Master 2 - Économétrie Appliquée et Statistiques

IAE Nantes - Université de Nantes

SVM et Réseaux de Neurones

Google Brain - Ventilator Pressure Prediction

Lucie VRIGNAUD

Cécile MOCHER

2020-2021

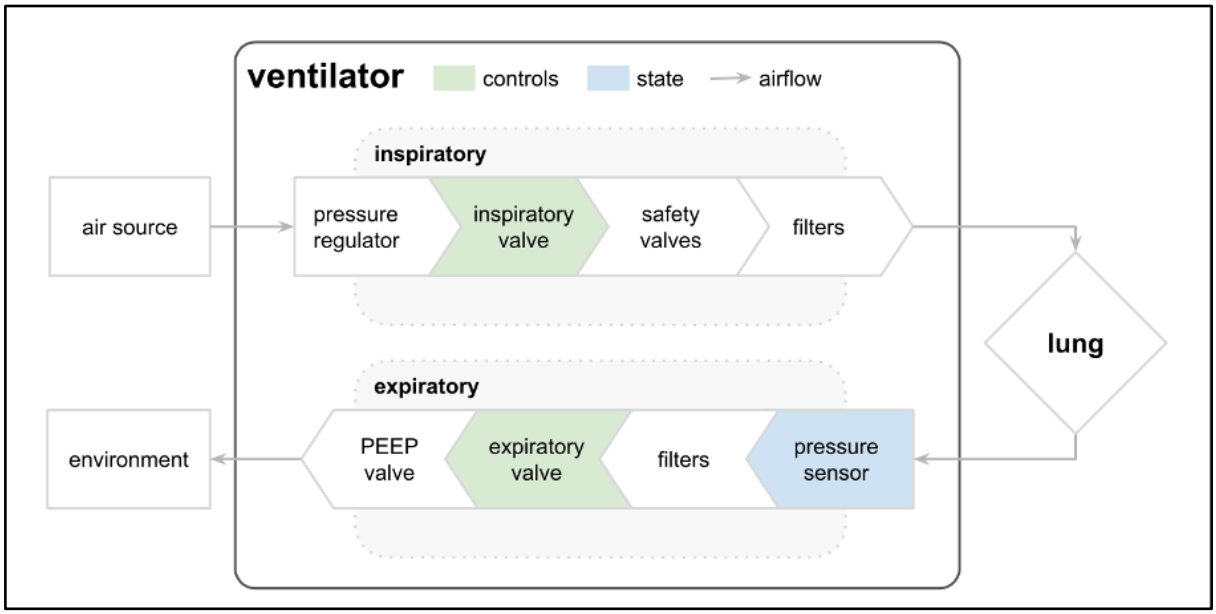
INTRODUCTION

Le Machine Learning est une application de l'intelligence artificielle qui propose aux systèmes informatiques la possibilité d'apprendre et de s'améliorer automatiquement à partir d'une expérience effectuée au préalable. Cette méthode se concentre sur le développement de programmes informatiques pouvant accéder aux données et utiliser celles-ci pour apprendre d'eux même, afin d'être appliqués sur des données inédites. L'objectif principal est de permettre aux algorithmes d'apprendre automatiquement sans intervention humaine ni assistance, et d'ajuster les actions en conséquence. Il existe différentes méthodes d'apprentissages automatiques. Ces dernières sont souvent classées comme supervisées ou non supervisées. Dans notre étude nous allons nous intéresser plus précisément aux algorithmes d'apprentissages supervisés, afin de répondre à notre problématique.

Dans ce travail, notre but est de participer à la compétition Kaggle "Google Brain - Ventilator Pressure Prediction". Dans cette compétition, nous devons prédire la pression des voies aériennes dans le circuit respiratoire à chaque étape de temps.
Notre objectif global est de simuler un respirateur connecté au poumon d'un patient sous sédation. En tenant compte des attributs pulmonaires et d'autres contraintes, nous devons simuler un ventilateur mécanique qui peut alléger le fardeau des cliniciens.
Les caractéristiques d'entrée sont les paramètres pulmonaires et les attributs du respirateur.

Les données du respirateur utilisées dans cette compétition ont été produites à l'aide d'un respirateur en circuit ouvert modifié, connecté à un poumon d'essai à souffle artificiel via un circuit respiratoire. Le schéma ci-dessous illustre la configuration, avec les deux entrées de contrôle surlignées en vert et la variable d'état (pression des voies aériennes) à prédire en bleu.
La première entrée de commande est une variable continue de 0 à 100 représentant le pourcentage d'ouverture de l'électrovanne inspiratoire pour laisser entrer l'air dans les poumons (c'est-à-dire que 0 est complètement fermé et aucun air n'est admis et 100 est complètement ouvert). La deuxième entrée de commande est une variable binaire représentant si la valve expiratoire est ouverte (1) ou fermée (0) pour laisser sortir l'air.

Figure 1 : Fonctionnement d'un respirateur



Source : https://www.kaggle.com/c/ventilator-pressure-prediction/data

Comme le montre cette figure, nous retrouvons ci-après les deux attributs décrivant les conditions d'un patient :

• R - attribut pulmonaire indiquant à quel point les voies respiratoires sont restreintes (en cmH2O/L/S). Physiquement, il s'agit du changement de pression par changement de débit (volume d'air par temps). Intuitivement, on peut imaginer faire exploser un ballon en le gonflant avec une paille. Nous pouvons changer R en changeant le diamètre de la paille, un R plus élevé étant plus difficile à souffler.

• C - attribut pulmonaire indiquant le degré de conformité du poumon (en ml/cmH2O). Physiquement, c'est le changement de volume par changement de pression. Intuitivement, on peut imaginer le même exemple de ballon. Nous pouvons changer C en changeant l'épaisseur du latex du ballon, avec un C plus élevé ayant un latex plus fin et plus facile à souffler.

Puis nous avons les paramètres du respirateur :

• u_in - l'entrée de commande de l'électrovanne inspiratoire. Plages de 0 à 100.

• u_out - l'entrée de commande de l'électrovanne d'expiration. Soit 0, soit 1.

Enfin nous retrouvons la variable cible que nous devons prédire :

• pression - la pression des voies aériennes mesurée dans le circuit respiratoire (mesurée en cmH2O) pour chaque ‘time_step’ donné de la série.

Ces données sont organisées par identifiants :

• id - identifiant de pas de temps unique sur l'ensemble d'un fichier

• breath_id - pas de temps unique pour les respirations

Dans le cadre de ce dossier nous utiliserons trois méthodes de classification : la régression logistique, les machines à support vectoriels (SVM linéaire et non linéaire) et les réseaux de neurones (ANN). Nous allons chercher à estimer du mieux possible la pression des voies aériennes à l’aide d’un modèle robuste de machine learning. Afin de faire de la classification, nous avons classifié la variable de la pression en 2 classes distinctes présentées ensuite.

