neuralcode

January 20, 2022

1 Introduction

1.1 Contexte

Les neurones du cerveau communiquent des informations en générant des séquences d'impulsions électriques (événements binaires, appelés pics). Ces séquences de pics (binaires) provenant de populations de neurones sont connues pour transporter des informations sur l'état actuel de l'activité cérébrale d'un animal. Cependant, les séquences de pics des neurones individuels, agissant comme un code neuronal, devraient également refléter le traitement de l'information dans le cerveau. Comprendre ce code et prédire l'état comportemental ou une action à partir de l'activité neuronale sous-jacente est l'un des problèmes centraux des neurosciences quantitatives. La possibilité d'utiliser les données de stimulation d'un seul neurone pour en déduire l'état du cerveau peut réduire considérablement le nombre d'expériences que les biologistes doivent réaliser. À un niveau plus fondamental, le défi aidera à quantifier la façon dont le code neuronal unicellulaire reflète l'état du cerveau.

1.2 Objectif

L'objectif du challenge est de classifier l'état d'activité cérébrale d'un animal sur la base des schémas d'activation de ses neurones individuels. À cette fin, les participants reçoivent des enregistrements de séquences de pics neuronales provenant de l'hippocampe de rats. À chaque séquence de pics de l'ensemble de données correspond une étiquette d'état d'activité (deux états du cerveau, étiquetés STATE1 ou STATE2). Il s'agit donc d'un problème de classification binaire, où chaque échantillon de données est une série temporelle et où les participants doivent prédire à quelle classe appartient un échantillon de série temporelle donné.

1.3 Description des données

Ce problème est un problème de classification supervisée, les participants reçoivent donc un certain nombre d'échantillons de données avec les étiquettes correspondantes.

Dans les fichiers de données d'entrée, chaque ligne est définie par un ID unique et contient un échantillon de séries temporelles de 50 valeurs chacune. Ces séries temporelles sont des séquences de temps d'apparition des pics (unités de temps arbitraires). Chaque ID est lié à l'enregistrement d'un seul neurone dans un certain état. Chaque ligne du fichier de données d'entrée contient également un numéro d'identification de cellule appelé `neuron_id`.

La première ligne de ce fichier d'entrée est l'en-tête, et les colonnes sont séparées par des virgules. La première colonne correspond à ID : numéro d'identification de la ligne, il est lié à l'ID de l'étiquette fournie dans le fichier de sortie. La deuxième colonne correspond au numéro d'identification du

neurone (`neuron_id`). Les 50 colonnes suivantes correspondent aux valeurs consécutives des temps d'apparition des pics.

Voici un exemple de fichier d'entrée :

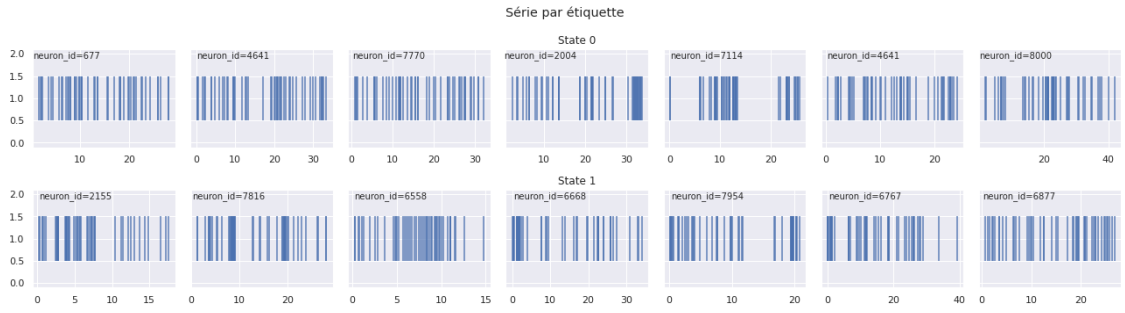
ID	neuron_id	timestamp_0	timestamp_1	timestamp_2	...	timestamp_50
1	2717	0.2406	0.6191	0.9372		5.6905
2	668	0.5332	0.5410	2.6912	...	3.9846
3	7678	0.3259	0.3863	0.4667		5.7914
4	668	0.1846	0.2874	0.6987		7.3247

