S8_3_Méthodes supervisées

Table of contents

1 Introduction

Méthodes supervisées : visent à prédire une variable **quantitative** ou **catégorielle** à partir d'une collection de variables prédictrices (qui peuvent être de n'importe quel type : quantitatives, catégorielles, etc.).

Stratégie d'analyse en deux temps :

1. Construire le modèle prédicteur
2. Évaluer ses performances prédictives (*étape particulièrement importante en raison du risque de surapprentissage au cours de la construction du modèle*).

> **!** Important
>
> **OBJECTIF**
> **PRÉDIRE** une variable **QUANTITATIVE** ou **CATÉGORIELLE** à partir d'un **ensemble de variables explicatives**.
>
> **2 TEMPS**
> **Construction** du modèle → **Évaluation** des performances prédictives

2 Malédiction du surapprentissage

2.A Définition du surapprentissage

Surapprentissage = surajustement (*overfitting* en anglais) :

- Principal problème des méthodes de classification automatique utilisées en science des données.
- Modèle trop complexe : excellentes performances prédictives sur l'échantillon d'entraînement, mais réduites sur un autre échantillon.
- Apprentissage de la structure du bruit de l'échantillon d'entraînement au lieu de l'information réellement contenue dans l'échantillon.
- Mais va reproduire la structure originale de ce bruit sur tout nouvel échantillon, ce qui est voué à l'échec car la structure du nouveau bruit sera totalement différente.

2.B Paramètres et hyperparamètres

En gros : les paramètres englobent les paramètres + les hyperparamètres.

Paramètres : estimés automatiquement par l'algorithme d'apprentissage à partir de l'échantillon d'entraînement (par exemple : coefficients de régression dans un modèle linéaire, maximum de vraisemblance pénalisé, etc.).

Hyperparamètres : fixés *a priori* par le statisticien ou informaticien avant l'apprentissage (par exemple : choix de la famille de modèles, liste des variables prédictives, modalités de codage des variables, paramètres de la famille de modèles, etc.).

- Famille de modèles choisie dans la liste définie au préalable par le statisticien ou informaticien (exemples : forêt aléatoire, machine à vecteurs de support ou régression logistique).

- Paramètres de la famille de modèles choisie (exemples : nombre d'arbres de la forêt aléatoire, taille minimale des nœuds terminaux, nombre maximal de nœuds terminaux, nombre de variables tirées au sort et nombre d'observations tirées au sort pour construire chaque arbre, graine pseudo-aléatoire, etc.).
- Liste des variables prédictives fournies à l'algorithme.
- Diverses modalités de codage de ces variables (exemples : recodage d'une variable quantitative en catégorielle, regroupement de modalités de variables catégorielles, codage en contraste « 0 / 1 » ou autre).

2.C Méthodes de choix des hyperparamètres

3 façons de choisir un hyperparamètre :

1. Choix pré-planifié
2. Choix automatique guidé par les données (*data driven*)
3. Choix manuel guidé par les données

Exemple :

- On veut construire un modèle prédictif de complications post-opératoires (variable binaire du type : oui/non) sur des données de registre comprenant une centaine de variables explicatives préopératoires.
- On choisit de faire une régression logistique (estimée par la méthode du maximum de vraisemblance) et on identifie 30 variables explicatives potentiellement pertinentes.
- les variables présentant une association significative dans une régression logistique univariée seront intégrées dans le modèle logistique multivarié.
- Une variable qui perd sa significativité lorsqu'elle est intégrée dans le modèle multivarié restera quand même dans le modèle.
- Des tests d'interactions seront réalisés sur toutes les paires de variables afin de vérifier la validité du modèle final.

Les hyperparamètres peuvent donc être classés selon leur mode de choix :

1. *Hyperparamètres à choix pré-planifié* : famille de modèles de régression logistiques, estimateur du maximum de vraisemblance, liste des 30 variables candidates.
2. *Hyperparamètres à choix automatique guidé par les données* : choix des variables intégrées dans le modèle logistique multivarié (ce choix repose sur des régressions logistiques univariées devant conduire à des tests significatifs).
3. *Hyperparamètres à choix manuel guidé par les données* : test des 45 interactions possibles entre les 10 variables incluses dans le modèle.
   Les variables impliquées dans des interactions significatives seront intégrées dans le modèle final (risque donc de "surinterprétation" des résultats en leur donnant un sens clinique, et donc de les inclure dans le modèle final).

2.D Approches pour corriger le surapprentissage

2 corrections possibles du surapprentissage :

1. Correction palliative :

- ne vise pas à supprimer ni réduire le surapprentissage,
- seulement correction de l'évaluation des performances prédictives réelles du modèle afin qu'elles ne soient pas surestimées.

2. Correction préventive :

- par le choix d'hyperparamètres appropriés,
- vise à générer un modèle ayant un surapprentissage le plus modéré possible tout en évitant le sous-apprentissage.

En pratique :

- Le traitement palliatif est toujours nécessaire (sauf si nombre de paramètres et hyperparamètres très faibles par rapport à la taille de l'échantillon).
- **Il faut donc savoir quand faire un traitement PRÉVENTIF** :
  - pas utile si tous les hyperparamètres peuvent être choisis par choix pré-planifié (modèles simples avec peu de paramètres).
  - utile voire nécessaire si nombre de paramètres potentiel est élevé et phénomène de surapprentissage potentiellement important.

2.D.1 Approches palliatives pour corriger le surapprentissage

2.D.1.1 Échantillons d'entraînement et de test

Méthode la plus simple =

- Construire le modèle sur un échantillon d'entraînement.
- Évaluer les performances prédictives sur un échantillon test indépendant de l'échantillon d'entraînement.
- En ayant figé tous les hyperparamètres et paramètres du modèle sur l'échantillon d'entraînement avant de tester ses performances prédictives sur l'échantillon test.
- Donc les 2 phases (modélisation sur l'échantillon d'entraînement et évaluation des performances sur l'échantillon test) doivent être complètement séparées, sans aucun aller-retour possible entre les deux !
- Sinon : l'échantillon test se met à participer à l'apprentissage et le surapprentissage n'est plus complètement corrigé.
- Dans le pire des cas, les performances prédictives peuvent alors être surestimées de manière majeure.

Autodiscipline à mettre en place :

1. Créer 3 fichiers séparés : échantillon d'entraînement, variables prédictives de l'échantillon test, variable à prédire de l'échantillon test.
2. Étape de modélisation :

- Rédiger un plan de modélisation avec un script estimant les paramètres du modèle sur l'échantillon d'entraînement.
- Rédiger un rapport d'analyse précisant les choix des hyperparamètres retenus et tous les paramètres du modèle final obtenu.
  - Le modèle doit être unique à ce stade.
  - Si plusieurs modèles différents ont été évalués, le meilleur d'entre eux doit être retenu, sur la base de critères objectifs ou subjectifs.
- Une fois cette étape de modélisation finie, l'ensemble des paramètres du modèle est figé et il ne faut plus y revenir.

3. Étape de prédiction et d'évaluation des performances prédictives
   - Importer le fichier d'échantillon test avec les variables prédictives et la variable à prédire.
   - À l'aide des paramètres obtenus dans le rapport de modélisation ou dans le fichier d'exportation des paramètres, on calculera les prédictions du modèle.
4. Confronter les prédictions du modèle aux valeurs réelles de la variable à prédire grâce au troisième fichier.
5. Rédiger un rapport d'analyse des performances prédictives du modèle.

Pour constituer l'échantillon test, plusieurs options sont possibles :

- Séparer la BDD initiale en deux échantillons : entraînement et test (par exemple, respectivement 70% et 30% de l'échantillon initial).
  - On parle alors d'échantillon test **interne** car ce dernier est issu de la même population que l'échantillon d'entraînement (avec, de ce fait, des spécificités communes fortes liées, par exemple, aux critères de sélection ou aux instruments de mesures utilisés).
  - Cette méthode permet de corriger le surapprentissage lié au bruit aléatoire présent dans l'échantillon, mais ne corrige pas le surapprentissage lié aux spécificités très particulières de l'échantillon.
  - Il faudra alors recourir à un échantillon test **externe**, collecté sur une période différente, dans des centres différents, avec des pratiques différentes.
  - Problème : perte d'observations utilisables pour l'échantillon d'entraînement, baissant alors un peu la précision des estimations des paramètres et réduisant les performances prédictives du modèle.

2.D.1.2 Validation croisée

validation = test croisé

Consiste à réaliser de multiples séparations entre échantillon d'entraînement et échantillon test.

Exemple :

- proécudre de validation croisée à k parties (k-fold cross-validation) (c'est à dire k échantillons test différents, avec k \10 le plus souvent).
- l'échantillon initial est découpé en k parties disjointes de taille égales ou presque égales.
- modèle entrainé sur réunion de toutes les parties sauf la partie N°1, puis performances estimées sur la partie N°1.

- puis entrainé sur toutes les parties sauf la partie N°2, puis performances estimées sur la partie N°2.
- répété pour toutes les parties de sorte que chacune des k parties se retrouve une fois et une seule dans l'échantillon de test.
- le modèle final est en revanche entraîné sur l'échantillon complet !!!

Au mieux : *leave-one-out cross-validation* (k = n, où n est le nombre d'observations de l'échantillon complet).

- Autant de validation que d'observations dans l'échantillon initial.
- À chaque fois, on entraîne le modèle sur n - 1 observations et on teste sur l'observation laissée de côté.

Problèmes de la validation croisée

- Impossible si certains hyperparamètres sont soumis à un choix manuel guidé par les données (parce que tout choix d'hyperparamètre fait partie de l'entraînement et doit donc être totalement automatisé dans la procédure d'entraînement que l'on applique k fois dans la validation croisée à k parties).
- Risque =
  - fixer l'hyperparamètre comme s'il avait été soumis à un choix pré-planifié alors qu'il a été en réalité guidé par les données d'une manière non automatisable.
  - ou oublier d'intégrer un hyperparamètre à choix automatique guidé par les données dans la procédure automatique d'entraînement.

Pour que la validation croisée soit statistiquement valide, il faut

- une maîtrise de la distinction entre différents types de choix (pré-planifié, automatique guidé par les données et manuel guidé par les données).
- une maîtrise du *pipeline* de programmation de la procédure d'entraînement au sens large : **automatique**, **identique**, **répété k fois**, **sur un sous-échantillon différent à chaque fois**.

Autres problèmes liés à la méthode *leave-one-out et "n = 1 observation"*

1. On ne peut pas calculer toutes les statistiques de performances prédictives, mais seulement celles qui disposent d'un estimateur non biaisé sur un échantillon de n = 1 observation.

En gros : avec un jeu de données de taille n :

- on retire 1 seule observation (échantillon test) à chaque itération,
- on entraîne le modèle sur les n-1 autres,
- on prédit la valeur de cette seule observation mise de côté.
- On répète ça n fois (chacun joue une fois le rôle de test).

Certaines statistiques peuvent être calculées observation par observation, puis moyennées → ça marche.

Possibles : **moyenne de la valeur absolue de l'erreur** (différence entre valeur prédite et valeur réelle en valeur absolue) ou **erreur quadratique moyenne** (carré de la différence entre valeur prédite et valeur réelle, qui permet de pénaliser davantage les grosses erreurs et d'avoir un chiffre

toujours positif)

Impossibles : **AUC - ROC** (aire sous la courbe ROC) ou **sensibilité/spécificité** (nécessitent plusieurs points pour être définies).

2. On peut faire une courbe ROC en faisant une validation croisée à 10 parties (10-fold), en calculant l'aire sous la courbe ROC séparément sur chacune des parties omises (et elle n'est pas biaisée si elle est correctement effectuée).

- Ici : pas de leave-one-out, mais du k-fold, par exemple 10-fold.
  - Découpe les données en 10 blocs (à peu près de taille égale).
  - À chaque fois :
    - on entraîne le modèle sur 9 blocs,
    - on teste sur le 10 bloc.
  - Donc chaque bloc contient 1/10ème des observations : possible de calculer une courbe ROC sur ce bloc (plusieurs prédictions, plusieurs vrais positifs / vrais négatifs, on peut tracer la ROC et en tirer une AUC).
  - On obtient alors 10 valeurs d'AUC (une par bloc test), qu'on peut moyenner pour obtenir une AUC de validation croisée.
  - Il reste quand même un biais lié à la "graine" de découpage aléatoire en 10 blocs : chaque découpage aléatoire peut donner des résultats légèrement différents.
  - On peut réduire ce biais en répétant plusieurs fois la procédure (découpage aléatoire en 10 blocs, calcul de l'AUC moyenne) et en moyennant les AUC obtenues sur ces répétitions (ça s'appelle *repeated k-fold cross-validation*).

3. Bootstrap pour corriger le surapprentissage

- Il s'agit de considérer notre échantillon initial comme s'il s'agissait d'une population infinie.
- En gros : on a un seul échantillon, on fait comme si c'était une population, et on simule le fait de ré-échantillonner dedans.
- 1. Créer un échantillon bootstrapé avec n observations tirées au sort avec remise parmi les observations de l'échantillon initial
  - certains individus peuvent être tirés plusieurs fois, ce qui peut "déformer" l'échantillon bootstrapé par rapport à l'échantillon initial.
- 2. Entraîner le modèle sur cet échantillon bootstrapé (avec les mêmes contraintes d'automatisation que pour la validation croisée).
- 3. Calculer les statistiques de performance prédictive du modèle entraîné, sur l'échantillon bootstrapé ainsi que sur l'échantillon initial.
- En répétant l'expérience un grand nombre de fois (par exemple 1000 fois), il est possible de calculer le biais de surajustement comme la différence de performance moyenne entre les statistiques obtenues sur l'échantillon bootstrapé et l'échantillon initial.
  - $Perf^{(b)}_{\text{boot}}$ = Performance sur l'échantillon bootstrapé : tendance à être optimiste car le modèle a été entraîné dessus.
  - $Perf^{(b)}_{\text{orig}}$ = Performance sur l'échantillon initial : plus réaliste car le modèle n'a pas été entraîné dessus.

