SAFSOUF, Khalifa MANSOURI et Franck POIRIER. « Experimental Design of Learning Analysis Dashboards for Teachers and Learners ». In : *Proceedings of the Eighth ACM Conference on Learning@ Scale*. 2021, p. 347-350.
- [12] Mohammed SALIHOUN. « State of Art of Data Mining and Learning Analytics Tools in Higher Education ». In : *International Journal of Emerging Technologies in Learning (iJET)* 15.21 (2020), p. 58-76.
- [13] Michalis XENOS, Christos PIERRAKEAS et Panagiotis PINTELAS. « A survey on student dropout rates and dropout causes concerning the students in the Course of Informatics of the Hellenic Open University ». In : *Computers & Education* 39.4 (2002), p. 361-377.

I

Glossaire

Hubble HUman oBservatory Based on analysis of e-learning traces est un projet d'observatoire national pour la construction et le partage de processus d'analyse de données massives, issues des traces laissées dans des environnement de type e-learning. Voir : [Projet Hubble](#). 10

Learning Analytics Collecte et analyse des données associées aux étudiants dans le but d'améliorer leurs enseignements. 8

Machine Learning Apprentissage automatique permettant aux machines d'apprendre à partir de données. 8

Massive Open Online Courses Formations en ligne ouvertes à tous. 1

modèle psychométrique Mesure des caractéristiques psychologiques des individus notamment par l'utilisation de tests.. 11

Multilayer Perceptron Perceptron multicouche en français, est un type de réseaux de neurones composés de plusieurs couches, possédant chacun un nombre variables de neurones artificiel. Les neurones sont connectés à tous les neurones de la couche précédente et de la couche suivante. Aujourd'hui ce type de réseaux de neurones est utilisé avec le principe de rétropropagation et permet de résoudre des problèmes non linéaires.. 17

Multiple Linear Regression Régression linéaire multiple en français, est une méthode statistique de régression permettant de lier une variable dépendante à plusieurs variables indépendantes, contrairement à la régression linéaire simple qui cherche à expliquer le lien avec une seule variable indépendante.. 17

Open Source Désignation des logiciels respectant les critères de l'Open Source Initiative, c'est-à-dire la libre redistribution, l'accès au code source et l'autorisation de création de travaux dérivés. voir https://fr.wikipedia.org/wiki/Open_source. 12, 13

Radial Basis Function Fonction de base radiale en français, est une fonction ϕ vérifiant l'égalité $\phi(x) = \phi(\|x\|)$. La gaussienne est par exemple une fonction de base radiale. . 17

Support Vector Machine Machine à vecteurs de support en français, est une technique appliquée à des problèmes de classifications dans un premier temps, puis à de la régression. Cette technique se base sur la théorie de Vapnik-Chervonenkis. . 17

tableaux de bord Outils d'observation graphique permettant de visualiser différents indicateurs de performance. 10

traces Données issues d'interaction entre une plateforme numérique et les utilisateurs. 9

Transfer Learning Apprentissage par transfert en français, est une technique de transmission des connaissances d'un modèle à un autre. Cette technique est notamment utilisé avec les réseaux de neurones, pour entraîner de nouveaux modèles avec de plus faible jeu de données.. 25

J

Acronymes

APA Average Prediction Accuracy. 18, 19

ENT Espace Numérique de Travail. 8, 9, 13

IMD Indices of Multiple Deprivation. 14

LA Learning Analytics. 8–12, 14–16, 21, *Glossaire* : Learning Analytics

ML Machine Learning. 8, 11, 12, 24, 25, *Glossaire* : Machine Learning

MLP Multilayer Perceptron. 17–19, *Glossaire* : Multilayer Perceptron

MLR Multiple Linear Regression. 17, 18, *Glossaire* : Multiple Linear Regression

MOOCs Massive Open Online Courses. 1, 2, 8, 10, 13–15, 20, 25, 27, *Glossaire* : Massive Open Online Courses

PAP Percentage of Accurate Predictions. 18, 19

RBF Radial Basis Function. 17, 18, *Glossaire* : Radial Basis Function

SVM Support Vector Machine. 17–19, *Glossaire* : Support Vector Machine

Prédiction de décrochage dans le milieu universitaire à l'aide de réseaux de neurones :

Hugo LE BECHENNEC

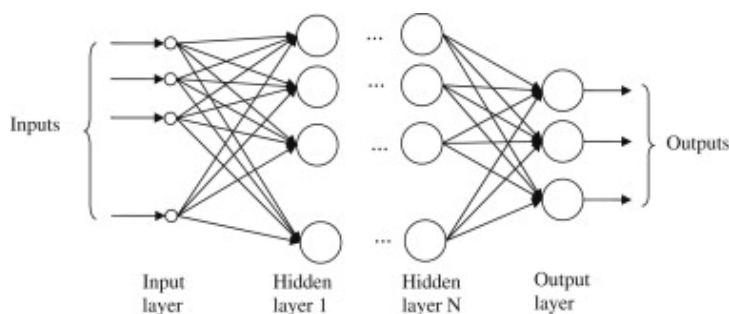
Encadrement : Sabine Barrat



En collaboration avec Polytech

Objectifs

L'objectif est de prédire les étudiants en risque de décrochage dans le milieu universitaire. Un premier objectif est d'entraîner un réseau de neurones sur des données issues de MOOCs. En cas de succès, l'objectif suivant est de mettre en place une plateforme numérique que les universités françaises pourrait exploiter pour mettre en place des actions pour les personnes en risque de décrochage, et ce au moyen de visualisation issues des prédictions.



