INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

GMM 4e année
Mathématiques Appliquées
Année scolaire 2021-2022

MACHINE LEARNING
RAPPORT DE PROJET

Julie Tryoen

Table des matières

1 Introduction **5**

2 Analyse des données **9**
2.1 Statistique descriptive unidimensionnelle 9
2.2 Statistique descriptive multidimensionnelle 10
2.3 Analyse en composantes principales 10

3 Modélisation **15**
3.1 Stratégie générale 15
3.2 Méthodes d'apprentissage 17
3.2.1 Modèles linéaires 17
3.2.2 Analyse discriminante linéaire et SVM linéaire 19
3.2.3 Méthodes à noyaux 21
3.2.4 Arbres de classification et de régression 23
3.2.5 Agrégation et forêts aléatoires 24
3.2.6 Réseaux de neurones 25

4 Application au problème étudié **29**
4.1 Problème de régression 30
4.2 Problème de classification 31
4.2.1 Comparaison des méthodes sur un couple d'échantillons . . . 34
4.2.2 Comparaison des méthodes par validation croisée MC 35

5 Conclusion **39**

Chapitre 1

Introduction

L'objectif de ce projet est de prédire la quantité de pluie à un endroit donné le jour suivant, en se basant sur un échantillon de données météorologiques, observées le jour courant d'une part, et prédites pour le jour suivant d'autre part. On utilise pour cela le fichier **rain.txt** qui contient 688 observations météorologiques d'une station donnée, fournies par Météo France pour le challenge *Défi IA 2022* et extraites de la base de données disponible sur le github de Météo France.

On adopte le point de vue du Machine Learning. Il s'agit donc de résoudre un problème d'apprentissage supervisé à partir d'un échantillon de données. On cherche ainsi à estimer le lien entre des variables en entrée, appelées variables explicatives, et des variables en sortie, appelées variables de réponse, à partir de cet échantillon d'entrées/sorties. Cela permet ensuite d'effectuer une prévision associée à de nouvelles données en entrée.

Les variables explicatives sont les suivantes :

- Les paramètres météorologiques observés le jour courant :
  - `date` : la date du jour courant ;
  - `ff` : la vitesse du vent (en $m.s^{-1}$) ;
  - `t` : la température (en Kelvin K) ;
  - `td` : le point de rosée (en Kelvin K) ;
  - `hu` : l'humidité (en %) ;
  - `dd` : la direction du vent (en degrés) ;
  - `precip` : la quantité totale de précipitation (en $kg.m^{-2}$).
- Les paramètres météorologiques de prévision pour le jour suivant donnés par le modèle AROME de Météo France :
  - `ws_arome` : la vitesse du vent (en $m.s^{-1}$) ;
  - `p301_arome` : la direction du vent (en degrés) ;
  - `u10_arome` et `v10_arome` : les composantes du vent U (d'ouest en est) et V (du sud au nord) au niveau vertical de $10m$;

— `t2m_arome` : la température au niveau vertical de $2m$ (en K) ;
— `d2m_arome` : le point de rosée au niveau vertical de $2m$ (en K) ;
— `r_arome` : l'humidité (en %) ;
— `tp_arome` : la quantité totale de précipitation (en $kg.m^{-2}$) ;
— `msl_arome` : la pression au niveau de la mer (en Pa).

Les variables de réponse sont quant à elles :

- `rain` (quantitative) : la quantité totale de pluie pour le jour suivant (en $kg.m^{-2}$) ;
- `rain_class` (qualitative) : une variable catégorielle créée artificiellement composée de trois classes de pluie pour le jour suivant qui sont `no_rain` (si `rain` $= 0$), `low_rain` (si $0 <$ `rain` $<= 2$), et `high_rain` (si `rain` > 2).

Le problème considéré peut être résolu de deux manières différentes, selon que l'on considère comme variable de réponse la quantité totale de pluie ou la classe de pluie pour le jour suivant. Dans le premier cas, il s'agit d'un problème de régression, alors que dans le deuxième cas il s'agit d'un problème de classification.

On note :

- $\boldsymbol{x} = (x^1, \ldots, x^p) \in \mathcal{X}$ le vecteur des variables explicatives ($p = 16$ ici). Ces variables sont toutes quantitatives et appartiennent à $\mathbb{R}$, exceptée la première qui est qualitative.
- $y \in \mathcal{Y}$ la variable de réponse, quantitative dans $\mathbb{R}$ dans le cas du problème de régression, catégorielle composée de trois classes dans le cas du problème de classification.

L'objectif est donc d'estimer le lien entre le vecteur d'entrée $\boldsymbol{x}$ et la variable de sortie y : $y = f(\boldsymbol{x})$, avec $f : \mathcal{X} \to \mathcal{Y}$.
On dispose pour cela d'un échantillon composé de N observations de données de type entrée/sortie ($N = 688$ ici) :

$$\boldsymbol{d}^N = \{((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$$

où, pour tout $i = 1, \ldots, N$, $\boldsymbol{x}_i = (x_i^1, \ldots, x_i^p) \in \mathcal{X}$ et $y_i \in \mathcal{Y}$.
On note ce N-échantillon

$$\boldsymbol{D}^N = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_N, Y_N)\},$$

de loi de distribution inconnue P sur $\mathcal{X} \times \mathcal{Y}$.
Une règle de prédiction est alors une fonction mesurable $\hat{f} : \mathcal{X} \to \mathcal{Y}$, construite à partir de cet échantillon, qui associe la sortie $\hat{f}(\boldsymbol{x}) \in \mathcal{Y}$ à l'entrée $\boldsymbol{x} \in \mathcal{X}$. Elle dépend de $\boldsymbol{D}^N$ et est donc aléatoire.

Avant cette étape de modélisation, une étude exploratoire préliminaire doit être réalisée afin de contrôler et de comprendre les données. C'est l'objet de la première partie de ce rapport.

La seconde partie de ce rapport consiste à présenter la stratégie générale utilisée pour élaborer une règle de prédiction optimale et à décrire les différentes méthodes d'apprentissage existantes.

Ces méthodes sont finalement appliquées dans une troisième partie sur le problème étudié, successivement aux deux approches, régression et classification. Différents modèles sont construits et les règles de prédiction associées sont comparées, pour finalement dégager un modèle optimal permettant de prédire la quantité de pluie le jour suivant.

Pour réaliser ce projet, j'ai utilisé le langage de programmation Python et la librairie *scikit-learn*. Ce projet aurait également pu être réalisé avec le logiciel R. Python conduit à des résultats moins complets et moins interprétables mais avec une vitesse d'exécution plus rapide.

Chapitre 2

Analyse des données

L'objectif de cette première partie est de contrôler et de comprendre les données. Plus précisément, il s'agit de s'assurer de la bonne cohérence des données, de proposer d'éventuelles transformations et d'analyser les structures de corrélations ou plus généralement de liaisons entre les variables, les groupes d'individus ou les observations.

La variable catégorielle `date` est d'abord convertie en une variable `mois`, ce qui permet d'obtenir une variable catégorielle composée de douze classes.

On s'intéresse ensuite aux variables quantitatives. On réalise successivement une analyse statistique descriptive unidimensionnelle, une analyse descriptive multidimensionnelle puis une analyse en composantes principales.

2.1 Statistique descriptive unidimensionnelle

Pour chacune des variables quantitatives, on calcule les indicateurs statistiques unidimensionnels et on trace le boxplot et l'histogramme associés aux observations. Cela nous permet d'avoir une idée de la distribution de la variable et de noter la symétrie ou non de celle-ci. Pour certaines méthodes à venir de modélisation, telles que les modèles linéaires, il est en effet nécessaire d'avoir des distributions proches de distributions gaussiennes et donc relativement symétriques.

En observant les boîtes à moustache et les histogrammes sur la Figure 2.1, on remarque que deux variables en entrée, `precip` et `tp_arome`, ainsi que la variable de sortie, `rain`, présentent des distributions très asymétriques à droite avec un grand nombre de valeurs égales à zero. Il s'agit en fait des variables associées aux quantités de précipitation. Pour palier à ce problème, on propose la transformation de variable $X \mapsto \log(1 + X)$, qui permet de décentrer légèrement la distribution vers la droite. Les variables sont alors renommées en `Lprecip`, `Ltp_arome` et `Lrain`.