- Différence permet d'estimer le biais de sur-ajustement (over-optimism) : $Perf_{\text{boot}}^{(b)} - Perf_{\text{orig}}^{(b)}$ quantifie de combien les performances sur les données d'entraînement sont trop optimistes.

4. Difficulté pratique : séparation train/test
   - Le bootstrap et la validation croisée impliquent de programmer simultanément les étapes d'entraînement et de test.
   - Il faudrait, dans l'idéal, écrire le script entier en aveugle de la moindre donnée, après avoir bien défini quels hyperparamètres étaient à choix automatique guidé par les données et quels hyperparamètres étaient pré-définis.

2.D.2 Approches préventives pour réduire le surapprentissage

Méthodes similaires mais objectifs différents :

- Garantir le meilleur compromis entre surapprentissage et sous-apprentissage.
- Estimer correctement les performances prédictives.

Approches préventives :

- Pas utile en cas de surapprentissage modéré, notamment avec des modèles simples (exemple : régression logistique).
- Utile voire nécessaire pour les modèles les plus complexes (exemple : réseaux de neurones).

Prévention du surapprentissage :

- Choix d'hyperparamètres rendant le modèle le plus parcimonieux possible, par exemple en pénalisant les paramètres (régression lasso) ou en limitant le nombre total de paramètres utilisés (taille minimale des nœuds terminaux pour un arbre de classification).
- Objectif : limiter l'apprentissage artificiel du bruit de l'échantillon.
- Problème : hyperparamètres configurés pour une parcimonie extrême conduisent à un sous-apprentissage, avec un modèle capturant insuffisamment l'information contenue dans les données et donc, le plus souvent, à des performances prédictives médiocres.

Exemple de pipepline ;

- Premier jeu d'hyperparamètres choisi de manière arbitraire.
- Premier modèle entraîné sur l'échantillon d'entraînement
  - le surapprentissage sera potentiellement massif (le modèle apprend tout, y compris le bruit aléatoire).
  - Performances prédictives de ce modèle évaluées sur l'échantillon de validation, sans surapprentissage à ce stade (car l'échantillon de validation n'a pas servi à l'entraînement mais seulement à l'évaluation).
- Second temps : jeu d'hyperparamètres progressivement amélioré (on choisit un nouvel hyperparamètre),
  - en entraînant toujours les modèles correspondant sur l'échantillon d'entraînement,
  - en évaluant les performances prédictives sur l'échantillon de validation.

- Un jeu d'hyperparamètres optimal est finalement obtenu
  - avec, cependant, des performances prédictives largement surestimées, car un surapprentissage de l'échantillon de validation a été effectué via la sélection itérative des hyperparamètres (en effet, on a utilisé les données de validation pour choisir les hyperparamètres).
    - par ex si on trouvait une AUC à 0.75 avec un hyperparamètre, puis 0.80 avec un autre hyperparamètre, on va choisir le deuxième hyperparamètre donc on s'est servi des données de validation pour choisir l'hyperparamètre.
- La prévention du surapprentissage limite donc le surapprentissage sur l'échantillon d'entraînement (parce que les hyperparamètres sont choisis pour limiter le surapprentissage), mais en engendre un sur l'échantillon de validation (parce que les hyperparamètres sont choisis en fonction des performances sur l'échantillon de validation) et nécessite donc d'être combiné au traitement palliatif du surapprentissage avec le recours à un échantillon test.

1. Avant optimisation → hyperparamètres arbitraires → surapprentissage massif sur train
2. Après optimisation préventive → hyperparamètres ajustés → surapprentissage réduit sur train

MAIS surapprentissage créé sur validation

3. Donc → performances sur validation sont trop optimistes → il faut un troisième jeu complètement indépendant : test

Validation croisée préventive :

Le traitement préventif du surapprentissage peut utiliser les mêmes types d'échantillons que le traitement palliatif :

- validation interne (= échantillon de validation),
- validation externe (= échantillon test).
- validation croisée ou bootstrap (= échantillon de validation mais avec plusieurs séparations entre entraînement et validation).
  - validation croisée interne (= validation croisée sur l'échantillon de validation seulement, en ayant une partie train interne et une partie test interne à chaque itération de la validation croisée),
  - validation croisée externe (= découpage en blocs au niveau global, où chaque bloc sert à tester, à tour de rôle, un modèle réglé sans jamais utiliser les observations de ce bloc pour l'entraînement).

3 Analyse discriminante linéaire

Objectif : classifier des individus en fonction de variables explicatives. Par exemple, prédire le sexe en fonction des dimensions d'un os de mâchoire.

- On prend l'ensemble des combinaisons linéaires possibles des variables explicatives.
- On cherche la combinaison linéaire qui maximise la séparation entre les groupes (ici, hommes et femmes) = qui maximise la différence de moyennes (= **signal**)
- Tout en minimisant la variant à l'intérieur des groupes (ici, hommes et femmes) (= **bruit**).

- En faisant varier les coefficients de la combinaison linéaire.

$$Y = a_1 X_1 + a_2 X_2 + ... + a_p X_p$$

Avec Y = score construit à partir des variables explicatives $X_1, X_2, ..., X_p$ et des coefficients $a_1, a_2, ..., a_p$.

Dans la situation ou :

- 2 groupes (exemple : hommes et femmes)
- Les variables prédictrices suivent une loi multinormale dans chaque groupe (c'est à dire que dans chaque groupe, les caractéristiques se distribuent selon une loi multinormale.)

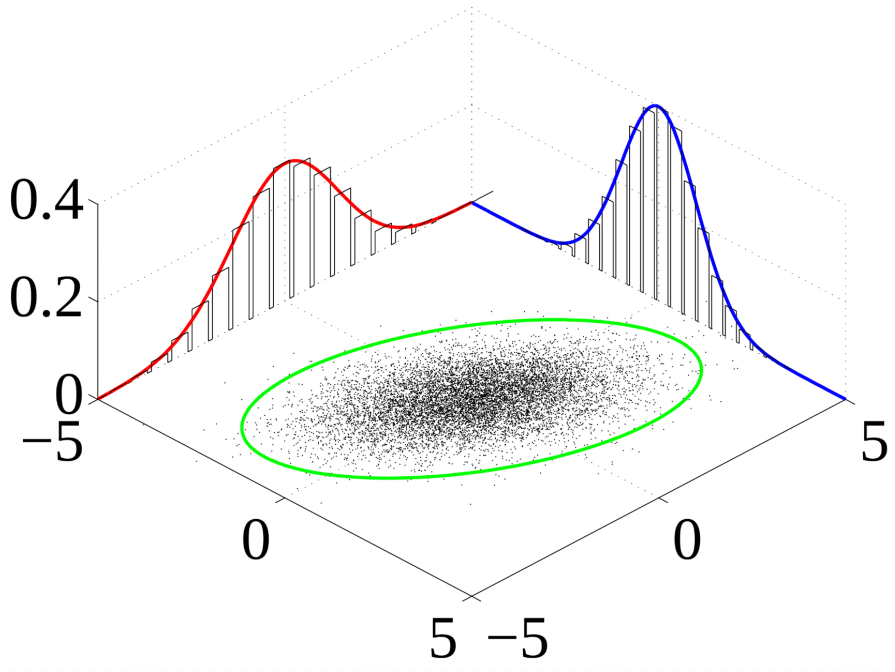


Figure 1: Représentation d'une loi normale multivariée.
Les courbes rouge et bleue représentent les lois marginales.
Les points noirs sont des réalisations de cette distribution à plusieurs variables.

-
- Les matrices variance-covariance sont identiques entre les groupes (c'est à dire que la dispersion et les corrélations entre variables sont les mêmes dans chaque groupe :
  - donc en gros les deux nuages de point ont la même forme (même ellipse) dans chacun des groupes, mais des centres différents (moyennes différentes).)
- Dans ce cas : (loi multinormale, même matrice variance-covariance) :
  - La combinaison linéaire optimale est obtenue en résolvant un problème aux valeurs propres (c'est à dire en trouvant les coefficients $a_1, a_2, ..., a_p$ qui maximisent le rapport signal/bruit).
  - La frontière de décision entre les deux groupes est une droite (d'où le terme "linéaire" dans "analyse discriminante linéaire").

  - La classification des individus se fait en fonction de quel côté de la droite ils se trouvent.
  - et c'est la méthode de classification optimale, à condition que l'échantillon soit représentatif de la prévalence en population réelle (selon la loi de Bayes = la probabilité a priori d'appartenance à chaque groupe doit être prise en compte dans la classification finale).

Quand il n'y a que 2 groupes à discriminer (c'est à dire une classification binaire type homme vs femmes) :

- La variable Y à expliquer prend 2 valeurs (par exemple Y=1 pour homme ou Y=0 pour femme).
- Les variables explicatives $X_1, X_2, ..., X_p$ (par exemple tailles, poids, mesures osseuses…).
- L'objectif : utiliser $X_1, X_2, ..., X_p$ pour deviner la valeur de Y en construisant une fonction discriminante linéaire.

Dans ce cas : la fonction discriminante (qui permet de classer les individus) est une combinaison linéaire des variables explicatives et peut être représenté sous la forme d'un **hyperplan**.

> **Note**
>
> NB : un hyperplan est un espace de dimension n-1 le nombre de variable explicatives (n) :
>
> - Si on a 2 variables explicatives, l'hyperplan est une droite (dimension 1 dans un plan de dimension 2).
> - Si on a 3 variables explicatives, l'hyperplan est un plan (dimension 2 dans un espace de dimension 3).
> - En général, dans un espace de dimension n, un hyperplan est un sous-espace affine de dimension n-1.

L'hyperplan peut être obtenu par l'estimation d'un modèle de régression linéaire où Y (codée 0/1) est expliquée par les variables prédictrices $X_1, X_2, ..., X_p$.

$$Y = a_0 + a_1 X_1 + \cdots + a_p X_p + \varepsilon$$

C'est un modèle de régression linéaire standard, avec :

- a_0 : l'ordonnée à l'origine (intercept = valeur de Y quand toutes les X_i sont nulles)
- $a_1, \ldots, a_p$: les coefficients attachés à chaque variable
- ε : le terme d'erreur (ce qui n'est pas expliqué par les X_i)

Le seuil permettant d'attribuer 0 ou 1 à Y à partir du score dépend de la proportion de sujets pour lesquels Y vaut 1 ou 0.

Le seuil doit être déterminé en fonction 1/ des différences de proportions observées sur le modèle linéaire et 2/ de la prévalence des groupes dans la population réelle.

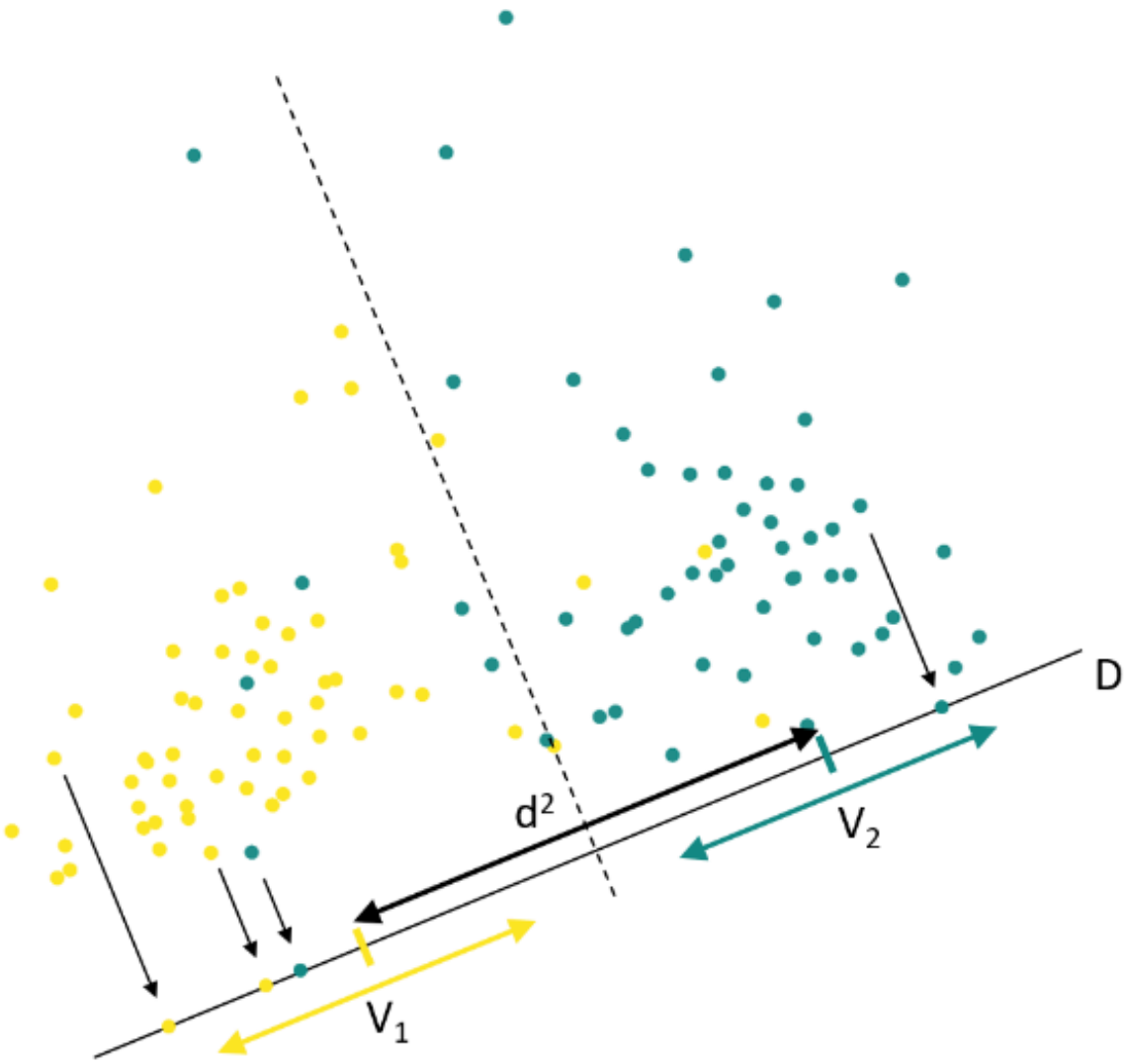


Figure 2: Analyse discriminante linéaire selon Fisher dans le cas simple où il y a 2 groupes (points jaunes et points verts) et 2 variables (le problème est donc plan).

