PSL UNIVERSITÉ PARIS

MIDO
MATHÉMATIQUES ET INFORMATIQUE
DE LA DÉCISION ET DES ORGANISATIONS

Dauphine UNIVERSITÉ PARIS

Projet de l'apprentissage statistique
Diagnostic de la maladie de Parkinson

Étudiants :

Mengru CHEN
Chuyao LU
Karl MEKIE CHAMI
Dossomguimon TUO

Responsable :

Yating LIU

2021 - 2022

Table des matières

L'OBJECTIF du projet est de déterminer les individus potentiellement atteints de la maladie de Parkinsons à partir des données sur les enregistrements vocaux. Il s'agit d'une maladie neurologique chronique dégénérative très fréquent chez les plus de 45 ans qui provoque des tremblements ou des raideurs musculaires, mais dont les causes sont encore inconnues. Si à partir d'un pré diagnostic, on puisse déterminer la présence ou non de la maladie chez l'individu, on peut alors essayer de retarder l'évolution du Parkinson.

Pour cela, nous allons procéder à plusieurs méthodes de *machine learning* pour la prédiction. Nous mettrons en place les modèles K-NN, régression logistique, arbre de décision et SVM. L'enjeu est ensuite de comprendre le fonctionnement ces modèles, estimer et comparer leur efficacité pour choisir le meilleur qui est adapté à notre étude. Au cours du développement, nous détecterons également les différents caractéristiques (sur les enregistrements vocaux) présents chez les individus atteints de la maladie de Parkinson.

Les données que nous allons utiliser sont issues du site `https://archive.ics.uci.edu/ml/datasets/parkinsons`.

1 Préparation des données

Avant tout modélisation, il est indispensable de réaliser une bonne préparation des données. Nous avons à notre disposition 195 enregistrements vocaux issues de 32 individus (et non pas 31 comme annoncé sur le site), chaque individu ayant réalisé 6 à 7 enregistrements. Les analyses réalisés ont permis de repérer 22 caractéristiques telle que la fréquence ou l'amplitude de la voix. Cette base de donnée est stockée dans un dataframe `parkinsons`. Par ailleurs, après vérification nous n'avions pas remarqué de données manquantes ni de doublons.

On considère que les données sont sous la forme :

$$\mathcal{D}_n = \{(X_i, Y_i)\,,\ i = 1, \dots, n\}$$

où $X_i \in \mathcal{X} = \mathbb{R}^p$, $Y_i \in \mathcal{Y} = \{0, 1\}, n = 195$ et $p = 22$.

Ici, $(Y_i)_{i=1,\dots,n}$ représente les données de la variable d'intérêt `status`, qui signifie l'état de santé des individus et $(X_i)_{i=1,\dots,n}$ représente les données des autres 22 variables.

- si `status` $= 1$, l'individu est atteint de la maladie de Parkinson ;
- si `status` $= 0$, l'individu n'est pas atteint de la maladie de Parkinson.

Même si le nombre de *features* dans notre cas n'est pas gigantesque, il se trouve que nombreux sont très corrélés. Il nous semble intéressant de comparer l'efficacité des modèles à l'issu de ces 2 bases de données. En particulier, nous avons supprimé 10 variables explicatives très corrélés $\sigma_{X_i,X_j} \geq 0.9$ avec $i \neq j$. Ce nouveau dataframe sera nommée `parkinsons_sv`.

Par la suite, nous avons séparé, pour chacune des deux bases de données en 4 sous ensembles.

- pour `parkinsons` : `X_train`, `X_test`, `Y_train`, `Y_test`
- pour `parkinsons_sv` : `Xsv_train`, `Xsv_test`, `Ysv_train`, `Ysv_test`

Pour notre étude, nous sommes fixés à 33% des données pour l'ensemble de test, c'est-à-dire `X_test` et `Xsv_test` contiennent 65 individus choisis aléatoirement parmi l'ensemble.

Étant donnée qu'on a pas beaucoup plus de *features* que le nombre d'individus $p < n$ et que (comme on le fera plus tard) le risque de sur apprentissage n'est pas un point de blocage dans notre cas, il n'est donc pas nécessaire de faire une régularisation.