2.2 Statistique descriptive multidimensionnelle

On poursuit l'analyse de données en observant les structures de corrélation entre les variables. Pour cela, on réalise les graphiques des nuages de points et de la matrice de corrélation (Figure 2.2).

Ces graphiques suggèrent une corrélation linéaire entre certaines variables : `ff` et `ws_arome` ; `t`, `td`, `t2m_arome` et `d2m_arome` ; `hu` et `r_arome`. Par contre, la variable de sortie `Lrain` n'est pas corrélée linéairement avec les variables en entrée, ce qui présage de résultats non satisfaisants pour les modèles linéaires de régression.

2.3 Analyse en composantes principales

On réalise pour terminer une analyse en composantes principales sur les variables quantitatives en entrée. Le graphique de décroissance de la variance expliquée (ébouli des valeurs propres) et les diagrammes à moustaches des variables principales (Figure 2.3) suggèrent de retenir une dimension quatre pour les approximations.

On représente sur les graphiques de la Figure 2.4 les individus par leurs projections sur deux des trois premiers vecteurs principaux avec trois couleurs différentes selon que l'individu correspond à la classe de pluie `no_rain`, `low_rain` ou `high_rain`. On remarque que les trois groupes d'individus se mélangent et on pressent donc qu'il va être difficile de discriminer les trois classes lors de l'utilisation des méthodes de classification.

Pour finir, on représente les variables initiales sur deux des trois premiers axes factoriels sur les graphiques de la Figure 2.5. On observe que la variable `msl_arome` est vraiment décorrélée des autres variables. De plus, on retrouve les corrélations entre les variables mentionnées dans la section précédente, en particulier la corrélation entre les variables `t`, `td`, `t2m_arome` et `d2m_arome`, qui sont très bien représentées par leur projection sur les deux premiers axes factoriels.

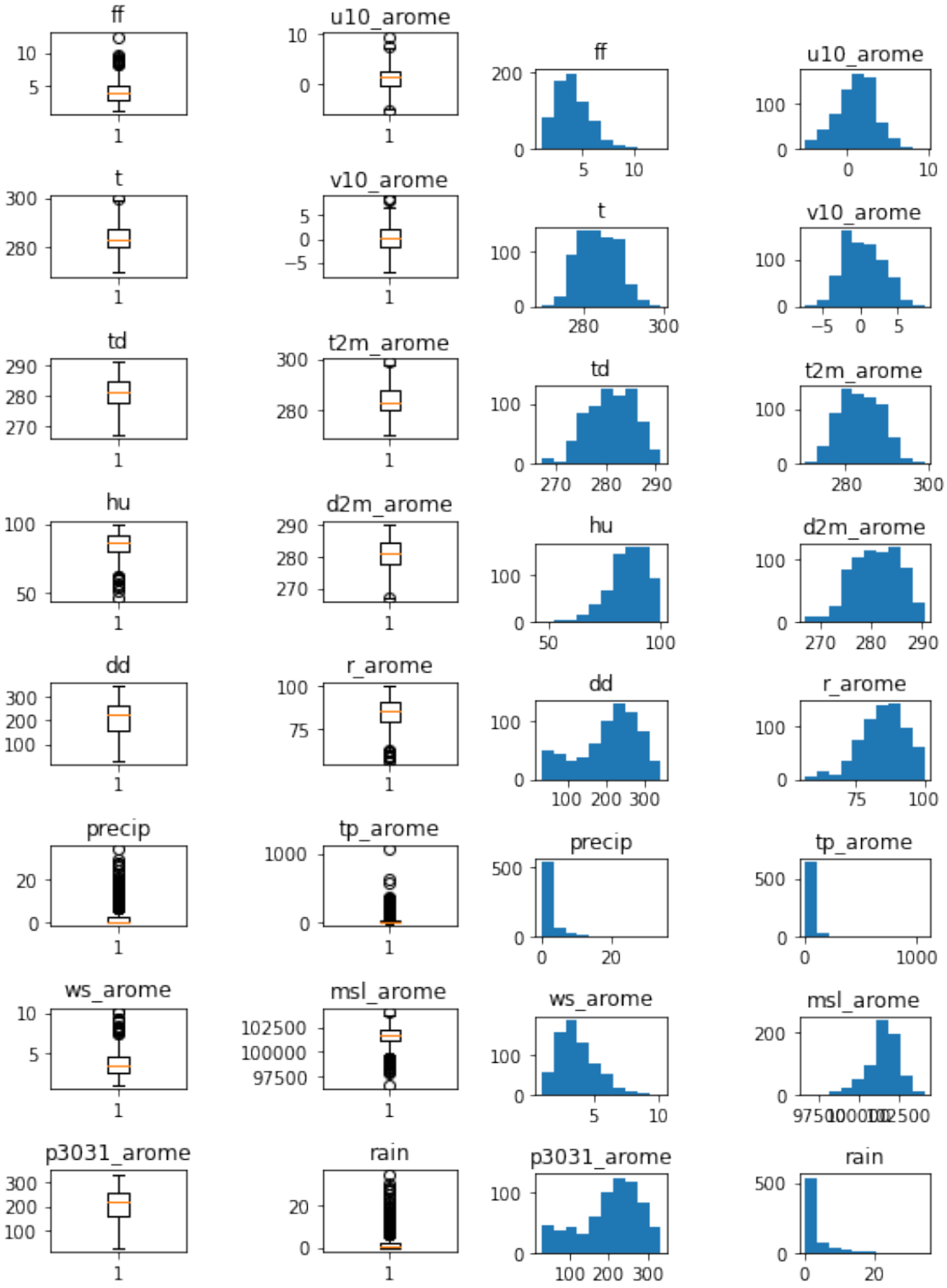

FIGURE 2.1 – Diagrammes à moustaches et histogrammes des variables quantitatives

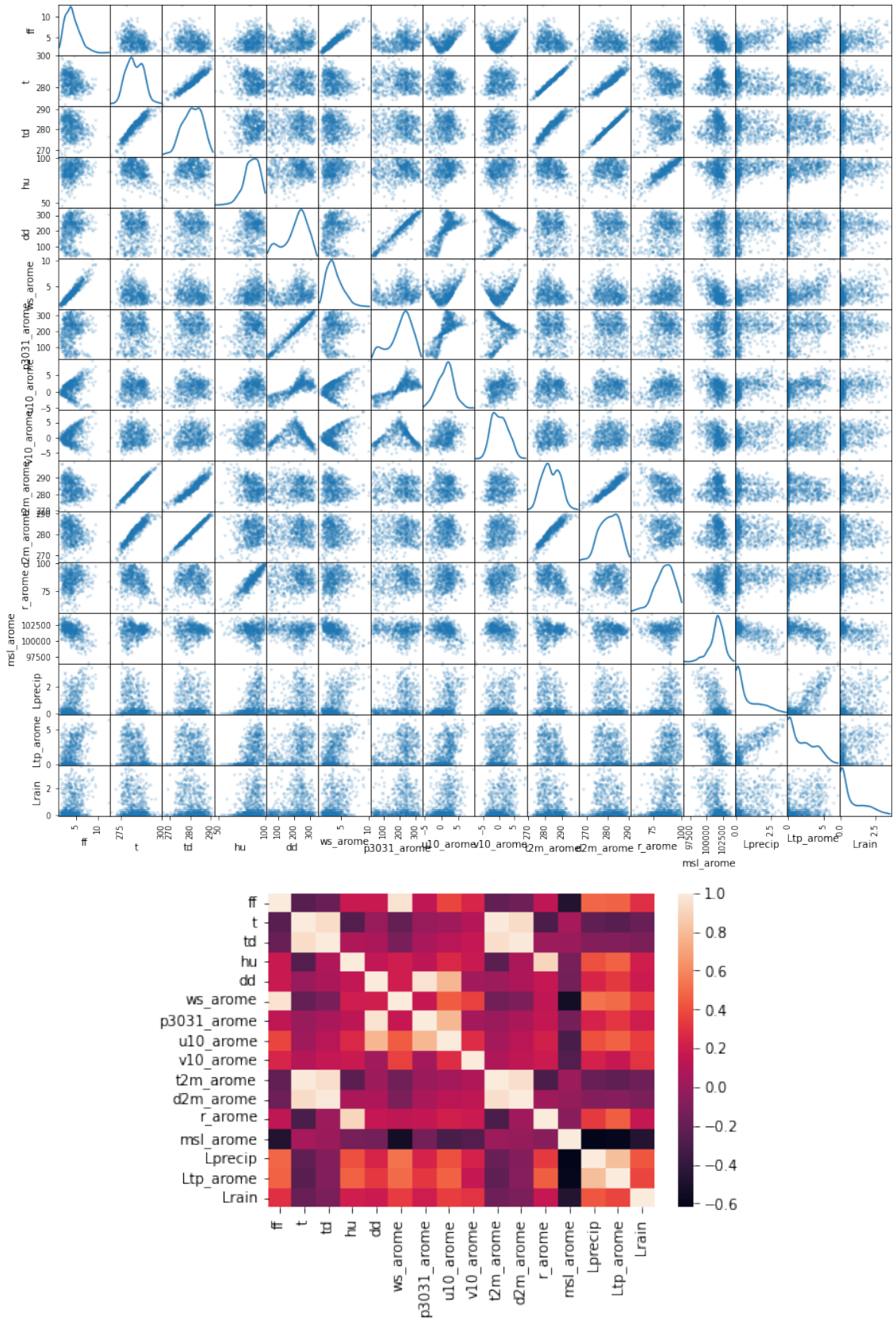

FIGURE 2.2 – Nuages de points et matrices de corrélation des variables quantitatives

