Data Mining
TP1 - Sélection de modèles

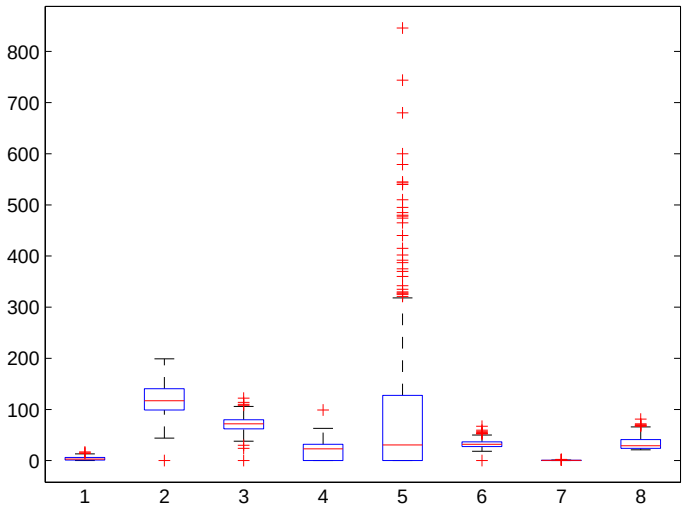
Rémi MUSSARD - Thomas ROBERT

Question 1

Effectuer une brève analyse statistique des données : moyenne, écart-type de chaque variable.

```
M = mean(X)
Med = median(X)
ET = std(X)
figure();
boxplot(X);
```

```
M =
    3.8451  120.8945   69.1055   20.5365   79.7995   31.9926    0.4719   33.2409

Med =
    3.0000  117.0000   72.0000   23.0000   30.5000   32.0000    0.3725   29.0000

ET =
    3.3696   31.9726   19.3558   15.9522  115.2440    7.8842    0.3313   11.7602
```



Ces diverses valeurs permettent d'avoir une vue globale des données.

Question 2

Implémenter une méthode de k-ppv (k plus proche voisins) avec une distance euclidienne.

On utilise la fonction knn fournie qui renvoie les prédictions et les distances calculées à partir des données d'entrées.

Question 3

Séparer aléatoirement l'ensemble des données en un ensemble d'apprentissage et un ensemble de test en respectant au mieux la proportion des classes. L'ensemble de test ne sera utilisé qu'une fois. Nota : voir la fonction splidata.m sur Moodle.

La fonction splitdata permet de séparer les données en respectant la proportion des classes. On sépare ainsi notre ensemble de données en deux ensembles de même taille.

```
[xapp, yapp, xtest, ytest] = splitdata(X, y, 0.5);
```

