INF8215 – Intelligence artificielle: méthodes et algorithmes

Automne 2021

TP3 - Vinho Verde

Groupe 05 - Équipe *les dindons* sur Kaggle

1960266 – Yanis Toubal

1947497 – Yuhan Li

Soumis à : Quentin Cappart

Dimanche 10 décembre 2021

Pour ce travail, nous avons expérimenté avec deux approches assez différentes. La première est basée sur les arbres de décision (decision tree) alors que la deuxième est basée sur les réseaux de neurones (Neural network). Dans les deux cas, nous avons utilisé la librairie *scikit-learn* qui nous a été facile à prendre en main. Le modèle basé sur les arbres de décision utilise le *RandomForestClassifier* alors que le modèle basé sur les réseaux de neurones utilise le *MLPClassifier* de cette librairie.

1. Prétraitement des attributs (Feature design)

Tout d'abord, avant d'entamer officiellement l'étape de prétraitement, nous nous sommes familiarisés avec le *dataset* de vin. Nous avons pu nous apercevoir qu'il y a 14 colonnes, soit 13 attributs et 1 variable représentant ce qu'on veut prédire (qualité). En tout, il y avait non-loin de 5000 entrées, ce qui représente un assez petit *dataset*. Une caractéristique qui nous a frappé est que la colonne pour l'attribut *couleur* contenait des valeurs sous forme de chaîne de caractères, plutôt qu'en chiffre. Cela nous a causé des erreurs lorsqu'on essayait d'appliquer des calculs statistiques sur les données. Pour remédier à ce problème, nous avons converti les chaînes de caractères en type *categorical* ce qui a permis de convertir les *strings* en chiffres. Cela suit la même logique qu'un type *enum*.

Pour ce qui est des étapes effectuées lors du prétraitement, nous avons d'abord vérifié s'il y a avait des valeurs manquantes. C'est une chose qui n'est pas rare lors des traitements de données, en particulier dans un large *dataset.* Heureusement pour nous, ce dataset n'en avait pas et donc nous n'avons eu rien de plus à faire sur ce point. Ensuite, nous avons observé la corrélation entre les variables sous forme de matrice, ce qui nous a permis de noter quelques faits intéressants. Par exemple, l'alcool (colonne 12) semble de loin être l'attribut le plus corrélé à la qualité, donc on suppose fortement qu'il sera l'attribut le plus important pour estimer la qualité. Aussi, nous avons observé que l'attribut *total sulfur dioxide* (colonne 8) est grandement corrélé avec les attributs *color* (colonne 1) et *free sulfur dioxide* (colonne 7). C'est pourquoi, il s'agit là d'un attribut que nous avons considéré possible à enlever. Toutefois, en enlevant cet attribut, la précision globale du modèle à baisser. Nous avons donc gardé tous les attributs.

Pour le modèle basé sur les arbres de décision, nous n'avons pas fait de prétraitement sur les données. Lors de nos tests, nous avons tenté d'appliquer différentes transformations sur les données, comme par exemple enlever l'attribut *total sulfur dioxide* (colonne 8) ou encore normaliser les données. Ces transformations n'ont pas amélioré les résultats. Les raisons qui peuvent expliquer ce phénomène seront abordées dans la section 2 (Méthodologie).

Pour le modèle basé sur les réseaux de neurones, nous avons appliqué une normalisation des données à l'aide du *MinMaxScaler.* Les données sont donc converties en valeurs dans l'intervalle de 0 et 1 à l'aide du minimum et du maximum de chaque colonne. La normalisation sert principalement à ramener les données sur une même échelle pour pouvoir les traiter uniformément. [1] Dans notre cas, la différence entre la précision sans normalisation et avec normalisation est de ≅25%, ce qui est assez significatif.