Le fichier d'entrainement contient l'étiquette `TARGET` pour chaque ID, où la `TARGET` est l'étiquette de l'état du cerveau. La première ligne de ce fichier est l'en-tête et les colonnes sont séparées par des virgules. Les colonnes correspondent respectivement au numéro d'identification de la ligne et à la valeur de l'état cérébral réel pour cette ligne spécifique, comme le montre l'exemple suivant :

ID	TARGET
1	0
2	1
3	0
4	1

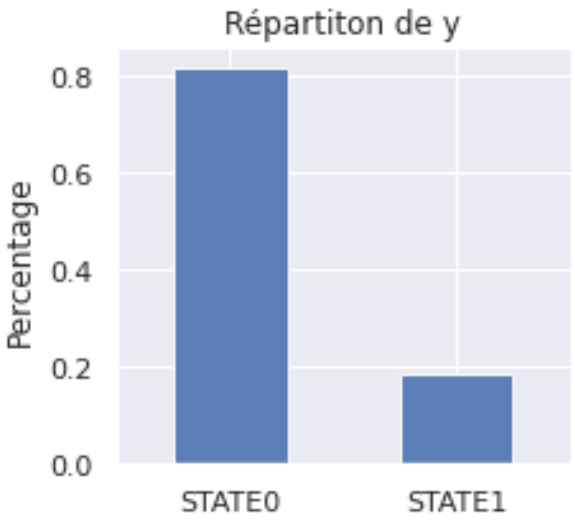
où `TARGET` égale à 0 correspond à l'état "STATE1", et `TARGET` égale à 1 correspond à l'état "STATE2". Le même neurone (avec le même "neuron_id") peut se trouver à la fois dans l'état "STATE1" et dans l'état "STATE2", en fonction du moment exact où la série de pointes est extraite de l'enregistrement.

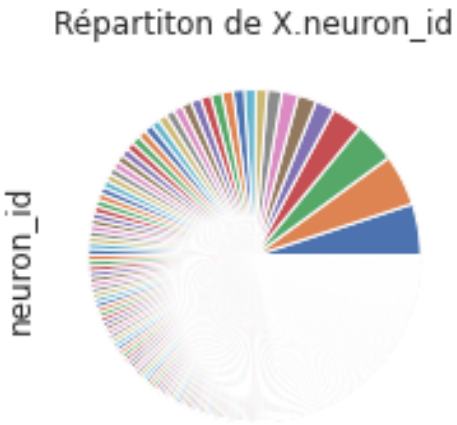
La métrique utilisée dans ce défi pour désigner le participant gagnant est le score kappa de Cohen. Le choix de cette métrique est motivé par la présence d'un déséquilibre de la distribution des classes dans l'ensemble de données.



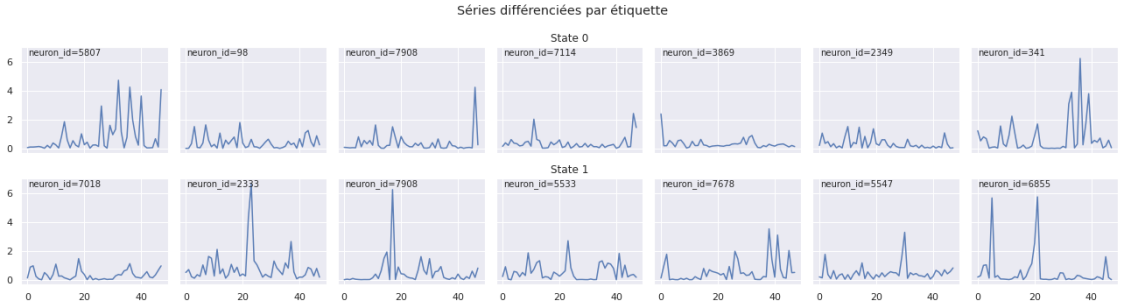
Il est bien difficile de discriminer les états à l'oeuil nu...

Affichons quelques informations sur le jeu de données. On a environ 80% des relevés correspondant à l'étiquette `STATE0` et 20% à l'étiquette `STATE1`. Quelques neurones sont sur-représentés.





L'étape de prétraitement fondammentale est la différanciation des séries de timestamps pour chaque individus. En effet, cela permet de mettre en exergue les variations de fréquances dans l'apparition des pics. C'est sur cette base que nous avons appliqué toutes sortes de méthodes. Nous représentons ci-dessous quelques séries différenciées en fonction de la variable `TARGET`.



Nous avons par la suite calculé des statistiques sur ces séries pour chaque individu.