Les points sont projetés sur la droite D.
V1 est la variance des points projetés du groupe 1,
V2 la variance des points projetés du groupe 2,
d2 est le carré de la distance entre la projection du barycentre des points du groupe 1 et la projection du barycentre des points du groupe 2.

La droite D est considérée comme optimale quand d2/(V1+V2) est maximum. La droite discriminante, ici en pointillés, est orthogonale à D.

3.A Exemple 1 en R

- Fichier `vox`: contient des données acoustiques de voix humaines de 244 patients déprimés hospitalisés.

- Objectif : prédire le sexe des patients (homme/femme) à partir de caractéristiques acoustiques de leur voix.
  - Mesures :
    * hauteur de la voix (fréquence fondamentale moyenne `moyf0`)
    * fluctuations de cette hauteur de voix (écart-type de f0 : `sdf0`)

Mise en forme des données :

1. création d'un nouveau `df` appelé `vox.sex` avec les variables d'intérêt (`sexe`, `moyf0`, `sdf0`) et suppression des lignes contenant des valeurs manquantes.
2. division des données en échantillon d'entraînement et échantillon de test (50% des données dans chaque échantillon, tirage aléatoire avec `set.seed(2)` pour reproductibilité).
   - la fonction `sample()` permet de tirer au sort les indices des lignes à inclure dans l'échantillon d'entraînement.
   - `1:nrow(vox.sex)` génère une séquence de tous les indices de lignes du data frame `vox.sex`.
   - `as.integer(nrow(vox.sex)/2)` calcule la moitié du nombre total de lignes, convertie en entier.
3. création des échantillons d'entraînement (`v.train`) et de test (`v.test`) en utilisant les indices tirés au sort.
   - `v.train` contient les lignes correspondant aux indices dans `training`.
   - `v.test` contient les lignes restantes (celles qui ne sont pas dans `training`) car utilisation de l'opérateur négatif `-`.

```
# création d'un nouveau data frame avec les variables d'intérêt et suppression des
 ↪  lignes contenant des valeurs manquantes
vox.sex <- na.omit(data.frame(sexe=vox$sexe, moyf0=vox$moyf0, sdf0=vox$sdf0))

# division des données en échantillon d'entraînement et échantillon de test
set.seed(2)
training <- sample(1:nrow(vox.sex), as.integer(nrow(vox.sex)/2))

# échantillon d'entraînement et échantillon de test
v.train <- vox.sex[training, ]
v.test <- vox.sex[-training, ]
```

L'échantillon initial de 224 sujets sans NA est séparé en deux échantillons de 112 sujets chacun : un échantillon d'entraînement (`v.train`) et un échantillon de test (`v.test`).

** Construction du modèle d'analyse discriminante linéaire sur l'échantillon d'entraînement `v.train` :**

1. Charger la bibliothèque `MASS` qui contient la fonction `lda()` pour l'analyse discriminante linéaire.
2. Utiliser la fonction `lda()` pour construire le modèle discriminant linéaire.

- La formule `sexe ~ moyf0 + sdf0` indique que l'on cherche à prédire la variable `sexe` en fonction des variables explicatives `moyf0` et `sdf0`.
- L'argument `data = v.train` spécifie que le modèle doit être entraîné sur l'échantillon d'entraînement.

3. Stocker le modèle entraîné dans l'objet `discrim`.
4. Évaluer les performances prédictives du modèle sur l'échantillon de test `v.test` :
   - Utiliser la fonction `predict()` pour obtenir les prédictions du modèle sur l'échantillon de test.
   - Extraire la classe prédite avec `$class`.
   - Comparer les classes prédites aux classes réelles (`v.test$sexe`) en utilisant la fonction `table()` pour créer une matrice de confusion.

```
discrim <- lda(sexe ~ moyf0 + sdf0, data = v.train)
table(predict(discrim, v.test)$class, v.test$sexe)
```

```
   1  2
1 45  7
2  1 59
```

La matrice de confusion s'interprète :

- Les lignes correspondent aux classes prédites par le modèle (1 = homme, 2 = femme).
- Les colonnes correspondent aux classes réelles dans l'échantillon de test.

Résultats :

- Classification correcte de 104 sujets sur 112 (45 + 59).
- Taux de classification correcte = 104 / 112 92.9%.

Pour obtenir la fonction discriminante linéaire (coefficients des variables explicatives) :

$$Y = a + b\,\text{moyf0} + c\,\text{sdf0}$$

L'obtention de l'expression serait utile pour connaître l'importance relative de chaque variable dans la classification.

Et permettrait aussi de tracer la frontière de décision dans le plan défini par `moyf0` et `sdf0` dont l'équation de la droite est :

$$\text{sdf0} = -\frac{a}{c} - \frac{b}{c}\,\text{moyf0}$$

```r
# obtention des coefficients de la fonction discriminante linéaire
bdisc <- discrim$scaling[1] #discrim$scaling donne les coefficients
cdisc <- discrim$scaling[2]

# calcul des moyennes des variables explicatives dans l'échantillon d'entraînement
 ↪  pour obtenir l'ordonnée à l'origine (a)
m.moyf0 <- mean(v.train$moyf0)
m.sdf0 <- mean(v.train$sdf0)

# calcul de l'ordonnée à l'origine a de la fonction discriminante linéaire (sert à
 ↪  tracer la droite de décision car done l'intercept)
adisc <- -(m.moyf0*bdisc+m.sdf0*cdisc)

# calcul de l'ordonnée et du coefficient directeur de la droite de décision
-c(adisc/cdisc, bdisc/cdisc)
```

```
[1] 144.4782984  -0.7649609
```

Représentation du modèle de régression linéaire $\text{sexe} = a + b\,\text{moyf0} + c\,\text{sdf0}$:

```r
rlin <- lm(sexe~moyf0+sdf0, data=v.train)
arlin <- coef(rlin)[1] - mean(v.train$sexe)
brlin <- coef(rlin)["moyf0"]
crlin <- coef(rlin)["sdf0"]
-c(arlin/crlin, brlin/crlin)
```

```
(Intercept)       moyf0
144.4782984  -0.7649609
```

En gros : l'analyse discriminante linéaire et la régression linéaire donnent la même droite de décision dans le cas binaire (2 groupes à discriminer).

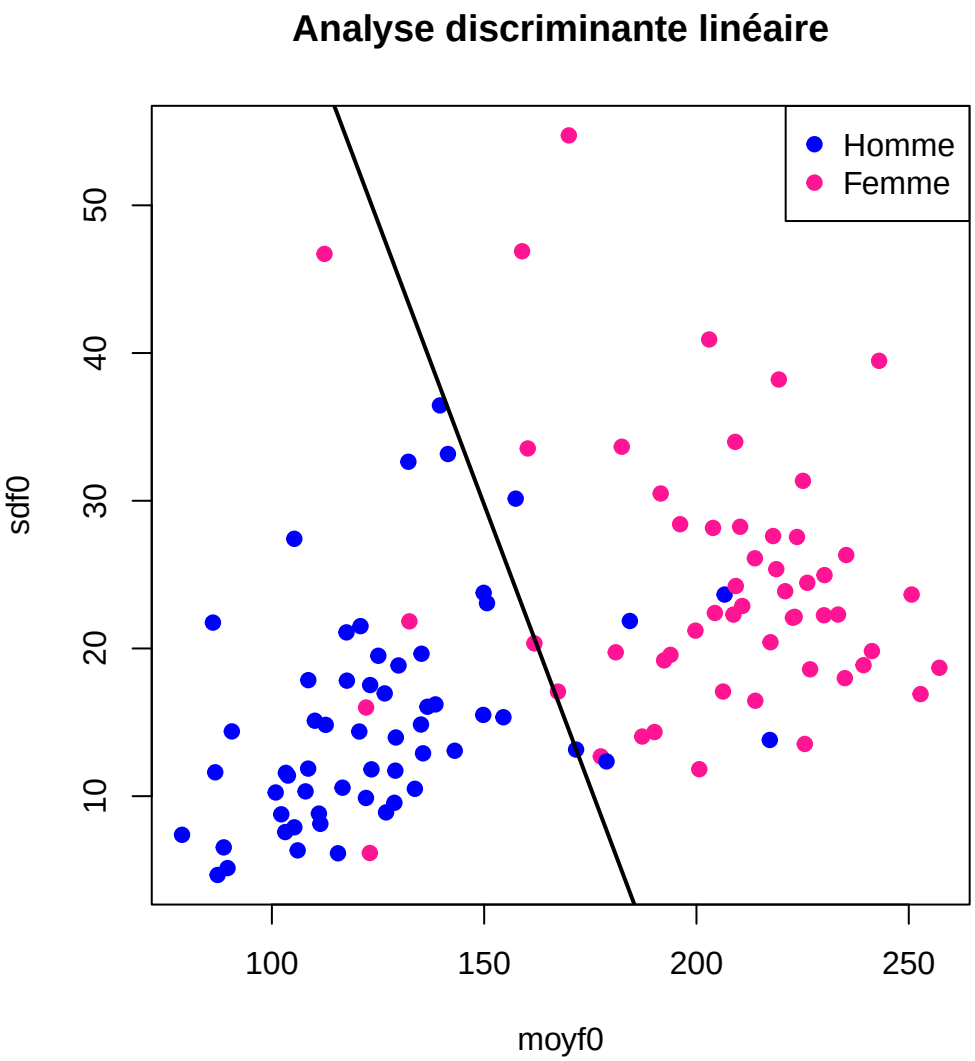
On a quand même fait une régression linéaire multiple même si Y étant binaire, la normalité des résidus n'est pas respectée.

Mais en fait : aucun test statistique n'a été fait, et aussi parce que la correspondance suggère que la régression linéaire multiple peut être utilisée pour obtenir la fonction discriminante linéaire dans le cas binaire.

Représentation graphique de l'analyse discriminante linéaire :

```r
plot(
    v.train$moyf0,
    v.train$sdf0,
    col=ifelse(v.train$sexe==1, "blue", "deeppink"),
    xlab="moyf0", ylab="sdf0",
    pch=19,
    main="Analyse discriminante linéaire")
```

```r
#droite de décision
abline(-adisc/cdisc, -bdisc/cdisc, col="black", lwd=2)

# légende
legend(
    "topright",
    legend=c("Homme", "Femme"),
    col=c("blue", "deeppink"),
    pch=19)
```



3.B Exemple 2 en R

Pour prédire une variable catégorielle avec plus de 2 modalités (classification multiple) :

A partir du fichier `abd` qui recueille des signes et symptômes cliniques de patients souffrant de douleurs abdominales aux urgences.