Notre étude se présente de la manière suivante : tout d’abord nous allons expliquer brièvement la partie théorique des différentes méthodes que nous allons utiliser afin de répondre à la problématique. Dans une seconde partie, nous effectuons des statistiques descriptives de nos données. Dans une troisième partie, nous appliquerons ces méthodes à notre échantillon. Enfin, nous discuterons des résultats et du meilleur modèle obtenu.

MÉTHODOLOGIE

Initialement, notre variable à prédire était continue mais nous avons transformé celle-ci en variable binaire afin d’appliquer les méthodes vues en cours. Nous allons les présenter brièvement dans cette partie.

• Régression logistique

La régression logistique est un modèle mathématique qui combine un ensemble de variables prédictives avec une variable aléatoire binomiale. Elle est couramment utilisée dans le domaine de l'intelligence artificielle et du machine learning. Elle est considérée comme l'un des modèles d'analyse multivariée les plus simples à déchiffrer et analyser.

Un modèle de classification est un modèle qui classe chaque instance d'un jeu de données dans une catégorie. On distingue deux grands types de modèle de classification :

- Binaire : il n'y a que deux catégories possibles, par exemple : la potabilité d'un échantillon d'eau (potable ou non potable).
- Multi-classe : il y a au moins trois catégories, par exemple : la météo un jour donné (pluie, soleil ou neige).

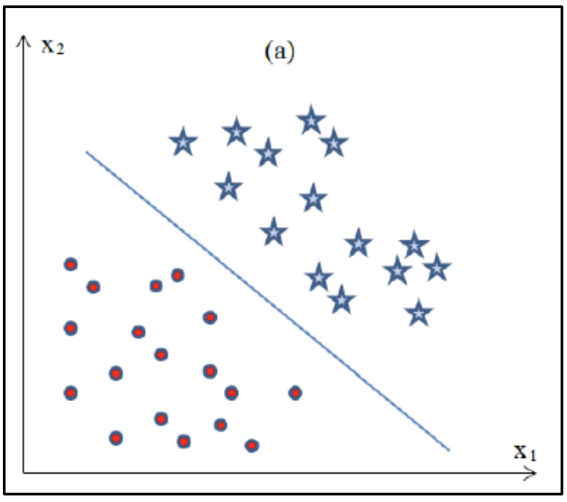
La régression logistique peut s'appliquer à ces deux types de classification mais nous nous concentrerons ici sur son application à une classification binaire.

● SVM

- ***Le cas linéaire***

La méthode des SVM est un algorithme qui peut être utilisé à des fins de classification et de régression. Il s'agit d'une méthode qui peut être adaptée à une relation linéaire ou non linéaire. De manière simple, les SVM reposent sur l'idée de trouver un hyperplan qui divise au mieux un jeu de données en deux. La représentation graphique ci-dessous explique cette méthode pour une relation linéaire :

Figure 2 : Exemple de séparateur linéaire

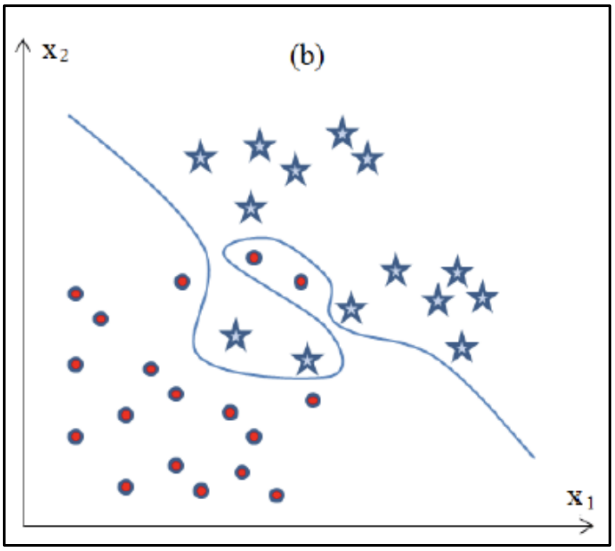


Source : https://www.researchgate.net/figure/Classification-SVM-a-SVM-lineaire-separation-par-une-ligne-droite-b-SVM-non_fig18_303899174

- ***Le cas non linéaire***

Dans le cas des SVM non linéaires, le problème est un peu plus complexe. En effet, les données ne sont pas toujours aussi propres et faciles à traiter. Il peut arriver que les données soient tellement mélangées qu'il en devient impossible de le séparer de manière linéaire. Ainsi, pour classer un jeu de données de ce type, il est nécessaire de passer à plan tridimensionnel. Cet agrandissement de l'espace de fonctionnalités est connu sous le nom de Kernel (noyau). Dans notre cas nous appliquerons le noyau Gaussien "RBF".
La séparation des classes est donc plus facile et nous pouvons constater ceci dans le graphique suivant :

Figure 3 : Exemple de séparateur non linéaire



Source : https://www.researchgate.net/figure/Classification-SVM-a-SVM-lineaire-separation-par-une-ligne-droite-b-SVM-non_fig18_303899174

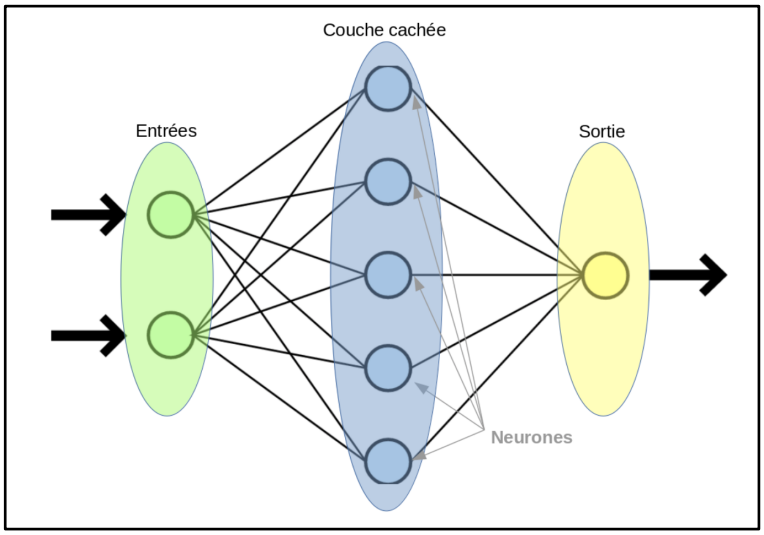
- ANN

L'algorithme des réseaux de neurones est un algorithme d'apprentissage supervisé de classificateurs binaires, séparant deux classes. Il s'agit d'un type de classifieur linéaire et du type de réseau de neurones artificiels le plus simple.
Dans le cas d'un classifieur binaire il ne peut y avoir que deux catégories de classification, et dans le cas d'un classifieur linéaire, les données d'entraînement doivent être classifiées en catégories correspondantes de sorte que toutes les données d'entraînement doivent être ordonnées dans ces deux catégories.

