

Compte rendu TP 2 AI : Réseaux et Modèles Bayésiens

1. Naive Bayes model with Gaussian, multinomial, or kernel predictors :

En statistique, les classificateurs Naive Bayes sont une famille de "classificateurs probabilistes" simples basés sur l'application du théorème de Bayes avec des hypothèses d'indépendance fortes (naïves) entre les caractéristiques. Ils font partie des modèles de réseau bayésiens les plus simples, mais couplés à l'estimation de la densité du noyau, ils peuvent atteindre des niveaux de précision plus élevés.

Les classificateurs naïfs de Bayes sont hautement évolutifs, nécessitant un certain nombre de paramètres linéaires dans le nombre de variables (caractéristiques / prédicteurs) dans un problème d'apprentissage. L'entraînement au maximum de vraisemblance peut être fait en évaluant une expression de forme fermée, qui prend un temps linéaire, plutôt que par une approximation itérative coûteuse comme utilisée pour de nombreux autres types de classificateurs.

Jusqu'à présent, la discussion a dérivé le modèle de caractéristiques indépendant, c'est-à-dire le modèle de probabilité naïf de Bayes. Le classifieur naïf de Bayes combine ce modèle avec une règle de décision. Une règle courante consiste à choisir l'hypothèse la plus probable

Les modèles naïfs de Bayes supposent que les observations ont une distribution multivariée étant donné l'appartenance à une classe, mais le prédicteur ou les caractéristiques composant l'observation sont indépendants. Ce cadre peut accueillir un ensemble complet de fonctionnalités tel qu'une observation est un ensemble de comptages multinomiaux.

Sur Matlab on a les fonctionnalités suivantes :

- **Create Naive Bayes Model** : Pour entraîner un modèle Bayes naïf, on utilise `fitcnb` dans l'interface de ligne de commande. Après l'entraînement, on estime les probabilités postérieures en passant le modèle et les données de prédicteur à prédire

- **Cross-Validate** : La validation croisée à k blocs, « k-fold cross-validation » : on divise l'échantillon original en k échantillons (ou « blocs »), puis on sélectionne un des k échantillons comme ensemble de validation pendant que les $k-1$ autres échantillons constituent l'ensemble d'apprentissage. Après l'apprentissage, on peut calculer une performance de validation. Puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les blocs prédéfinis. À l'issue de la procédure nous obtenons ainsi k scores de performances, un par bloc. La moyenne et l'écart type des k scores de performances peuvent être calculés pour estimer le biais et la variance de la performance de validation

Pour cela on a les fonctions suivantes :

<code>crossval</code>	Renvoie un classificateur Bayes naïf à validation croisée
<code>kfoldEdge</code>	Donne les bords de classification pour les observations non utilisées pour la formation
<code>kfoldLoss</code>	Calcul la perte de classification pour les observations non utilisées pour la formation
<code>kfoldfun</code>	Donne la fonction de validation croisée
<code>kfoldMargin</code>	Donne les marges de classification pour les observations non utilisées pour la formation
<code>kfoldPredict</code>	Prédire la réponse pour les observations non utilisées pour la formation

- **Measure Performance:** est le processus de collecte, d'analyse et / ou de rapport d'informations concernant les performances du classificateur bayes:

Pour cela on a les fonctions suivantes :

loss	Calcul la perte de classification pour le classificateur de Bayes naïf
resubLoss naïf	Calcul la perte de classification de resubstitution pour le classificateur Bayes naïf
logp Log Bayes	Calcul la densité de probabilité inconditionnelle pour le classificateur naïf de Bayes
compareHoldout nouvelles données	Comparez les précisions de deux modèles de classification à l'aide de nouvelles données
edgeClassification	Donne les bords des classificateur Bayes naïf
margin	Donne les marges de classification pour le classificateur de Bayes naïf
partialDependence	Calcule la dépendance partielle
plotPartialDependence conditionnelle individuelle (ICE)	Pour créer des graphiques de dépendance partielle (PDP) et d'espérance conditionnelle individuelle (ICE)
resubEdge naïf	Donne le bord de classification de resubstitution pour le classificateur Bayes naïf
resubMargin naïf	Reconstitution des marges de classification de pour le classificateur de Bayes naïf

- **Classify Observations:** En plus de la discrimination entre les groupes, l'analyse discriminante permet de classer les observations en groupes. La classification peut être effectuée en utilisant les variables de réponse observées ou plus communément les valeurs de la fonction discriminante linéaire. Nous commencerons par examiner la classification à l'aide de variables observées.

Pour cela on a les fonctions suivantes :

predict\resubPredict	Classifier les observations à l'aide du classificateur naïf de Bayes
----------------------	--

2. Train Classification Models in Classification Learner App:

On va utiliser le Classification Learner pour former des modèles de ces classificateurs: arbres de décision, analyse discriminante, machines vectorielles de support, régression logistique, voisins les plus proches, Bayes naïfs et classification d'ensemble. En plus des modèles d'entraînement, ça nous permet aussi d'explorer nos données, sélectionner des fonctionnalités, spécifier des schémas de validation et évaluer les résultats. on peut exporter un modèle vers l'espace de travail pour utiliser le modèle avec de nouvelles données ou générer du code MATLAB pour en savoir plus sur la classification par programmation.

La formation d'un modèle à Classification Learner comprend deux parties:

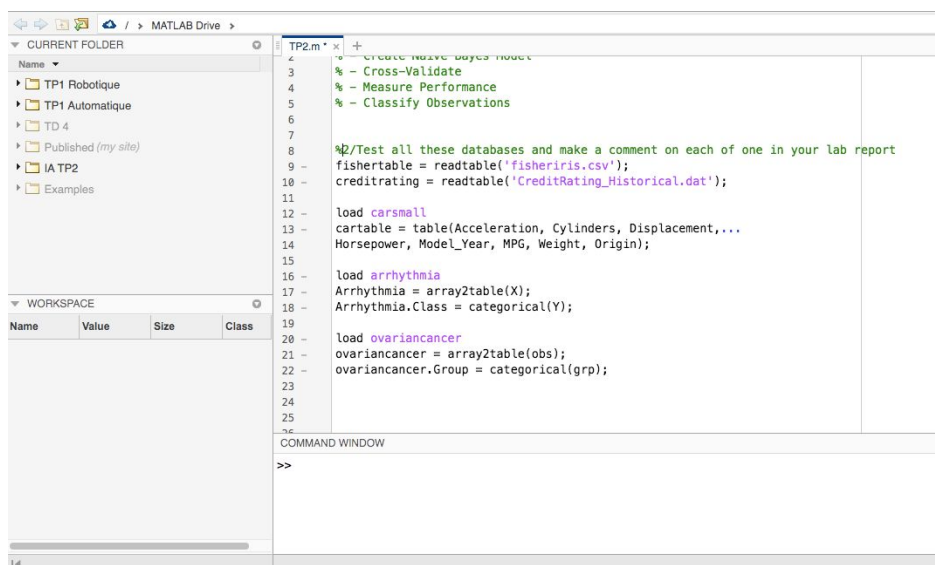
Modèle validé: Former un modèle avec un schéma de validation. Par défaut, l'application protège contre le surajustement en appliquant une validation croisée. On peut également choisir la validation d'exclusion. Le modèle validé est visible dans l'application.

Modèle complet: Former un modèle sur des données complètes sans validation. L'application entraîne ce modèle simultanément avec le modèle validé. Cependant, le modèle entraîné sur des données complètes n'est pas visible dans l'application. Lorsqu'on choisit un classificateur à exporter vers l'espace de travail, l'apprenant de classification exporte le modèle complet.