L'objectif est de prédire la cause principale de la douleur abdominale (appendicite, colique néphrétique, cholécystite, occlusion) à partir des autres variables cliniques (toutes binaires).

```
head(abd)
```

```
  hypoc.droit epigastre hypoc.gauche fid fig para.ombilic irrad.lomb
1           1         1            1   1   1            1          0
2           0         0            0   0   1            0          0
3           0         0            1   0   1            0          0
4           1         1            0   0   0            0          1
5           0         0            0   1   0            0          0
6           0         0            0   1   0            1          0
  irrad.descend arret.mat arret.gaz pollakiurie hematurie atcd.chir temperature
1             1         0         0           0         0         0           1
2             0         1         0           1         0         1           0
3             0         0         0           0         0         1           0
4             0         1         0           0         0         1           1
5             0         0         0           0         0         0           0
6             0         0         1           0         0         1           0
  agitation arret.respabd meteorisme defense murphy contracture tympanisme
1         0             0          0       0      0           0          0
2         1             0          0       0      0           0          0
3         0             0          0       0      0           0          0
4         0             1          0       0      1           0          0
5         0             0          0       0      0           0          0
6         0             0          1       0      0           0          1
  bruits.hyd.aer tr.droite tr.douglas   diagnostic
1              1         0          1  appendicite
2              0         1          1 colique.neph
3              0         0          0 colique.neph
4              1         0          0 cholecystite
5              1         1          0  appendicite
6              1         0          0    occlusion
```

1. Création des échantillons d'entraînement et de test (2/3 des observations dans l'échantillon d'entraînement, 1/3 dans l'échantillon de test).

```
set.seed(1)

# création d'un vecteur nabd contenant le nombre de lignes du data frame abd
nabd <- nrow(abd)

# création d'un échantillon d'entraînement contenant 2/3 des observations
training <- sample(1:nabd, floor(nabd*2/3))
# création des échantillons d'entraînement et de test
abd.train <- abd[training,]
abd.test <- abd[-training,]
```

2. Construction du modèle d'analyse discriminante linéaire sur l'échantillon d'entraînement.

```
# construction du modèle d'analyse discriminante linéaire sur l'échantillon
 ↪  d'entraînement.
# le fait de mettre diagnostic~. signifie que l'on utilise toutes les variables
 ↪  sauf diagnostic pour prédire diagnostic
abd.disc <- lda(diagnostic~., data=abd.train)
```

3. Évaluation des performances prédictives du modèle sur l'échantillon de test.

```
# évaluation des performances prédictives du modèle sur l'échantillon de test et
 ↪  affichage du tableau
tbl.abd <- table(predict(abd.disc, abd.test)$class,
abd.test$diagnostic)
tbl.abd
```

	appendicite	autre	cholecystite	colique.neph	occlusion	peritonite
appendicite	387	141	11	3	6	9
autre	80	187	24	11	24	11
cholecystite	6	51	147	5	1	10
colique.neph	4	16	2	41	2	1
occlusion	7	14	8	4	123	14
peritonite	17	3	4	0	5	41

4. Calcul du taux de classification correcte.

```
# sum(diag(tbl.abd))/sum(tbl.abd) permet de calculer le taux de classification
 ↪  correcte
sum(diag(tbl.abd))/sum(tbl.abd)
```

```
[1] 0.6521127
```

Le pouvoir prédictif est assez médiocre : seulement 65% des patients de l'échantillon de test sont correctement classés

5. Affichage des probabilités a posteriori d'appartenance à chaque classe pour les premières observations de l'échantillon de test.

```
# head(round(predict(abd.disc, abd.test)$posterior,3)) permet d'afficher les
 ↪  probabilités a posteriori d'appartenance à chaque classe pour les premières
 ↪  observations de l'échantillon de test
head(round(predict(abd.disc, abd.test)$posterior,3))
```

```
   appendicite autre cholecystite colique.neph occlusion peritonite
1        0.291 0.638        0.003        0.021     0.003      0.044
2        0.003 0.091        0.000        0.904     0.002      0.000
8        0.001 0.011        0.988        0.000     0.000      0.001
18       0.307 0.629        0.002        0.017     0.044      0.001
20       0.430 0.565        0.001        0.003     0.001      0.000
21       0.028 0.395        0.001        0.026     0.550      0.000
```

On peut calculer, pour chaque patient, les probabilités de chacune des pathologies telles qu'estimées par le modèle.

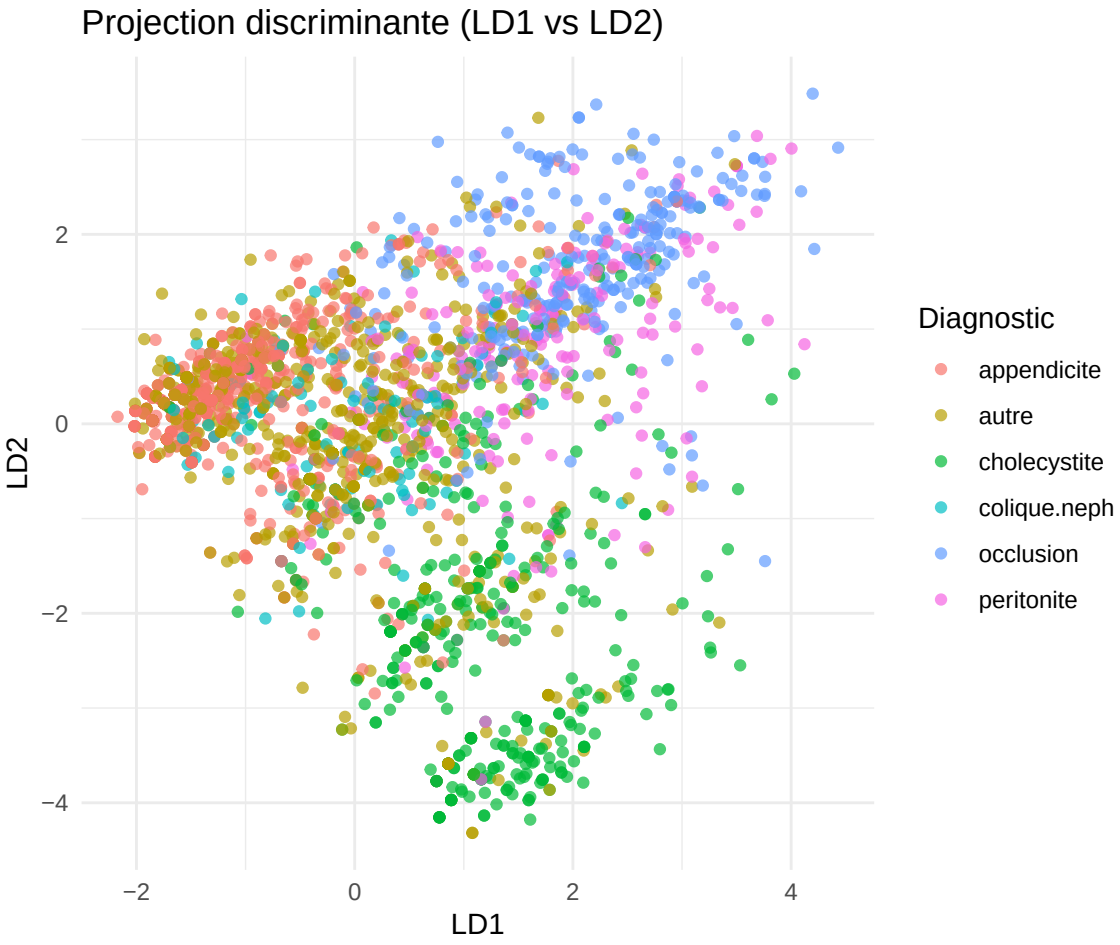
> **!** Important
>
> Attention, ces probabilités sont estimées selon une hypothèse de multinormalité des variables explicatives dans chaque groupe et d'égalité des matrices variance-covariance entre les groupes, ce qui est rarement le cas en pratique !!

Formule totale :

```r
#set.seed : pour reproductibilité du tirage aléatoire
set.seed(1)

# création d'un vecteur nabd contenant le nombre de lignes du data frame abd
nabd <- nrow(abd)

# création d'un échantillon d'entraînement contenant 2/3 des observations
training <- sample(1:nabd, floor(nabd*2/3))

# création des échantillons d'entraînement et de test
abd.train <- abd[training,]
abd.test <- abd[-training,]

# construction du modèle d'analyse discriminante linéaire sur l'échantillon
 ↪  d'entraînement.
# le fait de mettre diagnostic~. signifie que l'on utilise toutes les variables
 ↪  sauf diagnostic pour prédire diagnostic
abd.disc <- lda(diagnostic~., data=abd.train)

# évaluation des performances prédictives du modèle sur l'échantillon de test et
 ↪  affichage du tableau
tbl.abd <- table(predict(abd.disc, abd.test)$class,
abd.test$diagnostic)
tbl.abd

# sum(diag(tbl.abd))/sum(tbl.abd) permet de calculer le taux de classification
 ↪  correcte
sum(diag(tbl.abd))/sum(tbl.abd)
```

```r
# head(round(predict(abd.disc, abd.test)$posterior,3)) permet d'afficher les
 ↪  probabilités a posteriori d'appartenance à chaque classe pour les premières
 ↪  observations de l'échantillon de test
head(round(predict(abd.disc, abd.test)$posterior,3))
```

Représentation graphique en (en 3D dans le fichier HTML) :



4 Régression logistique

Objectif : modéliser la relation entre une variable dépendante binaire (Y = 0 ou 1) et un ensemble de variables explicatives (X1, X2, ..., Xp).

$$Y = a_0 + a_1X_1 + a_2X_2 + ... + a_pX_p + \varepsilon$$

Y étant binaire, il est plus classique d'utiliser un modèle logistique :

$$\text{logit}(P(Y = 1)) = a_0 + a_1X_1 + a_2X_2 + ... + a_pX_p$$

Pour chaque sujet, le score $a_0 + a_1X_1 + a_2X_2 + ... + a_pX_p$ permet d'estimer la probabilité que Y = 1.

Selon la valeur supérieure ou inférieure à 0.5, on peut classer le sujet dans la catégorie Y=1 ou Y=0.

Quand le nombre d'observations disponibles est suffisant, la régression logistique est un des meilleurs classifieurs pour variable binaire !

4.A Exemple en R

A partir du fichier `vox` qui contient des données acoustiques de voix humaines de 244 patients déprimés hospitalisés.

L'objectif est de prédire le sexe des patients (homme/femme) à partir de caractéristiques acoustiques de leur voix.

Formule : `rlog <- glm(2-sexe~moyf0+sdf0, data=v.train, family=binomial)`

- `rlog` : contiendra le modèle de régression logistique entraîné sur l'échantillon d'entraînement `v.train`.
- il faut écrire `glm(2-sexe)` car la variable `sexe` vaut 1 pour homme et 2 pour femme dans le df, or on veut modéliser la probabilité d'être une femme (Y=1) donc on fait 2-sexe pour que Y=1 corresponde aux femmes.
- `moyf0` et `sdf0` sont les variables explicatives.
- `data=v.train` indique que le modèle doit être entraîné sur l'échantillon d'entraînement.
- `family=binomial` spécifie que l'on utilise une régression logistique (car variable dépendante binaire).
- `ifelse(predict(rlog, v.test, type="response")>0.5, 2, 1)` : permet de prédire les classes (1 ou 2) pour chaque observation de l'échantillon de test `v.test` en utilisant le modèle `rlog`.
  - `predict(rlog, v.test, type="response")` calcule les probabilités prédites d'être une femme (Y=1) pour chaque observation de `v.test`.
  - `>0.5` compare ces probabilités au seuil de 0.5 pour décider si l'observation est classée comme femme (2) ou homme (1).

```
rlog <- glm(2-sexe~moyf0+sdf0, data=v.train, family=binomial)
summary(rlog)
```

```
Call:
glm(formula = 2 - sexe ~ moyf0 + sdf0, family = binomial, data = v.train)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 10.89261    1.92617   5.655 1.56e-08 ***
moyf0       -0.05531    0.01014  -5.452 4.97e-08 ***
sdf0        -0.08531    0.03725  -2.290    0.022 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 155.122  on 111  degrees of freedom
Residual deviance:  59.316  on 109  degrees of freedom
AIC: 65.316

Number of Fisher Scoring iterations: 6
```

```
table(ifelse(predict(rlog, v.test, type="response")>0.5, 2, 1), v.test$sexe)
```

```
    1  2
  1  1 59
  2 45  7
```

Résultats :

- Classification correcte de 104 sujets sur 112 (45 + 59).
- Taux de classification correcte = 104 / 112 92.9%.
- Performances prédictives identiques à celles de l'analyse discriminante linéaire dans cet exemple particulier.

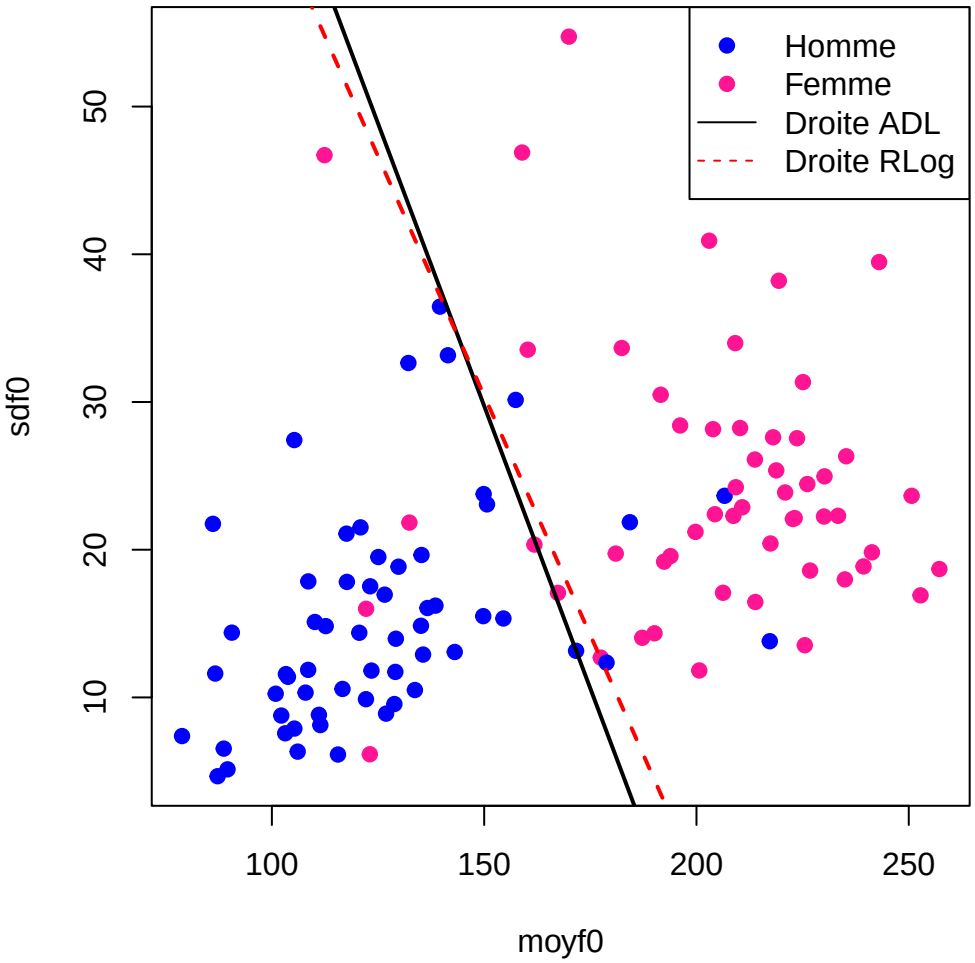
A partir de cet exemple, on peut tracer la droite discriminante logistique :
$\text{sdf0} = -\frac{a'}{c'} - \frac{b'}{c'}\,\text{moyf0}$

```
arlog <- rlog$coefficients["(Intercept)"] #coefficient [1] dans summary(rlog)
brlog <- rlog$coefficients["moyf0"] #coefficient [2]
crlog <- rlog$coefficients["sdf0"] #coefficient [3]
-c(arlog/crlog, brlog/crlog)
```

```
(Intercept)       moyf0
127.6831343  -0.6483722
```

Représentation graphique sur le même grphique de la regression logistique et de l'analyse discriminante linéaire :

```
plot(
    v.train$moyf0,
    v.train$sdf0,
    col=ifelse(v.train$sexe==1, "blue", "deeppink"),
    xlab="moyf0", ylab="sdf0",
    pch=19,
    main="Analyse discriminante linéaire et régression logistique")
#droite de décision de l'analyse discriminante linéaire
abline(-adisc/cdisc, -bdisc/cdisc, col="black", lwd=2)
#droite de décision de la régression logistique
abline(-arlog/crlog, -brlog/crlog, col="red", lwd=2, lty=2)
# légende
legend(
    "topright",
    legend=c("Homme", "Femme", "Droite ADL", "Droite RLog"),
    col=c("blue", "deeppink", "black", "red"),
    pch=c(19, 19, NA, NA),
    lty=c(NA, NA, 1, 2))
```



En conclusion : la régression logistique dichotomisée et l'analyse discriminante linéaire sont très proches et donnent de bons résultats en termes de classification.