Bien que cet algorithme ait développé l'intelligence artificielle de façon majeure, les limites techniques ont rapidement été atteintes. En effet, un perceptron à une seule couche peut uniquement séparer les classes si elles sont séparables de façon linéaire.

Ainsi, il a été découvert, par la suite, qu'un perceptron multicouche permettait de classifier des groupes qui ne sont pas séparables de façon linéaire. Il permet donc de résoudre des problèmes que les algorithmes à une seule couche ne peuvent pas résoudre. L'information circule de la couche d'entrée vers la couche de sortie. Tandis qu'un modèle monocouche ne dispose que d'une seule sortie pour toutes les entrées. Ainsi, chaque couche se compose d'un nombre de neurones variables, et les neurones de la dernière couche sont les sorties globales. C'est donc un réseau possédant différents types d'interconnexions, une couche d'entrée et plusieurs couches actives. Il est donc capable d'approximer toute fonction continue, à condition que l'on fixe convenablement le nombre de neurones dans la couche cachée.

Figure 4 : Exemple de réseau de neurones multicouches



Source : https://www.natural-solutions.eu/blog/histoire-du-deep-learning

STATISTIQUES DESCRIPTIVES

● Analyse de forme

Nous travaillons sur la base d'apprentissage composée des 6 036 000 observations et de 7 variables. Parmi nos variables, 4 sont de types "integer" et 3 de type “float”.
Nous avons procédé à la vérification d’absence de valeurs manquantes dans notre base de données, nous n’en trouvons pas, nous pouvons ainsi fonctionner avec la base de données intégrale.

● Analyse de fond

- ***Statistiques descriptives***

Dans le tableau ci-dessous sont représentées les statistiques descriptives des variables de notre base de données. Notre variable à expliquer “pressure” est continue

Tableau 1 : Statistiques descriptives

	breath_id	R	C	time_step	u_in	u_out	pressure
count	6036000.00	6036000.00	6036000.00	6036000.00	6036000.00	6036000.00	6036000.00
mean	62838.86	27.04	26.08	1.31	7.32	0.62	11.22
std	36335.26	19.60	17.15	0.77	13.43	0.49	8.11
min	1.00	5.00	10.00	0.00	0.00	0.00	-1.90
25%	31377.00	5.00	10.00	0.64	0.39	0.00	6.33
50%	62765.50	20.00	20.00	1.31	4.39	1.00	7.03
75%	94301.00	50.00	50.00	1.97	4.98	1.00	13.64
max	125749.00	50.00	50.00	2.94	100.00	1.00	64.82

Dans ce tableau :
* La variable “breath_id” n’est pas intéressante car elle reprend les valeurs d’identifiants.
* Pour les variables “R” et “C” on remarque que celles-ci semblent prendre uniquement trois possibles au vu des résultats affichés, avec respectivement (5, 20, 50) et (10, 20, 50).
* La variable “time_step” prend des valeurs entre 0 et 2.94.

* La variable “u_in” variant entre 0 et 100, selon le taux de pourcentage de l'électrovanne ouvert pour laisser passer l’air dans les poumons, à une valeur de 4.98 pour le troisième quartile. On imagine qu’en général ce taux reste autour d’une valeur proche 0 et non 100.
* La variable “u_out” semble bien, quant-à-elle, prendre 2 valeurs : 0 ou 1.
* La variable “pressure” varie entre -1.90 et 64.82, l’étendue entre ses deux valeurs étant assez importante, on comprend que les caractéristiques de cette étude varie beaucoup d’un patient à un autre.

- ***Corrélations***

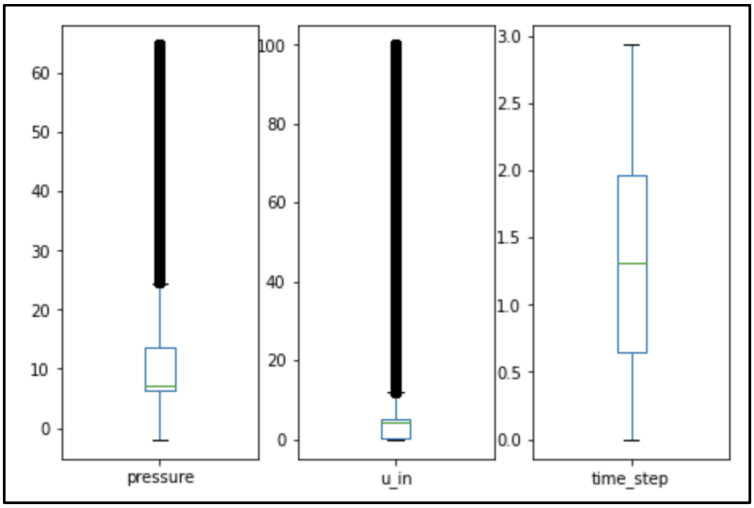
Tableau 2 : Corrélations entre les variables

	breath_id	R	C	time_step	u_in	u_out	pressure
breath_id	1.000000	0.001860	0.007222	-0.000213	-0.002378	-0.000100	-0.002394
R	0.001860	1.000000	-0.096070	-0.014535	-0.148120	-0.007594	0.015976
C	0.007222	-0.096070	1.000000	0.004936	0.151002	0.003720	-0.036727
time_step	-0.000213	-0.014535	0.004936	1.000000	-0.352276	0.839191	-0.524829
u_in	-0.002378	-0.148120	0.151002	-0.352276	1.000000	-0.416985	0.308136
u_out	-0.000100	-0.007594	0.003720	0.839191	-0.416985	1.000000	-0.614910
pressure	-0.002394	0.015976	-0.036727	-0.524829	0.308136	-0.614910	1.000000

Ce tableau des corrélations entre les variables de notre base ne montre pas de très forts liens entre certaines. Seulement la corrélation entre “u_out” et “time_step” est plus élevée que le reste.

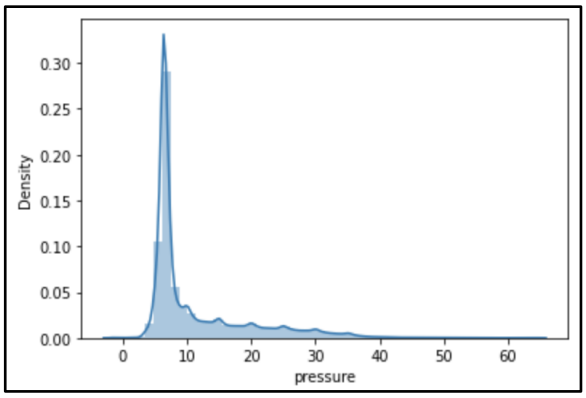
- ***Graphiques***

Figure 5 : Boîtes à moustaches des variables de types float



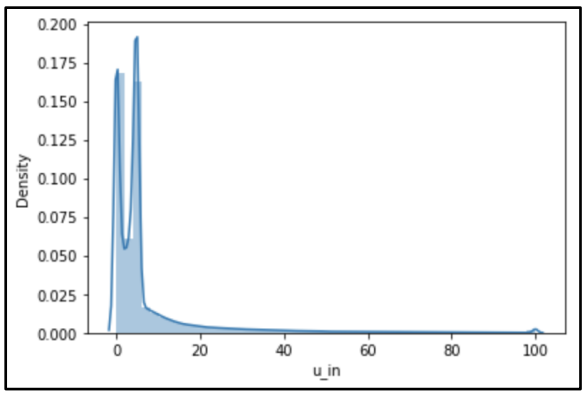
Nous remarquons un nombre important de valeurs atypiques pour les variables “pressure” et “u_in” contrairement à la variable “time_step” qui n’en possède aucune. Cela semble s’expliquer par les distributions, propres à chacune de ces variables, que nous allons afficher ci-après.