L'application nous affiche les résultats du modèle validé. Les mesures de diagnostic, telles que la précision du modèle, et les graphiques, tels qu'un nuage de points ou le diagramme de matrice de confusion, reflètent les résultats du modèle validé.

On peut entraîner automatiquement une sélection de ou tous les classificateurs, comparer les résultats de validation et choisir le meilleur modèle qui convient à notre problème de classification. Lorsqu'on choisit un modèle à exporter vers l'espace de travail, l'apprenant de classification exporte le modèle complet. Étant donné que l'apprenant de classification crée un objet de modèle du modèle complet pendant la formation, on ne rencontre aucun délai lorsqu'on exporte le modèle. On peut utiliser le modèle exporté pour faire des prédictions sur de nouvelles données.

On va d'abord importer nos données qu'on souhaite visualiser ainsi :



The screenshot shows the MATLAB interface with the following components:

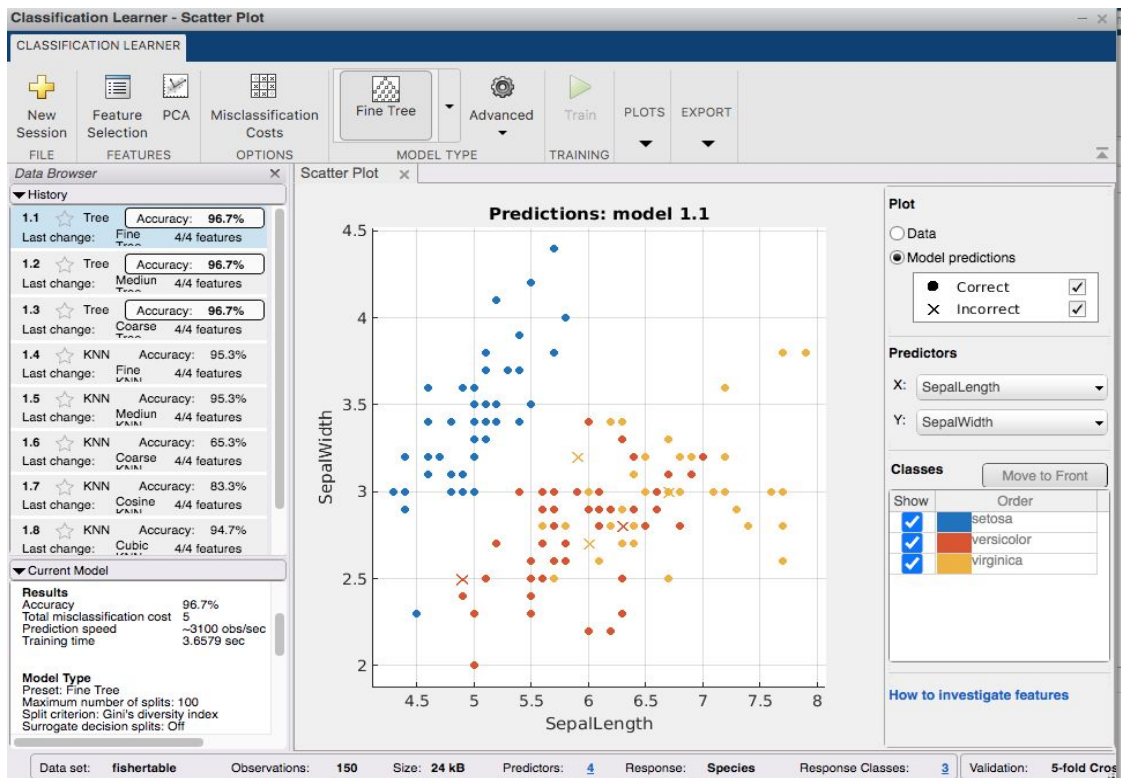
- CURRENT FOLDER:** Displays a directory structure including 'TP1 Robotique', 'TP1 Automatique', 'TD 4', 'Published (my site)', 'IA TP2', and 'Examples'.
- WORKSPACE:** A table with columns 'Name', 'Value', 'Size', and 'Class'. It is currently empty.
- Script Editor:** Contains MATLAB code for loading and preparing data:


```

1 % Create naive bayes model
2 % - Cross-Validate
3 % - Measure Performance
4 % - Classify Observations
5
6
7
8 %/Test all these databases and make a comment on each of one in your lab report
9 -
10 fishertable = readtable('fisheriris.csv');
11 creditrating = readtable('CreditRating_Historical.dat');
12
13 load carsmall
14 cartable = table(Acceleration, Cylinders, Displacement,...
15 Horsepower, Model_Year, MPG, Weight, Origin);
16
17 load arrhythmia
18 Arrhythmia = array2table(X);
19 Arrhythmia.Class = categorical(Y);
20
21 load ovariancancer
22 ovariancancer = array2table(obs);
23 ovariancancer.Group = categorical(grp);
24
25
26

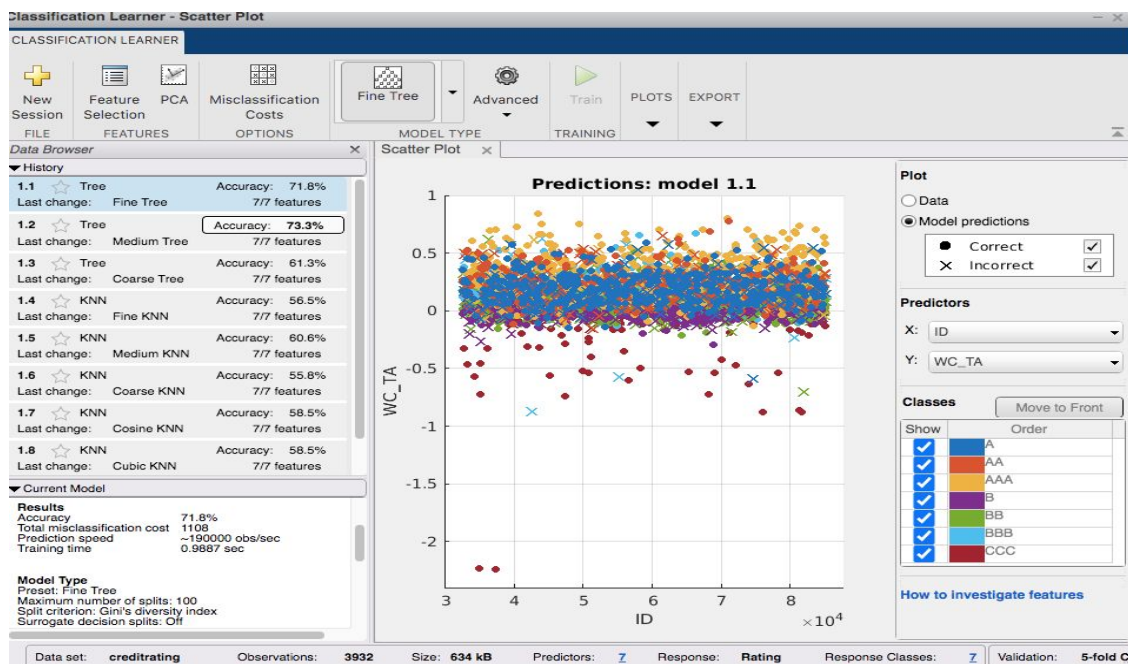
```
- COMMAND WINDOW:** Shows the prompt '>>'.

a. Fisher Iris data classification:



Le Classification Learner App ici nous affiche un nuage de points des prédictions du modèle 1.1 qui est un arbre de décision, en fonction de nos données (sepalWidth et sepalLength). l'App nous dit que c'est le modèle le plus précis à 96.7% d'accuracy : cela veut dire que si on veut savoir quel type de d'iris on a, le modèle va se tromper que 3.3 % des fois .

b. Credit Ranking data classification:



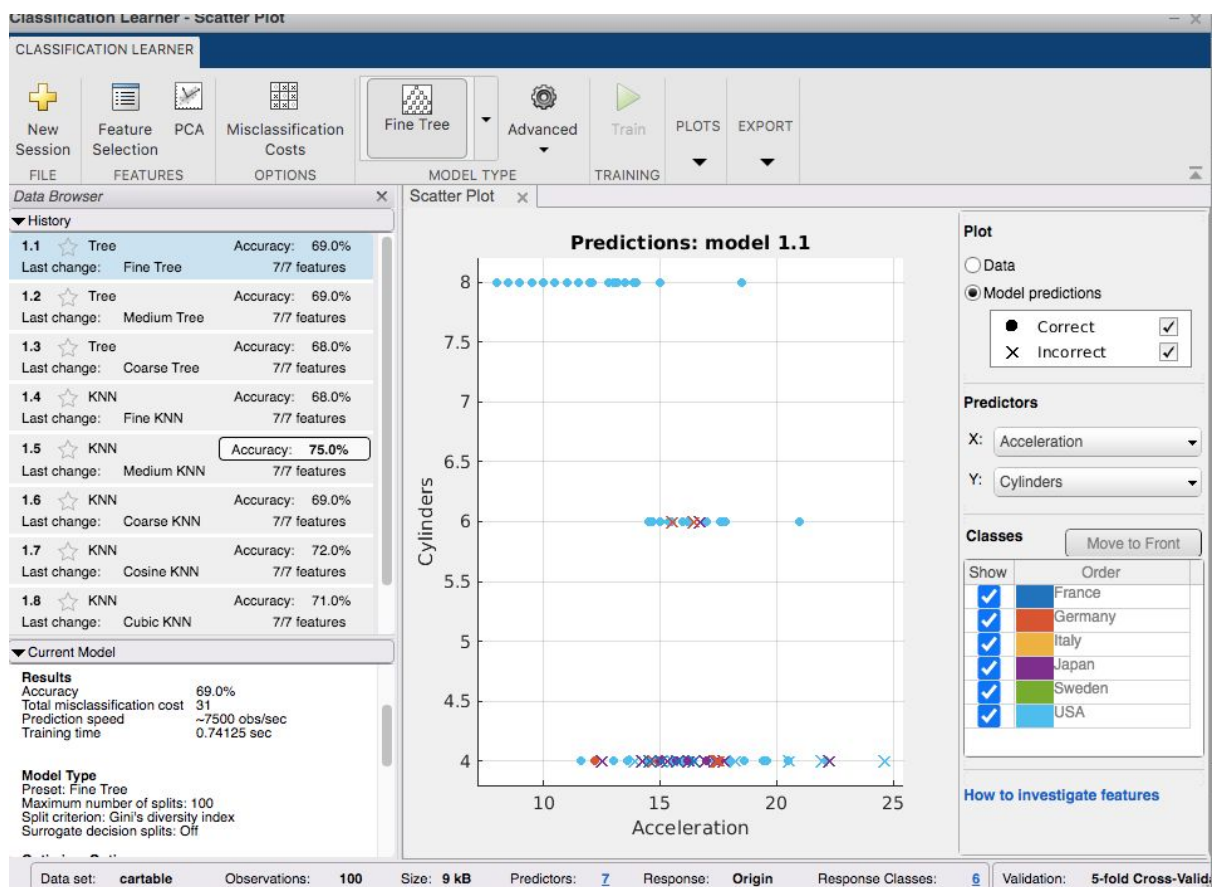
Cet exemple montre comment créer un outil de notation de crédit automatisé.

L'une des tâches fondamentales de la gestion du risque de crédit consiste à attribuer une note de crédit à un emprunteur. Les notes sont utilisées pour classer les clients en fonction de leur solvabilité perçue: de meilleures notes signifient des clients moins risqués; des grades similaires signifient un niveau de risque similaire. Les notes se divisent en deux catégories: le rating de crédit et le scoring de crédit. Les rating de crédit sont un petit nombre de classes distinctes, généralement étiquetées avec des lettres, telles que «AAA», «BB-», etc.

Le set de données qu'on a est constitué d'informations sur les ratios financiers et les secteurs industriels pour une liste de clients professionnels. La variable de réponse est constituée des notations de crédit (AAA, AA, A, BBB, BB, B, CCC) attribuées par une agence de notation.

Le Classification Learner App nous affiche encore une fois un nuage de points de nos prédictions du modèle 1.2 qui est un arbre de décision, en fonction de WC_TA (Working capital / Total Assets) et customer ID. L'App nous dit que c'est le modèle le plus précis à 73.3% d'accuracy : Cela veut dire que notre algorithme est correct à 73.3% en prédiction de risque de nos clients.

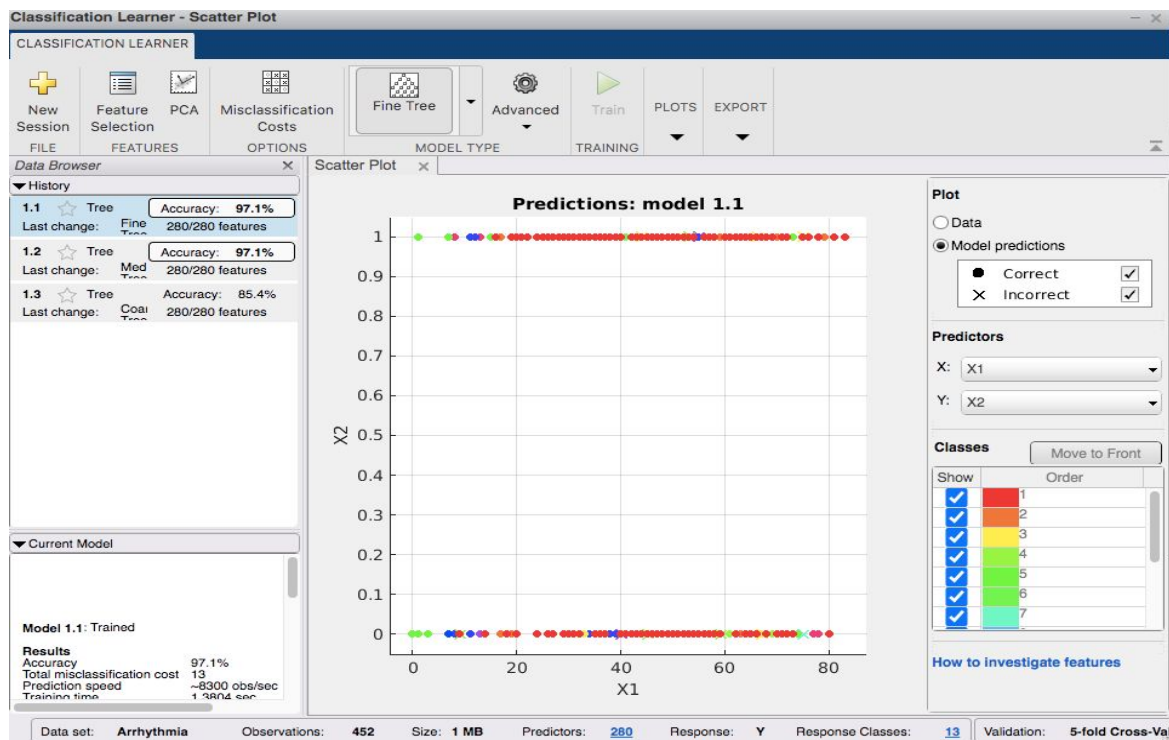
c. Cars data classification:



Ce jeu de données est constitué d'informations sur les mesures des voitures, dans les années 1970, 1976 et 1982. On va essayer de classer le pays d'origine à partir de ce dataset avec le classification learner app.