Mais quelques points importants à considérer :

- Les deux méthodes reposent sur des hypothèses différentes (multinormalité et égalité des matrices variance-covariance pour l'ADL, indépendance des observations et linéarité du logit pour la régression logistique).
- Les performances prédictives peuvent différer selon la nature des données et le respect ou non des hypothèses.
- Les erreurs de classement ne sont pas cliniquement équivalentes ni statistiquement équiprobables dans le contexte médical (c'est à dire que classer un malade comme non-malade n'a pas les mêmes conséquences que l'inverse).

Pour une maladie rare, le seuil de probabilité de 0,50 peut ne jamais être atteint (du fait que la maladie est rare donc la probabilité a priori est faible), ce qui peut conduire à classer systématiquement tous les sujets comme non-malades.

On peut alors ajuster le seuil de probabilité en fonction des conséquences cliniques des erreurs de classification, et de la probabilité a priori de la maladie dans la population cible (basée sur la

prévalence, dans la littérature).

5 Machine à vecteurs de support (SVM)

SVM pour Support Vector Machine :

- Repose sur les principes de l'analyse discriminante linéaire.
- Mais les dépasse avec une plus grande complexité d'utilisation.
- Trois étapes pour la présenter :

1. Dans un jeu de données où les groupes sont parfaitement séparables par une droite (ou un hyperplan dans un espace de dimension n) : l'objectif est de discriminer les deux groupes
    - Il existe une infinité de droites (ou hyperplans) séparant parfaitement les deux groupes.
    - Pour obtenir la droite (ou hyperplan selon le nombre de variables explicatives) optimale, on choisit celle qui maximise la distance (marge) entre la droite et les points les plus proches de chaque groupe .

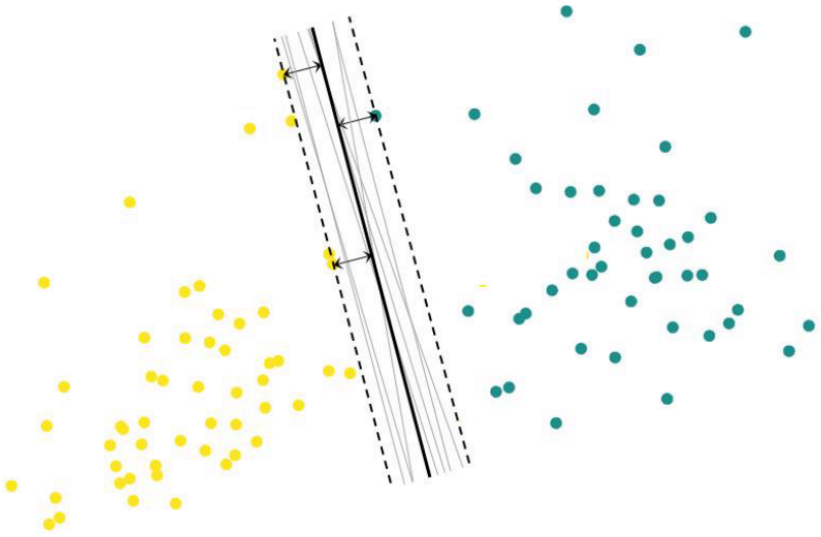


Fig. 3.4 — *Deux groupes de points sont parfaitement séparables par une infinité de droites discriminantes. Celle proposée par le SVM maximise l'écart minimal avec les points des deux groupes. Cette solution semble naturelle, elle est par ailleurs susceptible d'avoir de meilleures performances sur un échantillon de validation.*

2. En pratique, il est peu probable que les groupes soient parfaitement séparables par une droite (ou hyperplan).
    - Pour surmonter cette difficulté, on procède à la deuxième étape : changement de variables = **Kernel trick** :
        - On transforme les variables explicatives initiales en de nouvelles variables (par exemple, en utilisant des fonctions polynomiales ou des fonctions log : X devient $\log(X)$).
        - Dans ce nouvel espace de variables, il est plus probable que les groupes soient séparables, et cette approche permet de recourir à des fonctions discriminantes courbes et non plus linéaires.

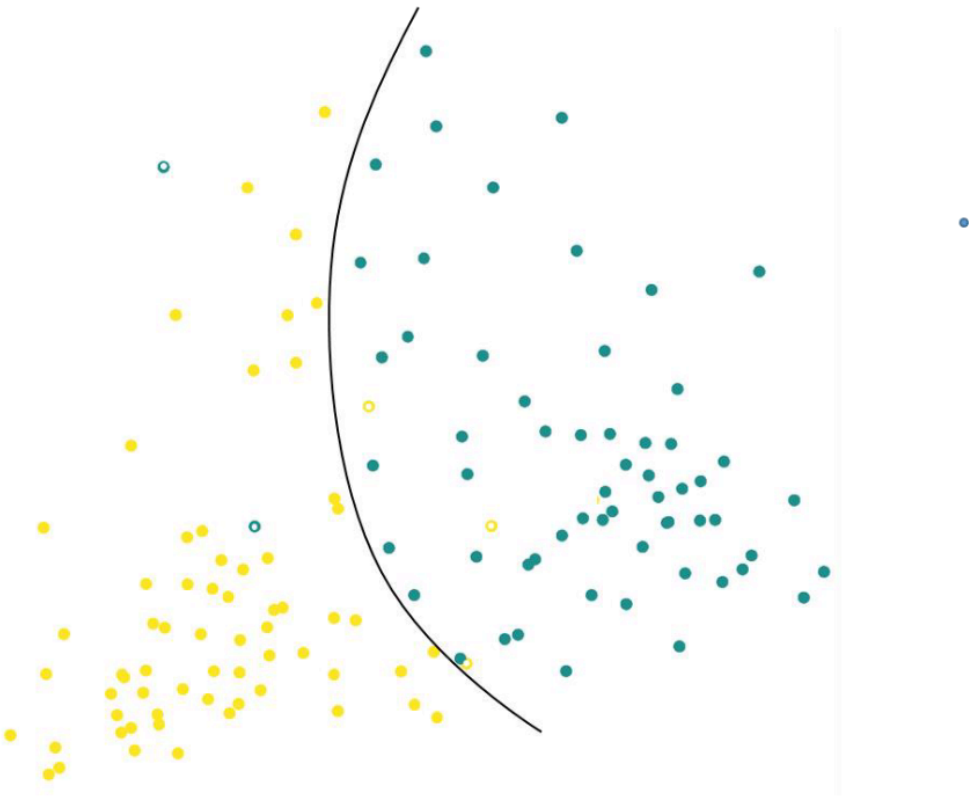


Fig. 3.5 — *SVM avec noyau non linéaire (la fonction discriminante n'est plus une droite) autorisant des observations mal classées (ici les 5 points colorés évidés). Ces observations sont intégrées dans l'algorithme sous forme d'une pénalité pondérée par un hyperparamètre que le statisticien devra choisir avec prudence.*

3. Si le *Kernel trick* ne permet pas de séparer parfaitement les groupes, on peut autoriser quelques erreurs de classification (points mal classés de chaque côté de la droite ou hyperplan).

- 3e étape : autoriser certaines observations à ne pas être correctement prédites
- Le compromis se fait en fonction d'un hyperparamètre correspondant au "coût" d'une erreur de classification
  - Si le coût est élevé : l'aglorithme privilégie une courbe sinueuse séparant bien les groupes sur les données d'apprentissage, mais avec des performances médiocres sur les données de validation
  - Si le coût est faible, la courbe discriminante est simple, avec de bonnes performances sur de nouvelles observations.

5.A Exemple en R

5.A.1 SVM linéaire

On utilise la fonction `svm()` du package `e1071` pour entraîner un modèle SVM.

```
svmlin <- svm(factor(sexe) ~ moyf0 + sdf0, data = v.train, kernel = "linear", cost
```

= 1, scale=FALSE)

- `factor(sexe)` : indique que la variable dépendante `sexe` est catégorielle (facteur).
- `moyf0 + sdf0` : variables explicatives utilisées pour prédire `sexe`.
- `data = v.train` : spécifie que le modèle doit être entraîné sur l'échantillon d'entraînement `v.train`.
- `kernel = "linear"` : indique que l'on utilise un noyau linéaire (droite de séparation).
- `cost = 1` : hyperparamètre contrôlant le compromis entre la complexité du modèle et les erreurs de classification.
  - c'est la valeur par défaut, mais dans le cas linéaire le choix du coût a peu d'impact (c'est surtout si on fait un truc curviligne)
- `scale=FALSE` : indique que les variables explicatives ne doivent pas être mises à l'échelle (normalisées).
  - Par défaut, `svm()` met les variables à l'échelle, mais ici on choisit de ne pas le faire.
  - comme ça on pourra comparer avec les autres méthodes (ADL, régression logistique) qui n'ont pas mis les variables à l'échelle.

```
svmlin <- svm(factor(sexe) ~ moyf0 + sdf0, data = v.train, kernel = "linear", cost
 ↪  = 1, scale=FALSE)
table(predict(svmlin, v.test), v.test$sexe)
```

```
   1  2
1 45  7
2  1 59
```

Résultats :

- Classification correcte de 104 sujets sur 112 (45 + 59).
- Idem que ADL et régression logistique dans cet exemple particulier.

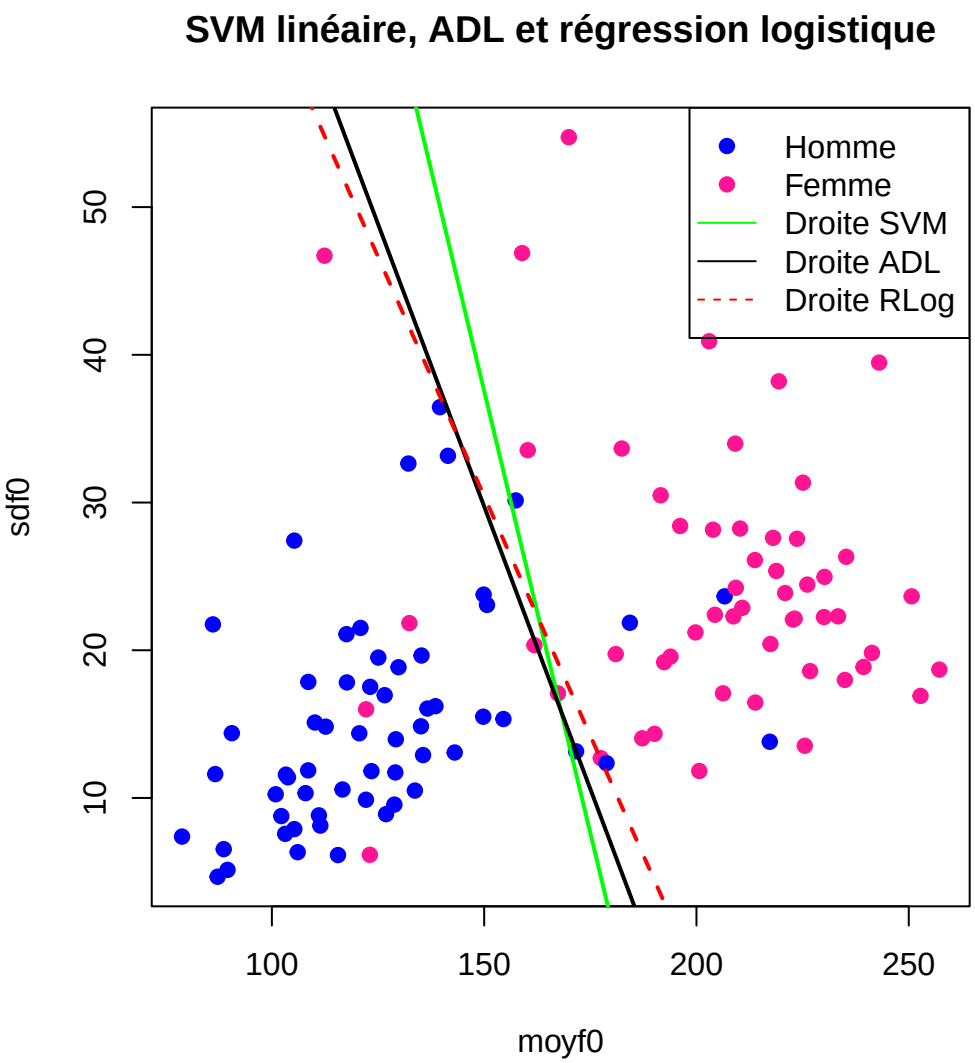
Représentation graphique de la SVM linéaire (la formule est complexe !!)

```
csvm <- drop(t(svmlin$coefs) %*% as.matrix(v.train[svmlin$index,"sdf0"]))
bsvm <- drop(t(svmlin$coefs) %*% as.matrix(v.train[svmlin$index,"moyf0"]))
asvm <- -svmlin$rho
-c(asvm/csvm, bsvm/csvm)
```

```
[1] 217.057139  -1.196721
```

```
plot(
    v.train$moyf0,
    v.train$sdf0,
    col=ifelse(v.train$sexe==1, "blue", "deeppink"),
    xlab="moyf0", ylab="sdf0",
```

```
    pch=19,
    main="SVM linéaire, ADL et régression logistique")
#droite de décision de la SVM linéaire
abline(-asvm/csvm, -bsvm/csvm, col="green", lwd=2)
#droite de décision de l'analyse discriminante linéaire
abline(-adisc/cdisc, -bdisc/cdisc, col="black", lwd=2)
#droite de décision de la régression logistique
abline(-arlog/crlog, -brlog/crlog, col="red", lwd=2, lty=2)
# légende
legend(
    "topright",
    legend=c("Homme", "Femme", "Droite SVM", "Droite ADL", "Droite RLog"),
    col=c("blue", "deeppink", "green", "black", "red"),
    pch=c(19, 19, NA, NA, NA),
    lty=c(NA, NA, 1, 1, 2))
```