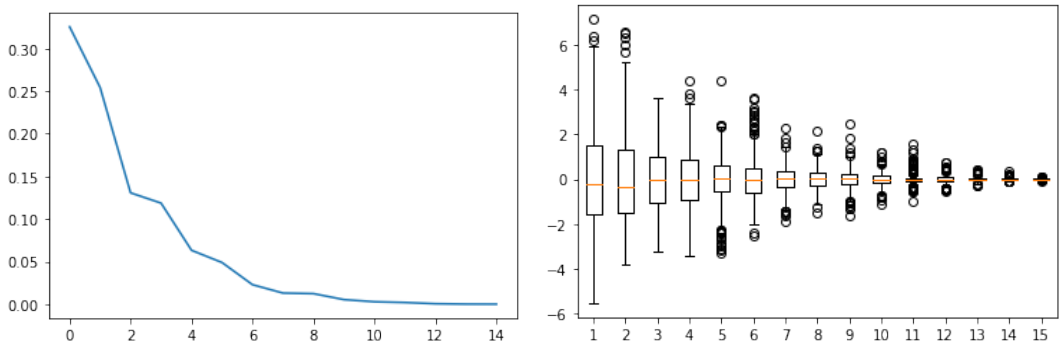

FIGURE 2.3 – Décroissance de la variance expliquée et diagrammes à moustaches des variables principales

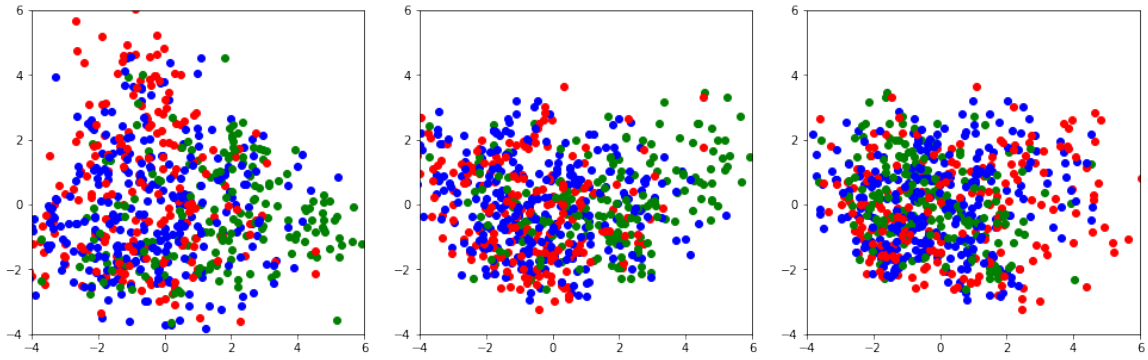

FIGURE 2.4 – Projection des individus sur les premier et deuxième vecteurs principaux à gauche, sur les premier et troisième vecteurs principaux au centre, sur les deuxième et troisième vecteurs principaux à droite. L'individu est coloré en rouge si la classe de pluie correspondante est `no_rain`, en bleu si la classe de pluie correspondante est `low_rain` et en vert si la classe de pluie correspondante est `high_rain`.

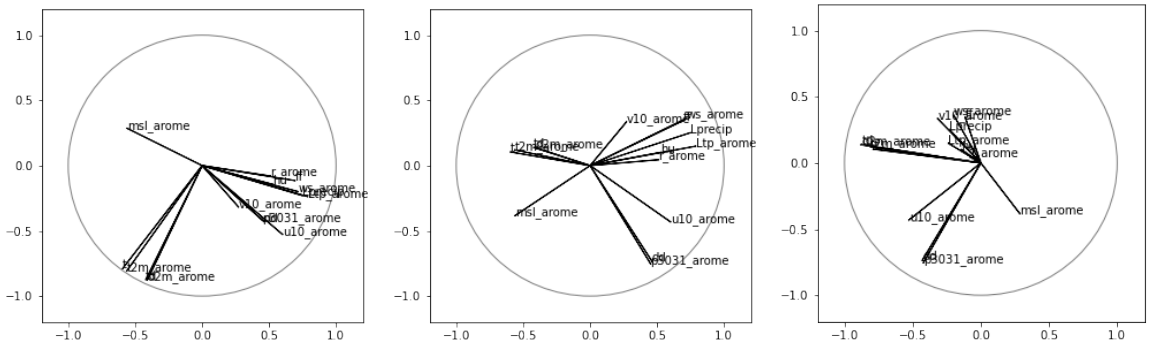

FIGURE 2.5 – Projection des variables initiales sur les premier et deuxième axes factoriels à gauche, sur les premier et troisième axes factoriels au centre, sur les deuxième et troisième axes factoriels à droite.

Chapitre 3

Modélisation

Après l'étape préliminaire d'analyse des données, on s'attaque au coeur du problème qui consiste à contruire une règle de prédition optimale $\hat{f} : \mathcal{X} \to \mathcal{Y}$ qui associe la sortie $\hat{f}(\boldsymbol{x}) \in \mathcal{Y}$ à l'entrée $\boldsymbol{x} \in \mathcal{X}$, à partir de l'échantillon observé $\boldsymbol{D}^N = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_N, Y_N)\}$.

Cette deuxième partie s'attache tout d'abord à expliquer la stratégie générale utilisée en apprentissage supervisé pour construire cette règle de prédiction optimale. Elle détaille ensuite les différentes méthodes d'apprentissage existantes.

3.1 Stratégie générale

La première étape est de séparer l'échantillon $\boldsymbol{D}^N$ en deux parties : l'échantillon d'apprentissage et l'échantillon test. On procède à un tirage aléatoire d'un échantillon test de n observations correspondant à environ un cinquième de l'échantillon initial, qui ne sera utilisé que lors de la dernière étape pour estimer l'erreur de prédiction et comparer les méthodes. La partie restante est l'échantillon d'apprentissage, qui va nous permettre de construire un modèle et une règle de prédiction. On le note $\boldsymbol{D}^n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$.

Pour chacune des méthodes décrites dans la section suivante, on construit un modèle en optimisant ses paramètres et/ou sa complexité de façon à minimiser une estimation "sans biais" de l'erreur de prédiction évaluée sur l'échantillon d'apprentissage.