Le Classification Learner App nous affiche encore une fois un nuage de points des prédictions du modèle 1.5 (il précise les prédictions correctes et celles qui ne le sont pas) en fonction de l'accélération et le cylindres des voitures. Le modèle est un nearest-neighbor classification/modèle classification KNN. l'App nous dit que c'est le modèle le plus précis à 75% d'accuracy : Cela veut dire que notre algorithme est correct à 75% en prédiction de l'origine d'une voiture .

d. Arrhythmia data classification :

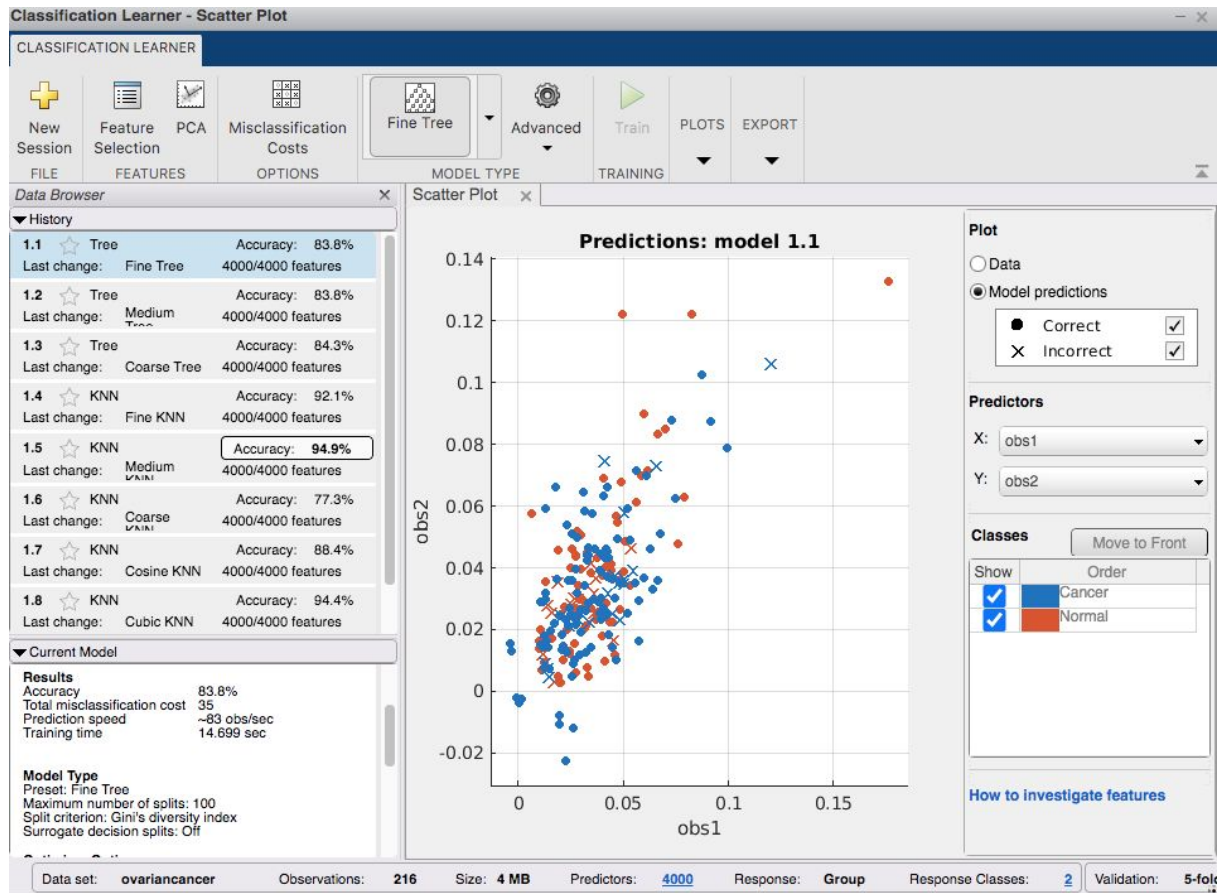


Ce jeu de données est constitué d'informations sur le patient et variables de réponse indiquant la présence et l'absence d'arythmie cardiaque.

Le Classification Learner App nous affiche encore une fois un nuage de points des prédictions du modèle 1.1 (il précise les prédictions correctes et celles qui ne le sont pas) qui est un modèle de classification d'arbres de décisions. l'App nous dit que c'est le modèle le plus précis à 97.1% d'accuracy : Cela veut dire que notre algorithme est correct à 97.1% en prédiction de la présence et l'absence d'arythmie cardiaque.

Ici la classification erronée d'un patient comme «normal» a des conséquences plus graves que les faux positifs classés comme «souffrant d'arythmie». Donc, dans les 2.9% des fausses prédictions (faux négatif et/ou faux positif) notre modèle doit optimiser les faux négatifs (les malades déclarer comme pas malades) .

e. Overance Cancer data classification:



L'objectif est de construire un classificateur qui puisse faire la distinction entre les patients cancéreux et les patients témoins à partir du tableau de protéines WCX2. Il comprend 95 contrôles et 121 cancers de l'ovaire.

Le Classification Learner App nous affiche un nuage de points des prédictions du modèle 1.5 (en précisant les prédictions correctes et celles qui ne le sont pas). Le modèle est un modèle de nearest-neighbor classification/modèle classification KNN. L'App nous dit que c'est le modèle le plus précis à 94.9% d'accuracy : Cela veut dire que notre algorithme est correct à 94.9% en prédiction des les patients cancéreux et les patients témoins.

Ici aussi la classification erronée d'un patient comme « patients témoins» a des conséquences plus graves que les faux positifs classés comme « patients cancéreux». Donc, dans les 5.1% des fausses prédictions (faux négatif et/ou faux positif) notre modèle doit optimiser les faux négatifs (les malades déclarer comme pas malades) .