2 Classification par K-NN

2.1 Présentation de la méthode K-NN

La méthode des K plus proches voisins (K-NN) développée pour la première fois par Evelyn Fix et Joseph Hodges en 1951, puis approfondie par Thomas Cover en 1968, est une méthode simple d'apprentissage supervisé non paramétrique.

L'idée de la méthode est simple et intuitive : pour un individu donnée, on dit qu'il appartient à la modalité M si la majorité de ses K plus proches voisins (partageant des caractéristiques similaires) appartiennent aussi à cette modalité.

Pour un $x \in \mathbb{R}^p$ fixé, on va réorganiser les données $(X_1, Y_1), \ldots, (X_n, Y_n)$ selon l'ordre croissant de $\|X_i - x\|$ où $\|\cdot\|$ est la distance euclidienne. Notons $i(x) = \{i_1(x), \ldots, i_n(x)\}$ une permutation de $\{1, \ldots, n\}$ telle que :

$$\underbrace{\left\|X_{i_1(x)} - x\right\|}_{\text{Le plus proche voisin de } x} \leq \underbrace{\left\|X_{i_2(x)} - x\right\|}_{\text{Le 2 ème proche voisin de } x} \leq \ldots \leq \underbrace{\left\|X_{i_n(x)} - x\right\|}_{\text{Le voisin le plus éloigné de } x}$$

Les données réorganisées sont :

$$\left\{\left(X_{i_1(x)}, Y_{i_1(x)}\right), \ldots, \left(X_{i_n(x)}, Y_{i_n(x)}\right)\right\}$$

La fonction de `0-1` classification est :

$$\widehat{f}_n(x) = \widehat{f}\,(x, \mathcal{D}_n) := \begin{cases} 1 & \text{si } \#\left\{Y_{i_j(x)} = 1,\ 1 \leq j \leq K\right\} \geq \#\left\{Y_{i_j(x)} = 0,\ 1 \leq j \leq K\right\} \\ 0 & \text{sinon.} \end{cases}$$

2.2 Résultats avec et sans sélection

Afin de faire la classification par K-NN, on a utilisé le package `KNeighborsClassifier` dans `sklearn` avec K = 5 par défaut.

Après avoir entraîné le modèle sur la base de donnée initiale `parkinsons`, on obtient les résultats suivants :

- Le temps nécessaire : 0.0748 sec
- Le taux de précision : 83.07%
- Le taux d'apprentissage : 88.46%.

Lorsqu'on entraîne le modèle sur la base de donnée après sélection de variables `parkinsons_sv`, on obtient les résultats suivants :

- Le temps nécessaire : 0.0768 sec
- Le taux de précision : 83.08%
- Le taux d'apprentissage : 88.46%.

On remarque qu'il n'a pas beaucoup de différence en terme de taux de précision, taux d'apprentissage et le temps d'exécution.

Cependant, le modèle ne nous permet pas de comprendre le comportement des *features* X_i sur la variable d'intérêt Y.

Nous nous sommes alors intéressé à étudier l'impact des paramètres de l'algorithme de K-NN sur le taux de précision, en particulier le nombre du K voisins et le weight. Dans le cas où `weight = uniform`, les deux bases de données atteints le maximum en terme de *accuracy* pour $K = 5$. Dans le cas `weight = distance`, on a plusieurs possibilité du choix de K pour avoir un taux de précision maximal.

