Premier Devoir : Apprentissage supervisé.

Lisez attentivement tout ce qui suit !

Pourquoi?

Le but de ce devoir est d'explorer certaines techniques d'apprentissage supervisé. Il est important de réaliser que la compréhension d'un algorithme ou d'une technique nécessite de comprendre comment il se comporte dans une variété de circonstances. A ce titre, il vous sera demandé de mettre en œuvre quelques algorithmes d'apprentissage simples et de comparer leurs performances.

Les moyens

Dans cette mission, vous passerez par :

1. Le processus d'exploration des jeux de donnée que vous avez choisis (**EDA**)
2. L'optimisation de l'algorithme que vous avez appris. (**Tuning**)
3. La rédaction d'une analyse approfondie de vos résultats. Tout ce qui compte dans ce devoir c'est l**'analyse**. Peu importe si vous implémentez vous-même des algorithmes d'apprentissage (**Analyse**).

Vous pouvez programmer dans n'importe quel langage que vous le souhaitez, et vous êtes autorisé à utiliser n'importe quelle bibliothèque que vous voulez à partir du moment où je peux être en mesure de **reproduire vos expériences**. La bibliothèque principale est sklearn en python (ou Weka en Java, cependant en entreprise python est largement le plus répandu pour les tâches de machine learning).

Les problèmes qui vous sont confiés

Vous devez implémenter **au moins** cinq algorithmes d'apprentissage. C'est-à-dire :

• Des arbres de décision (avec élagage, ou pruning en anglais)

• Les réseaux de neurones (artificial neural network)

• Boosting

• SVM (machines à vecteurs de support)

• K-voisins les plus proches (k-nearest neighbor)

Chaque algorithme est décrit en détail dans les **vidéos de cours** ainsi que **le livre de référence** (Machine Learning, Tom Mitchell) ainsi que dans les **documents que je partage en ressources** (blog, article..etc). En réalité, au lieu d'implémenter les algorithmes vous-même, vous pouvez (et je veux dire devriez) utiliser des librairies; cependant, si vous le faites, vous devez fournir un travail **particulièrement soigné**. Aussi, vous remarquerez que vous devez faire quelques manipulations pour obtenir de bons résultats, des graphiques et autres, donc même si vous utiliser le package d'un autre, vous devrez peut-être pouvoir le modifier de diverses manières.

Arbres de décision. Pour l'arbre de décision, vous devez implémenter ou voler un algorithme d'arbre de décision (et par voler je veux dire utiliser des librairies). Assurez-vous d'utiliser des arbres de décision avec **élagage** (par exemple random forest/forêt aléatoire). Vous n'êtes pas obligé d'utiliser le gain d'informations (par exemple, il existe quelque chose appelé l'indice de GINI qui est parfois utilisé pour ça) pour diviser les attributs/features, mais vous devez décrire **tout** ce que vous utilisez.

Les réseaux de neurones. Pour le réseau de neurones, vous devez implémenter ou voler votre type de réseau préféré ainsi que son algorithme d'entraînement. Vous pouvez utiliser des réseaux de nœuds avec autant de couches que vous le souhaitez et la fonction d'activation que vous jugez appropriée.

Boosting. Implémentez ou volez une version boostée de vos arbres de décision. Comme précédemment, vous devrez utiliser des arbres avec élagage, mais désormais parce que vous utilisez le boost, vous pouvez vous permettre d'être beaucoup plus agressif dans votre élagage.

SVM. Vous devez mettre en œuvre un SVM. Cela devrait être fait de telle manière que vous puissiez changer de fonctions noyaux. J'aimerais en voir au moins **deux.**

KNN, k-Les voisins les plus proches. Vous devez mettre en œuvre un KNN. Utilisez **différentes** valeurs de k.

Testing

En plus d'implémenter les algorithmes décrits ci-dessus, vous **devez concevoir deux problèmes de classification intéressants**. Vous pouvez obtenir les données où vous le voulez mais il vous est grandement conseillé de regarder les jeux de données sur Kaggle et UCI. Cependant lorsque vous choisissez vos jeux de données vous devrez expliquer pourquoi ils sont **intéressants**. Ces jeux de données sont conservés pour les autres devoirs.

DELIVRABLES

Vous devez soumettre :

1. un fichier nommé README.txt contenant des instructions pour exécuter votre code.

2. un fichier nommé *nom.prenom.TP1.analyse.pdf* contenant votre rapport.