3. Fisher's iris data set (out of the application) :

Le jeu de données comprend 50 échantillons de chacune des trois espèces d'iris (*Iris setosa*, *Iris virginica* et *Iris versicolor*). Quatre caractéristiques ont été mesurées à partir de chaque échantillon : la longueur et la largeur des **sépales** et des **pétales**, en centimètres. Sur la base de la combinaison de ces quatre variables, on a élaboré un modèle d'analyse de bayes naïf permettant de distinguer les espèces les unes des autres.

```
%2/Test this code and make a comment on it in the lab report:
load fisheriris
X = meas(:,3:4);
Y = species;
tabulate(Y)

Mdl = fitcnb(X,Y,'ClassNames',{'setosa','versicolor','virginica'})
setosaIndex = strcmp(Mdl.ClassNames,'setosa');
estimates = Mdl.DistributionParameters{setosaIndex,1}

figure
gscatter(X(:,1),X(:,2),Y);
h = gca;
xlim = h.XLim;
ylim = h.YLim;
hold on

Params = cell2mat(Mdl.DistributionParameters);
Mu = Params(2*(1:3)-1,1:2); % Extract the means
Sigma = zeros(2,2,3);

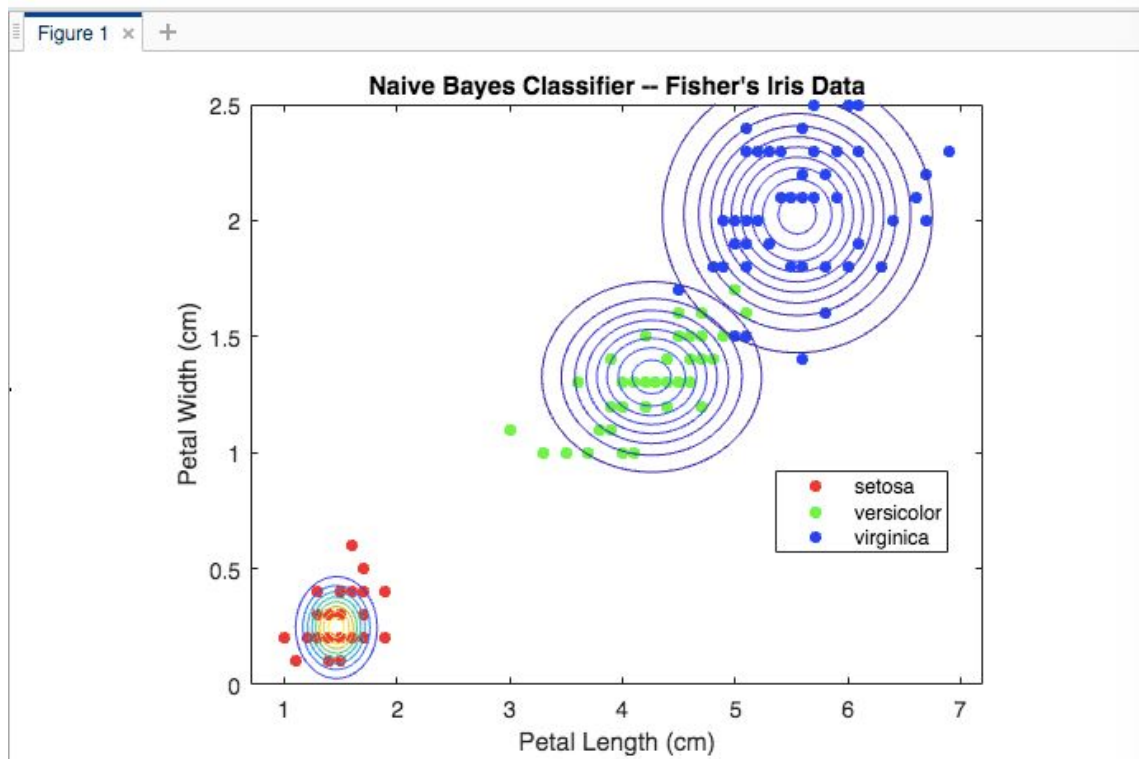
for j = 1:3
    Sigma(:,:,j) = diag(Params(2*j,:)).^2; % Create diagonal covariance matrix
    xlim = Mu(j,1) + 4*[-1 1]*sqrt(Sigma(1,1,j));
    ylim = Mu(j,2) + 4*[-1 1]*sqrt(Sigma(2,2,j));
    f = @(x,y) arrayfun(@(x0,y0) mvnpdf([x0 y0],Mu(j,:),Sigma(:,:,j)),x,y);
    fcontour(f,[xlim ylim]) % Draw contours for the multivariate normal distributions
end

h.XLim = xlim;
h.YLim = ylim;
title('Naive Bayes Classifier -- Fisher's Iris Data')
xlabel('Petal Length (cm)')
ylabel('Petal Width (cm)')
legend('setosa','versicolor','virginica')
hold off
```

On a d'abord chargé l'ensemble de données d'iris de Fisher ensuite on a fait l'apprentissage d'un classificateur naïf de Bayes.

Par défaut, le logiciel modélise la distribution des prédicteurs au sein de chaque classe en utilisant une distribution gaussienne ayant une moyenne et un écart type. On utilise la notation par points pour afficher les paramètres d'un ajustement gaussien particulier, par exemple, afficher l'ajustement pour la première entité dans *setosa*.

On va ensuite tracer les contours gaussiens. On trouve le résultat suivant :



Le schéma nous montre les régions de prédiction de notre modèle en fonction des données Petal Length et Petal Width. On peut voir qu'il y a une confusion entre les iris versicolor et virginica.

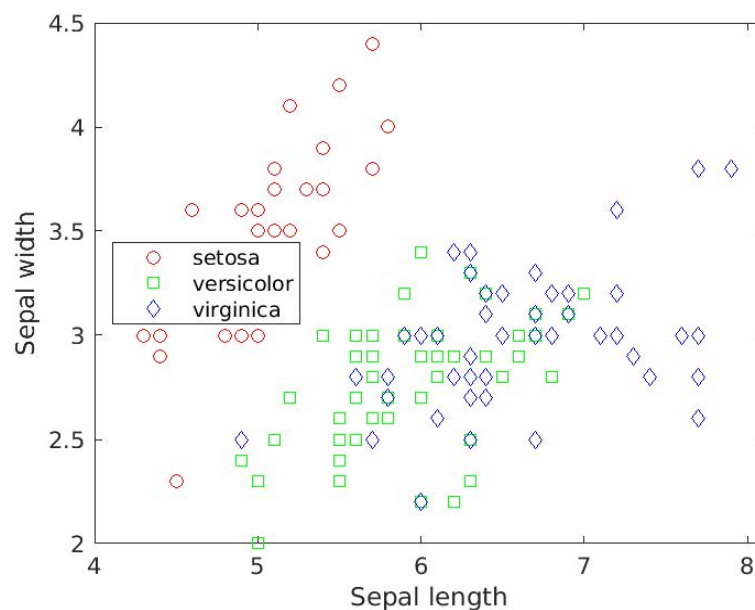
4. Naive Bayes Examples :

- Posterior Classification Probabilities:

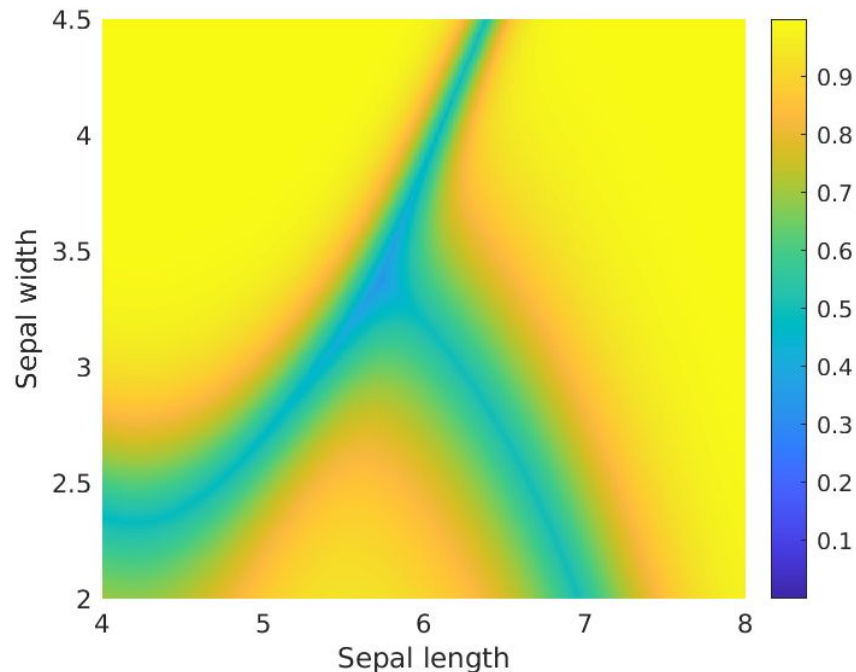
Cet exemple montre comment visualiser les probabilités de classification postérieure prédites par un modèle de classification de Bayes naïf.