Les observations situées sur les marges sont données par : `svmlin$index[abs(svmlin$coefs)!=1]`

Les observations écartées et utilisées comme pénalisation sont données par : `svmlin$index[abs(svmlin$coefs)==1`

Les coefficients des droites correspondant aux marges sont données par : `c(-(asvm+1)/csvm,-bsvm/csvm)` et `c(-(asvm-1)/csvm,-bsvm/csvm)`

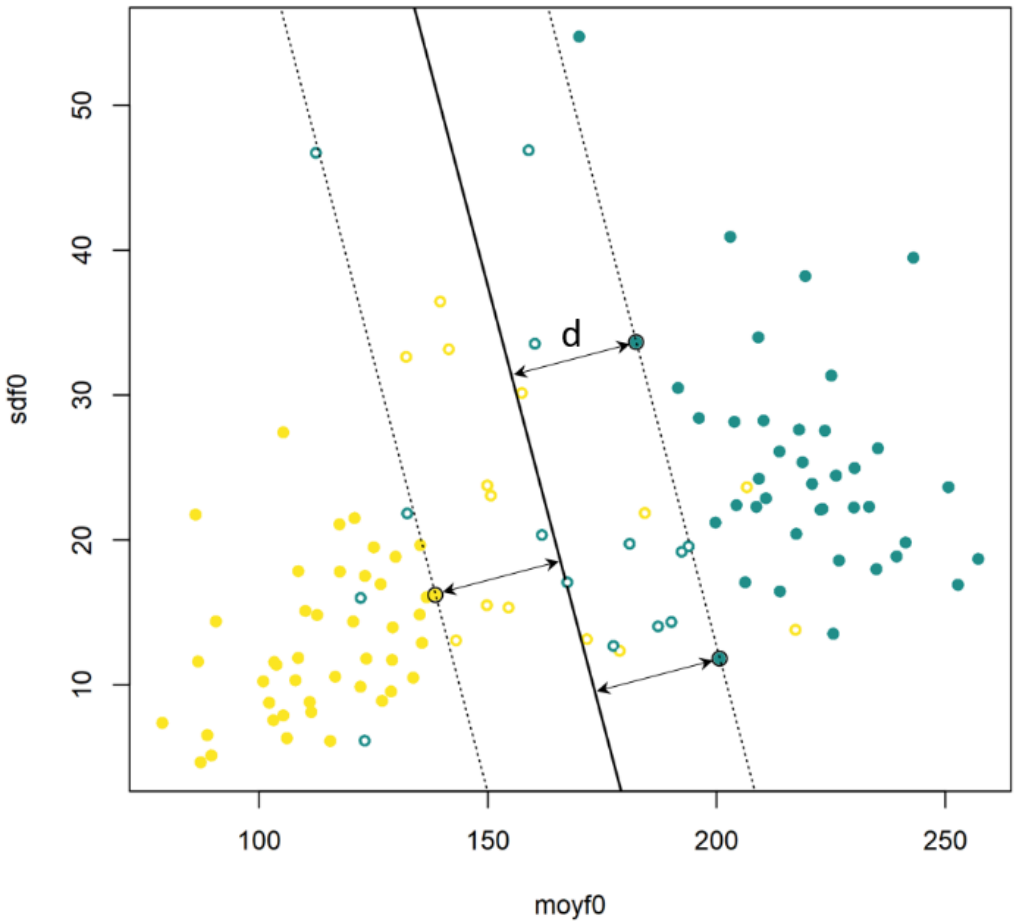


Fig. 3.6 — *SVM avec noyau linéaire. Les observations intégrées dans l'algorithme sous forme de pénalités sont représentées par des points colorés évidés. Ces pénalités sont utilisées pour déterminer la droite optimale qui maximise la marge d correspondant à la distance minimale entre la droite et les observations de chaque groupe (une fois enlevées les observations pénalisantes).*

5.A.2 SVM avec noyau différent

On peut utiliser la fonction `tune` qui permet de déterminer

- le noyau le plus adéquat,
- les hyperparamètres optimaux (coût, paramètre gamma pour noyau radial, degré pour noyau polynomial)

```
# Reproductibilité avec seed()
set.seed(3)

#Fonction tune
```

```
tune(
    svm, # fonction à optimiser
    factor(sexe) ~ moyf0 + sdf0, # formule du modèle
    data = v.train, # données d'entraînement
    ranges = list(kernel = c('linear','polynomial','sigmoid'), # types de noyaux à
     ↪  tester
    cost = c(0.5,1.0,2.0))) # valeurs de l'hyperparamètre coût à tester (un
     ↪  éventail plus large est testé en pratique)
```

```
set.seed(3)
tune(
    svm,
    factor(sexe) ~ moyf0 + sdf0,
    data = v.train,
    ranges = list(kernel = c('linear','polynomial','sigmoid'),
    cost = c(0.5,1.0,2.0)))
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 kernel cost
 linear    1

- best performance: 0.09848485
```

Sortie :

- best paramaters : kernel = linear, cost = 1

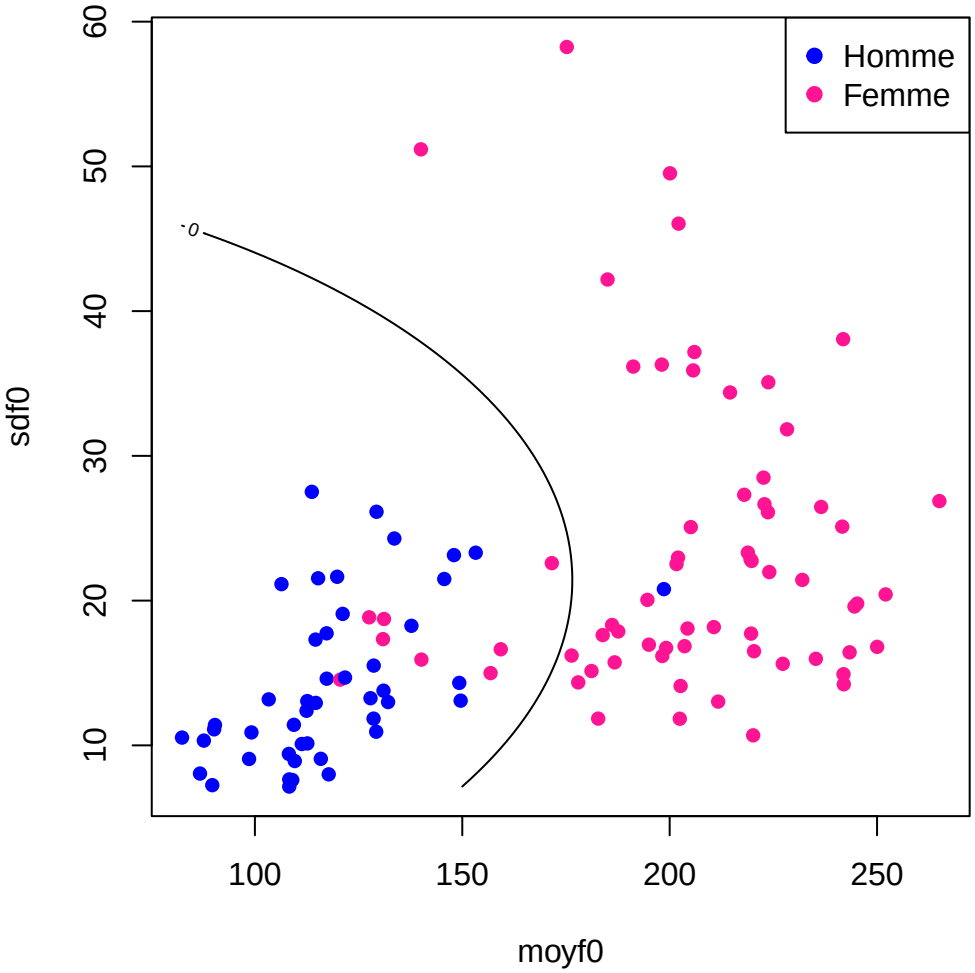
Mais si on veut regarder ce que ferait un noyau polynomial de degré 2 avec un coût de 2 :

```
svmpol <- svm(factor(sexe) ~ moyf0 + sdf0, data = v.train, kernel="polynomial",
 ↪  scale=TRUE, coef0=1, degree=2, cost=2)
table(predict(svmpol, v.test), v.test$sexe)
```

```
    1  2
  1 45  8
  2  1 58
```

Le nombre est moins bon (103 sujets correctement classés sur 112, soit 92.0%).
La fonction séparatrice est plus complexe à obtenir (car noyau polynomial de degré 2).

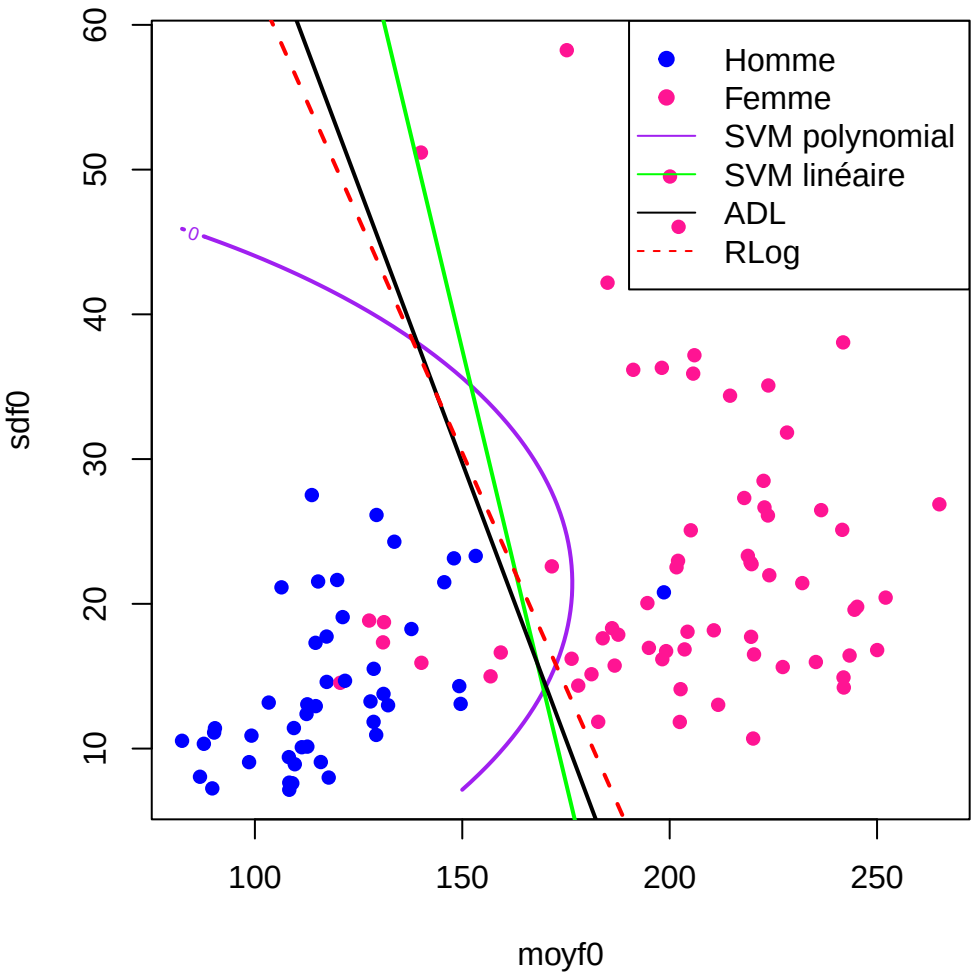
```r
couleur <- c("blue", "deeppink")

xg <- seq(from=min(v.test$moyf0),
            to=max(v.test$moyf0), length=100)

yg <- seq(from=min(v.test$sdf0),
            to=max(v.test$sdf0), length=100)

grid <- expand.grid(moyf0=xg, sdf0=yg)

predgrid <- predict(svmpol, newdata=grid, decision.values = TRUE)
score <- attr(predgrid, "decision.values")

plot(
    v.test$moyf0, v.test$sdf0,
    col = couleur[v.test$sexe],
    xlim = c(min(v.test$moyf0), max(v.test$moyf0)),
    ylim = c(min(v.test$sdf0), max(v.test$sdf0)),
    pch = 16,
    xlab = "moyf0",
    ylab = "sdf0"
)

contour(
    xg, yg,
    matrix(score, 100, 100),
    level = 0,
    add = TRUE
)

# légende
legend(
    "topright",
    legend=c("Homme", "Femme"),
    col=c("blue", "deeppink"),
    pch=19)
```