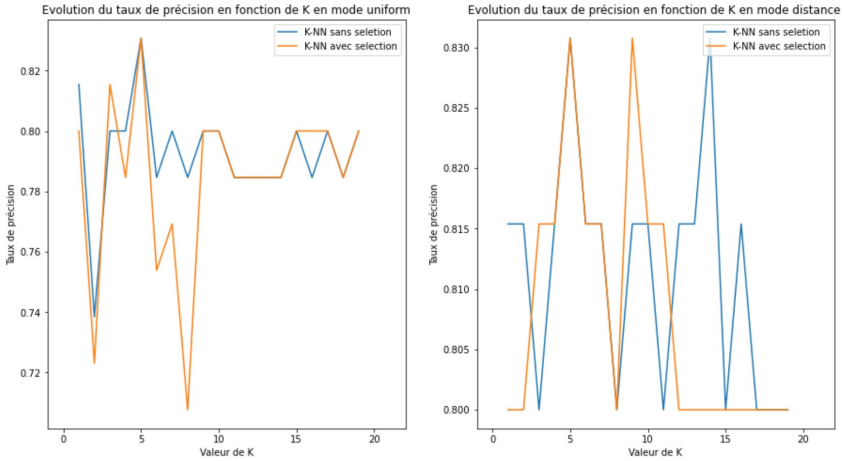

FIGURE 1 – Etude du modèle en fonction de l'évolution du nombre de voisins K-NN

3 Méthode de la régression logistique

3.1 Présentation du modèle

La méthode de la régression logistique est utilisé pour des problème de classification binaire comme dans notre cas. Le prédicteur de la régression pour une nouvelle donnée x_0 est :

$$f^{\beta,b}(x_0) = \begin{cases} 1 & \text{si } \Phi(x_0\beta + b) \geq 0.5 \\ 0 & \text{sinon.} \end{cases}$$

où Φ est la fonction sigmoïde définie par $\Phi(x) = \frac{1}{1+e^{-x}} \in]0, 1[$.

Quant aux paramètres β et b, le package `LogisticRegression` de `sklearn.linear_model` permet de calculer directement les solutions optimales qui minimisent la fonction de coût. Par exemple, sur la base de donnée `parkinsons`, on a en output :

- β = [-0.01, -0.01, 0.01, 0, 0, 0.05, 0.03, 0.15, 0.59, 6.01, 0.27, 0.44, 0.49, 0.82, -0.05, 0.13, -3.15, 3.12, 1.72, 1.81, 3.57, 1.87]
- $b = 1.18$

3.2 Résultats et interprétation

Comme dans le cas de K-NN, il n'y a pas de différence significative en terme de taux de précision, de taux d'apprentissage et de temps d'exécution lorsqu'on entraîne sur les modèles `parkinsons` ou `parkinsons_sv`.

Sur la base de donnée initiale `parkinsons` :

- Le temps nécessaire : 0.0483 sec
- Le taux de précision : 83.07%
- Le taux d'apprentissage : 86.92%.

Sur la base de donnée réduite `parkinsons_sv` :

- Le temps nécessaire : 0.0328 sec
- Le taux de précision : 81.53%
- Le taux d'apprentissage : 88.46%.

Par rapport à la méthode K-NN, l'algorithme n'apporte pas beaucoup d'amélioration sur le taux de précision. Il est néanmoins plus rapide en terme de temps.

On trouvait ensuite intéressant de comprendre les influences des *features* sur la présence ou non de la maladie de Parkinson sur l'individu.

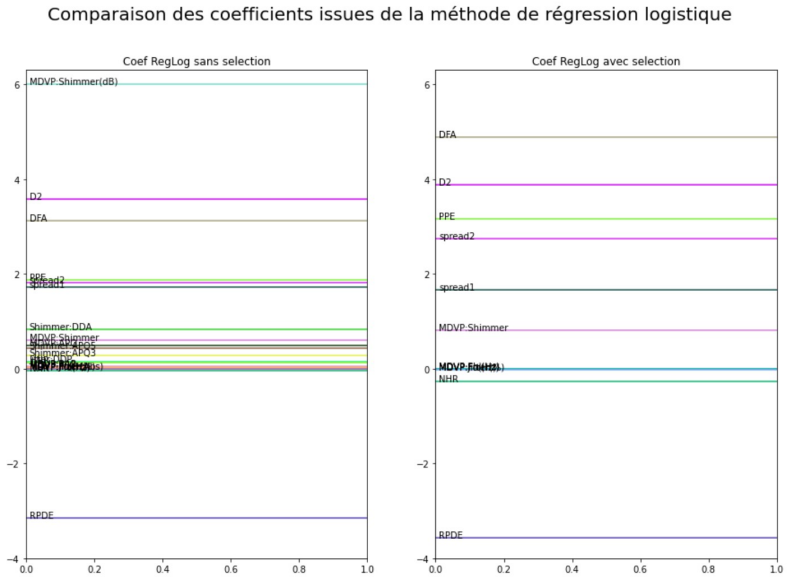

FIGURE 2 – Comparaison des coefficients de la méthode de régression logistique

Visuellement, les *features* les plus importants sont associés aux $|\beta_i|$ les plus importants. Par exemple, le β_{10} associé à la variable `MDVP:Shimmer(dB)` vaut 6.01. Cela signifie que plus la variation de l'amplitude de l'enregistrement est grande, plus l'individu diagnostiqué est susceptible d'avoir la maladie.