On définit pour estimer cette erreur une fonction de perte $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, telle que $l(y, y) = 0$ et $l(y, y') > 0$ pour $y \neq y'$. En régression, on utilise classiquement la perte $\mathbb{L}^p$: $l(y, y') = |y - y'|^p$, en particulier la perte $\mathbb{L}^2$ ou perte quadratique, et en classification binaire la perte 0-1 : $l(y, y') = \mathbb{1}_{y \neq y'}$.
En classification multiclasses - c'est le cas ici si on cherche à prédire l'une des

trois classes de pluie - on considère $Y \in \{1, \dots, K\}$. Les algorithmes fournissent généralement une estimation des probabilités $f_k(\boldsymbol{x}) = \mathbb{P}(Y = k|\boldsymbol{X} = \boldsymbol{x})$ et une règle de prédiction $\hat{f}(\boldsymbol{x}) = \text{argmax}_{k\in\{1,\dots,K\}}\hat{f}_k(\boldsymbol{x})$ proche de celle de Bayes, qui assigne à une entrée $\boldsymbol{x}$ la classe la plus probable, c'est-à-dire celle qui correspond à la probabilité estimée la plus grande. On utilise alors la fonction de perte de l'entropie croisée :

$$l(Y, \hat{f}(\boldsymbol{X})) = -\sum_{k=1}^{K} \mathbb{1}_{Y=k}\log(\hat{f}(\boldsymbol{X})).$$

Dans tous les cas, l'objectif est de minimiser l'espérance de la fonction de perte, appelée risque ou erreur de généralisation :

$$R_P(f) = \mathbb{E}_{(\boldsymbol{X},Y)\sim P}\left[l(Y, f(\boldsymbol{X}))\right],$$

où f est une règle de prédiction construite à partir de $\boldsymbol{D}^n$ et $(\boldsymbol{X}, Y)$ est indépendant de $\boldsymbol{D}^n$. Son estimateur empirique, appelé risque empirique ou erreur d'apprentissage :

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n}\left[l(Y_i, f(\boldsymbol{X}_i))\right],$$

est biaisé car il est calculé en utilisant $\boldsymbol{D}^n$ et sous-estime le risque.

Pour obtenir une estimation "sans biais" du risque, deux stratégies sont possibles. La première consiste à ajouter au risque empirique un terme de pénalité pour éviter la sélection de modèles trop complexes (phénomène de sur-ajustement), qui induisent une règle de prédiction avec un biais faible mais une variance importante. On peut par exemple chercher le modèle qui minimise le critère de Mallow, AIC ou BIC. La seconde consiste à estimer l'erreur de généralisation sur des données qui n'ont pas été utilisées durant la phase d'apprentissage, en ayant recours à des méthodes de validation croisée ou des méthodes *bootstrap*.

On utilise dans le projet la méthode de validation croisée *K-fold* qui fonctionne de la manière suivante : on découpe aléatoirement l'échantillon d'apprentissage en K sous-échantillons de tailles sensiblement identiques ($K = 10$ en général). Chacun de ces folds va être utilisé successivement comme échantillon de validation. Lorsque le fold k est l'échantillon de validation, on construit le modèle correspondant aux $K-1$ autres folds et on évalue ensuite la fonction de perte de ce modèle sur chaque élément du fold k. Cette opération est réalisée pour tous les folds k et on calcule ensuite une estimation globale de l'erreur. Finalement, le modèle optimal parmi la collection de modèles correspondant à des paramètres ou des niveaux de complexité différents est celui qui minimise cette estimation d'erreur de généralisation obtenue par validation croisée.

La dernière étape est d'apprécier la performance de ce modèle optimal en utilisant l'échantillon test pour évaluer l'erreur de généralisation. Pour cela, on calcule les prédictions associées aux données en entrée de l'échantillon test et on procède

comme suit. Dans le cas du problème de régression, on trace le graphique des résidus et on calcule l'erreur moyenne quadratique MSE et le coefficient R^2 entre les valeurs prédites et observées. Dans le cas du problème de classification, on détermine le taux de mal classés et la matrice de confusion. On peut également tracer la courbe ROC lorsqu'il s'agit d'un problème de classification binaire. Ces différents indicateurs vont nous permettre *in fine* de comparer les méthodes entre elles et de dégager celle qui est la plus adaptée au problème considéré.

3.2 Méthodes d'apprentissage

Maintenant que la stratégie générale a été expliquée, détaillons les méthodes d'apprentissage existantes.

3.2.1 Modèles linéaires

Régression linéaire

On considère dans cette section une variable de réponse quantitative à valeur réelle Y, à exprimer en fonction du vecteur des p variables explicatives $\boldsymbol{X} = (\boldsymbol{X}^1, \ldots, \boldsymbol{X}^p)$, et on fait l'hypothèse que la fonction de régression $\mathbb{E}[Y|\boldsymbol{X}]$ est linéaire par rapport à $\{\mathbf{1}, \boldsymbol{X}^1, \ldots, \boldsymbol{X}^p\}$, où $\mathbf{1}$ est le vecteur de $\mathbb{R}^n$ avec toutes les composantes égales à 1.

Le modèle linéaire est alors classiquement défini par :

$$Y_i = \beta_0 + \beta_1 X_i^1 + \cdots + \beta_p X_i^p + \epsilon_i, \text{ pour tout } i = 1, \ldots, n,$$

où $(X_i^1, \ldots, X_i^p, Y_i)$ correspond à la i-ième observation du n-échantillon $\boldsymbol{D}^n$ et les ϵ_i sont des variables aléatoires i.i.d. centrées de variance σ^2 et indépendantes de $\boldsymbol{X}$. Ces erreurs peuvent être supposées gaussiennes. De plus, le vecteur des paramètres inconnus $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)'$ est constant.

En notant $\boldsymbol{Y}$ le vecteur de terme général Y_i, $\boldsymbol{\epsilon}$ le vecteur de terme général ϵ_i et $\mathbb{X}$ la matrice de terme général X_i^j avec pour première colonne le vecteur $\mathbf{1}$, la formulation matricielle du modèle linéaire est :

$$\boldsymbol{Y} = \mathbb{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

Un estimateur de $\boldsymbol{\beta}$ est obtenu en minimisant le carré de la somme des résidus (critère des moindres carrés) :

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta} \in \mathbb{R}^{p+1}} ||\boldsymbol{Y} - \mathbb{X}\boldsymbol{\beta}||_2^2,$$

dont on peut calculer une solution explicite. La réponse du modèle associée à une nouvelle entrée $\boldsymbol{X}_0 = (1, X_0^1, \ldots, X_0^p)$ peut alors être prédite de la manière suivante :

$$\widehat{Y_0} = \boldsymbol{X}_0 \hat{\boldsymbol{\beta}}.$$

Cet estimateur est sans biais mais sa variance peut être importante. Il est donc préférable de considérer des modèles biaisés mais avec une variance moindre. Pour cela, deux stratégies peuvent être employées.

La première est d'induire une réduction du nombre de variables explicatives et par conséquent de simplifier le modèle. Une sélection de variables peut ainsi être opérée comme suit : on définit une classe de modèles, chacun correspondant à un sous-ensemble de variables explicatives utilisées pour définir le modèle linéaire. On retient alors le modèle qui maximise le coefficient R^2 ajusté, ou qui minimise le critère de Mallow ou BIC.

La seconde stratégie consiste à réduire les valeurs des paramètres du modèle $\beta_1, \ldots, \beta_p$ en leur ajoutant des contraintes : il s'agit des méthodes de régression Ridge ou Lasso. À noter que la méthode Lasso induit également une sélection de variables car certains paramètres sont mis à zéro.

Les estimateurs de Ridge et de Lasso sont obtenus en minimisant le critère des moindres carrés avec un terme additif de pénalité, respectivement l_2 pour Ridge et l_1 pour Lasso :

$$\hat{\boldsymbol{\beta}}_\lambda^R = \text{argmin}_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \left(||\boldsymbol{Y} - \mathbb{X}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_2^2 \right),$$

$$\hat{\boldsymbol{\beta}}_\lambda^L = \text{argmin}_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \left(||\boldsymbol{Y} - \mathbb{X}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_1 \right).$$

Ces estimateurs dépendent d'un paramètre positif λ à calibrer, en ayant recours par exemple à la méthode de validation croisée. L'estimateur de Ridge a une solution explicite, tandis que l'estimateur de Lasso est obtenu en résolvant un problème d'optimisation. C'est ce dernier qui sera utilisé dans le projet.

Régression logistique

Le pendant des méthodes de régression linéaire pour les problèmes de classification est la méthode de régression logistique. Elle s'applique aux problèmes de classification binaire, mais peut être généralisée aux problèmes de classification à K classes : on construit un classifieur pour chacune des classes contre toutes les autres et on prédit la classe pour une nouvelle entrée par un vote majoritaire - la classe retenue est celle pour laquelle la probabilité estimée est la plus grande.