2 Feature engineering

Dans le but d'obtenir un premier modèle, nous construisons des variables descriptives sur les incréments du jeu de données initial. Dans un premier temps, les variables calculées ont été les suivantes: 'min', 'max', 'sum', 'mean', 'median', 'std', 'Q10', 'Q25', 'Q75','Q90', 'skew', 'kurtosis', 'mad', 'entropy'.

Puis nous avons par la suite calculé pour chaque `neuron_id` la mediane de ces statistiques. Ces variables permettent d'obtenir le comportement median des nerones. Elles sont importantes si on fait l'hypothèse que les neurones n'ont pas tous le même comportement. C'est à dire que l'on peut supposer par exemple que les neurones n'ont pas tous les mêmes seuils d'activation. Ainsi, pour

deux neurones ayant des graphiques avec les mêmes intervalles inter-pics (IIP), il se peut qu'ils ne soient pas dans le même état. On ajoute également pour chaque individus le nombre d'occurrences pour le champ `neuron_id`.

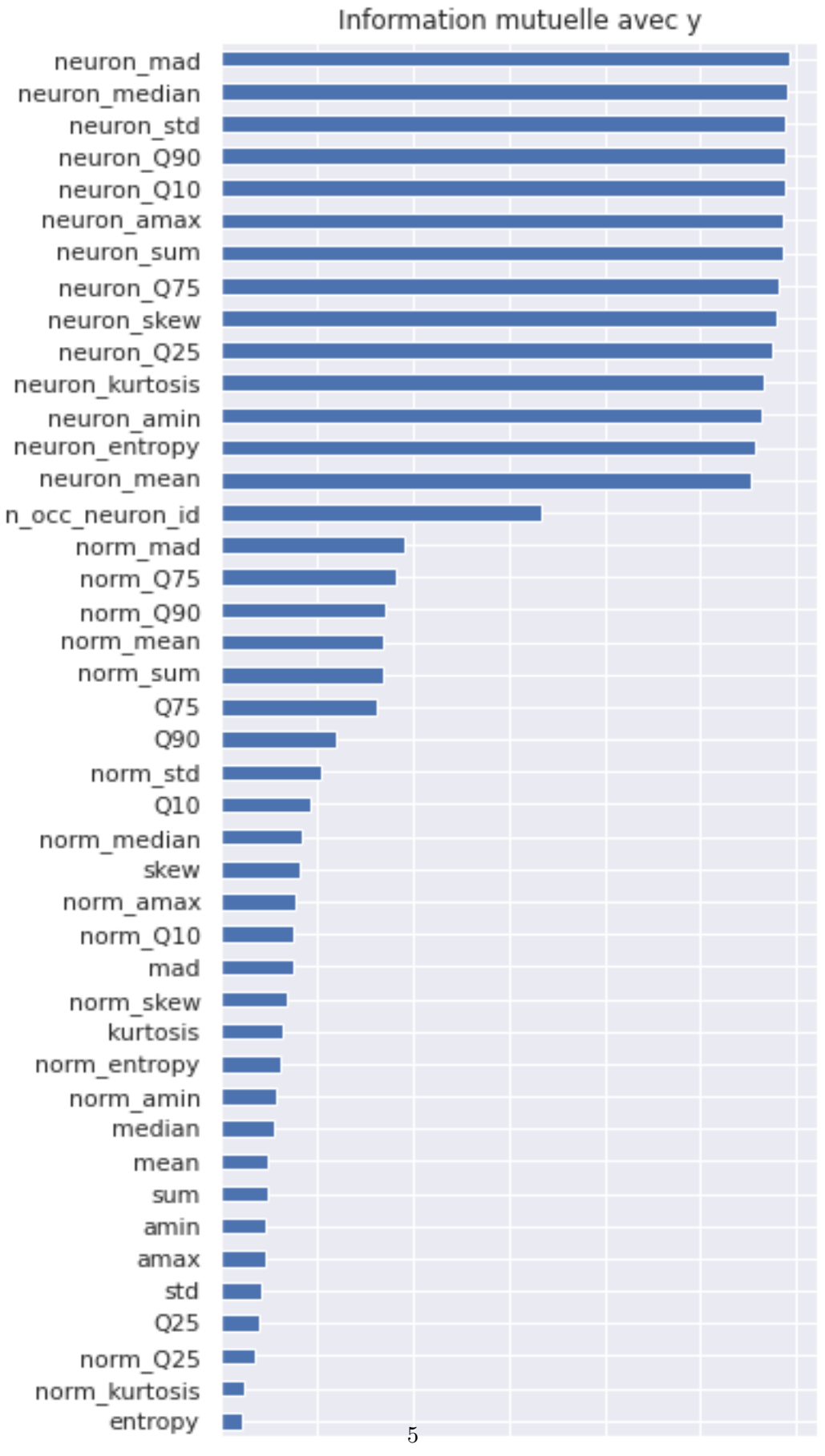
Pour finir de constituer le jeu de données, on ajoute le rapport entre les 14 variables initiales et leur mediane. De plus, on retire l'information `neuron_id` car les identifiants des neurones du jeu d'entrainement sont différents des neurones du jeu de test.