X est une matrice numérique qui contient deux mesures de pétales pour 150 iris. Y est un tableau de cellules de vecteurs de caractères contenant les espèces d'iris correspondantes.

On va visualiser les données à l'aide d'un nuage de points. On regroupe les variables par espèce d'iris. On obtient :



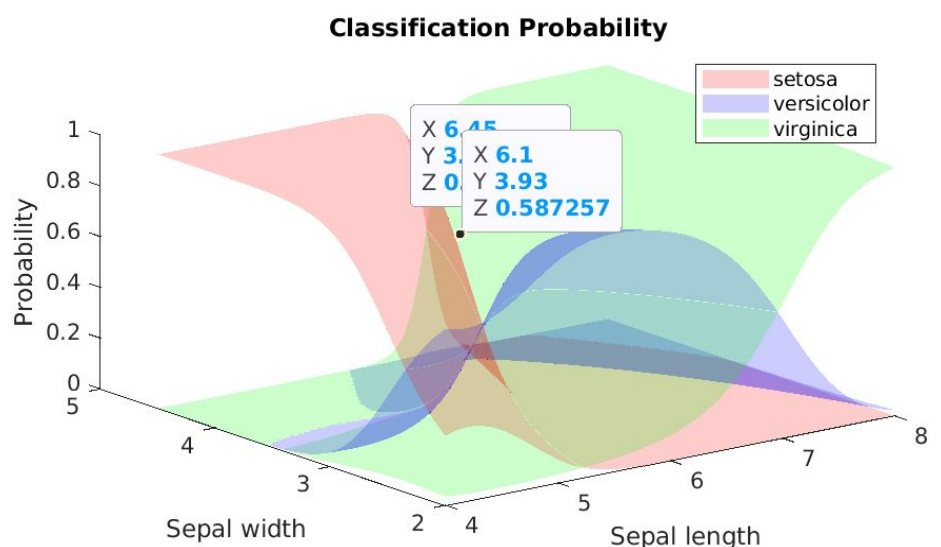
On va donc faire l'apprentissage d'un classificateur naïf de Bayes. Ensuite on crée une grille de points couvrant tout l'espace dans certaines limites des données. Les données en $X(:, 1)$ sont comprises entre 4,3 et 7,9. Les données en $X(:, 2)$ sont comprises entre 2 et 4,4. On prédit les espèces d'iris et les probabilités de classe postérieure de chaque observation dans XGrid en utilisant mdl. Et enfin on trace la distribution de probabilité postérieure pour chaque espèce :



Plus une observation se rapproche de la surface de décision, moins il est probable que les données appartiennent à une certaine espèce.

On trace les distributions de probabilité de classification individuellement:

On remarque ici la probabilité que notre iris appartienne à une certaine espèce en fonction de sepal width et sepal length.



- Classification:

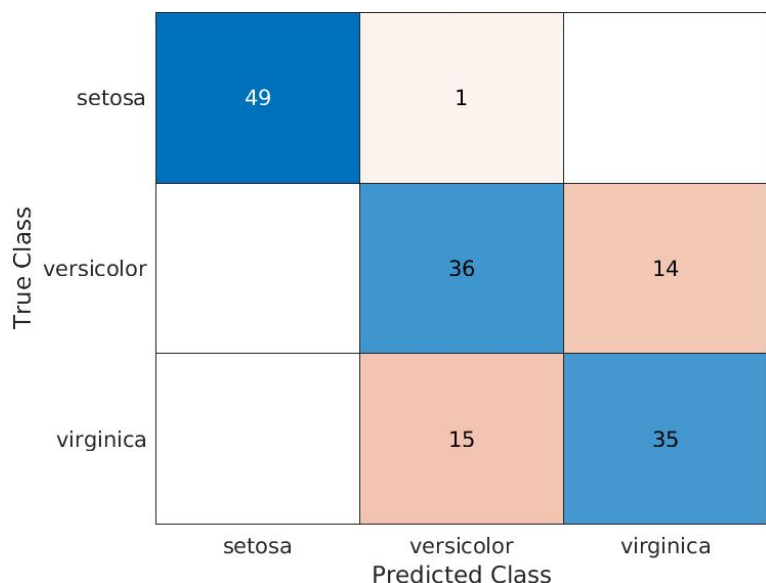
Cet exemple montre comment effectuer une classification à l'aide d'une analyse discriminante, de classificateurs Bayes naïfs et d'arbres de décision. On suppose que nous avons un ensemble de données contenant des observations avec des mesures sur différentes variables (appelées prédicteurs) et leurs étiquettes de classe connues. Si on obtient des valeurs de prédicteur pour de nouvelles observations, pouvons-nous déterminer à quelles classes ces observations appartiennent probablement? C'est le problème de la classification.

Supposons qu'on mesure un sépale et un pétale d'un iris et qu'on doit déterminer son espèce sur la base de ces mesures. Une approche pour résoudre ce problème est connue sous le nom d'analyse discriminante.

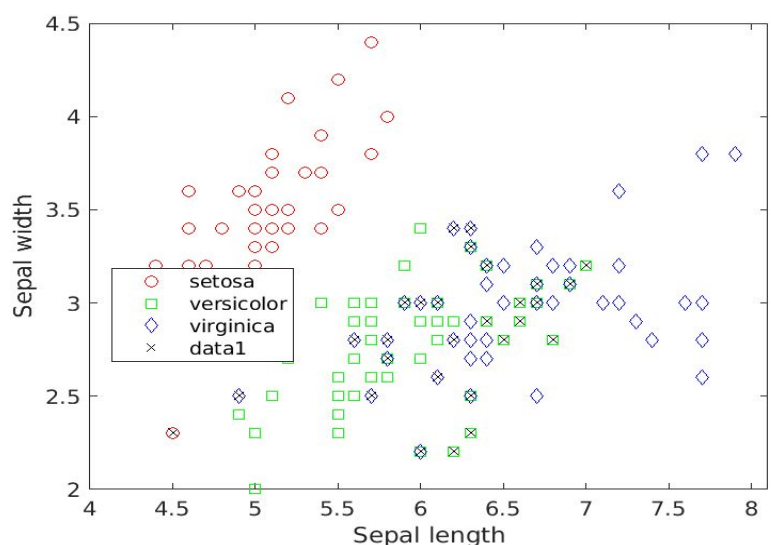
La fonction `fittediscr` peut effectuer une classification en utilisant différents types d'analyse discriminante. On classe les données à l'aide de l'analyse discriminante linéaire par défaut (LDA).

Pour une meilleure lecture, on calcule la matrice de confusion sur l'ensemble d'apprentissage. Une matrice de confusion contient des informations sur les étiquettes de classe connues et les étiquettes de classe prédites. D'une manière générale, l'élément (i, j) dans la matrice de confusion est le nombre d'échantillons dont l'étiquette de classe connue est la classe i et dont la classe prédite est j . Les éléments diagonaux représentent des observations correctement classées.

La matrice de confusion nous dit que sur les 150 observations d'apprentissage, 20% ou 30 observations sont mal classées par la fonction discriminante linéaire.



On peut voir lesquels en traçant X à travers les points mal classés.



