Aix*Marseille université

M1 Informatique - 2015/2016

Initiation à l'Apprentissage automatique

Examen (durée : 2 heures)

Tous documents autorisés. Vous pouvez consulter la documentation en ligne de `sklearn`. Les questions sont indépendantes.

Le jeu de données Les données de travail [1] mettent en relation huit attributs décrivant des caractéristiques architecturales d'immeubles résidentiels (`surface des murs`, `surface des toits`, `orientation`, etc) et 2 attributs cibles : les `charges de chauffage` et les `charges de climatisation` de ces immeubles. Le problème d'apprentissage consiste à prédire les attributs cibles, c'est-à-dire les charges, en fonction des attributs descriptifs.

Des données ont été recueillies pour 768 immeubles. Toutes les données sont numériques. Les attributs descriptifs sont décrits dans le tableau suivant [2] :

Attribut	Nombre de valeurs différentes
Relative compactness	12
Surface area	12
Wall area	7
Roof area	4
Overall height	2
Orientation	4
Glazing area	4
Glazing area distribution	6

Les charges de chauffage et de climatisation comprennent respectivement 586 et 636 valeurs diférentes.

Vous pouvez consulter les données dans le fichier suivant `http://pageperso.lif.univ-mrs.fr/~francois.denis/IAAM1/data.csv`. Les 8 premières colonnes correspondent aux attributs descriptifs et les deux dernières, aux charges de chauffage et de climatisation (dans cet ordre).

Pour les charger en Python, vous pourrez utiliser le code suivant :

```
import numpy as np
data = np.loadtxt("./data.csv")
X = data[:,:-2]
Y = data[:,-2:]
Yheat = Y[:,0]
Ycool = Y[:,1]
```

1. tirées de A. Tsanas, A. Xifara : *Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools*, Energy and Buildings, Vol. 49, pp. 560-567, 2012
2. la signification précise de chacun de ces attributs n'a pas d'importance pour la suite.

Vous devez rendre deux types de documents :

1. Un programme source, écrit en Python et utilisant la bibliothèque `sklearn`, qui rassemble le code **commenté** permettant de répondre aux questions ci-dessous.
2. Une copie d'examen (papier) sur laquelle vous écrivez les réponses numériques que vous avez obtenues, ainsi que tous, les commentaires qui vous sont demandés ou qui vous semblent appropriés.

Vous remettrez la copie et le programme source à votre surveillant en fin d'épreuve, selon un protocole qu'il vous indiquera.

1 Un problème de régression

Dans un premier temps, on ne va s'intéresser qu'à la variable `Yheat` : la charge de chauffage.

1. Réservez 25% des exemples pour le test et 75% pour l'apprentissage. Apprenez une fonction de régression linéaire f_{heat} permettant de prédire les charges de chauffage en fonction des attributs descriptifs. Affichez le score [3] de cette fonction de régression calculé sur l'échantillon test. Expliquez précisément ce que représente ce score.
2. On peut tenter d'améliorer le score en éliminant un attribut. Pour cela, on peut procéder par validation croisée : pour chaque attribut, on évalue le score obtenu en le supprimant. Puis on décide d'éliminer l'attribut pour lequel le score est le moins élevé. Écrivez le code permettant de trouver l'attribut à éliminer. On pourra utiliser l'instruction `Xnew = np.delete(X_train, i, axis=1)` qui permet d'enlever la i-ème colonne d'un tableau. Quel est cet attribut ?
3. Est-ce une bonne idée d'éliminer un attribut ? Comparez les scores obtenus avec élimination et sans élimination. Que recommandez vous de faire ?

2 Un problème de classification

On peut transformer le problème initial en un problème de classification. Par une méthode de clustering [4], on peut répartir les charges de chauffage et de climatisation en 3 classes : *faibles*, *moyennes*, *élevées*. Les données se répartissent alors de la manière suivante :

Classe	code de la classe	nombre d'exemples	Charges totales moyennes
élevée	0	189	73.6
moyenne	1	185	56.3
faible	2	394	29.7

Les étiquettes des classes se trouvent dans le fichier `http://pageperso.lif.univ-mrs.fr/~francois.denis/IAAM1/labels`. Le code Python permettant de charger ce fichier :

```
target = np.loadtxt('./labels', dtype='int')
```

Nous allons comparer plusieurs méthodes d'apprentissage :

1. les k-plus proches voisins

3. Méthode prédéfinie de la classe `LinearRegression`.
4. Vous n'avez pas besoin de savoir comment elle fonctionne pour traiter l'exercice.

   — `from sklearn.neighbors import KNeighborsClassifier`
   `clf = KNeighborsClassifier(n_neighbors= XXX)`
   — Hyperparamètre à régler : `n_neighbors`.
2. les arbres de décision
   — `from sklearn import tree`
   `clf = tree.DecisionTreeClassifier()`
   — Hyperparamètre à régler : aucun.
3. SVM à noyau linéaire
   — `from sklearn.svm import SVC`
   `clf = SVC(kernel = 'linear')`
   — Hyperparamètre à régler : aucun.
4. SVM à noyau rbf
   — `from sklearn.svm import SVC`
   `clf = SVC(gamma= XXX, kernel = 'rbf')`
   — Hyperparamètre à régler : `gamma`.

Questions

1. Écrivez le code permettant :
   — de séparer les données de travail en des données d'apprentissage et de test,
   — de sélectionner les hyperparamètres des algorithmes d'apprentissage sur l'échantillon d'apprentissage par validation croisée,
   — d'afficher les scores de ces algorithmes évalués sur l'échantillon test.
2. Pour chacune des méthodes, calculez l'intervalle à 95% de confiance auquel le score doit appartenir. Rappel : si `e` est l'erreur estimée, le rayon de l'intervalle d'erreur à la confiance de 95% est égal à : $1.96\sqrt{e(1-e)/N}$, où N est le nombre d'exemples sur lequel l'évaluation a été faite.
3. Commentez les résultats obtenus. Quel algorithme recommanderiez vous d'utiliser ? Avec quelles garanties de résultat ?
4. (Bonus). Les deux meilleurs algorithmes sont-ils départageables avec le test de McNemar (introduit au TP2) ?