La version avec la sélection de variable nous a pas permis d'améliorer le taux de précision. Cependant, il nous apporte une meilleure visibilité en terme d'interprétation des variables.

3.3 Algorithme de la descente de gradient stochastique (SGD)

Dans le même temps, nous avons essayé d'écrire un algorithme permettant la minimisation de la fonction de perte par la méthode de la descente de gradient stochastique mini-batch `SGD_logistic_reg` à la main afin de comprendre le mécanisme qui se trouve derrière.

$$R^{Emp,D_n} = -\sum_{i=1}^{n} \left(Y_i log\left(g^{\beta,b}\left(x_i\right)\right) + \left(1 - Y_i\right) log\left(1 - g^{\beta,b}\left(x_i\right)\right)\right)$$

avec

$$g^{\beta,b}(x) = \Phi(x\beta + b)$$

Malheureusement, les résultats ne sont pas convaincants en terme convergence, ni en terme de précision. Le taux de précision obtenu est de seulement 73.84%. De plus, les coefficients à l'issu de l'algo sont très différents de ceux obtenus du package `sklearn`.

4 Arbre de décision

4.1 Généralité sur les arbres de décision

Un arbre de décision est un schéma représentant les résultats possibles d'une série de choix interconnectés. Il permet à une personne d'évaluer différentes actions possibles en fonction de leur coût, leur probabilité et leurs bénéfices. Il emploie une représentation hiérarchique de la structure des données sous forme des séquences de décisions (tests) en vue de la prédiction d'un résultat ou d'une classe. Chaque décision est classée selon l'ordre d'importance. Chaque individu (ou observation), qui doit être attribué(e) à une classe, est décrit(e) par un ensemble de variables qui sont testées dans les nœuds de l'arbre. Les tests s'effectuent dans les nœuds internes et les décisions sont prise dans les nœuds feuille.

- *Nœud racine* : l'accès à l'arbre se fait par ce nœud ;
- *Nœuds internes* : les nœuds qui ont des descendants (ou enfants), qui sont à leur tour des nœuds.
- *Nœuds terminaux (ou feuilles)* : nœuds qui n'ont pas de descendant.

On parcourt l'arbre de façon suivante : on se déplace vers la gauche si la condition dans le nœud est vérifiée sinon vers la droite. Pour bien comprendre cette notion d'arbre, nous nous intéressons à l'exemple simple du Titanic.

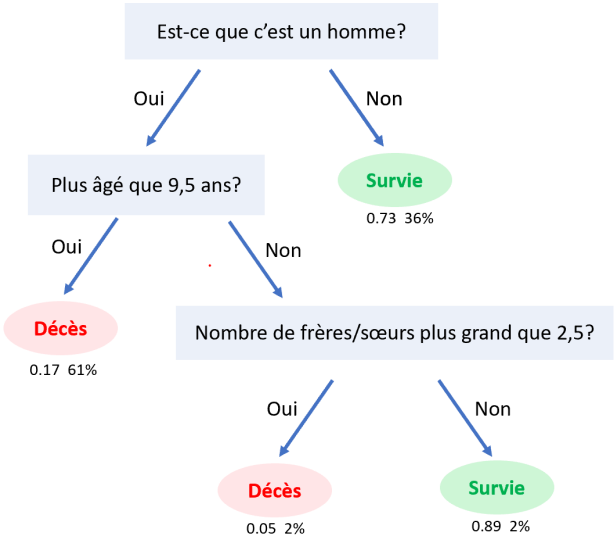

FIGURE 3 – Exemple de l'arbre de décision sur les données des survivants du Titanic

Dans cette exemple, là partir de l'arbre, on peut conclure que les hommes qui ont plus de 9 ans ont plus de chance de décéder, alors que les femmes, enfants ainsi que les familles ayant moins de 3 enfants ont plus de chance de survivre au naufrage.

4.2 Résultats du modèle

L'entraînement du modèle s'est fait à partir du package `DecisionTreeClassifier` de `sklearn.tree`. Les résultats obtenus sur les deux bases de données sont alors présenté ci-dessous.