On obtient alors au total un train set de 16635 individus et 43 variables descriptives.

> Pistes non concluantes / à creuser: Dans un premier temps, nous avions tenté de générer ces caractéristique à l'aide de modules tels que `tsfresh`ou encore `tsfel`. Cependant, les data sets augmantés n'ont pas obtenus de meilleurs résultats que le data set de référence sur les jeux de validation. Peut-être que l'ajout d'uniquement de quelques variables spectrales et de variables liées aux coefficients d'autocorrélations (voir `tsfel` feature domain) donneraient de meilleurs résultats.

3 Feature selection

Afin d'améliorer les performances du modèle, nous avons tenté de retirer les variables les moins pertinantes. Pour représenter leur importance nous avons utilisé deux méthodes. La prémière est d'entrainer un modèle basé sur la construction d'arbre de décision et d'observer la contribution de chaque variable via le paramètre `feature_importances_`. La seconde méthode est de calculer l'information mutuelle entre les X et y. Nous obtenons alors le résultat ci-desous.


Information mutuelle avec y
neuron_mad
neuron_median
neuron_std
neuron_Q90
neuron_Q10
neuron_amax
neuron_sum
neuron_Q75
neuron_skew
neuron_Q25
neuron_kurtosis
neuron_amin
neuron_entropy
neuron_mean
n_occ_neuron_id
norm_mad
norm_Q75
norm_Q90
norm_mean
norm_sum
Q75
Q90
norm_std
Q10
norm_median
skew
norm_amax
norm_Q10
mad
norm_skew
kurtosis
norm_entropy
norm_amin
median
mean
sum
amin
amax
std
Q25
norm_Q25
norm_kurtosis
entropy
0.00
0.02
0.04
0.06
0.08
0.10
0.12

Les variables partageant le plus d'information avec `y` sont les variables d'aggrégation sur les statistiques des neurones, la variable `n_occ_neuron_id`, puis les ratio puis les variables simples.

Le meilleur score sur le test set a été obtenu en gardant l'ensemble des variables.

4 Model sélection

Trois types d'algorithmes ont été testés: RandomForestClassifier, XGBoostClassifier, LGBMClassifier. Nous avons opter pour le dernier car rapide à "fitter", rapide à optimiser et donnant d'aussi bons résultats comparativement aux optimisations faites avec le XGBoostClassifier.

Au sujet de l'optimisation, lors de la recherche des meilleurs hyper-paramètres nous avons utilisé la méthode `GroupedKFold` de `sklearn` afin d'avoir une indépendance entre le jeu d'entrainement et le jeu de validation. En effet, si on entraine un modèle sur des données contenant les mêmes `neuron_id` que les données de validation alors ces deux jeux ne sont pas indépendants. Dans ce cas, nous obtenons un score (très) optimiste sur le jeu de validation. Score qui s'éffrondre sur le jeu de test pour cause d'overfitting.

Suite aux nombreux tests, le modèle ayant obtenus le meilleurs score est le suivants: `LGBMClassifier(max_depth=6, num_leaves=9, class_weight="balanced")`.

5 Conclusion

Notre modèle a obtenu la première place du classement public du Challenge Data de l'ENS à la date du 20 janvier 2022 sous le pseudo *amtmr* et avec un score de 0.5074.

Parmis les axes d'amélioration nous pouvons citer l'ajout de nouvelles variables descriptives sur les incréments. De plus, l'approche via classification de séries temporelles (avec des algorithmes classiques tel que les plus proches voisins ou des réseaux de neuronnes LSTM ou bien des algorithmes moins connus tel que SAX-VSM) n'ont pas donné de bons résultats. Néanmoins, il est probable que la raison de cet échec soit du à la faible maitrise de ces algorithmes par l'auteur.