- **Comparaison and visualization Decision Surfaces of Different Classifiers :**

Dans cet exemple, on compare les différents algorithmes de classification. Et on trace leurs surfaces de décision.

a. Naive Bayes Classifiers :

Les classificateurs Naive Bayes sont parmi les classificateurs les plus populaires. Bien que l'hypothèse d'indépendance conditionnelle de classe entre les variables ne soit pas vraie en général, les classificateurs Bayes naïfs se sont avérés bien fonctionner en pratique sur de nombreux ensembles de données.

Pour cet ensemble de données, le classificateur naïf de Bayes avec une estimation de la densité du noyau obtient une erreur de restitution et une erreur de validation croisée plus petites que le classificateur naïf de Bayes avec une distribution gaussienne.

b. Linear and Quadratic Discriminant Analysis:

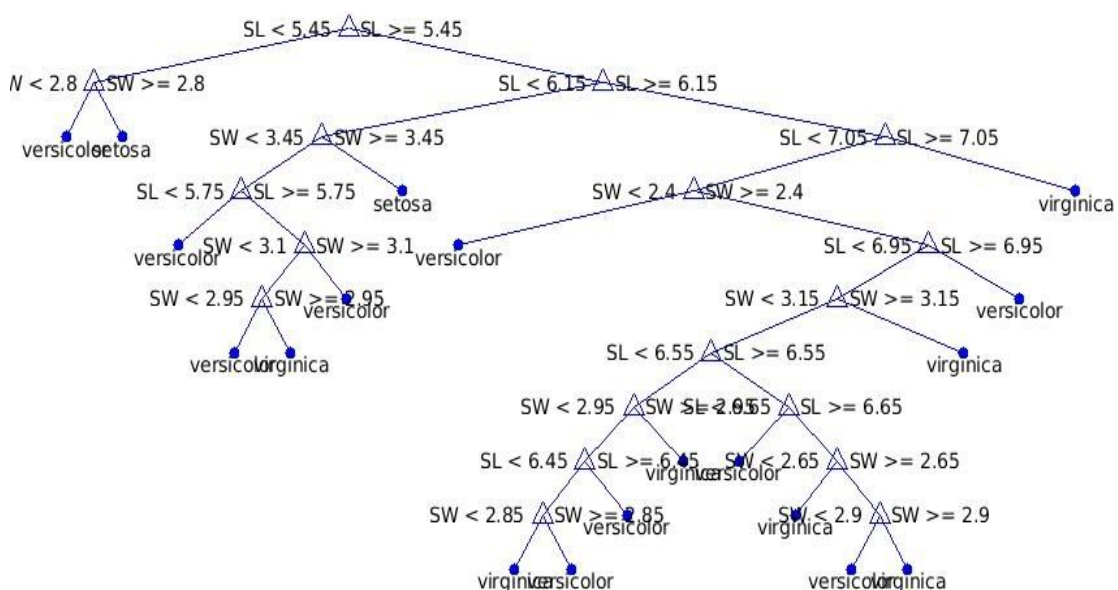
Pour certains ensembles de données, les régions des différentes classes ne sont pas bien séparées par des lignes. Lorsque tel est le cas, l'analyse discriminante linéaire n'est pas appropriée. Au lieu de cela, vous pouvez essayer l'analyse discriminante quadratique (QDA) pour nos données. QDA a une erreur de validation croisée légèrement plus grande que LDA. Cela montre qu'un modèle plus simple peut obtenir des performances comparables ou meilleures qu'un modèle plus compliqué.

c. Decision Tree:

Un autre algorithme de classification est basé sur un arbre de décision. Un arbre de décision est un ensemble de règles simples, telles que «si la longueur du sépale est inférieure à 5,45, classez le spécimen comme *setosa*». Les arbres de décision ne sont pas non plus paramétriques car ils ne nécessitent aucune hypothèse sur la distribution des variables dans chaque classe.

La fonction `fitctree` crée un arbre de décision.

Une façon de visualiser l'arbre de décision consiste à dessiner un diagramme de la règle de décision et des attributions de classe:

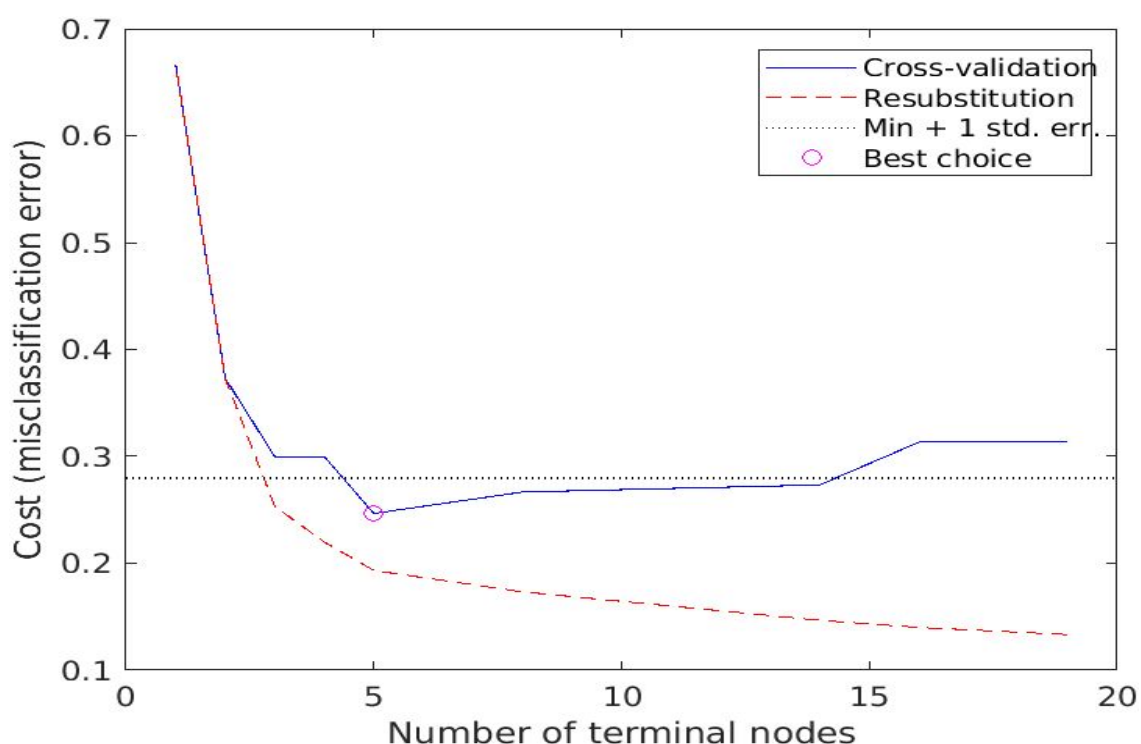


Cet arbre à l'aspect encombré utilise une série de règles de la forme " $SL < 5,45$ " pour classer chaque spécimen dans l'un des 19 nœuds terminaux. Pour déterminer l'affectation des espèces pour une observation, on commence au nœud supérieur et on applique la règle. Si le point satisfait à la règle, on prend le chemin de gauche, sinon on prend le chemin de droite. En fin de compte, on atteint un nœud terminal qui attribue l'observation à l'une des trois espèces.