Idem mais ajouter logistique et ADL et SVL linéaire :

```
plot(
    v.test$moyf0, v.test$sdf0,
    col = couleur[v.test$sexe],
    xlim = c(min(v.test$moyf0), max(v.test$moyf0)),
    ylim = c(min(v.test$sdf0), max(v.test$sdf0)),
    pch = 16,
    xlab = "moyf0",
    ylab = "sdf0"
)
contour(
    xg, yg,
    matrix(score, 100, 100),
    level = 0,
    add = TRUE,
    col="purple",
    lwd=2
)
#droite de décision de la SVM linéaire
abline(-asvm/csvm, -bsvm/csvm, col="green", lwd=2)
```

```
#droite de décision de l'analyse discriminante linéaire
abline(-adisc/cdisc, -bdisc/cdisc, col="black", lwd=2)
#droite de décision de la régression logistique
abline(-arlog/crlog, -brlog/crlog, col="red", lwd=2, lty=2)
# légende
legend(
    "topright",
    legend=c("Homme", "Femme", "SVM polynomial", "SVM linéaire", "ADL", "RLog"),
    col=c("blue", "deeppink", "purple", "green", "black", "red"),
    pch=c(19, 19, NA, NA, NA, NA),
    lty=c(NA, NA, 1, 1, 1, 2))
```



6 Arbres de classification (CART)

Arbres de classification : similaires aux arbres diagnostiques utilisés en médecine.

Face à un prurit : 1/ localisé ou diffus ? 2/ Si localisé : cuir chevelu ou ailleurs ? 3/ Si diffus : rechercher signes dermatologiques précis.

L'algorithme CART formalise cette approche.

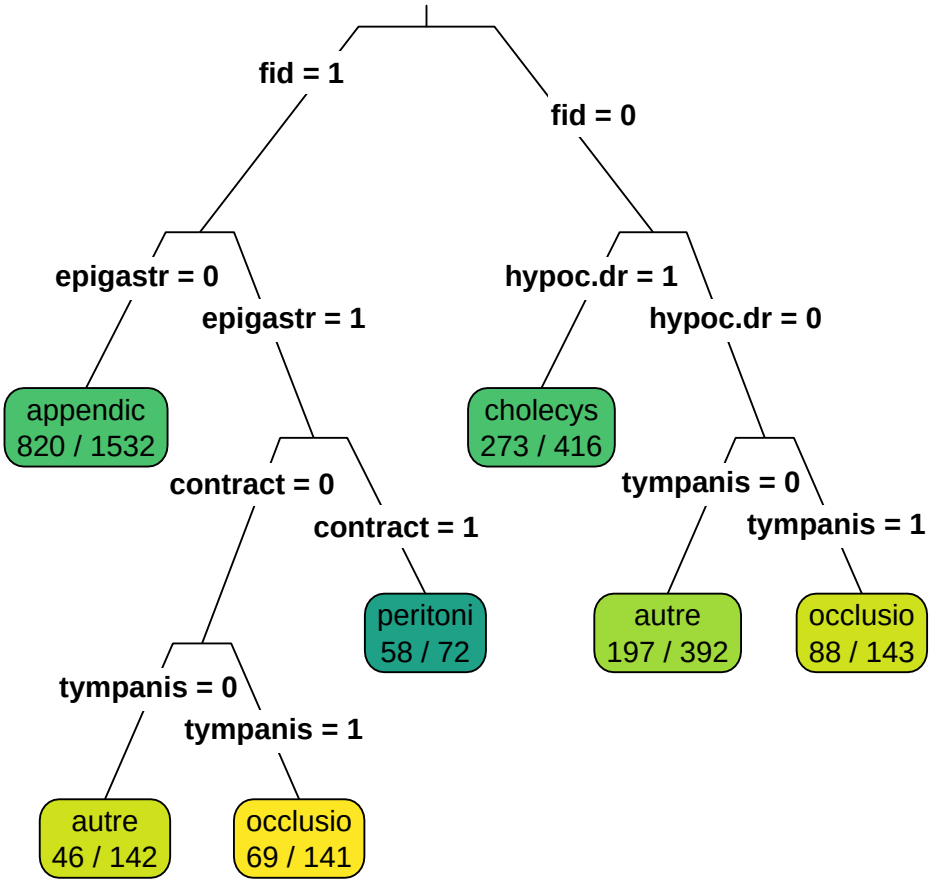
- À chaque étape, chaque signe et symptôme sont considérés sous une forme binaire.
  - Pour une variable catégorielle, chaque modalité sera tour à tour opposée aux autres.
  - Pour une variable quantitative, toutes les dichotomies possibles seront examinées.
- La segmentation retenue sera celle qui conduit à deux sous branches contenant chacune la population de patients la plus homogène en terme de diagnostic.
- Si cette approche est répétée à l'infini, l'arbre serait d'une complexité incomcompatible avec son utilisation pratique !
- Se limiter à un arbre simple = éviter le surapprentissage (overfitting) qui ferait qu'il ne serait pas transposable à de nouvelles données.
- Aussi, pas une méthode inférentielle : pas de tests statistiques ou d'intervalles de confiance.

6.A Exemple en R

Fonctions `rpart()` et `rpart.plot()` des packages `rpart` et `rpart.plot` pour construire un arbre de classification.

On reprend le jeu de données `abd` qui recueille des signes et symptômes cliniques de patients souffrant de douleurs abdominales aux urgences.

```
arbre.abd <- rpart(diagnostic ~ ., data = abd.train, method = "class")
prp(
    arbre.abd,
    type = 3,
    clip.right.labs = FALSE,
    branch = .3,
    box.col = viridis(15)[9:15],
    extra=2)
```



La fonction `prp()` permet de visualiser l'arbre de classification, les options sont très riches !!

Des techniques de validation croisées sont disponibles à partir des fonctions `printcp()` et `prune()`, et sont susceptibles de produire des arbres plus performant sur un échantillon de validation.

```
cart.abd <- table(
                predict(
                    arbre.abd,
                    newdata=abd.test,
                    type="class"),
                abd.test$diagnostic)
sum(diag(cart.abd))/sum(cart.abd)
```

```
[1] 0.5556338
```

Le pourcentage de patients bien diagnostiqué dans l'échantillon de test est modeste : 56%!

C'est séduisant mais en pratique, c'est décevant et moins bien qu'une analyse discriminante linéaire ou une régression logistique.

7 Forêts aléatoires (Random Forest)

Problème : instabilité des arbres de classification (instabilité = petites variations dans les données d'entraînement peuvent conduire à des arbres très différents).

Principe des forêts aléatoires et du boosting pour stabiliser les arbres de classification :

- Plusieurs arbres rudimentaires sont construits selon des modalités différentes.
- Chaque arbre pris isolément a des performances médiocres.
- Considérés dans leur ensemble, ils peuvent s'avérer redoutablement efficaces.

Avec l'approche du *boosting* :

- Les observations les plus mal classées par les arbres disponibles à un instant t ont un poids de plus en plus important.
- Les nouveaux arbres, même s'ils sont simples dans leur forme, peuvent être plus performants pour les observations difficiles à classer.

Avec l'approche des *forêts aléatoires* (random forest) :

- Un grand nombre d'arbres sont construits à partir de sous-échantillons du jeu de données initial.
- Les sous-échantillons sont réalisés :
  - en termes d'observations (environ 2/3 sont tirées au sort à chaque fois (1)).
  - en termes de variables (s'il y a p variables en tout, $\sqrt{p}$ seront tirées au sort à chaque embranchement pour participer au choix réalisé par l'algorithme).
  - Chaque arbre est construit de manière indépendante des autres.

L'outil final regroupe l'ensemble des arbres ainsi obtenus.

C'est la prédiction majoritaire qui sera proposée comme réponse finale de l'algorithme.

7.A Exemple en R

Fonction `randomForest()` du package `randomForest` pour construire une forêt aléatoire.

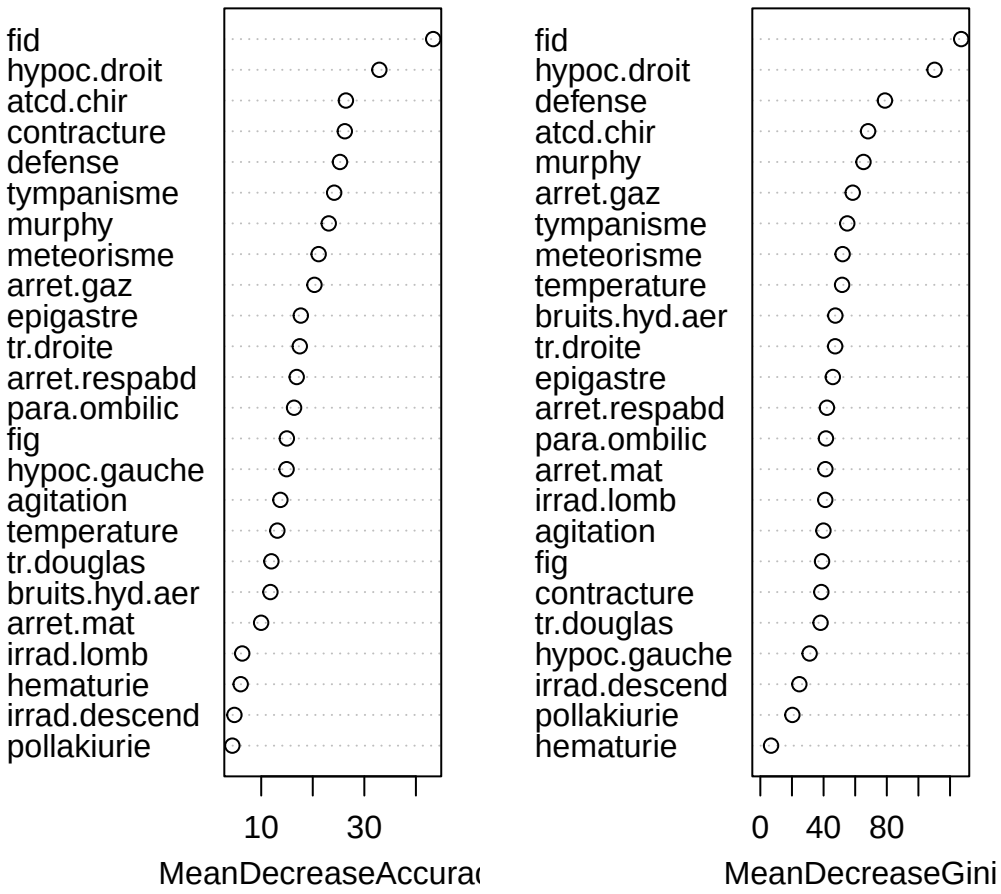
On reprend le jeu de données `abd` qui recueille des signes et symptômes cliniques de patients souffrant de douleurs abdominales aux urgences.

```
set.seed(4)
model.rf <- randomForest(
                    factor(diagnostic) ~ .,
                    data = abd.train,
                    ntree = 200,
                    importance=TRUE)
```

A ce moment : model.rf contient 200 arbres de classification construits sur des sous-échantillons de `abd.train`.

Pour les représenter, on peut utiliser la fonction `varImpPlot()` qui affiche l'importance des variables dans la classification.

```
varImpPlot(model.rf)
```



À gauche : l'indicateur correspond à la diminution du pourcentage d'observations bien classées quand la variable concernée est éliminée du jeu de données ;

A droite : il s'agit de la diminution du niveau moyen de pureté des segmentations réalisées.

Les deux donnent des résultats comparables : douleurs à la fosse iliaques droite et à l'hypochondre droit sont à chaque fois au premier plan.

Évaluation des performances prédictives du modèle sur l'échantillon de test `abd.test` :

```
round(model.rf$confusion,3)
```

```
             appendicite autre cholecystite colique.neph occlusion peritonite
appendicite          773   113           10            3         7         30
autre                431   307           98            4        46          7
cholecystite          28    45          281            2        14         14
colique.neph          30    50            5           24        10          2
occlusion             20    42            9            0       215         10
peritonite            24    23           20            0        22        119
             class.error
```

```
appendicite        0.174
autre              0.656
cholecystite       0.268
colique.neph       0.802
occlusion          0.274
peritonite         0.428
```

La colonne `class.error` donne le taux d'erreur de classification pour chaque pathologie.

Pour obtenir le taux de classification correcte global sur l'échantillon de test :

```
rf.abd <- table(abd.test$diagnostic, predict(model.rf, newdata=abd.test))
sum(diag(rf.abd))/sum(rf.abd)
```

```
[1] 0.628169
```

C'est mieux : 63% des patients de l'échantillon de test sont correctement classés, contre 56% avec un seul arbre de classification.

Mais en vrai, c'est toujours moins bien que l'ADL qui avait mieux performé (65%).