Figure 6 : Représentation graphique de la variable “pressure”



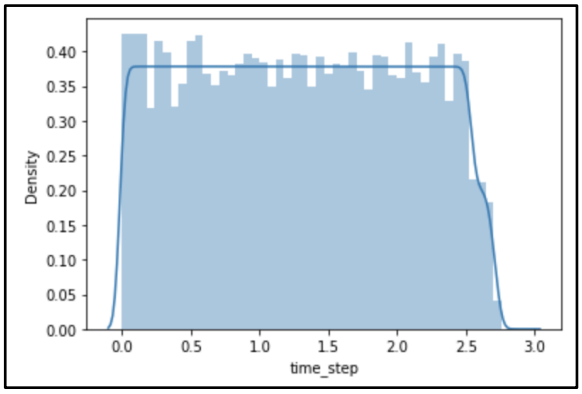
Nous comprenons facilement avec la représentation de la densité de cette variable, que peu d’observations se trouvent au-delà de 10 cmH2O pour cette variable, celles-ci se trouvant en masse avant cette valeur.

Figure 7 : Représentation graphique de la variable “u_in”



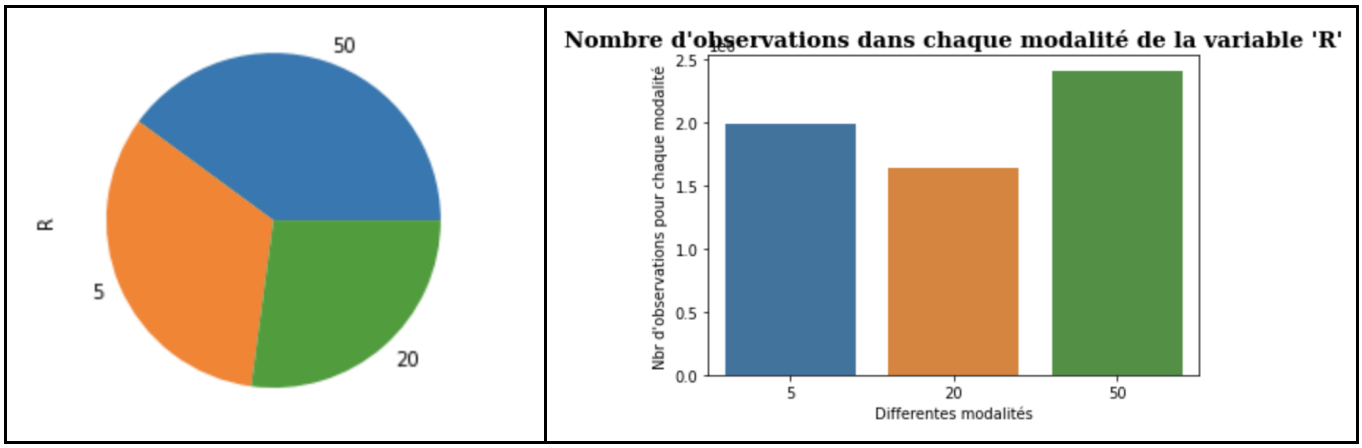
De même pour la variable “u_in”, sa densité est majoritairement répartie entre 0 et 10 sur une plage allant de 0 à 100, les valeurs après 10 sont très étalées et peu nombreuse comme remarqué sur le tableau des statistiques descriptives où le 3ème quartile de cette variable vaut 4,98.

Figure 8 : Représentation graphique de la variable “time_step”



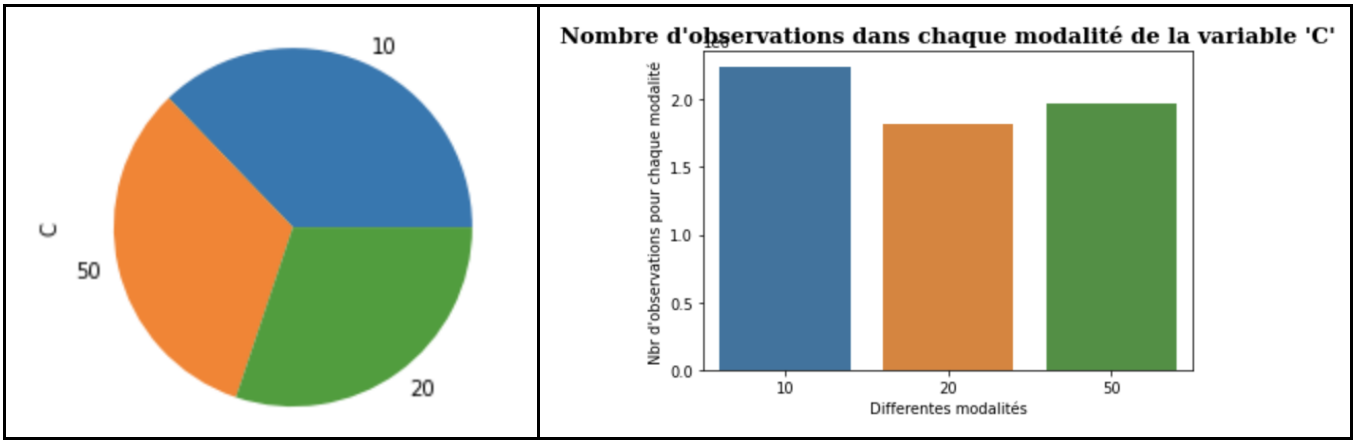
Contrairement aux deux variables précédentes, la densité de “time_step” est en un bloc avec quelques variations mais celles-ci ne permettent de permettre une quelconque anomalie parmi les observations. Cela confirme l’absence de valeur atypique sur la boîte à moustaches.

