TP3 (comptant pour 8%)

IFT-4102 et IFT-7025 : Techniques avancées en Intelligence Artificielle

À rendre pour le 13 avril 2022 à 23h59

Introduction

Dans ce travail pratique, vous devez implémenter deux techniques d'apprentissage machine : les K plus proches voisins et la classification naïve bayésienne. La meilleure façon d'apprendre c'est de programmer sans avoir recours aux bibliothèques déjà prêtes à faire tout le travail d'apprentissage (telle que `scikit-learn`). Vous allez donc programmer de A à Z ces deux méthodes, et par la suite vous pourrez valider votre implémentation en comparant les résultats obtenus avec ceux obtenus en utilisant les méthodes de `scikit-learn`. Il est donc interdit d'utiliser toute bibliothèque ou tout programme déjà fait pour l'implémentation. Si jamais vous voulez utiliser une partie déjà faite il vous faut demander l'autorisation à Mathieu Pagé-Fortin. Afin d'évaluer l'efficacité de ces techniques, vous devrez entraîner et tester votre code sur les ensembles de données réels fournis.

Pour toute aide veuillez vous adresser à Mathieu Pagé-Fortin.

1 Ensembles de données (Datasets)

Les datasets fournis ici proviennent tous du site UCI [1] (un dépôt de datasets d'apprentissage machine). Sous la catégorie classification, nous avons sélectionné trois datasets qui, selon nous, donneront une diversité de résultats avec les méthodes d'apprentissage automatique sélectionnées.

Note : tous ces datasets sont fournis dans le dossier `Code/datasets` en attaché.

1.1 Classification des Iris (Iris Dataset)

Ce dataset [2] classe les fleurs en trois différentes classes d'iris en fonction de la taille des différentes caractéristiques (longueur du sépale, largeur du sépale, longueur du pétale et largeur du pétale). Il s'agit d'un problème de classification simple et classique. C'est un mélange de classes linéairement séparables et inséparables.

Le fichier contenant le dataset s'appelle `bezdekIris.data` et se trouve dans ce répertoire [3].

1. `http://archive.ics.uci.edu/ml/datasets.html`
2. `https://archive.ics.uci.edu/ml/datasets/Iris`
3. `http://archive.ics.uci.edu/ml/machine-learning-databases/iris/`

1.2 Prédiction de l'appréciation du vin blanc portugais *Vinho Verde*

Afin d'étudier l'association entre la composition chimique du vin et son appréciation par les consommateurs, Cortez et al. [2009] ont recueilli deux ensembles de données sur le *vinho verde* rouge et blanc, un vin produit dans le nord du Portugal. Nous nous intéresserons au dataset du vin blanc. Pour ce TP, nous avons réorganisé le jeu de données en deux classes : *bon* et *mauvais*. Il s'agira donc de prédire si un vin est plutôt bon ou mauvais étant donné sa composition chimique (voir l'annexe 1 pour plus de détails sur les caractéristiques).

Le fichier contenant le dataset est `binary-winequality-white.csv` et se trouve dans le dossier `Code/datasets`. Pour plus de détails sur le dataset original, vous pouvez consulter ce répertoire [4].

1.3 Prédiction de l'âge des ormeaux (*abalones*)

L'âge des ormeaux (*abalones*) est généralement déterminé en prélevant un échantillon de leur coquille et en comptant au microscope le nombre d'anneaux qui s'y trouvent, un procédé assez fastidieux. On voudrait plutôt être capable de déterminer leur âge à partir de caractéristiques physiques plus facilement mesurables. Le dataset original regroupe donc 8 caractéristiques physiques de 4177 ormeaux, ainsi que leur nombre d'anneaux (l'âge en années se calcule simplement en ajoutant 1.5 au nombre d'anneaux). Pour ce TP, nous avons divisé les données en 3 classes qui correspondent à des intervalles du nombre d'anneaux (voir l'annexe 1 pour plus de détails).

Le fichier contenant le dataset est `abalone-intervalles.csv` et se trouve dans le dossier `Code/datasets` en attaché. Pour plus de détails sur le dataset original, vous pouvez consulter ce répertoire [5].

2 Techniques d'apprentissage automatique à développer

Cette section décrit les techniques d'apprentissage automatique que vous aurez à implémenter par vous même pour mettre en œuvre et tester les datasets fournis.
En apprentissage automatique, plusieurs termes sont utilisés pour évaluer les performances des différentes techniques. Nous trouvons par exemple l'*Accuracy* (exactitude) qui est tout simplement le pourcentage des exemples bien classifiés par rapport au nombre total des exemples présentés à l'algorithme.