L'idée est de considérer un modèle linéaire pour les probabilités en ayant recours à une fonction de lien bijective entre $[0; 1]$ et $\mathbb{R}$, la plus utilisée étant la fonction logit et son inverse la fonction sigmoid. On considère $\mathcal{X} = \mathbb{R}^p$ muni du produit scalaire usuel et $\mathcal{Y} = \{-1, 1\}$ et on cherche à estimer

$$\pi_{\boldsymbol{\beta}}(\boldsymbol{x}) = \mathbb{P}_{\boldsymbol{\beta}}(Y = 1 | \boldsymbol{X} = \boldsymbol{x}) = \frac{\exp(\langle \boldsymbol{\beta}, \boldsymbol{x} \rangle)}{1 + \exp(\langle \boldsymbol{\beta}, \boldsymbol{x} \rangle)}, \text{ pour tout } \boldsymbol{x} \in \mathcal{X}, \text{ avec } \boldsymbol{\beta} \in \mathbb{R}^p.$$

Soit $\boldsymbol{D}^n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ un n-échantillon d'observations. La distribution de $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ sachant $\boldsymbol{X} = \boldsymbol{x}$ avec $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$ est alors

une distribution de Bernouilli et le paramètre $\boldsymbol{\beta}$ est estimé en maximisant la vraisemblance conditionnelle de $\boldsymbol{Y}$ sachant $\boldsymbol{X}$. Plus précisément, on détermine par un algorithme de Newton-Raphson l'estimateur $\hat{\boldsymbol{\beta}}$ qui satisfait $S(\boldsymbol{Y}, \hat{\boldsymbol{\beta}}) = 0$, où S est la fonction score correspondant au gradient du logarithme de la vraisemblance. Finalement, la règle de prédiction associée est le classifieur $\hat{f}(\boldsymbol{x}) = \text{sign}(\left\langle \hat{\boldsymbol{\beta}}, \boldsymbol{x} \right\rangle)$, qui associe -1 ou 1, et donc une des deux classes, à une entrée $\boldsymbol{x} \in \mathcal{X}$.

À noter qu'une sélection de variables ou une procédure de sélection de modèle peut être opérée comme en régression linéaire, ce qui peut s'avérer utile en particulier lorsque le nombre de variables explicatives est important.

3.2.2 Analyse discriminante linéaire et SVM linéaire

Les deux méthodes développées dans cette section sont définies pour des problèmes de classification.

Analyse discriminante linéaire

On considère une variable de réponse quantitative à K classes $Y \in \{1, \dots, K\}$ et on cherche à estimer les probabilités $f_k(\boldsymbol{x}) = \mathbb{P}(Y = k | \boldsymbol{X} = \boldsymbol{x})$, de manière à définir une règle de prédiction $\hat{f}(\boldsymbol{x}) = \text{argmax}_{k \in \{1,\dots,K\}} f_k(\boldsymbol{x})$ proche de celle de Bayes.

On fait l'hypothèse que la distribution de $\boldsymbol{X}$ sachant $Y = k$ est une distribution normale multivariée, de moyenne $\boldsymbol{\mu}_k$ et de matrice de covariance $\boldsymbol{\Sigma}_k$. Pour l'analyse discriminante linéaire, on fait l'hypothèse supplémentaire que la matrice de covariance est la même pour tout k : $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$. Dans ce cas, on a :

$$\log f_k(\boldsymbol{x}) = C(x) + \delta_k,$$

où $C(x)$ ne dépend pas de la classe et δ_k s'exprime en fonction de $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}$ et $\log(\pi_k)$, avec $\pi_k = \mathbb{P}(Y = k)$. La règle de Bayes assigne à $\boldsymbol{x} \in \mathcal{X}$ la classe $f^*(\boldsymbol{x})$ qui maximise $\delta_k(\boldsymbol{x})$.

La règle de décision est par conséquent construite à partir d'un n-échantillon $\boldsymbol{D}^n$ en calculant, pour tout k, des estimateurs $\hat{\pi}_k$, $\hat{\boldsymbol{\mu}}_k$ et $\hat{\boldsymbol{\Sigma}}$, et on assigne à $\boldsymbol{x} \in \mathcal{X}$ la classe $\hat{f}(\boldsymbol{x})$ qui maximise $\hat{\delta}_k(\boldsymbol{x})$. La frontière de séparation entre deux classes k et l est alors un hyperplan. Les paramètres $\boldsymbol{\mu}_k$ et $\boldsymbol{\Sigma}$ du modèle peuvent être optimisés par une procédure de sélection de modèle par validation croisée.

Finalement, l'hypothèse de matrice de covariance constante peut être relaxée en faisant dépendre celle-ci de la classe. On obtient alors une fonction de discrimination quadratique.

SVM linéaire

Tout comme la régression logistique et l'analyse discriminante, les méthodes SVM (machines à vecteurs de support en français) sont des méthodes de discrimi-

nation basées sur l'échantillon d'apprentissage, cependant elles ne font pas d'hypothèse sur la distribution de celui-ci - pour rappel la régression logistique se base sur un modèle binomial et l'analyse discriminante sur un modèle gaussien. Elles s'appliquent aux problèmes de classification binaire, mais peuvent être généralisées aux problèmes de classification à K classes par vote majoritaire.

On considère $\mathcal{X} = \mathbb{R}^p$ muni du produit scalaire usuel et $\mathcal{Y} = \{-1, 1\}$ et on fait l'hypothèse dans un premier temps que l'échantillon d'apprentissage $\boldsymbol{d}^n = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ est linéairement séparable en deux classes, c'est-à-dire qu'il existe $\boldsymbol{w} \in \mathbb{R}^p$ et $b \in \mathbb{R}$, tel que $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$ est un hyperplan séparateur ($\boldsymbol{w}$ est un vecteur orthogonal). La règle de classification choisie est

$$f_{\boldsymbol{w},b}(\boldsymbol{x}) = \mathbf{1}_{\langle \boldsymbol{w},\boldsymbol{x} \rangle + b \geq 0} - \mathbf{1}_{\langle \boldsymbol{w},\boldsymbol{x} \rangle + b < 0},$$

et l'hyperplan choisi est celui qui maximise la marge γ entre les deux classes : cet hyperplan est appelé hyperplan canonique et vérifie $\gamma = \frac{1}{||\boldsymbol{w}||}$.

Trouver cet hyperplan revient à :

$$\text{minimiser } \frac{1}{2}||\boldsymbol{w}||^2 \text{ sous la contrainte } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 \text{ pour tout } i.$$

Il s'agit d'un problème classique d'optimisation convexe avec des contraintes linéaires, pour lequel il existe un unique minimiseur global. Il peut être résolu par l'approche duale en introduisant des multiplicateurs de Lagrange $\boldsymbol{a} = (\alpha_1, \ldots, \alpha_n)$ et est équivalent à résoudre le problème dual :

$$\text{maximiser } \theta(\boldsymbol{a}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$

sous la contrainte $\sum_{i=1}^{n} \alpha_i y_i = 0$ et $\alpha_i \geq 0$, pour tout i.
La solution $\boldsymbol{a}^*$ du problème dual est alors obtenue avec des algorithmes d'optimisation classique et les solutions $\boldsymbol{w}^*$ et b^* du problème primal en sont déduites. L'hyperplan séparateur retenu pour la règle de discrimination a pour équation $\langle \boldsymbol{w}^*, \boldsymbol{x} \rangle + b^* = 0$ avec $\boldsymbol{w}^* = \sum \alpha_i^* y_i \boldsymbol{x}_i$ et la règle de classification est la suivante :

$$\hat{f}(\boldsymbol{x}) = \mathbf{1}_{\sum y_i \alpha_i^* \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + b^* \geq 0} - \mathbf{1}_{\sum y_i \alpha_i^* \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + b^* < 0}.$$

De par les conditions de Karush-Kuhn-Tucker du problème dual, seuls les $\boldsymbol{x}_i$ tels que $\alpha_i > 0$ interviennent dans la définition de l'hyperplan ; ils sont appelés vecteurs supports et sont situés sur la frontière définie par la marge maximale. Cette méthode est donc efficace en terme de complexité quand la solution du problème dual comporte un nombre restreint de vecteurs supports. Cependant, le principal inconvénient de cette méthode, en plus de considérer que les classes sont linéairement séparables, est qu'elle est très sensible aux observations aberrantes.