2. Méthodologie

Pour le modèle basé sur les arbres de décision nous avons opté pour le *RandomForestClassifier*. Ce type de classificateur utilise plusieurs arbres de décision, dont le nombre peut être spécifié en hyper paramètre. Le classificateur va attribuer aléatoirement une partie des données à chaque arbre puis, lorsque vient le temps de prédire une valeur, va prendre la moyenne des résultats de chaque arbre pour donner une prédiction. [2] Pour ce qui est de la séparation des données d'entraînement et de validation, nous avons décidé de ne pas avoir d'ensemble de validation et d'utiliser 100% des données pour l'entraînement. Ce choix s'explique par le fait que nous avons utilisé deux méthodes de validation, soit le

cross-validation (cv) ainsi que le *out-of-bag error* (OOB). En testant, nous nous sommes aperçus qu'on obtient de meilleurs résultats plus on a des données d'entraînement. La technique de cross-validation consiste à diviser les données en k parties. Ensuite, k-1 parties seront utilisées comme ensemble d'entraînement et la partie restante sera l'ensemble de validation.[3] Au final, cela nous donnea k scores de précision et en faisant leur moyenne on peut avoir une valeur proche de la précision de notre modèle. Pour ce qui est de la technique OOB, celle-ci consiste à utiliser les données qui n'ont pas été utilisées pour l'entraînement comme données de validation pour chaque arbre de décision. Puis, elle se veut faire la moyenne des résultats de chaque arbre de décision. [4] Pour ce qui est du déséquilibre des classes, nous en avons remarqué un important notamment entre les classes 6 et 5 qui sont très nombreuses, ainsi que les classes 9 et 3 qui sont très peu nombreuses. Après un peu de recherche sur le sujet, nous avons pu trouver que le *oversampling*, c'est-à-dire ajouter des données des classes peu représentées, ou le *undersampling*, c'est-à-dire enlever des données des classes très représentées, sont de bonnes stratégies pour ce type de problème. Toutefois, nous n'avons pas eu beaucoup de succès en les implémentant, ce qui fait en sorte que nous n'avons pas géré le déséquilibre des classes. Dans le cas du RandomForestClassifier, nous n'avons pas eu besoin d'employer une stratégie de régularisation, car nous n'avons pas remarqué de surapprentissage. Aussi, ce type de classificateur a moins tendance à surprendre avec un nombre d'arbre de décision assez grand. Pour les hyperparamètres, nous avons utilisé le GridSearchCV pour trouver la combinaison qui donnait les meilleurs résultats et au final nous avons seulement gardé l'hyperparamètre n_estimators à une valeur de 1000.

Pour ce qui est du modèle basé sur le réseau de neurones, nous avons choisis une couche contenant 11 neurones (la raison pour ce nombre sera expliquée plus tard). Pour ce qui est de la séparation des données d'entraînement et de validation, nous avons suivi le standard du 80/20, c'est-à-dire que 80% des données ont été utilisées pour l'entraînement et 20% ont été utilisées pour la validation. Tout comme dans le classificateur précédent, les tentatives pour gérer le déséquilibre des classes avec l'oversampling et l'undersampling ont échoué et donc rien n'a été fait sur ce point. Aussi, nous n'avons pas observé de surapprentissage lors de nos tests et donc nous n'avons pas utilisé de stratégie de régularisation. Toutefois, il aurait été une bonne pratique d'inclure une telle stratégie pour un réseau de neurones. Pour les hyperparamètres, nous avons également utilisé le GridSearchCV. Par contre, comparativement aux RandomForestClassifier, il y avait beaucoup plus d'hyperparamètres d'importance à considérer. Donc, au final, la meilleure configuration a été 11 neurones dans 1 couche cachée, 50 itérations sur l'ensemble d'entraînement, une fonction d'activation de type ReLU, un solveur de type SGB et un learning rate de 0,3. Cela a requis un temps très important pour essayer les différentes combinaisons possibles et donc il aurait été surement préférable d'utiliser un autre type de CV plutôt que GridSearch qui fait une recherche exhaustive.

3. Résultats

Tableau 1: Résultats de la cross-validation avec k = 10 pour le RandomForestClassifier

Accuracy de la qualité									
0.6396	0.6681	0.6571	0.6857	0.6571	0.6462	0.6418	0.6806	0.6850	0.6542

Moyenne: 0.6615
Le score OOB pour le RandomForestClassifier est de 0.6769.
Le score dans l'ensemble test (Kaggle) est de 0.6800.