1. Faites une recherche et définissez **dans vos mots** les termes suivants :
   - La précision (Precision)
   - Le rappel (Recall)
   - Le F1-score
   - La matrice de confusion (Confusion matrix). Dans votre définition, expliquez comment l'exactitude, la précision, le rappel et le F1-score peuvent tous être extraits à partir de la matrice de confusion.

Remarques :

4. `https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/`
5. `http://archive.ics.uci.edu/ml/datasets/Abalone`

— Vous pouvez voir sur Wikipedia. C'est à vous de faire un petit résumé et d'inclure dans votre rapport une définition de chaque terme.
— En général, les termes précision, rappel et F1-score sont définis pour les problèmes de classification binaire. Pour les datasets utilisés dans ce TP, la classification n'est pas toujours binaire. Dans ce cas il convient de calculer la précision, le rappel et le F1-score **pour chaque classe** (une approche de un-contre-tous).

2.1 K plus proches voisins (K-nearest neighbors)

Pour cette méthode vous devez implémenter la technique des K *plus proches voisins* vue en cours, introduite au *chapitre 18.8.1* et également expliquée sur Wikipédia [6].

1. Mentionnez la métrique de distance que vous avez choisie, et justifiez votre choix.
2. Donnez le pseudo-code de l'algorithme.
3. Évaluez cette technique sur l'ensemble de test en donnant pour chaque classe :
   — L'exactitude (Accuracy)
   — La précision (Precision)
   — Le rappel (Recall)
   — Le F1-score
   — La matrice de confusion (Confusion matrix)
4. Pour le choix de K :
   — **IFT-4102 seulement :** utilisez $K = 5$ (vous avez aussi la possibilité de tester avec d'autres valeurs de K).
   — **IFT-7025 seulement :** utilisez la validation croisée (cross-validation) pour déterminer la *meilleure valeur* de K. Expliquez votre démarche pour la validation croisée et détaillez, dans la mesure du possible, votre démarche.
   Pour la validation croisée et pour chacun des K variant entre $K_{Min}, \ldots, K_{Max}$, on divise l'échantillon original en L échantillons, puis on sélectionne un des L échantillons comme ensemble de validation et les $(L-1)$ autres échantillons constitueront l'ensemble d'apprentissage. On calcule alors l'erreur. Puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les $(L-1)$ échantillons qui n'ont pas encore été utilisés pour la validation du modèle. L'opération se répète ainsi L fois (généralement 10 fois) pour qu'en fin de compte chaque sous-échantillon ait été utilisé exactement une fois comme ensemble de validation. La moyenne des L erreurs est enfin calculée pour estimer l'erreur de prédiction. On choisit alors le K entre $K_{Min}, \ldots, K_{Max}$ qui donne la plus faible erreur.
5. Utilisez maintenant la classe de `scikit-learn` [7] pour entraîner un classifieur K plus proches voisins. Comparez les résultats avec ceux obtenus avec votre implémentation.

6. `https://fr.wikipedia.org/wiki/Recherche_des_plus_proches_voisins`
7. `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`

2.2 Classification Naïve Bayésienne

Ce type de classification a été vue en cours, et est également décrit dans le *chapitre 20.2* du livre et sur Wikipédia [8].

1. Donnez le pseudo-code de l'algorithme.
2. Implémentez ce modèle et entraînez le sur chacun des datasets.
3. Évaluez cette technique sur l'ensemble de test en donnant pour chaque classe :
   — L'exactitude (Accuracy)
   — La précision (Precision)
   — Le rappel (Recall)
   — Le F1-score
   — La matrice de confusion (Confusion matrix)
4. Utilisez maintenant la classe de `scikit-learn` [9] pour entraîner un classifieur naïf bayésien. Comparez les résultats avec ceux obtenus avec votre implémentation.

3 Directives

Commencez par voir de plus près le code (voir dossier `Code`) qui vous est donné. Vous devez compléter les fonctions de chargement des datasets. Ensuite, implémentez les deux techniques en suivant le squelette des classes tel qu'imposé dans les fichier python en attaché.