Figure 9 : Représentation graphique de la variable “R”



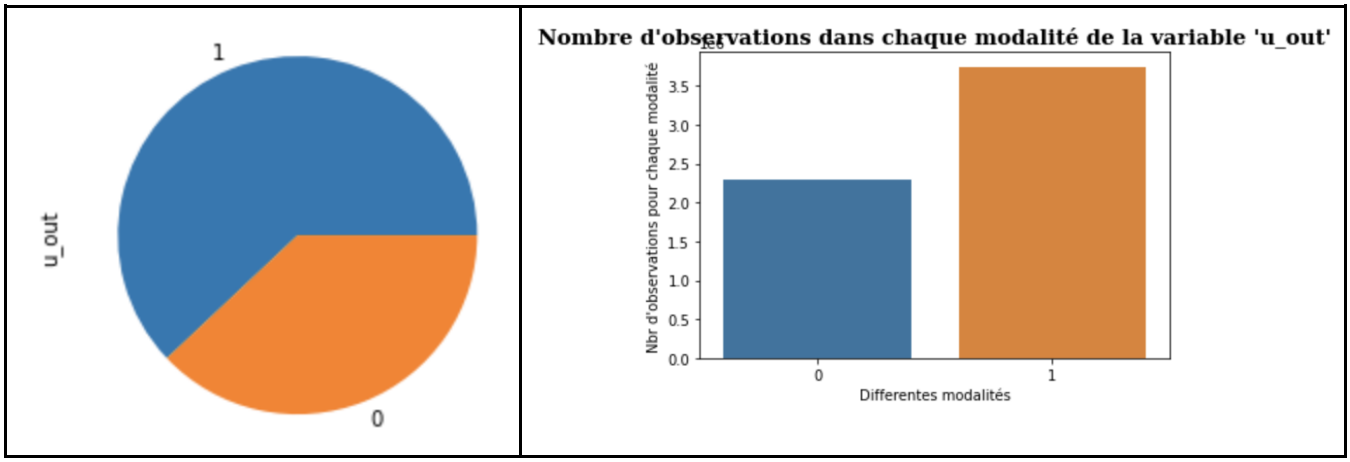
La répartition de la variable “R” est relativement égale entre les 3 valeurs que prend celle variable. On remarque tout de même une majorité de valeur pour 50 et une minorité pour 20.

Figure 10 : Représentation graphique de la variable “C”



La répartition de la variable “C” ressemble grandement à celle de la variable “R”. Ici, on retrouve une majorité de valeur pour 10 et une minorité pour 20.

Figure 11 : Représentation graphique de la variable “u_out”



Pour la variable binaire “u_out”, nous retrouvons davantage d’observations pour la modalité 1, correspondant à l'ouverture de l'électrovanne expiratoire.

Pour la suite de l’étude nous avons transformé la variable quantitative “pressure” en une variable catégorielle prenant 2 modalités :

— 0 si Médiane de “pressure” ≤ 7,03

— 1 si Médiane de “pressure” > 7,03

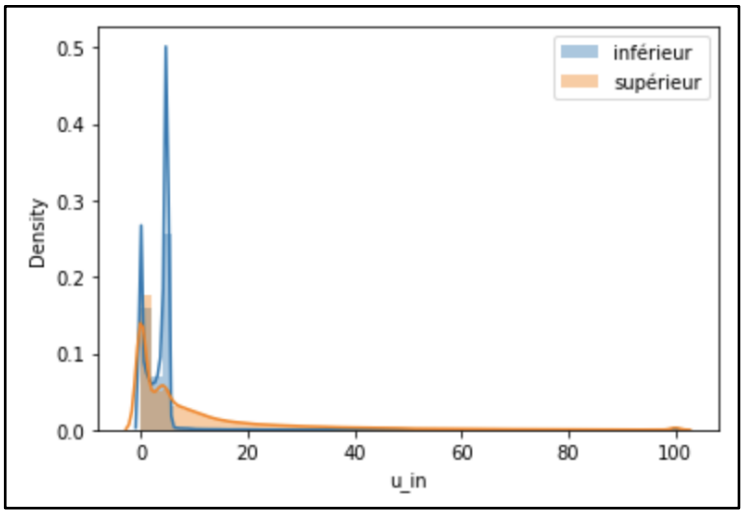
Nous retrouvons cette variable, que nous avons nommée “y”, en dernière colonne du tableau ci-après.

Tableau 3 : Aperçu de la base données avec la nouvelle variable “y”

	breath_id	R	C	time_step	u_in	u_out	pressure	y
0	1	20	50	0.000000	0.083334	0	5.837492	0.0
1	1	20	50	0.033652	18.383041	0	5.907794	0.0
2	1	20	50	0.067514	22.509278	0	7.876254	1.0
3	1	20	50	0.101542	22.808822	0	11.742872	1.0
4	1	20	50	0.135756	25.355850	0	12.234987	1.0

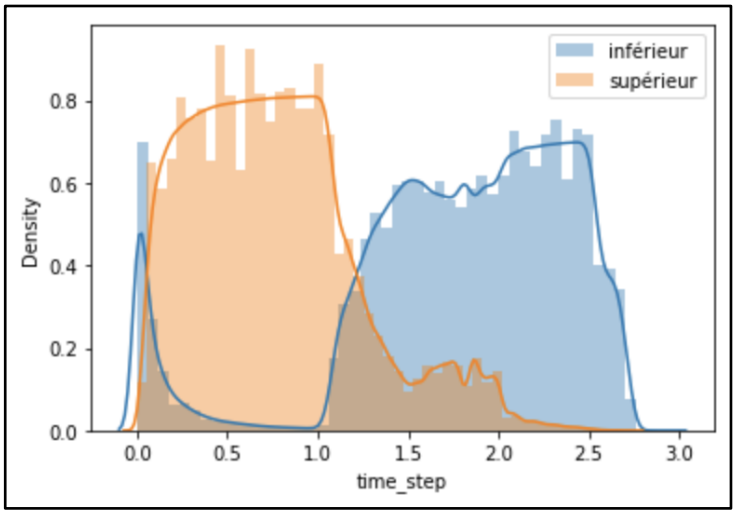
C’est ainsi que nous avons créé deux sous-ensembles de notre base de données de départ en disposant d’un côté, d’une sous-base prenant en compte “y = 0”, et de l’autre, une deuxième sous-base prenant en compte “y = 1”.
À partir de cette décomposition nous avons pu afficher et analyser les graphiques suivants.

Figure 12 : Représentation de la variable “u_in” selon les deux modalités de “y”



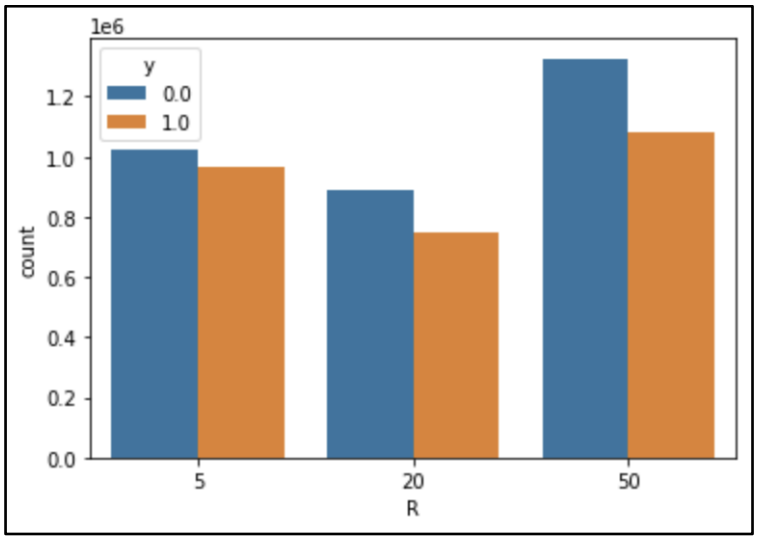
La distribution de la variable “u_in” est plus étalée pour le sous-ensemble supérieur et plus dense pour la sous-ensemble inférieur. Cette différence nous amène à supposer l’impact de la variable “u_in” sur la variable “y” à prédire.