Voici les 9 *features* X_i qui ont les plus grands coefficients d'importances. La variable `spread1` (qui correspond à la 19^e *feature* X_{19}) représente 36.86%, cela signifie qu'il est le noeud racine de l'arbre de décision.

	Importance des features
spread1	0.368615
DFA	0.131277
MDVP:Fhi(Hz)	0.117769
D2	0.115383
Shimmer:APQ3	0.110310
MDVP:Fo(Hz)	0.093107
MDVP:Flo(Hz)	0.035299
MDVP:Jitter(Abs)	0.028239
Shimmer:DDA	0.000000

FIGURE 4 – Importance des variables explicatives

Pour la base de données complète `parkinsons` :

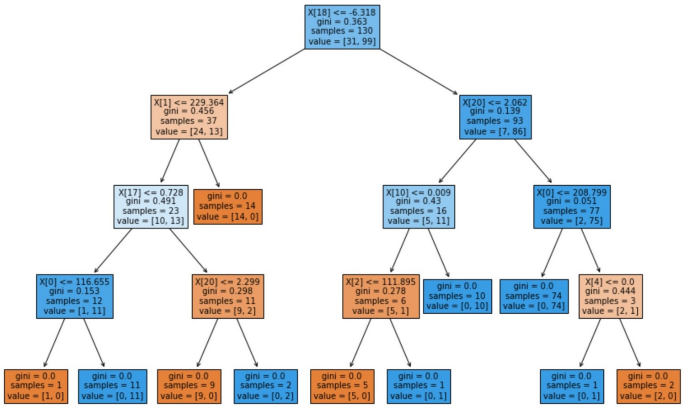

FIGURE 5 – Arbre de décision sur les données (`X_train`, `Y_train`)

- Le temps nécassaire : 0.0471 sec
- Le taux de précision : 92.31%
- Le taux d'apprentissage : 100%.

Pour la base de données réduite `parkinsons_sv` : Nous obtenons un arbre de décision très similaire à la précédente avec les caractéristiques suivantes :

- Le temps nécassaire : 0.0631 sec
- Le taux de précision : 92.31%
- Le taux d'apprentissage : 100%.

Dans ce cas de figure, l'entraînement sur la base de donnée réduite est inefficace puisque le temps d'exécution a augmenté alors que le taux de précision est resté identique.

4.3 Conclusion sur les arbres de décisions

Le modèle de l'arbre de décision est :

- facile à implémenter ;
- avec un grand taux de précision ;
- et un temps d'exécution très rapide.

Nous avons ensuite jugé intéressant de tracer la frontière de décision avec seulement 2 variables les plus importants (à savoir `spread1` et `DFA`) pour mieux interpréter les résultats du modèle.

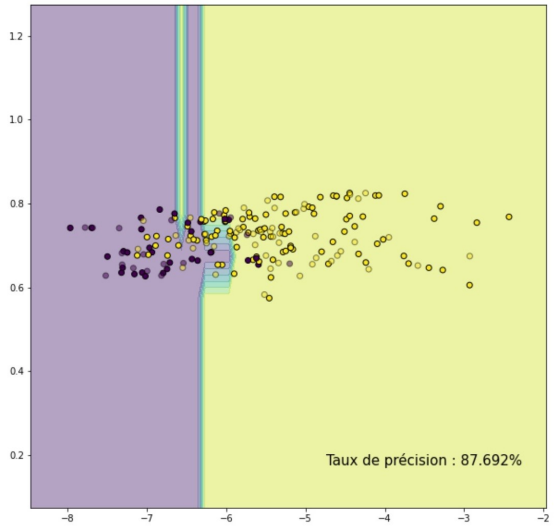

FIGURE 6 – Frontière de décision sur (`X_train`, `Y_train`)

On remarque alors que les individus ayant un `spread1` supérieure à -6.318, c'est-à-dire qui ont une variation de la fréquence fondamentale plus élevée sont plus risqués à la maladie de Parkinson.

5 Machine à Vecteurs de Support (SVM)

5.1 Généralité sur le modèle SVM

Les machines à vecteurs de support (SVM) sont des modèles d'apprentissage supervisé utilisés pour la classification, ce qui justifie l'usage de ce modèle pour notre problème de classification binaire.