Pour palier à ce problème, on autorise certains points à être dans la marge, même du mauvais côté. Pour cela, on introduit la variable *slack* $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)$ et

la contrainte $y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1$ devient $y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i$, avec $\xi_i \geq 0$, pour tout i. Si $\xi_i \in [0; 1]$, le point est bien classé mais dans la région définie par la marge et si $\xi_i > 1$, le point est mal classé. La marge est alors appelée marge souple. Le problème primal devient alors :

$$\text{minimiser } \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n} \xi_i$$

sous les contraintes $y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i$ et $\xi_i \geq 0$ pour tout i.
C est un paramètre positif à calibrer, qui va contrôler la tolérance aux erreurs de classification. La solution du problème dual $\boldsymbol{a}^*$, avec $0 \leq \alpha_i^* \leq C$, pour tout i, est à nouveau obtenue avec des algorithmes d'optimisation classique. On en déduit les solutions $\boldsymbol{w}^*$, b^* et ξ^* du problème primal, puis l'hyperplan séparateur et la règle de classification correspondants.

3.2.3 Méthodes à noyaux

Machines à vecteurs de support (SVM)

En général, l'échantillon n'est pas linéairement séparable et l'utilisation de la méthode SVM linéaire, même avec une marge souple, n'est pas performante. La procédure de classification peut être rendue plus flexible en envoyant les observations des variables d'entrée $\boldsymbol{x}_i$, pour $i = 1, \ldots, n$ dans un espace de Hilbert $\mathcal{H}$ via une transformation non linéaire définie par une fonction ϕ, de sorte que le nouvel échantillon obtenu $\{\phi(\boldsymbol{x}_i), y_i\}$ puisse être considéré comme linéairement séparable. La méthode SVM linéaire est alors appliquée sur ce nouvel échantillon.

En pratique, l'espace $\mathcal{H}$ et la fonction ϕ n'ont pas besoin d'être connus. En effet, la règle de classification devient :

$$\hat{f}(\boldsymbol{x}) = \mathbf{1}_{\sum y_i \alpha_i^* \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle + b^* \geq 0} - \mathbf{1}_{\sum y_i \alpha_i^* \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle + b^* < 0},$$

où $\boldsymbol{a}^* = \{\alpha_1, \ldots, \alpha_n\}$ est la solution du problème dual dans l'espace $\mathcal{H}$. De fait, elle ne dépend de la fonction ϕ que par l'intermédiaire des produits scalaires de la forme $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle$ et $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$. L'astuce consiste à définir une fonction $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, appelée noyau, telle que $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$, qui vérifie la condition de Mercer : la matrice de terme général $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ doit être symétrique et définie positive. L'espace $\mathcal{H}$ associé est appelé espace de Hilbert à noyau reproduisant (RKHS). Le noyau usuel employé en SVM est le noyau gaussien : $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\sigma^2}}$, où l'écart-type σ est un paramètre à calibrer.

La règle de prédiction retenue est obtenue en minimisant un risque empirique convexifié avec une pénalité pour une fonction perte l convexe (en général on choisit celle associée à la hinge loss) :

$$f^* = \text{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} l(y_i, f(\boldsymbol{x}_i)) + \frac{1}{2\lambda n}||\boldsymbol{w}||^2,$$

où $\mathcal{F} = \{\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b, \boldsymbol{w} \in \mathbb{R}^p, b \in \mathbb{R}\}$ et λ est un paramètre positif de pénalisation à calibrer.

Régression à vecteurs de support (SVR)

Les méthodes à noyaux peuvent également être utilisées dans le cadre des problèmes de régression. On considère $\mathcal{X} = \mathbb{R}^p$ muni du produit scalaire usuel et $\mathcal{Y} = \mathbb{R}$. La méthode de régression à vecteurs de support ϵ (ϵ-SVR) est basée sur la recherche d'une fonction f la plus "plate" possible telle que, pour tout i, l'erreur commise entre $f(\boldsymbol{x}_i)$ et y_i soit au plus égale à ϵ.

Dans le cas linéaire, un prédicteur est : $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$, avec $\boldsymbol{w} \in \mathbb{R}^p$ et $b \in \mathbb{R}$ et le problème d'optimisation convexe associé est :

$$\text{minimiser } \frac{1}{2}||\boldsymbol{w}||^2 \text{ sous les contraintes, pour tout } i,$$

$$y_i - (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \leq \epsilon \text{ et } -y_i + (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \leq \epsilon.$$

De cette façon, on accepte les erreurs plus petites que ϵ, mais on n'autorise pas les erreurs supérieures à ϵ. Pour accorder plus de flexibilité à la méthode, on introduit comme en SVM des variables *slack* $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)$ et $\boldsymbol{\xi'} = (\xi'_1, \ldots, \xi'_n)$ et on résout le problème d'optimisation convexe :

$$\text{minimiser } \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}(\xi_i + \xi'_i) \text{ sous les contraintes, pour tout } i,$$

$$y_i - (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \leq \epsilon + \xi_i \text{ et } -y_i + (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \leq \epsilon + \xi'_i \text{ , avec } \xi_i, \xi'_i \geq 0.$$

Comme précédemment, seuls les produits scalaires $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle$ et $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$ interviennent dans la solution, ce qui permet de définir un prédicteur f non linéaire en ayant recours à un noyau.

Régression du noyau au sens des moindres carrés (KRLS)

Une autre méthode de régression basée sur les noyaux est la procédure de régression du noyau au sens des moindres carrés. Soit $\boldsymbol{D}^n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ un n-échantillon d'observations avec $\boldsymbol{X}_i \in \mathbb{R}^p$ et $Y_i \in \mathbb{R}$. Le prédicteur est cherché sous la forme $f(\boldsymbol{x}) = \sum_{i=1}^{n} c_j k(\boldsymbol{X}_j, x)$, avec $\boldsymbol{c} \in \mathbb{R}^n$ et k un noyau défini positif.

Soit $\boldsymbol{K}$ la matrice de terme général $K_{i,j} = k(\boldsymbol{X}_i, \boldsymbol{X}_j)$. Un estimateur de $\boldsymbol{c}$ est obtenu en minimisant un critère des moindres carrés pénalisé :

$$\hat{\boldsymbol{c}} = \text{argmin}_{\boldsymbol{c} \in \mathbb{R}^n} \sum_{i=1}^{n}(Y_i - f(\boldsymbol{X}_i))^2 + \lambda||f||^2_{\boldsymbol{K}},$$

avec $||f||^2_{\boldsymbol{K}} = \sum_{i,j=1}^{n} c_i c_j k(\boldsymbol{X}_i, \boldsymbol{X}_j)$ et λ un paramètre positif à calibrer.

Une solution explicite de $\hat{\boldsymbol{c}}$ peut être calculée. La réponse du modèle associée à une nouvelle entrée $\boldsymbol{x} \in \mathcal{X}$ est alors prédite de la manière suivante :

$$\hat{f}(\boldsymbol{x}) = \sum_{j=1}^{n} \hat{c}_j k(\boldsymbol{X}_j, \boldsymbol{x}).$$

3.2.4 Arbres de classification et de régression

L'algorithme CART (Classification and Regression Trees) est une méthode non paramétrique utilisée aussi bien en régression qu'en classification. Il s'agit d'un algorithme récursif, basé sur des arbres, qui partitionne l'espace des variables en entrée et prédit sur chaque élément de la partition un modèle simple : une fonction constante en régression et une seule classe en classification. La règle de prédiction peut être représentée par un arbre facilement interprétable.

Une première procédure simple et naturelle est la méthode des k-plus proches voisins (k-NN). Soit $\boldsymbol{D}^n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ un n-échantillon d'observations. Un prédicteur est construit pour une nouvelle entrée $\boldsymbol{x} \in \mathcal{X}$ à partir des k plus proches observations $\{\boldsymbol{X}_{(1)}, \ldots, \boldsymbol{X}_{(k)}\}$ et de leurs sorties associées $\{Y_{(1)}, \ldots, Y_{(k)}\}$, où le paramètre k est à calibrer pour éviter un sur-ajustement. Dans un contexte de régression, la prédiction au point $\boldsymbol{x}$ est obtenue en prenant la moyenne des observations $\{Y_{(1)}, \ldots, Y_{(k)}\}$, tandis qu'en classification on choisit la classe la plus représentée parmi $\{Y_{(1)}, \ldots, Y_{(k)}\}$ (vote majoritaire).

L'algorithme CART utilise la même idée de moyenne locale et de vote majoritaire mais la règle de prédiction est construite en tenant compte des valeurs des Y_i. La première étape consiste à construire un arbre maximal en effectuant un découpage binaire récursif sur une seule variable de sorte que les groupes d'observations soient les plus homogènes possibles. La seconde étape consiste quant à elle à réduire la complexité et le surajustement du modèle en élaguant l'arbre complet.