8 Régressions pénalisées (lasso)

Rappel : formule de la régression linéaire multiple classique :

$$Y = a_0 + a_1X_1 + a_2X_2 + ... + a_pX_p + \varepsilon$$

L'objectif est de trouver des coefficients a_i qui minimisent la variance résiduelle $var(\varepsilon)$ = la somme des carrés des résidus dans une régression linéaire (résidus = différences entre valeurs observées et valeurs prédites par le modèle).

Ainsi, les a_i expliquent "au mieux" Y.

On peut donc :

- Intégrer des indices de parcimonie dans le modèle pour éviter le surapprentissage (overfitting) par "régression pas à pas" (stepwise regression).
- Utiliser des régressions pénalisées (lasso, ridge, elastic net) qui contraignent les coefficients a_i à être petits (proches de 0) voire nuls.
  - Lasso (Least Absolute Shrinkage and Selection Operator) : pénalise la somme des valeurs absolues des coefficients (norme L1).
  - *Ridge : pénalise la somme des carrés des coefficients (norme L2)*
  - *Elastic net : combinaison des deux pénalisations*

Dans la pénalisation par lasso : l'idée est de considérer que si un terme a_iX_i n'expplique pas grand chose de Y, on peut contraindre a_i à être nul, afin d'avoir une meilleure estimation de Y en ne retenant que les variables explicatives les plus importantes.

Mathématiquement : on détermine les a_i qui minimisent en faisant :

$$\frac{1}{2}var(\varepsilon) + \lambda(|a_1| + |a_2| + ... + |a_p|)[1]$$

Ou λ est un hyperparamètre qui contrôle la pénalisation, fixé par le statisticien

L'objectif est que cette somme soit la plus petite possible.

Donc on fait :

- $\frac{1}{2}var(\varepsilon)$: minimiser la variance résiduelle ($\frac{1}{2}$ est juste un facteur de normalisation)
- $\lambda(|a_1| + |a_2| + ... + |a_p|)$: pénaliser la somme des valeurs absolues des coefficients. Plus un coefficient est grand, plus il est pénalisé par λ.

→ Si un des facteurs X_i a un pouvoir prédictif fort, la réduction de la variance résiduelle associée va compenser la pénalisation de $|a_i|$.

On peut faire ça aussi pour une régression logistique ou de Poisson, en remplaçant la variance résiduelle par l'opposé de la log-vraisemblance.

- en régression logistique : logit de $p = P(Y = 1)$
- en régression de Poisson : log de λ (paramètre de la loi de Poisson)
- NB : déviance = -2 * log-vraisemblance

La régression pénalisée **LASSO** (Least Absolute Shrinkage and Selection Operator) est une méthode d'estimation statistique essentielle, particulièrement utile dans les contextes de science des données et de *big data* où le risque de surapprentissage (ou sur-ajustement) est élevé„.

Voici une synthèse détaillée de son principe, de sa méthodologie et de son comportement, basée sur les sources fournies.

8.A Synthèse de la Régression Pénalisée par Lasso (NLM)

8.A.1 Contexte : Le Problème du Surapprentissage

La régression linéaire classique (méthode des moindres carrés) cherche à déterminer les coefficients qui **minimisent la variance résiduelle**.

Cependant, si le nombre de variables explicatives est très grand par rapport au nombre d'observations, cette approche devient trompeuse.

Un modèle trop complexe risque d'apprendre la structure du bruit aléatoire de l'échantillon d'entraînement, ce qui se traduit par des performances excellentes sur cet échantillon, mais médiocres sur un échantillon de validation indépendant (situation de **surapprentissage**).

Le Lasso est une approche préventive visant à corriger ce phénomène,.

8.A.2 Principe et Méthode du Lasso

Le Lasso est une technique de régression pénalisée.

L'idée centrale est que si un terme explicatif $a_i X_i$ n'explique la variable réponse Y que de manière marginale, il est préférable de considérer que son coefficient a_i est nul pour obtenir une meilleure estimation prédictive de Y.

8.A.2.1 La Fonction d'Objectif

Au lieu de simplement minimiser la variance résiduelle, la méthode Lasso minimise une somme composée de deux termes :

$$\text{Objectif} = \text{minimiser} : \frac{1}{2}\text{Var}(\varepsilon) + \lambda \left(\sum_{i=1}^{p} |a_i| \right)$$

1. **Terme de Qualité d'Ajustement :** $\frac{1}{2}\text{Var}(\varepsilon)$ (la moitié de la variance résiduelle, comme dans la régression linéaire standard).
2. **Terme de Pénalisation (Pénalité L_1) :** $\lambda \left(\sum_{i=1}^{p} |a_i| \right)$, qui est proportionnel à la somme des valeurs absolues des coefficients a_i.

8.A.2.2 Rôle du Paramètre de Pénalisation (λ)

- Le paramètre λ est la **constante de pénalité**, fixée par le statisticien (c'est un **hyper-paramètre**),.
  - Si $\lambda = 0$, le Lasso est équivalent à un modèle linéaire ordinaire.
  - Si λ est très grand, tous les coefficients a_i sont annulés, et il ne reste que l'ordonnée à l'origine a_0, qui est alors égale à la moyenne générale de Y,.

- Pour une valeur intermédiaire de λ, la pénalité contraint les coefficients ayant un faible pouvoir prédictif à se rapprocher de zéro, et certains sont même annulés.
- L'ordonnée à l'origine (a_0) n'est généralement pas pénalisée.

8.A.2.3 Généralisation aux Modèles Linéaires Généralisés (GLM)

Le principe de pénalisation peut être généralisé à des modèles non linéaires, tels que la **régression logistique** ou la **régression de Poisson**.

Dans ces cas, le terme de variance résiduelle $0,5 \times \text{Var}(\varepsilon)$ est remplacé par **l'opposé de la log-vraisemblance** dans l'équation d'optimisation.

8.A.3 Avantages Clés du Lasso

La pénalisation apporte deux avantages majeurs :

1. **Correction du Surapprentissage et Amélioration de la Capacité Prédictive :**
    - En contraignant la complexité du modèle, le Lasso évite le surajustement aux données d'entraînement et améliore sa capacité à prédire des résultats sur de nouvelles données.
2. **Parsimonie et Sélection Automatique de Variables :**
    - Le Lasso a la capacité de rendre nuls les coefficients des variables peu pertinentes.
    - Cela permet de baser la prédiction sur un nombre réduit de variables, ce qui est très utile lorsque la collecte de données est coûteuse, chronophage ou cliniquement invasive.

De plus, lors de simulations complexes avec de nombreuses variables et un faible ratio observations/variables, le Lasso a démontré une **meilleure performance prédictive moyenne** (moins de 10% d'erreurs) que l'Analyse Discriminante Linéaire (LDA, 19% d'erreurs), la Régression Logistique (GLM, 25% d'erreurs) ou même la régression par analyse pas-à-pas (*stepwise*, 13% d'erreurs),.

8.A.4 Comportement Face aux Variables et Inconvénients

8.A.4.1 Comportement face à la Colinéarité

Lorsque des variables explicatives sont fortement corrélées (colinéarité) :

- La régression par analyse pas-à-pas a tendance à retenir **une seule des variables** et à ignorer l'autre, souvent de manière "chaotique",.
- Le Lasso, au contraire, préfère **partager l'effet** en attribuant un coefficient non nul et de même amplitude normalisée aux deux variables.

Si l'objectif principal est d'obtenir le modèle le plus parcimonieux possible (avec le moins de variables), la régression pas-à-pas pourrait être plus pertinente, bien que le Lasso soit généralement préféré pour sa robustesse face au surapprentissage.

8.A.4.2 2. Comportement face aux Variables selon leur Type

- **Variables Catégorielles :**
    - Le choix de la modalité de référence (pour le codage 0/1) d'une variable catégorielle peut **influencer de manière substantielle les résultats du modèle Lasso**.

- De plus, le Lasso peut annuler certaines des $k-1$ variables binaires générées pour une variable catégorielle à k modalités, mais en conserver d'autres.

- **Groupes de Variables Liées :**
  - Le Lasso standard ne prend pas en compte le fait que certaines variables doivent être considérées en bloc (par exemple, les indicateurs binaires d'une seule variable catégorielle).
  - Par exemple : catégories socio-professionnelles codées en 1 = Cadre, 2 = Employé, 3 = Ouvrier, transformées en deux variables binaires avec Cadre comme référence.
  - Il existe une extension, le *group lasso*, qui corrige ce problème.

8.A.4.3 3. Limites du Lasso pour l'Interprétation

Les coefficients a_i obtenus par le Lasso **ne sont pas destinés à être interprétés** dans le cadre de modèles explicatifs (visant à comprendre les déterminants d'un phénomène).

Ils proviennent d'un apprentissage automatisé ciblant l'efficacité prédictive.

Le fait que le codage de la variable catégorielle puisse affecter les coefficients confirme que le modèle ne doit pas être utilisé pour des conclusions explicatives,.

8.A.5 V. Choix de la Constante de Pénalité (λ)

La constante λ est un **hyperparamètre** qui doit être choisi avec prudence.

- **Méthode de Choix Courante :**
  - La valeur de λ est le plus souvent choisie par **validation croisée**,.
  - Cette méthode évalue le taux d'erreur de classement du modèle pour différentes valeurs de λ et choisit la solution qui minimise ce taux,.

- **Risque de Perte de Parcimonie :**
  - La validation croisée peut être assimilée à un test de significativité déguisé.
  - Sur de grands échantillons, elle peut conduire à la sélection d'un modèle incluant un grand nombre de variables pour des bénéfices prédictifs infimes, entraînant une perte de la parcimonie recherchée.

- **Arbitraire Humain :**
  - Par défaut, le Lasso pénalise toutes les variables (sauf l'intercept) de la même manière, indépendamment de leur coût de collecte ou de leur nature.
  - Il est possible d'adapter la pénalité en modifiant l'échelle des variables, mais tout réglage manuel de λ introduit un **arbitraire humain** dans le processus d'apprentissage ; ce qui rend impossible l'utilisation ultérieure de méthodes comme la validation croisée ou le *bootstrap* pour l'évaluation des performances du modèle final (on ne va pas estimer des fluctuations d'échantillonage sur un truc basé sur un choix humain)

8.A.6 VI. Positionnement par rapport à d'Autres Modèles Supervisés

Le Lasso excelle dans la tâche de **prédiction** en présence d'un grand nombre de variables explicatives et d'un risque élevé de surapprentissage, là où les méthodes traditionnelles échouent,.

Modèle Comparé	Points de Comparaison / Supériorité du Lasso
Régression Linéaire/Logistique (GLM)	Les GLM classiques peuvent produire un ajustement parfait sur l'échantillon d'entraînement (surapprentissage) et des erreurs majeures sur un échantillon de validation lorsque le nombre de variables est élevé. Le Lasso évite ce surapprentissage et est plus stable lorsque la taille de l'échantillon diminue.
Analyse Pas-à-Pas (Stepwise)	Les analyses pas-à-pas sont "à éviter à tout prix" en général, car elles biaissent considérablement les résultats, bien qu'elles puissent être plus pertinentes que le Lasso si l'objectif est de maximiser la parsimonie au sens de réduire au minimum le nombre de variables retenues. Le Lasso montre de meilleures performances prédictives que le Stepwise lors de simulations avec beaucoup de variables.
Analyse Discriminante Linéaire (LDA)	La LDA, bien que simple, souffre d'un surapprentissage important en cas de nombreuses variables, conduisant à des performances médiocres là où le Lasso reste stable,.

En conclusion, le Lasso est un outil précieux pour la **construction de modèles prédictifs robustes et parcimonieux**, notamment en présence de *big data*„ mais il est moins adapté aux analyses explicatives nécessitant une interprétation précise de chaque coefficient.

Note

EXEMPLE DE VARIABLES CATÉGORIELLES LIÉES
Lorsqu'une variable explicative est catégorielle et possède k modalités (par exemple, "Niveau d'étude" : Primaire, Secondaire, Supérieur), elle est transformée par les logiciels en $k-1$ **variables binaires (indicatrices)** pour être incluse dans un modèle.
Exemple : Statut socio-professionnel (3 modalités)
Si nous avons la variable Z avec trois modalités :

- **A : Cadre (Modalité de référence)**
- **B : Employé**
- **C : Ouvrier**

Elle est transformée en deux variables binaires (X_{Employ} et $X_{Ouvrier}$) :

Observation	Statut Z	X_{Employ}	$X_{Ouvrier}$
1	Cadre	0	0
2	Employé	1	0
3	Ouvrier	0	1

Dans ce cas, X_{Employ} et $X_{Ouvrier}$ sont intrinsèquement **liées** puisqu'elles représentent ensemble la variable Z.
Le **Lasso standard**, qui applique une pénalité sur chaque coefficient individuellement, peut alors :

- Garder les coefficients pour X_{Employ} et $X_{Ouvrier}$ (si l'effet est fort).
- **Annuler le coefficient de** X_{Employ} mais garder celui de $X_{Ouvrier}$.

Le problème est que cette situation **n'a pas vraiment de sens** dans un modèle explicatif. Le Lasso a seulement réduit la complexité pour des raisons prédictives, mais il a laissé un modèle où la variable catégorielle Z est partiellement représentée, ce qui compromet la parsimonie souhaitée, car l'enlèvement d'une partie du bloc sans enlever l'autre est contre-intuitif.
L'extension **"group lasso"** a été développée spécifiquement pour corriger cette limite en considérant ces variables comme un **bloc** unique lors de l'application de la pénalité, garantissant que soit toutes les variables du groupe sont conservées, soit toutes sont annulées.
En conclusion, alors que le LASSO excelle pour la prédiction et la sélection de variables indépendantes, il peut se comporter de manière ambiguë face aux variables catégorielles recodées en blocs binaires.