Figure 13 : Représentation de la variable “time_step” selon les deux modalités de “y”



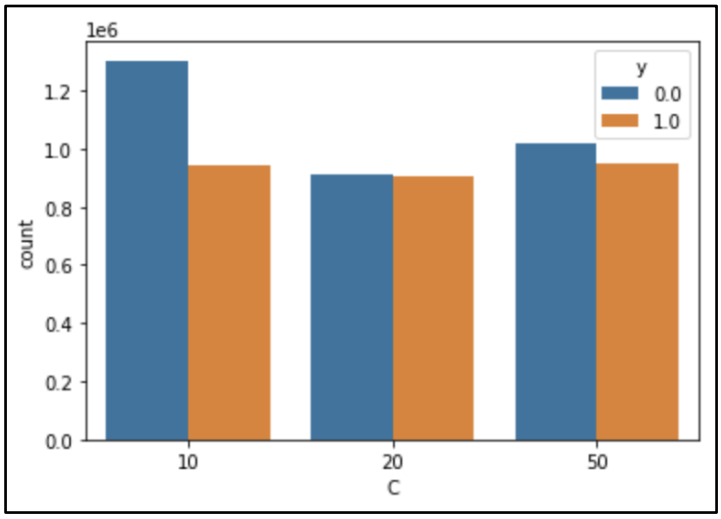
On observe de nouveau deux distributions différentes selon les deux sous-ensembles séparés par la médiane de la variable “pressure”. Ainsi, d’après ce graphique nous supposons que la variable “time_step” aura un impact sur la prédiction de la variable binaire “y”.

Figure 14 : Représentation de la variable “R” selon les deux modalités de “y”



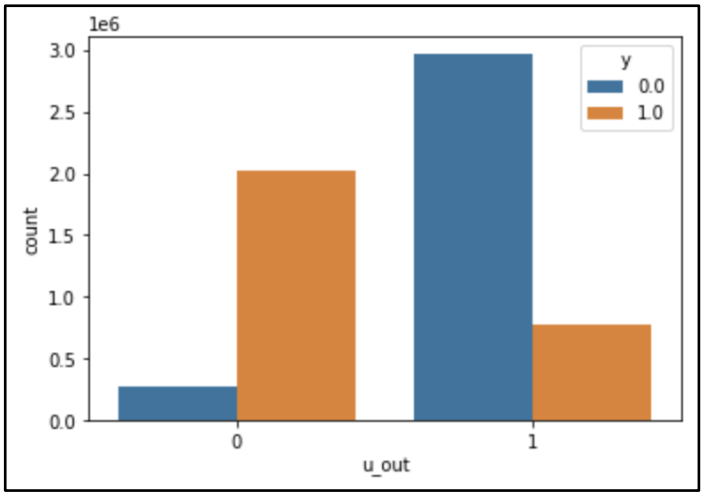
La répartition parmi les 3 valeurs possibles de la variable “R” des deux sous-ensembles est relativement identique.

Figure 15 : Représentation de la variable “C” selon les deux modalités de “y”



La répartition parmi les 3 valeurs possibles de la variable “C” des deux sous-ensembles est relativement identique sauf pour la valeur 10 où on remarque davantage d’effectif pour “y = 0”.

Figure 16 : Représentation de la variable “u_out” selon les deux modalités de “y”



Lorsque la variable “y” vaut 0, c’est-à-dire que la pression est dans la partie inférieure de notre base, la variable “u_out” est majoritairement de “1”, c'est-à-dire que l’électrovanne d’expiration est ouverte. Cela fonctionne à l’inverse lorsque “y” vaut 1. Nous supposons que cette variable aura un impact sur la prédiction de notre variable “y”.

MODÉLISATIONS

Pour nos modélisations nous allons les effectuer sur un échantillon de 2000 observations. Les résultats ne variant pas énormement entre 2000 observations et plus de 6 millions, nous préférons appliquer ces modèles sur un plus petit jeu de données.

- ## Régression logistique

Nous avons commencé par prédire notre variable à expliquer via un modèle de régression logistique.

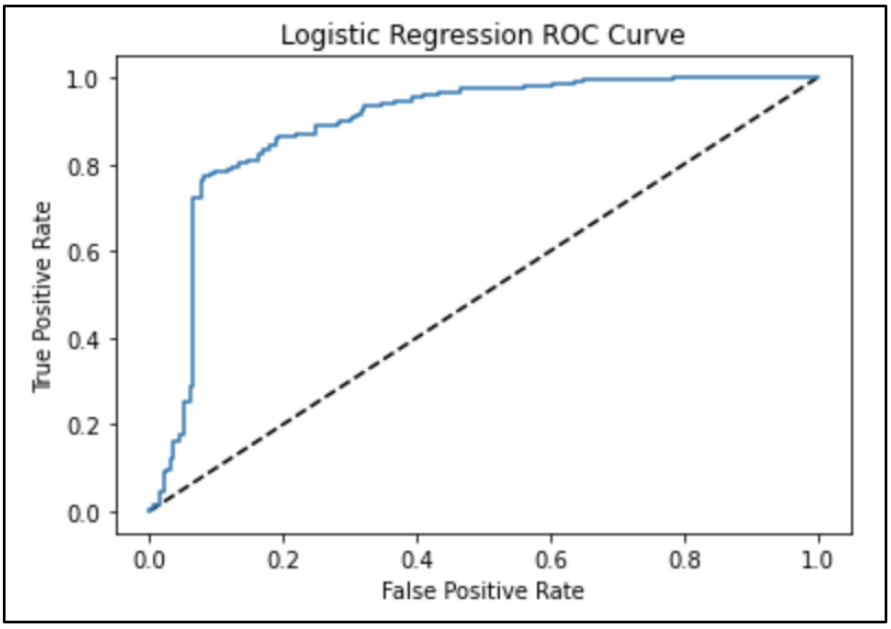
Tableau 4 : Résultats de notre modèle de régression logistique

```
Accuracy: 0.8425
AUC: 0.890406162464986
Confusion Matrix: [[177  19]
 [ 44 160]]
              precision    recall  f1-score   support

         0.0       0.80      0.90      0.85       196
         1.0       0.89      0.78      0.84       204

    accuracy                           0.84       400
   macro avg       0.85      0.84      0.84       400
weighted avg       0.85      0.84      0.84       400

Tuned Model Parameters: {'logreg__C': 31.622776601683793}
```

Nous retrouvons la valeur de R2 à 0.8425, cela est un très bon indicateur de qualité pour notre modèle de régression logistique.