Mise en œuvre

Le projet se découpe en deux parties. Une partie entraînement du réseau de neurones qui a pour objectif de prédire au mieux les décrochages sur une base de données issues de MOOCs. Une seconde partie d'implémentation d'une plateforme numérique depuis laquelle les enseignants peuvent se connecter et consulter des visualisations obtenus à partir des prédictions, permettant ainsi de mettre en place des actions pour les étudiants.

	P prédiction Non passé/Raté	P prédiction Réussi/Mention
Non passé/Raté réel	3819	1792
Réussi/Mention réel	33	5765

Résultats obtenus

Le meilleur modèle a permis d'avoir 84% de précision sur la base de tests. Ce résultat est déjà très intéressant puisque quasiment l'intégralité des étudiants prédits comme étant en décrochage le sont réellement comme l'on peut le constater sur la matrice de confusion. Des améliorations sont toutefois faisables au vu de l'écart que l'on peut constater entre la précision de la base de validation et de la base d'entraînement.

pic/best_model_1



Prédiction de décrochage dans le milieu universitaire à l'aide de réseaux de neurones :

Hugo LE BECHENNEC

Encadrement : Sabine Barrat

Objectifs

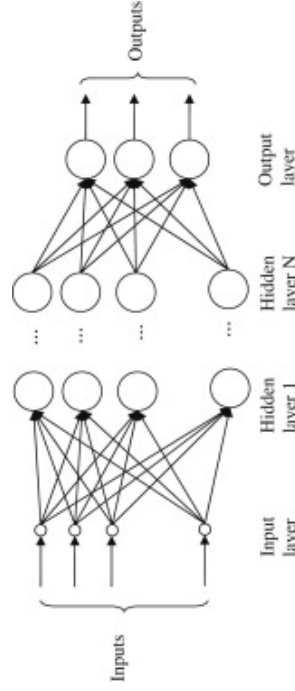
L'objectif est de prédire les étudiants en risque de décrochage dans le milieu universitaire. Un premier objectif est d'entraîner un réseau de neurones sur des données issues de MOOCs. En cas de succès, l'objectif suivant est de mettre en place une plateforme numérique que les universités françaises pourraient exploiter pour mettre en place des actions pour les personnes en risque de décrochage, et ce au moyen de visualisation issues des prédictions.

Mise en œuvre

Le projet se découpe en deux parties. Une partie entraînement du réseau de neurones qui a pour objectif de prédire au mieux les décrochages sur une base de données issues de MOOCs. Une seconde partie d'implémentation d'une plateforme numérique depuis laquelle les enseignants peuvent se connecter et consulter des visualisations obtenus à partir des prédictions, permettant ainsi de mettre en place des actions pour les étudiants.

Résultats obtenus

Le meilleur modèle a permis d'avoir 84% de précision sur la base de tests. Ce résultat est déjà très intéressant puisque quasiment l'intégralité des étudiants prédits comme étant en décrochage le sont réellement comme l'on peut le constater sur la matrice de confusion. Des améliorations sont toutefois faisables au vu de l'écart que l'on peut constater entre la précision de la base de validation et de la base d'entraînement.



Non passé/Raté réel	Prédiction Non passé/Raté	Prédiction Réussi/Mention
Réussi/Mention réel	3819	1792
	33	5765



Prédiction de décrochage dans le milieu universitaire à l'aide de réseaux de neurones

Résumé

Ce rapport concerne le projet de recherche et développement réalisé par tous les étudiants de 5ème année à Polytech Tours. Ce projet traite de la prédiction de décrochage dans le milieu universitaire à l'aide de réseau de neurones. Il se découpe en deux parties, l'entraînement du réseau de neurones jusqu'à atteindre des performances acceptables, et la réalisation d'un outil de visualisation afin d'aider les enseignants à prendre des décisions relatives aux étudiants à risques. Ce rapport est décomposé en 7 parties. Premièrement l'introduction du projet, présentant les objectifs et enjeux notamment. Ensuite, une description générale du projet est réalisée, ainsi qu'un état de l'art du domaine. L'analyse et la conception du projet sont ensuite détaillées, ainsi que sa mise en œuvre. Enfin un bilan sur les tâches réalisées et restant à faire est décrit suivi d'une annexe.

Mots-clés

Prédiction, décrochage, réseaux de neurones

Abstract

This report concerns the research and development project done by every 5th year students of Polytech Tours. This project is about predicting dropout in college using neural networks. It is divided in two parts, the neural network training to achieve acceptable performance and creating a visualization tool to help teachers to make decisions based on at-risk students.

This report contains 7 parts. First, the introduction of the project, presenting the objectives and stakes. Then, a general description of the project is made, as well as a state of the art of the field. The analysis and conception of the project are then detailed, followed by its implementation. Finally, a review of the realized and unrealized tasks is described, followed by an annex.

Keywords

Prediction, dropout, neural network

Entreprise

Polytech



Tuteur entreprise

Sabine BARRAT (Professeure)

Étudiant

Hugo LE BECHENNEC (DI5)

Tuteur académique

Sabine BARRAT