On peut observer que, malgré l'absence d'un ensemble de validation pour le modèle, les deux techniques de validation donnent des résultats très proches de l'ensemble test. Pour ce qui est de la

méthode de cross-validation, la raison pour laquelle la moyenne est sous-estimée est que l'ensemble d'entraînement est plus petit (et, du même coup, l'ensemble de validation est plus grand) que celui du vrai modèle et il est normal que la valeur de précision donnée soit un peu plus petite. Pour ce qui est de la méthode de OOB, ce score était très proche de la précision du modèle sur l'ensemble de test (Kaggle) à +/- 0.5% près.

Tableau 2: Résultat de la cross-validation avec k = 10 pour le MLPClassifier

Accuracy de la qualité									
0.5467	0.5385	0.5220	0.5357	0.5412	0.5302	0.5330	0.5124	0.5372	0.5317

Moyenne: 0.5329
Le score sur l'ensemble de validation (20%) est de 0.58.
Le score dans l'ensemble test (Kaggle) est de 0.4964.

On peut observer un score nettement plus bas que le classificateur précédent. On peut également observer une différence importante entre les résultats des validations et le résultat sur l'ensemble de test (Kaggle). Cela peut être dû à de nombreuses raisons, mais on suspecte fortement qu'il y a du surapprentissage.

4. Discussion

Pour ce qui est de l'approche utilisant le RandomForestClassifier, nous avons trouvé très simple et rapide de créer un modèle fonctionnel capable de faire de très bonnes prédictions (autour de 60%) avec un minimum de configuration. Aussi, à travers nos différents tests, nous avons pu nous rendre compte que cette méthode ne nécessitait pas beaucoup de pré-traitement des données. Ensuite, avec un peu d'optimisation du modèle nous sommes arrivés à un très bon score de 68% avec l'ensemble test (Kaggle). Par contre, nous suspectons qu'on aurait pu améliorer encore davantage ce score notamment en poussant plus loin la recherche sur le pré-traitement des données et sur le déséquilibre des classes. Malgré que nous pensons que le score peut être amélioré, nous ne pensons pas qu'il puisse être augmenté considérablement. En effet, nous avons observés qu'à partir d'un certain seuil (qui était de 66% pour nous), il fallait beaucoup d'effort simplement pour augmenter la précision de 0.5-1% et donc nous suspectons que la limite théorique de ce modèle (avec les mêmes données) n'est pas très loin de ce qu'on a trouvé. Un dernier point que nous avons noté est que les performances du modèle sont grandement affectées selon le nombre d'arbres de décision utilisé pour donner une prédiction et donc à partir de quelques milliers d'arbres le modèle peut devenir assez long à entraîner.

Pour ce qui est de l'approche utilisant le réseau de neurones, nous avons trouvé qu'il y avait beaucoup de configuration possible. Cela nous pousse à croire que ce type de modèle est très malléable et peut être entraîné pour avoir de très bons résultats. Toutefois, nous avons pu voir que ce genre de modèle est très sensible à la qualité des données et donc il faut passer un temps important à transformer les données afin d'augmenter la précision du modèle. Plus encore, il y a de nombreux hyperparamètres d'importance et donc il est assez long de trouver une combinaison optimale. Aussi, ce type de modèle nécessite généralement beaucoup de données afin de bien fonctionner. Dans notre cas, la taille du dataset était assez petite ce qui n'a pas été très bénéfique pour le réseau de neurones. Au final, nous avons pu obtenir une précision proche de 50% ce qui n'est pas si mal, mais cela reste assez faible si on compare le 68% du modèle basé sur les arbres de décisions.

Références

1. Jaitley, U. (7 octobre 2018). *Why Data Normalization is necessary for Machine Learning models,* Medium | Urvashi Jaitley. https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029

2. Jaitley, UPedregosa et al. (2011). Scikit-learn: *Machine Learning in Python | sklearn.ensemble.RandomForestClassifier,* Scikit Learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

3. Jaitley, UPedregosa et al. (2011). *Machine Learning in Python | 3.1. Cross-validation: evaluating estimator performance*, Scikit Learn. https://scikit-learn.org/stable/modules/cross_validation.html

4. Bhatia, N. (26 juin 2019). *What is Out of Bag (OOB) score in Random Forest?*. Towards Data Science.https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710