Un SVM est un classificateur entièrement défini par un hyperplan (généralisation des plans dans $\mathbb{R}^3$ et des droites $\mathbb{R}^2$). En d'autres termes, étant donné les données d'apprentissage étiquetées (apprentissage supervisé), l'algorithme défini un hyperplan séparateur optimal qui catégorise les individus.

Deux cas de figures se présentent dans l'élaboration de l'hyperplan séparateur :

Cas 1 : avec une possibilité de séparation linéaire

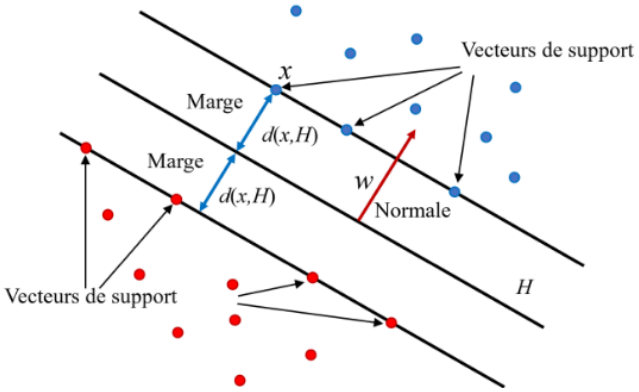

FIGURE 7 – Hyperplan séparateur linéaire

Dans le cas où il y a une possible séparation linéaire le but des SVMs est de maximiser la *marge* qui est définie par des vecteurs de supports. Les SVMs sont ainsi qualifiés de *séparateurs à vaste marge*, les *vecteurs de support* étant les données les plus proches de la frontière. Cette notion de frontière suppose que les données soient linéairement séparables, ce qui est rarement le cas.

Cas 2 : avec une impossibilité de séparation linéaire

Ce cas beaucoup plus fréquent nécessite quelques transformations de l'espace ou un changement de variables. Afin de remédier au problème de l'absence de séparateur linéaire, l'idée de l'astuce du ***noyau*** est de reconsidérer le problème dans un espace de dimension supérieure. Dans ce nouvel espace, il est alors probable qu'il existe une séparation linéaire.

Prenons exemple d'un cas simple. Si on place les points (0,0), (0,1), (1,0) et (1,1) dans un plan à deux dimensions, on obtient la figure suivante :

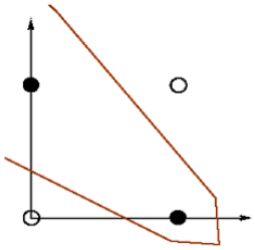

FIGURE 8 – Cas non linéairement séparable

Bien que dans ce cas de figure il est en effet impossible d'avoir un séparateur linéaire on remarque que :

Si on prend une fonction polynomiale

$$\begin{array}{rccc} \phi : & \mathbb{R}^2 & \longrightarrow & \mathbb{R}^3 \\ & (x,y) & \longmapsto & (x,y,xy) \end{array}$$

$$\begin{array}{rcl} \phi(0,0) & = & (0,0,0) \\ \phi(0,1) & = & (0,1,0) \\ \phi(1,0) & = & (1,0,0) \\ \phi(1,1) & = & (1,1,1) \end{array}$$

On passe d'un espace de dimension 2 à un espace de dimension 3, on obtient un problème en trois dimensions linéairement séparable, où l'hyperplan séparateur est le plan en rouge.

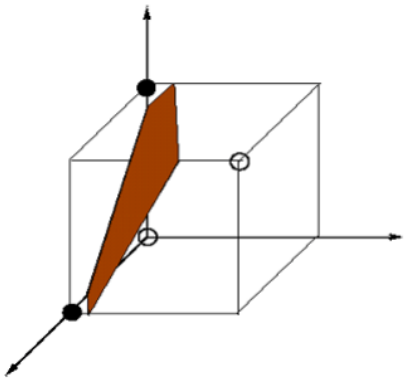

FIGURE 9 – Linéairement séparable après transformation

5.2 Présentation des résultats

De chacune des bases de données `parkinson` et `parkinson_sv` (base de données obtenues après études de corrélations), nous avons créé 4 différents modèles à partir du package `svm.SVC` de `sklearn`. Ceux-ci ne se distinguant les uns aux autres que par les noyaux (le choix du type de noyau du modèle étant à l'origine de la transformation à faire, comme dans l'exemple précédent où il s'agissait d'un noyau polynomial).