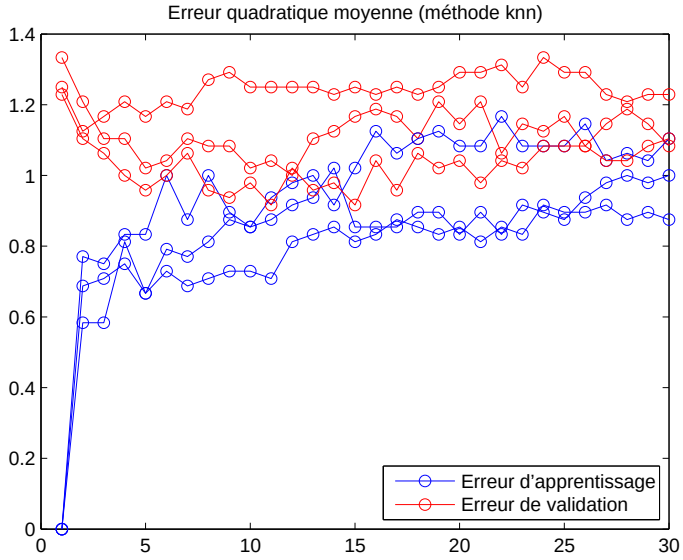
Question 4

Séparer l'ensemble d'apprentissage en 2 ensembles : un autre ensemble d'apprentissage et un ensemble de validation. Tester votre méthode k-ppv sur l'ensemble d'ensemble d'apprentissage et l'ensemble de validation pour différentes valeurs de k ? Tracer une courbe de l'erreur d'apprentissage et une courbe de l'erreur de validation en fonction de k. Quelle est la valeur de k qui donne la plus faible erreur en validation ?

```
figure();

% on execute le code 3 fois pour voir son insatabilité
for i=1:3
    % init de la matrice des erreurs pour chaque k de 1 à 30
    kmax = 30;
    errApp = zeros(kmax, 1);
    errVal = zeros(kmax, 1);

    % découpage en jeu d'apprentissage et de validation
    [xapp2, yapp2, xval2, yval2] = splitdata(xapp, yapp, 0.5);

    % pour chaque k
    for k = 1:kmax

        % prédiction
        [ypredApp, Dist] = knn(xapp2, xapp2, yapp2, k);
        [ypredVal, Dist] = knn(xval2, xapp2, yapp2, k);

        % calcul d'erreur quadratique moyenne pour le k choisi
        errApp(k) = mean((ypredApp - yapp2).^2);
        errVal(k) = mean((ypredVal - yval2).^2);
    end

    % affichage des erreurs
    plot(errApp, 'o-');
    hold on;
    plot(errVal, 'or-');
    title('Erreur quadratique moyenne (méthode knn)');
    leg = legend('Erreur d''apprentissage', 'Erreur de validation');
    set(leg,'Location','SouthEast');

    % meilleure valeur de k en validation

    [erreurMin, bestk] = min(errVal);
    fprintf('Meilleur k par méthode knn : %i\n',bestk);
end
```

```
Meilleur k par méthode knn : 2
```

```
Meilleur k par méthode knn : 12
Meilleur k par méthode knn : 11
```



Avec cette méthode, on constate que la valeur de k trouvée est très dépendante du découpage aléatoire des données qui a été réalisé, entre ensemble de test et ensemble d'apprentissage.

Pour résoudre ce problème, on peut utiliser la méthode de validation croisée qui permet d'utiliser toute les données pour réaliser les tests et l'apprentissage de façon "rotative".

Question 5

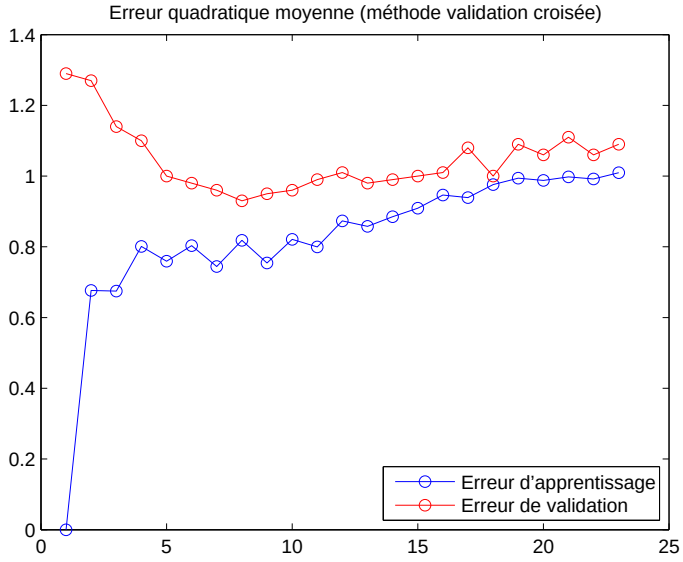
Refaire l'expérience en utilisant une méthode de validation croisée sur les données d'apprentissage crées à la question 3. Quelle est la meilleure valeur de k ? Nota : voir sur Moodle la fonction SepareDataNfoldCV.m.

On utilise la fonction SepareDataNfoldCV pour découper l'ensemble d'apprentissage en plusieurs blocs afin d'y appliquer la méthode de validation croisée.

```
% constantes
kmax = 23;
Nfold = 20;
errApp = zeros(kmax, Nfold);
errVal = zeros(kmax, Nfold);

% pour chaque bloc
for NumFold = 1:Nfold
    % séparation des données
    [xapp2, yapp2, xval2, yval2] = SepareDataNfoldCV(xapp, yapp, Nfold, NumFold);

    % pour chaque k
    for k = 1:kmax

        % prédiction
        [ypredApp, Dist] = knn(xapp2,xapp2,yapp2,k);
        [ypredVal, Dist] = knn(xval2,xapp2,yapp2,k);

        % calcul d'erreur quadratique moyenne pour le k choisi
        errApp(k, NumFold) = mean((ypredApp - yapp2).^2);
        errVal(k, NumFold) = mean((ypredVal - yval2).^2);
    end
end

% erreur d'apprentissage et de validation moyenne pour chaque k
errAppK = mean(errApp')';
errValK = mean(errVal')';
```

```
28
29 % affichage des erreurs
30 figure();
31 plot(errAppK, 'o-');
32 hold on;
33 plot(errValK, 'or-');
34 title('Erreur quadratique moyenne (méthode validation croisée)');
35 leg = legend('Erreur d''apprentissage', 'Erreur de validation');
36 set(leg,'Location','SouthEast');
37 hold off;
38
39 % meilleure valeur de k en validation
40 [erreurMin, bestk] = min(errValK);
41 fprintf('Meilleur k par méthode de validation croisée : %i\n',bestk);
```

```
Meilleur k par méthode de validation croisée : 8
```



On voit effectivement que la méthode de la validation croisée est plus stable.