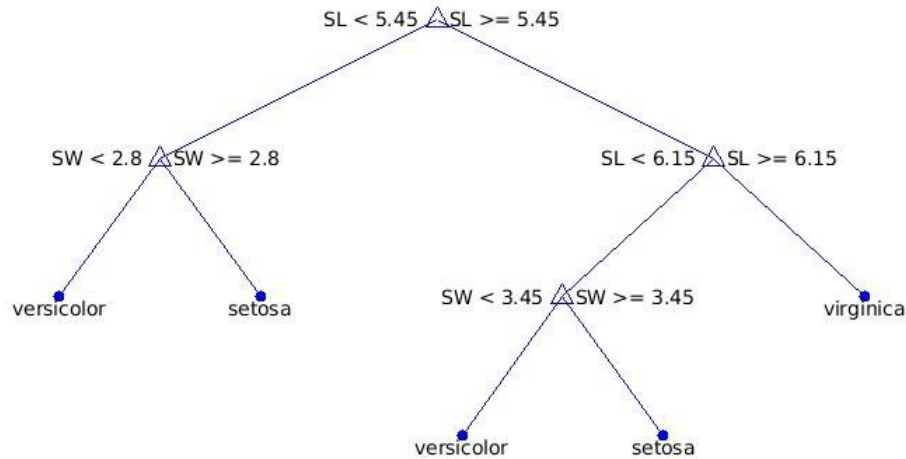
Pour l'algorithme d'arbre de décision, l'estimation de l'erreur de validation croisée est significativement plus grande que l'erreur de restitution. Cela montre que l'arborescence générée surpasse l'ensemble d'apprentissage. En d'autres termes, il s'agit d'un arbre qui classe bien l'ensemble d'apprentissage d'origine, mais la structure de l'arborescence est sensible à cet ensemble d'apprentissage particulier de sorte que ses performances sur les nouvelles données sont susceptibles de se dégrader. Il est souvent possible de trouver un arbre plus simple qui fonctionne mieux qu'un arbre plus complexe sur de nouvelles données.

On essaye d'élaguer l'arbre. On calcule d'abord l'erreur de substitution pour différents sous-ensembles de l'arborescence d'origine. Ensuite on calcule l'erreur de validation croisée pour ces sous-arbres. Un graphique montre que l'erreur de restitution est trop optimiste. Il diminue toujours à mesure que la taille de l'arbre augmente, mais au-delà d'un certain point, l'augmentation de la taille de l'arbre augmente le taux d'erreur de validation croisée. On peut le montrer sur le graphique en calculant une valeur seuil égale au coût minimum plus une erreur standard. Le "meilleur" niveau calculé par la méthode cvloss est le plus petit arbre sous ce seuil.

Une règle simple serait de choisir l'arbre avec la plus petite erreur de validation croisée. Bien que cela puisse être satisfaisant, on veut peut-être utiliser un arbre plus simple s'il est à peu près aussi bon qu'un arbre plus complexe. Pour cet exemple, on prend l'arbre le plus simple qui se trouve dans une erreur standard du minimum. C'est la règle par défaut utilisée par la méthode cvloss de ClassificationTree.



On affiche l'arbre élagué:

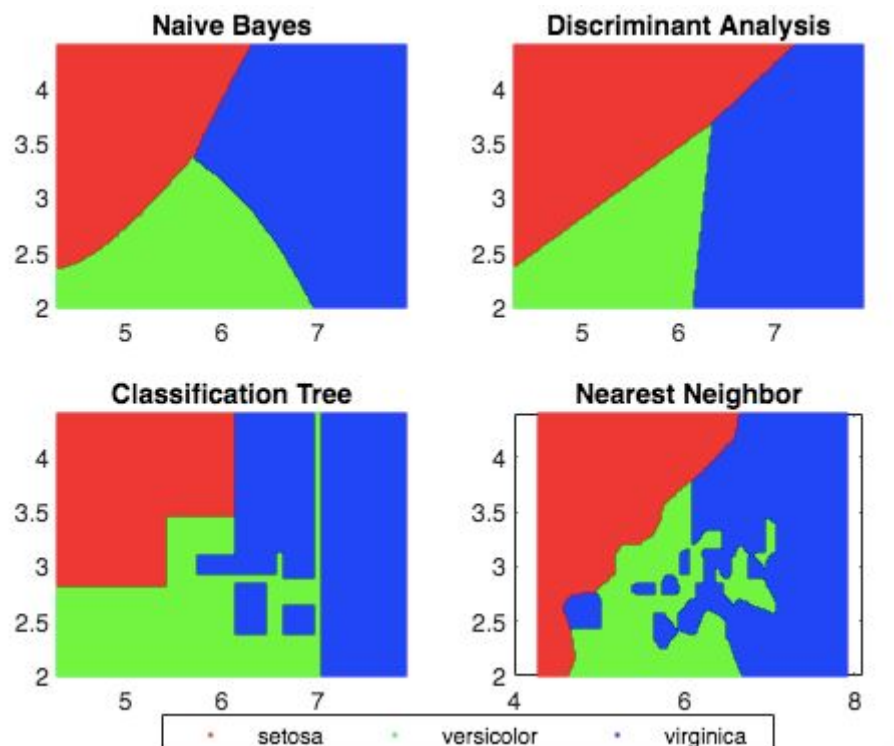


d. *k*-nearest neighbor classifier:

Étant donné un ensemble X de n points et une fonction de distance, la recherche de k -plus proche voisin (kNN) vous permet de trouver les k points les plus proches dans X d'un point d'interrogation ou d'un ensemble de points Y . La technique de recherche kNN et les algorithmes basés sur kNN sont largement utilisés comme règles d'apprentissage de référence. La simplicité relative de la technique de recherche kNN facilite la comparaison des résultats d'autres techniques de classification avec les résultats kNN.

e. visualization Decision Surfaces of Different Classifiers:

Chaque algorithme de classification génère différentes règles de prise de décision. Une surface de décision peut nous aider à visualiser ces règles.



5. Réseau Bayésiens:

a. Problème 1 : Diagnostics médicaux contradictoires

La résolution du Problème 1 a été faite sur MATLAB en utilisant le module Bayes Net Toolbox (BNT)

b. Problème 2 : Contrôles antidopage

on définit les éléments suivant

$$\begin{cases} D: \text{"Disqualifier"} \\ S: \text{"Saint"} \\ T_1: \text{"test 1 positif"} \\ T_2: \text{"test 2 positif"} \end{cases}$$

on a donc :

$$\begin{cases} P(S) = 0,9 \\ P(T_1) = 0,94 \\ P(T_2) = 0,9 \end{cases}$$

$$\begin{aligned} P(D) &= P(D \cap S) + P(D \cap \bar{S}) \\ &= P(S) \cdot P(D/S) + P(\bar{S}) \cdot P(D/\bar{S}) \end{aligned}$$

on a :

$$\begin{aligned} P(D/\bar{S}) &= P(T_1 \cap T_2) - P(T_1 \cap T_2) \\ &= P(T_1) + P(T_2) - P(T_1 \cap T_2) \end{aligned}$$

$$P(D/\bar{S}) = 0,994 \Rightarrow P(D/S) = 0,006$$

Donc

$$P(D) = 0,9 \cdot 0,006 + 0,994 \cdot 0,1$$

$$P(D) = 0,14 \%$$

6. Conclusions

Ce TP montre comment effectuer une classification avec l'outil MATLAB à l'aide des fonctions Statistiques et Machine Learning Toolbox ainsi que le classification learner App .

Le TP n'est pas censé être une analyse idéale des données. En fait, l'utilisation d'autres mesures peut conduire à une meilleure classification.

De plus, le TP ne vise pas à comparer les forces et les faiblesses de différents algorithmes de classification mais nous apprend à bien choisir notre modèle en fonction de ce que l'on cherche.

Il est peut-être instructif d'effectuer l'analyse sur d'autres ensembles de données et de comparer différents algorithmes. Il existe également des fonctions Toolbox qui implémentent d'autres algorithmes de classification.