Rapport d'analyse des données

Projet StarCraft 2 Play Prediction Challenge

Objectif :

Ce projet a pour objectif l'entrainement d'une intelligence artificielle pour reconnaitre un joueur à partir de ses touches claviers lors d'une partie.

Dataset :

Pour ce faire, nous disposons de plusieurs dataset. Nous avons deux fichiers 'TRAIN' et deux fichiers 'TEST'. Chaque catégorie a un dataset avec les datas simplifé, qui contient la touche appuyée à l'instant t, et une dimension temporelle qui indique la durée du jeu en seconde. Le deuxième dataset contient plus d'informations. IL contient toutes les informations présentes dans le premier dataset, mais aussi des éléments extérieurs au joueur comme des évènements du jeu. Pour notre code, j'ai choisi le dataset simplifié, contenant que les informations propres au joueur.

Prétraitement des données :

Le but du prétraitement de mes données est d'obtenir un jeu de donnée encodé représentant uniquement les touches saisis par le joueur dans l'ordre chronologique.

Lors de l'import des données, la fonction read_csv de pandas ne marche pas car le fichier csv a quelques erreurs. Je me sers donc d'une fonction personnalisée qui vérifie bien que chaque ligne commence par 'http' (l'identifiant du joueur) pour m'assurer que la ligne est valide et n'a pas d'erreur, puis elle divise bien chaque valeurs avec le séparateur « , ».

Une fois les données importées, je supprime les valeurs qui indique la temporalité, et je décale les autres valeurs, pour ne pas avoir de « trous » avec des valeurs None au milieu des touches. Une fois cette étape faite, mon jeu de donnée comporte bien les données que je souhaite envoyer à mon modèle.

Enlever les variables commençant par 't' me permettent de réduire la dimension du dataframe, j'enlève donc les dernières colonnes qui ne contiennent que des valeurs nulles.

Une fois le dataframe propre, je crée un dictionnaire à partir de toutes les valeurs comprises dans mon dataframe, pour pouvoir ensuite encoder mon jeu de donnée avec les entiers correspondant. Certaines colonnes sont mal encodées, il y a donc une fonction qui reprend ces colonnes et recommencent l'encodage.

Une fois encodé, il faut équilibrer les classes car nous avons un problème avec des classes déséquilibrés. Certaines classes ont 58 échantillons et d'autres n'en ont que 4. Nous utilisons donc un SMOTE pour augmenter à 50 échantillons toutes les classes et nous réduisons également les classes avec plus de 50 échantillons.

Nous diminuions ensuite le nombre de colonne pour réduire le temps d'entrainement de notre modèle et nous séparons notre jeu de donnée en valeurs d'entrainement et de test.

Choix et entrainement du modèle :

Pour mon modèle, j'ai fait le choix d'utiliser un LSTM. Ce modèle va permettre l'analyse de mes données en prenant en compte leur dimension temporelle (ordre chronologique) pour trouver des patterns de touches propres à chaque joueur.

Le modèle a beaucoup évolué dans le temps. J'ai d'abord commencé avec un modèle simple, avec une seule couche de neurone, pas de dropout, et pas de normalisation. Une fois le modèle plus maitrisé et comprit, j'ai ajouté des neurones à chaque couche, j'ai ajouté du dropout, de la normalisation, et j'ai augmenté le nombre d'epoch.

Evaluation de l'entrainement :

Pour évaluer le modèle, j'utilise la métrique « f1 score ». Plus j'augmente le nombre de neurone, le nombre de dimension, et le nombre d'epoch, plus le modèle est performant, mais le temps d'entrainement augmente également de manière considérable.

Optimisation du modèle :

Les environnements de googlecolab ne permettent pas l'entrainement d'un tel modèle, ils se déconnectent s'il n'y a plus d'activité. Anaconda peut permettre un tel entrainement, mais j'ai choisi de ne pas laisser tourner un modèle pendant plus de 10h, dans un souci d'éthique de consommation d'énergie. J'ai donc entrainé de plus petit modèle, avec 512 et 256 neurones sur la première et deuxième couche, et avec une dimension pour le dataframe d'entrées de 10 000 lignes et 100 colonnes donc 100 touches.

Conclusion et perspectives :

Notre modèle a plusieurs axes d'améliorations, comme expliqué précédemment, nous pouvons ajouter des couches LSTM et des neurones pour trouver des patterns plus complexes, ajouter de la dimension à nos données d'entrée, augmenter le nombre d'epoch, changer le mode de compilation, ajouter du dropout a chaque couche pour augmenter la flexibilité du modèle, mais dans le cadre de mon projet, j'ai choisi de m'arrêter à ce stade. D'autres moyens complètement différents peuvent être exploité, comme l'occurrence de chaque touche par joueur, ou l'utilisation du dataset avec plus d'informations. Cependant mon modèle m'a permit d'obtenir un score de 0.76 (f1 score) avec le fichier test. Ce score peut encore être améliorer, mais il est déjà relativement bien.