Remarque ci-dessous : je dois pouvoir accéder à votre code et à vos données. Donc veuillez me zipper tout votre travail (données, code, output, rapport, README..etc)

Votre fichier *nom.prenom.TP1.analyse.pdf* doit contenir :

1. une **description** de vos problèmes de classification, et pourquoi vous pensez qu'ils sont intéressants. Réfléchissez bien à propos de ça. Pour être intéressant, les problèmes doivent être **non triviaux d'une part**, mais capables d'**admettre les comparaisons et l'analyse des différents algorithmes d'autre part**. Par exemple si je vois que vos algorithmes ont pour la majorité d'entre eux le même comportement sur votre jeu de donnée c'est que votre travail de classification et/ou que votre jeu de donnée n'est pas intéressant.

2. les taux d'erreur d'entraînement et de test que vous avez obtenus en exécutant les différents algorithmes d'apprentissage sur votre problème. Au minimum, vous devriez **inclure des graphiques** qui montrent les performances à la fois de l'entraînement et données de test en fonction de la taille de l'entraînement (notez que cela implique que vous devez concevoir un problème de classification qui a plus qu'une quantité insignifiante de données) et - pour les algorithmes qui sont itératifs - temps d'entraînement/itérations. Ces deux types de graphiques sont appelés courbes d'apprentissage (learning curve en anglais).

3. **L'analyse** de vos résultats. Pourquoi avez-vous obtenu les résultats que vous avez obtenus ? Comparer et contraster les différents algorithmes. Quel type de changements pourriez-vous apporter à chacun de ces algorithmes pour améliorer leur performance? Quel runtime ? Combien d'Itérations ? **La validation croisée** pourrait-t-elle être utile ? (et si c'était le cas, pourquoi ne l'avez-vous pas mis en œuvre ?) ? Quelle performance était due aux problèmes de classification(la façon dont vous avez conçu votre problème de classification) que vous avez choisis? Quelles métriques d'évaluation avez-vous choisis ? En quoi répondent-t-elles à votre problème de classification ? Qu'en est-il des valeurs que vous choisissez pour les taux d'apprentissage, les critères d'arrêt, méthodes d'élagage, et ainsi de suite..etc (et pourquoi votre analyse ne montre-t-elle pas de résultats pour les différentes valeurs vous avez choisis ? S'il vous plaît, **regardez-en plus d'un**. Et s'il vous plaît assurez-vous de le **comprendre**, cela ne rentre en compte que si les résultats sont significatifs) ? Quel algorithme a le mieux fonctionné ? Que veut dire « mieux » dans votre problème de classification ?

Soyez créatif et réfléchissez à autant de questions que possible et à autant de réponses que possible.

Critère d'Evaluation

Vous serez évaluez principalement sur la **qualité** de votre analyse. Je veux savoir ce que vous avez appris des différentes ressources que j'ai centralisées pour vous et comment vous vous en êtes servies pour concevoir votre problème de classification et l'appliquer à votre jeu de données. Dans la vraie vie c'est comme ça qu'on travaille, on n'essaie pas de faire rentrer les données dans un corpus théorique avec théorèmes et démonstrations mais en raisonnant de façon empirique à l'aide d'arguments empiriques**. Le nombre de page MAXIMUM** de votre rapport doit être au nombre de 14 (hors annexes).

Le Standard 2021

Bien évidemment on ne fait pas du machine learning en 2021 comme on le faisait il y a quatre ans. Aussi je tiens particulièrement à vous signaler que certains éléments de votre analyse seront pris en compte **en plus de ce que j'ai cité plus haut**. Il faudra donc être attentif sur le discord à ce sujet. En 2021 il ne s'agit plus d'avoir des bons modèles avec de bonnes métriques à optimiser mais également de bien calibrer les modèles (**vos classifications sont-t-elles justes ?**) voir quelles features (notamment pour les algorithmes black box, ou boite noire en français, comme les modèles boosting ou réseau de neurone) ont été vraiment **utilisés** dans l'apprentissage **(comment vos classificateurs ont appris ?)..etc** Je vous présenterai les librairies et algorithmes qui vous permettent d'expliquer mieux vos modèles et algorithmes.

BONUS :

Pour ceux qui auront terminé en avance, le travail bonus (qui n'est absolument pas indispensable) du premier devoir est l'implémentation de l'algorithme **XGBoost** à votre problème de classification. C'est un algorithme particulièrement puissant et utilisé dans les compétitions kaggle avec succès. Quelle est sa particularité ? Pourquoi est-t-il si utilisé en compétition ? Comment se compare-t-il avec les modèles de boosting concurrents ? Enfin appliquez-le sur vos données ? Comment se compare-t-il avec votre boosting ? Quels hyper-paramètres ont-t-ils été les plus importants dans le tuning ? A vous de jouer.