La construction d'un arbre binaire consiste à déterminer une séquence de nœuds. Un nœud est défini par le choix d'une variable parmi les p variables explicatives et une division qui induit une partition en deux sous-ensembles d'observations. La division est définie par une valeur seuil si la variable sélectionnée est quantitative et deux groupes de modalités si la variable est qualitative. Un critère de division pour sélectionner la meilleure division parmi toutes les divisions possibles est défini, ainsi qu'une règle de nœud terminal qui transforme le nœud en feuille et lui associe une valeur prédite (moyenne ou vote majoritaire). On peut choisir de transformer un nœud en feuille lorsqu'il ne contient plus qu'une seule observation, lorsque tous les individus ont la même valeur de Y, ou lorsque le nombre d'observations qu'il contient est plus petit qu'une certaine valeur.

Le critère de division est basé sur la définition d'une fonction d'hétérogénéité D_κ pour un nœud κ. L'objectif est en fait de diviser les observations qui composent un nœud en deux groupes plus homogènes par rapport à la variable Y à expliquer.

Parmi toutes les divisions possibles du nœud κ en deux nœuds fils κ_L (nœud de gauche) et κ_R (noœud de droite), l'algorithme va retenir celui qui minimise la somme des hétérogénéités de ses nœuds fils $D_{\kappa_R} + D_{\kappa_L}$. Graphiquement, la longueur de chaque branche de l'arbre est proportionnelle à la réduction d'hétérogénéité induite par la division. La fonction d'hétérogénéité est définie en régression avec la variance empirique et en classification avec la déviance ou la concentration de Gini.

Finalement, une fois cet arbre maximal construit, on l'élague pour réduire la complexité du modèle. Pour cela, une séquence d'arbres A emboîtés, dite séquence de Breiman, est construite en s'appuyant sur un critère pénalisé par le nombre des feuilles de l'arbre $|A|$: $C(A) = \sum_{\kappa=1}^{|A|} D_\kappa + \gamma \times |A|$. Cette séquence est associée à une suite de valeurs du paramètre γ. Le paramètre γ est optimisé par validation croisée et l'arbre optimal A_{Opt} retenu est celui qui correspond au γ_{Opt} dans la séquence de Breiman.

Bien que présentant l'avantage de fournir une règle de prédiction facilement interprétable et une méthode de construction efficace, l'algorithme CART est très instable et non robuste au sens où un changement dans les données peut aboutir à des séquences de découpages très différentes. Cette variance importante de la règle de prédiction conduit par conséquent à de mauvais résultats en comparaison avec d'autres méthodes. Elle peut être réduite en utilisant une méthode d'agrégation.

3.2.5 Agrégation et forêts aléatoires

Le *bagging* est un algorithme qui permet de construire aléatoirement une famille de modèles ; il est utilisé dans le cadre de l'agrégation de modèles par *bootstrap*. Le principe du *bagging* peut s'appliquer à n'importe quelle méthode de prédiction mais est surtout intéressant et efficace pour les modèles instables car il permet de réduire la variance de la règle de prédiction. Il est donc naturellement utilisé pour améliorer la méthode précédente (CART).

Agrégation

Soit $\boldsymbol{Z} = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ un n-échantillon d'observations et $\hat{f}(.)$ le prédicteur associé à cet échantillon. Si on considère B sous-échantillons indépendants $\{\boldsymbol{Z}_b\}_{b=1,\ldots,B}$, le prédicteur d'agrégation de modèles est défini

— en calculant la moyenne de tous les prédicteurs si la variable Y à expliquer est quantitative :

$$\hat{f}_B(.) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_{\boldsymbol{Z}_b}(.);$$

— en utilisant un vote majoritaire si Y est qualitative :

$$\hat{f}_B(.) = \text{argmax}_j \text{card}\{b | \hat{f}_{\boldsymbol{Z}_b}(.) = j\}.$$

Agréger ainsi différents modèles indépendants permet de réduire la variance et donc l'erreur de prédiction. Cependant, il n'est pas réaliste de considérer B

sous-échantillons indépendants. La solution est de considérer à la place B sous-échantillons *bootstrap*, chacun obtenu par un tirage avec remise de n observations parmi les observations de l'échantillon initial. Les prédicteurs associés ne sont de fait plus indépendants ; la procédure permet néanmoins d'obtenir une réduction de la variance.

Forêts aléatoires

Dans le cas des modèles définis par des arbres binaires, cette procédure peut être améliorée en ajoutant à chaque nœud une sélection aléatoire de m variables explicatives parmi les p, pour lesquelles le découpage sera autorisé ; cela permet de rendre les arbres plus indépendants. Le nouveau modèle obtenu est appelé forêt aléatoire.

La procédure d'élagage décrite dans la section précédente n'est pas nécessaire ici, il suffit simplement de limiter la taille de l'arbre en imposant un nombre minimum d'observations par feuille, fixé à 5 par défaut. Le paramètre m est quant à lui fixé par défaut à $\frac{p}{3}$ pour un problème de régression et $\sqrt{p}$ pour un problème de classification mais peut être optimisé par validation croisée. Le nombre B d'arbres dans la forêt peut également être optimisé, soit par validation croisée, soit par l'évaluation itérative de l'erreur *out-of-bag*.

Les forêts aléatoires fournissent de bonnes performances de prédiction, sont faciles à implémenter et parallélisables sur chaque arbre. Elles sont cependant difficiles à interpréter. Des indices d'importance des variables explicatives $\boldsymbol{X}^j$ peuvent être calculés pour palier à ce problème. Le premier est le MDI (*Mean Decrease Impurity*), qui correspond à une moyenne de la décroissance d'hétérogénéité quand $\boldsymbol{X}^j$ est choisie pour un découpage. Le second est la MDA (*Mean Decrease Accuracy*), qui compare l'erreur *out-of-bag* avec celle obtenue en permutant les valeurs des variables $\boldsymbol{X}^j$ dans l'échantillon d'apprentissage - l'arbre ne donnant pas la bonne prédiction, l'erreur sera importante si la variable $\boldsymbol{X}^j$ est influente.

3.2.6 Réseaux de neurones

Les réseaux de neurones sont la brique de base de l'ensemble des méthodes dites d'apprentissage profond (*deep learning*), qui modélisent les données en utilisant des architectures complexes combinant différentes transformations non-linéaires des entrées. Plusieurs types d'architecture existent pour les réseaux de neurones, dont les perceptrons multicouches, qui sont les plus anciennes et les plus simples et qui vont être utilisés dans ce projet.

Un neurone artificiel est une fonction f_j d'une entrée $\boldsymbol{x} = (x_1, \ldots, x_d)$, définie par :

$$y_j = f_j(\boldsymbol{x}) = \phi(\langle \boldsymbol{w}_j, \boldsymbol{x} \rangle + b_j),$$

où $\boldsymbol{w}_j = (w_{j,1}, \ldots, w_{j,d})$ est un vecteur de poids, $b_j \in \mathbb{R}$ un biais et ϕ une fonction

d'activation.

Le perceptron multi-couches est une structure composée de plusieurs couches cachées de neurones, où la sortie d'un neurone d'une couche devient l'entrée d'un neurone de la couche suivante. La première couche est la couche d'entrée, viennent ensuite les couches cachées et enfin la dernière couche appelée couche de sortie. La couche de sortie est constitué d'un seul neurone en régression et en classification binaire, associé respectivement à la valeur de la variable quantitative y et à la probabilité $\mathbb{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x})$. En classification multi-classes, la couche de sortie comporte un neurone par classe, chacun associé à une probabilité $\mathbb{P}(Y = k|\boldsymbol{X} = \boldsymbol{x})$.

Les paramètres de l'architecture sont le nombre de couches et le nombre de neurones sur chaque couche (éventuellement à calibrer). Un exemple de perceptron correspondant à un problème de classification à quatre classes est donné sur la figure 3.1. La fonction d'activation ϕ des couches cachées est en général la fonction ReLU. La fonction d'activation de la couche de sortie, notée Ψ, est différente et dépend du type de problème que l'on traite : en régression, on utilise la fonction identité, en classification binaire la fonction sigmoid et en classification multi-classes la fonction softmax multidimensionnelle.