Ainsi nous avons fait pour chaque base de données des modèles avec des noyaux de types : linéaires, polynomiaux, sigmoïdaux et radiaux.

Après la création des différents modèles et le calcul des taux de précisions et d'apprentissages nous observons :

Pour la base de données complète :

Type de noyau	Linéaire	Polynomial	Sigmoïdal	Radial
Taux de précision *(en %)*	84.61	78.46	73.84	78.46
Taux d'apprentissage *(en %)*	87.69	86.15	76.15	83.07

Pour la base de données obtenue après sélection :

Type de noyau	Linéaire	Polynomial	Sigmoïdal	Radial
Taux de précision *(en %)*	84.61	76.92	73.84	78.46
Taux d'apprentissage *(en %)*	87.69	83.84	76.15	83.07

5.3 Interprétation et conclusion SVM

Au vu des résultats précédemment énoncés, de part la similarité des taux de précision et d'apprentissage par type de noyau pour des différentes bases, il vient que les variables délaissées après l'étude de corrélation n'influencent pas le modèle car pour un même type de noyau les taux de précisions sont très proches pour les deux bases de données.

En conclusion, nous observons que le modèle présentant le plus grand taux de précision est celui avec un noyau de type linéaire, ce qui laisse penser que nous sommes dans un cas avec séparabilité *(relativement-)* linéaire où les cas de `statuts=1` et `statuts=0` sont séparés par un hyperplan dont l'équation est régie par le modèle.

6 Conclusion et proposition des pistes d'amélioration

Avant d'aboutir à la conclusion, une des pistes pour améliorer le taux de précision est de considérer l'ensemble des modèles entraînés pour aboutir à une nouvelle prédiction.

L'idée est de prédire la valeur à 1 si la majorité des modèles donnent 1 en prédiction.

Dans notre cas, on considère pour chaque x à prédire :

$$Y^{new}(x) := \begin{cases} 1 & \text{si } \frac{1}{M}\sum_{m=1}^{M} Y^m(x) \geq 0.5 \\ 0 & \text{sinon.} \end{cases}$$

où :

- $Y^{new}(x)$ est la nouvelle prédiction associée à x (`Y_pred_new` dans le code)
- M est le nombre de modèles entraînés
- $Y^m(x)$ est la prédiction pour x d'un modèle particulier (`Y_pred_modele` dans le code)

Malheureusement, le taux de précision à l'issu de cette procédure est de seulement 86.15%. Cette valeur est certes plus élevée que la plupart des modèles que nous avons entraînés, mais reste toutefois inférieure à celle du modèle arbre décisionnelle.

Comparaison des modèles

Les précédents modèles que nous avons mis en oeuvre sont résumés dans le graphique ci-dessous. Nous présentons ici uniquement les résultats observés sur la base de données complète `parkinsons` puisque celle-ci donne des résultats meilleurs.

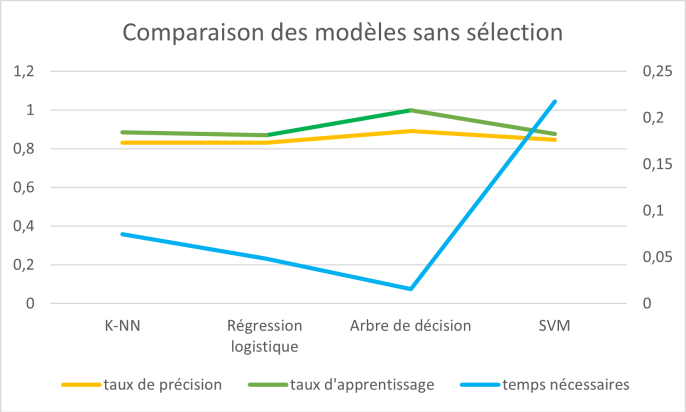

FIGURE 10 – Comparaison des modèles sans sélection

On conclut que l'arbre de décision est plus adapté à notre cas de prédiction de maladie. Son *accuracy* est de 92%, le plus élevé parmi tous les modèles que nous venons d'étudier. Il s'agit aussi du modèle qui a nécessité le moins de temps pour aboutir à l'entraînement du modèle.