— Complétez le fichier `load_datasets.py`, cela vous permettra de charger vos datasets.
— Lisez bien le fichier `classifieur.py` pour vous aider à implémenter les deux techniques d'apprentissage machine. Nommez le fichier selon la technique : `NaiveBayes.py` pour le modèle *Bayésien Naïf* et `Knn.py` pour la technique des *K plus proches voisis*.
— Compléter le fichier `entrainer_tester.py` pour lancer l'entraînement et le test de vos techniques, c'est le fichier principal pour l'exécution.

4 Livrables

Le travail doit être rendu dans un dossier compressé (.zip), contenant :

— README : un fichier texte contenant une brève description des classes, la répartition des tâches de travail entre les membres d'équipe, et une explication des difficultés rencontrées dans ce travail.
  S'il y a des erreurs ou des bogues dans votre code, mentionnez les, et expliquez les démarches que vous avez prises pour déboguer le code (cela vous permettra d'avoir des points même si votre code n'a pas fonctionné)
— Tout le code que vous avez écrit doit être remis dans le dossier `Code`, vous avez le droit de modifier le code que nous vous avons fournis, mais les noms des méthodes (tels que : `train(...)`, `predict(...)`, `evaluate(...)`, ...etc) doivent rester inchangés
— Un document PDF contenant :
  — Les réponses aux questions
  — Les discussions des résultats obtenus

8. `https://fr.wikipedia.org/wiki/Classification_na%C3%AFve_bay%C3%A9sienne`
9. `https://scikit-learn.org/stable/modules/naive_bayes.html`

— Une comparaison entre les deux techniques d'apprentissage en terme de performances : Temps d'exécution, Accuracy, Precision et Recall. (Faites un tableau récapitulatif).
— Une conclusion (mentionnez aussi les difficultés rencontrées)

Références

Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4) :547–553, 2009.

Annexe 1 : Caractéristiques des ensembles de données

Iris Dataset

- Nombre d'instances (entrainement et test) : 150 (50 instance dans chaque classe)
- Nombre d'attributs pour chaque instance : 4 numériques, et un définissant la classe à laquelle appartient l'instance (étiquette).
- **Détails sur les attributs :** chaque ligne dans le fichier du dataset représente une instance sous la forme :
  - `longueur du sépale` (cm)
  - `largeur du sépale` (cm)
  - `longueur du pétale` (cm)
  - `largeur du pétale` (cm)
  - `classe` : on en distingue 3 types
    - `Iris-setosa`
    - `Iris-versicolour`
    - `Iris-virginica`

Binary Wine quality

- Nombre d'instances (entraînement et test) : 2700
- Nombre d'attributs dans chaque instance : 12 y compris l'étiquette de la classe.
- **Détails sur les attributs** : Chaque ligne dans le dataset représente une instance sous la forme :
  - `acidité fixe` (grammes d'acide tartrique/dm^3)
  - `acide volatile` (grammes d'acide acétique/dm^3)
  - `acide citrique` (g/dm^3)
  - `sucre résiduel` (g/dm^3)
  - `chlorure de sodium` (g/dm^3)
  - `dioxyde de soufre libre` (mg/dm^3)
  - `dioxyde de soufre total` (mg/dm^3)
  - `densité` (g/cm^3)
  - `pH`
  - `sulfate de potassium` (g/dm^3)
  - `alcool` (vol. %)
  - `bon` $\in \{0, 1\}$: la classe à prédire. Pour ce TP, nous avons divisé le jeu de données original en deux classes :
    - 0 = Mauvais (qualité ≤ 5)
    - 1 = Bon (qualité ≥ 7)

Abalones

- Nombre d'instances (entraînement et test) : 4177
- Nombre d'attributs dans chaque instance : 8.
- **Détails sur les attributs** : Chaque ligne dans le dataset représente une instance sous la forme :
  - `Sexe` (mâle, femelle, enfant)

— `Longueur de la coquille` (mm)
— `Diamètre de la coquille` (mm)
— `Hauteur` (mm)
— `Poids total` (g)
— `Poids de la chair` (g)
— `Poids des viscères` (g)
— `Poids de la coquille` (g)
— `Intervalle du nombre d'anneaux` $\in \{0, 1, 2\}$: la classe à prédire. Nous l'avons divisé en 3 classes selon le nombre d'anneaux (n_{rings}) :
  — $0 : n_{rings} < 8$
  — $1 : n_{rings} \in [8, 14]$
  — $2 : n_{rings} > 14$