Une courbe ROC constitue un meilleur moyen d'évaluer la capacité d'un modèle de régression logistique binaire à classer correctement les observations. Nous l'affichons ci-dessous.

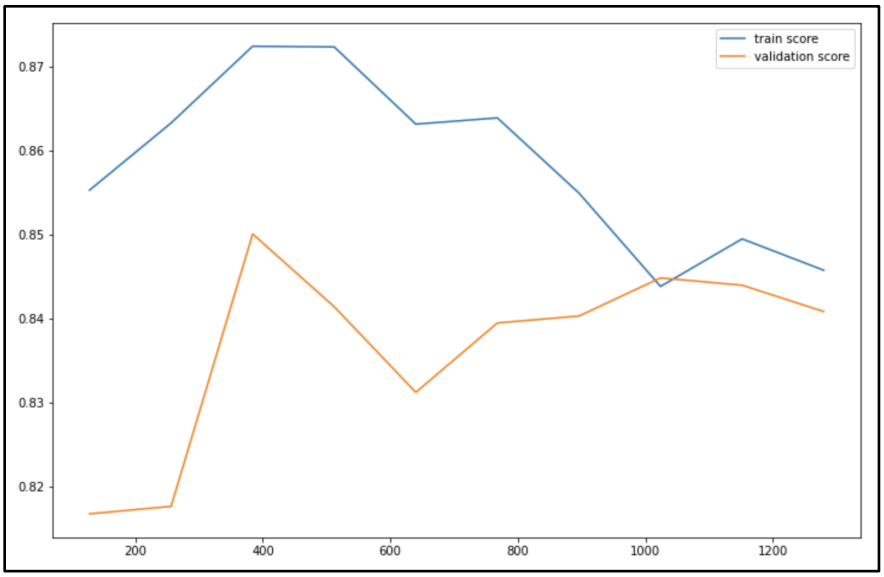
Figure 17 : Courbe ROC associée au modèle de régression logistique



La zone sous la courbe ROC (notée AUC) est une mesure de discrimination. Un modèle avec une zone importante sous la courbe ROC suggère qu'il est capable de prévoir correctement la valeur de la réponse de l'observation. Ici, comme affiché dans le tableau précédent, la valeur de AUC est de 0.89, cela signifie de nouveau que ce modèle est un bon modèle de prédiction.

Enfin, nous affichons le graphique des courbes d'apprentissage associées à cette modélisation.

Figure 18 : Courbes d'apprentissage associées au modèle de régression logistique



Le “train score” correspond à l’indicateur de performance du jeu d’apprentissage et le “validation score” correspond à celui du jeu de test.
Ayant peu d'observations dans l’estimation de ce modèle, nous nous attendions à ce que notre modèle ait des difficultés à prédire des données qu'il n'a pas étudié, or, on remarque également qu’avec un nombre croissant d'observations le score de validation s'améliore passant de 0.81 à 0.85 rapidement.
Nous observons que les deux courbes se rejoignent au niveau de 1000 observations, notre échantillon de 2000 observations est donc suffisant pour avoir une bonne estimation avec ce modèle. Mais cette intersection des courbes ne nous permet pas d’affirmer qu’elles se stabilisent pour la suite, n’ayant pas la capacité de calcul nécessaire sur nos ordinateurs pour pouvoir tester plus d'observations.

- SVM linéaire

Dans un second temps, nous avons appliqué la méthode des SVM linéaire sur notre jeu de données.

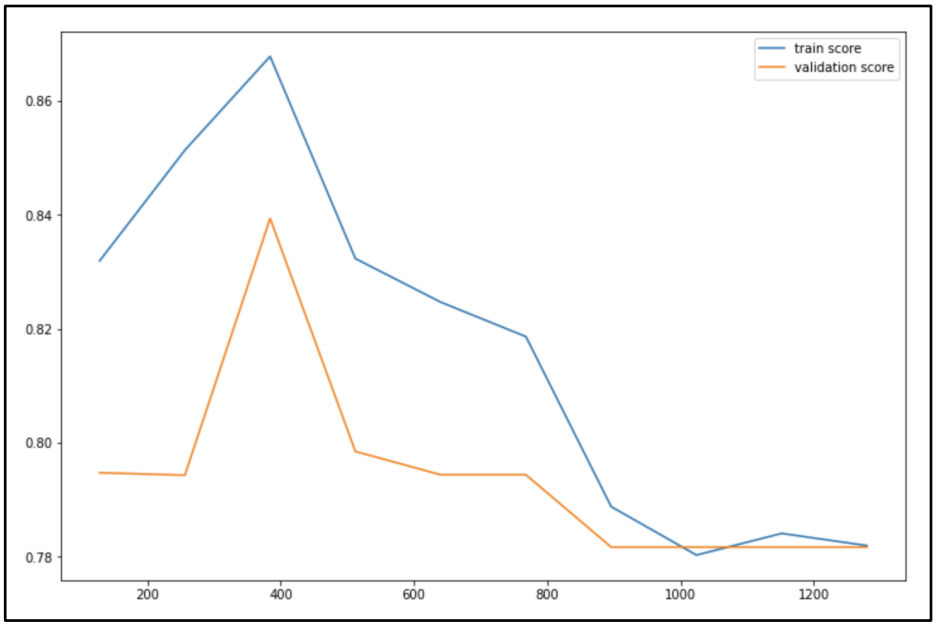
Tableau 5 : Résultats de notre modèle SVM linéaire

```
Accuracy: 0.8075
Confusion Matrix: [[183  13]
 [ 64 140]]
              precision    recall  f1-score   support

         0.0       0.74      0.93      0.83       196
         1.0       0.92      0.69      0.78       204

    accuracy                           0.81       400
   macro avg       0.83      0.81      0.81       400
weighted avg       0.83      0.81      0.80       400

Tuned Model Parameters: {'SVM__C': 1}
```

Nous retrouvons la valeur de R2 à 0.8075, cela est un très bon indicateur de qualité pour notre modèle SVM linéaire, mais c’est légèrement plus faible que pour la régression logistique.

De la même manière, nous affichons le graphique des courbes d’apprentissage.

Figure 19 : Courbes d'apprentissage associées au modèle SVM linéaire



De la même manière que pour les courbes d'apprentissage du modèle de régression logistique, nous observons que les deux courbes se rejoignent au niveau de 1000 observations, notre échantillon de 2000 observations est donc suffisant pour avoir une bonne estimation avec ce modèle. Cette fois-ci les deux courbes semblent se stabiliser davantage.

- SVM avec kernel RBF

Dans un troisième temps, nous avons appliqué la méthode des SVM avec kernel RBF sur notre jeu de données.

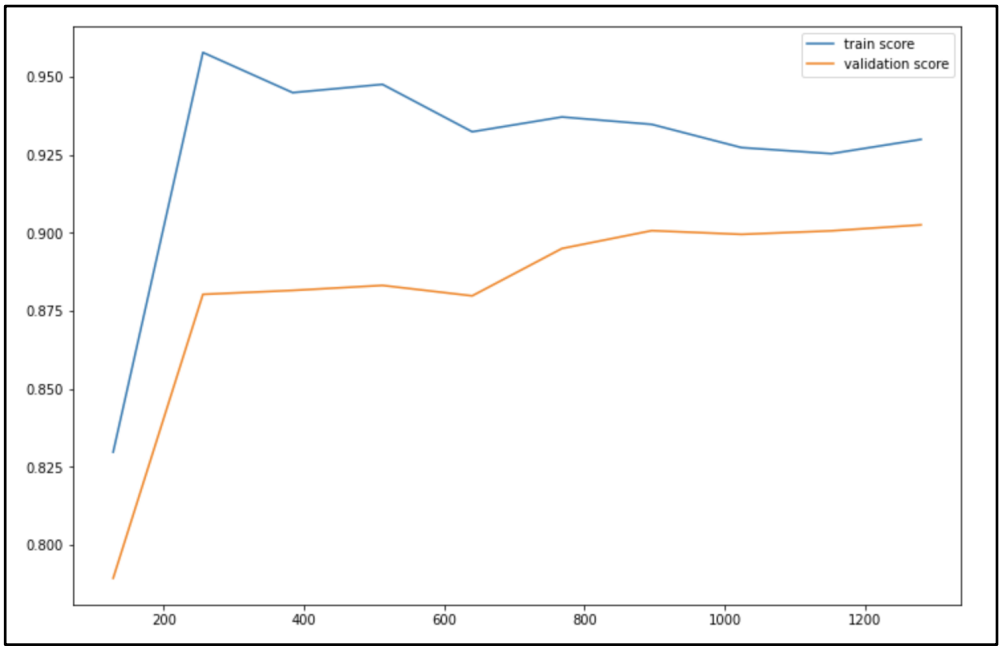
Tableau 6 : Résultats de notre modèle SVM à kernel RBF

```
Accuracy: 0.9075
Confusion Matrix: [[184  12]
 [ 25 179]]
              precision    recall  f1-score   support

         0.0       0.88      0.94      0.91       196
         1.0       0.94      0.88      0.91       204

    accuracy                           0.91       400
   macro avg       0.91      0.91      0.91       400
weighted avg       0.91      0.91      0.91       400

Tuned Model Parameters: {'SVM__C': 100, 'SVM__gamma': 0.1}
```

Nous retrouvons la valeur de R2 à 0.9075, cela est un très bon indicateur de qualité pour notre modèle SVM à kernel RBF. Il s'agit de notre meilleur résultat.

De la même manière, nous affichons le graphique des courbes d'apprentissage.

Figure 20 : Courbes d'apprentissage associées au modèle SVM à kernel RBF



Les courbes d'apprentissage du modèle SVM à kernel RBF ont une forme relativement identique. Elles ne se rejoignent pas mais semblent être en bonne voie pour se stabiliser par la suite. Nous concluons sur le fait qu'il nous manque des observations pour aboutir à des résultats optimaux pour ce modèle, or de nouveau, la capacité de calculs de nos ordinateurs nous a restreint sur la dimension de l'échantillon testé.

- ANN

Pour la méthode des réseaux de neurones, nous avons dans un premier temps calculé les valeurs optimales à appliquer aux paramètres de notre réseau de neurones afin d'optimiser les résultats.
Nous cherchons les paramètres optimaux pour :

- "batch_size" : correspondant au nombre d'échantillons qui seront propagés à travers le réseau,

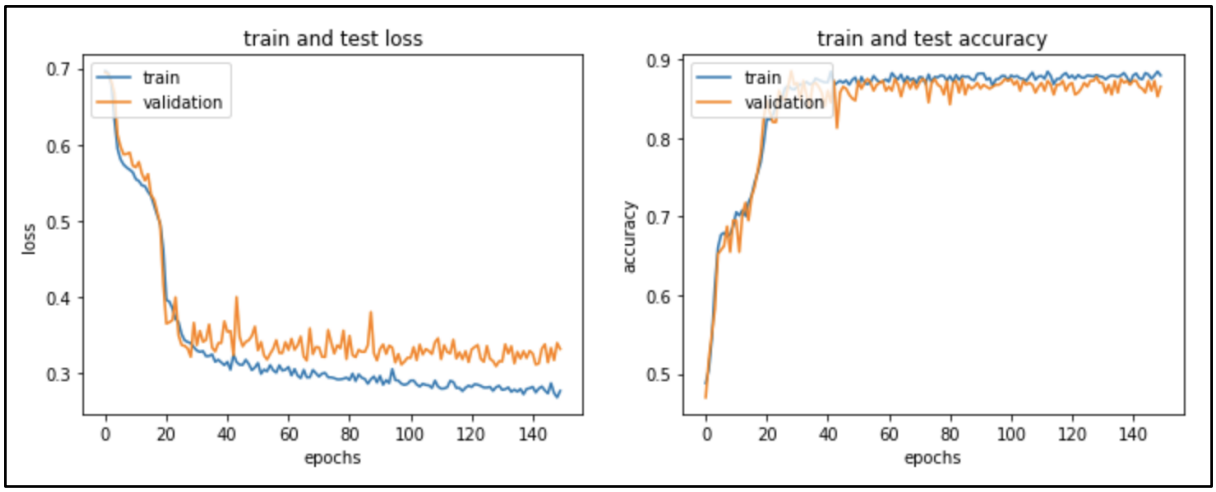
- “epochs” : 1 epoch, c'est lorsqu'un ensemble de données entier est transmis dans le réseau de neurones 1 seule fois,
- “optimizer” : fonction d’optimisation,
- “activation” : fonction d’activation,
- “nn” : nombre de neurones,
- “nl” : nombre de couches cachées (en plus celle ajoutée d’office dans notre modèle).

Notre modèle applique les paramètres optimaux suivants :

- batch_size = 10
- epochs = 150
- optimizer = Adam
- activation = tanh
- nl = 3
- nn = 10

La qualité d’ajustement du modèle est de 0.8719.

Figure 21 : Courbes associées au modèle ANN



Finalement, les courbes semblent se stabiliser à partir de 50 epochs, nous aurions donc pu nous arrêter à ce nombre de répétitions.

CONCLUSION

Dans cette étude, nous avons analysé notre jeu de données et nous avons mis en place des modèles de Machine Learning et de Deep Learning. Notre meilleur modèle est le modèle SVM à Kernel RBF d’une qualité de 90,75%.