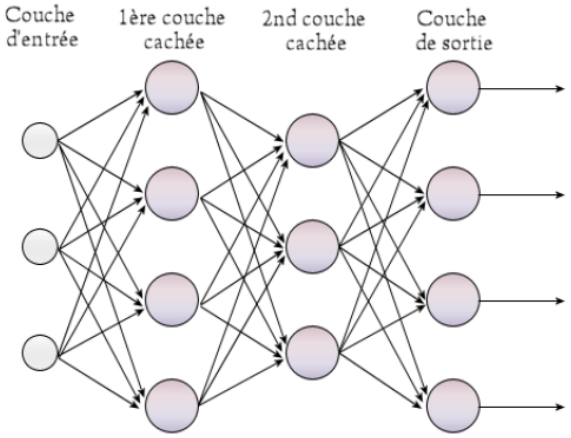

FIGURE 3.1 – Perceptron multi-couches - source : wikipedia

Soit L le nombre de couches cachées du perceptron. Pour $k = 1, \ldots, L + 1$, on note $\boldsymbol{W}^{(k)}$ la matrice des poids et $\boldsymbol{b}^{(k)}$ le vecteur des biais entre la $(k-1)$-ième et la k-ième couche. Ils constituent l'ensemble des paramètres du modèle, noté $\boldsymbol{\theta}$.

La formulation mathématique du perceptron multi-couches est la suivante :
On pose $\boldsymbol{h}^{(0)}(\boldsymbol{x}) = \boldsymbol{x}$.
Pour $k = 1, \ldots, L$ (couches cachées),

$$\begin{aligned}\boldsymbol{a}^{(k)}(\boldsymbol{x}) &= \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)}\boldsymbol{h}^{(k-1)}(\boldsymbol{x}) \\ \boldsymbol{h}^{(k)}(\boldsymbol{x}) &= \phi(\boldsymbol{a}^{(k)}(\boldsymbol{x})).\end{aligned}$$

Pour $k = L + 1$ (couche de sortie),

$$\boldsymbol{a}^{(L+1)}(\boldsymbol{x}) = \boldsymbol{b}^{(L+1)} + \boldsymbol{W}^{(L+1)}\boldsymbol{h}^{(L)}(\boldsymbol{x})$$
$$\boldsymbol{h}^{(L+1)}(\boldsymbol{x}) = \psi(\boldsymbol{a}^{(L+1)}(\boldsymbol{x})) := f(\boldsymbol{x}, \boldsymbol{\theta}).$$

On estime $\boldsymbol{\theta}$ à partir d'un échantillon d'apprentissage $\{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ en minimisant la perte empirique associée à une fonction de perte l pénalisé :

$$L_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} l\left(f(\boldsymbol{X}_i, \boldsymbol{\theta}), Y_i\right) + \lambda \sum_{k=1}^{L+1} ||\boldsymbol{W}^{(k)}||_F^2,$$

où $||\boldsymbol{W}||_F$ est la somme de Frobenius de la matrice $\boldsymbol{W}$ et λ un paramètre positif à calibrer. Pour rappel, on utilise traditionnellement la fonction de perte quadratique en régression, la fonction de perte 0-1 en classification binaire et la fonction de perte de l'entropie croisée en classification multiclasses.

Le critère $L_n(\boldsymbol{\theta})$ est minimisé avec un algorithme de descente de gradient stochastique, où le gradient est calculé en utilisant un algorithme de rétropropagation. Ce calcul n'est pas réalisé avec l'ensemble des n observations et à chaque itération, mais pour un sous-ensemble B d'observations de cardinal m (appelé *batch*), tirées aléatoirement et sans remise, et toutes les $\frac{n}{m}$ itérations. Au bout de ces $\frac{n}{m}$ itérations, appelées *epoch*, l'ensemble des n observations de l'échantillon est utilisé. L'algorithme de descente pour obtenir une estimation de $\boldsymbol{\theta}$ qui minimise $L_n(\boldsymbol{\theta})$ est réalisé pour un nombre d'*epochs* fixé. Le nombre d'*epochs* et la taille des *batchs* font partie des paramètres de l'algorithme qu'il convient d'optimiser. Cette procédure est appelée le *batch learning* et est spécifique aux algorithmes d'apprentissage profond.

Chapitre 4

Application au problème étudié

Dans cette partie, la majorité des méthodes décrites précédemment est appliquée au problème étudié, successivement aux deux approches, régression et classification.

On commence par diviser l'échantillon des observations en deux parties : on tire aléatoirement 150 observations (environ un cinquième de l'échantillon initial qui en comporte 688) pour constituer l'échantillon test. Le reste des observations constitue l'échantillon d'apprentissage. On note $(\boldsymbol{X}_i, Y_i)_{i \in I_{app}}$ l'échantillon d'apprentissage et $(\boldsymbol{X}_i, Y_i)_{i \in I_{test}}$ l'échantillon test.

Une standardisation des données ou normalisation est ensuite appliquée à l'échantillon d'apprentissage. Cela est indispensable à certaines méthodes, par conséquent cette étape est systématiquement exécutée. Les mêmes paramètres (moyennes et écarts-types) estimés sur l'échantillon d'apprentissage sont utilisés pour normaliser l'échantillon test.

Pour chacune des méthodes, on détermine le modèle optimal $\hat{f}_{\boldsymbol{\Gamma}}$ en calibrant certains paramètres de la méthode, notés $\boldsymbol{\Gamma}$, grâce une procédure de validation croisée 10-*folds* sur l'échantillon d'apprentissage $(\boldsymbol{X}_i, Y_i)_{i \in I_{app}}$.
Les méthodes testées et les paramètres calibrés dans le notebook associé à ce projet sont les suivants (les notations correspondent à celles de la partie précédente) :

- Régression linéaire et logistique avec pénalisation LASSO : la pénalité λ ;
- Régression et machines à vecteurs de support avec noyau gaussien : la pénalité λ et l'écart-type du noyau gaussien ;
- k-plus proches voisins : le nombre de voisins ;
- Arbre CART : profondeur maximale de l'arbre - pas de pénalisation de la complexité par le nombre de feuilles dans *scikit-learn* ;
- Forêt aléatoire : le nombre d'arbres dans la forêt et le nombre m de variables parmi lesquelles le découpage est autorisé, tirées aléatoirement lors de la construction de chaque nœud ;
- Réseau de neurones : le nombre de couches cachées et de neurones et la

pénalité λ - on pourrait aussi calibrer le nombre d'*epochs* et la taille des *batchs*.

On quantifie ensuite l'erreur de généralisation en calculant pour $i \in I_{test}$ les prédictions $\widehat{Y}_i = \hat{f}_{\mathbf{\Gamma}}(\boldsymbol{X}_i)$, qu'on compare avec les valeurs observées $(Y_i)_{i \in I_{test}}$ correspondantes. La méthode de comparaison diffère selon les deux approches et est détaillée dans chacune des sections. Finalement, on confronte les deux approches et les différents modèles obtenus pour essayer de dégager un modèle optimal de prédiction de la quantité de pluie le jour suivant.

4.1 Problème de régression

L'erreur de généralisation peut être quantifiée dans le cas d'un problème de régression en se basant sur les indicateurs suivants :

— le graphique des résidus $\left(Y_i - \widehat{Y}_i\right)$ en fonction de $\widehat{Y}_i$ pour $i \in I_{test}$;

— l'erreur moyenne quadratique : $MSE = \sum_{i \in I_{test}} \left(Y_i - \widehat{Y}_i\right)^2$;

— le coefficient R^2 :

$$R^2 = \frac{\sum_{i \in I_{test}} \left(\widehat{Y}_i - \overline{Y}\right)^2}{\sum_{i \in I_{test}} \left(Y_i - \overline{Y}\right)^2}.$$

Ce coefficient est compris entre 0 et 1 et correspond au quotient de la variabilité expliquée et de la variabilité totale. Plus R^2 est proche de 1, meilleure est la qualité de la prédiction et donc du modèle.

Le tracé des résidus est présenté dans la Figure 4.1 pour chacune des méthodes - à noter que les valeurs sont celles de la variable `Lrain`, soit $\log(1 + \texttt{rain})$ et que la transformation inverse n'a pas éte appliquée. Force est de constater que les résultats sont très mauvais. En effet, une bonne prédiction implique un nuage de points dans l'ensemble centrés et alignés autour de zéro, qui atteste d'un biais faible et d'une variance globalement constante des résidus. Ici on observe des résidus éloignés de zéro et de plus en plus importants au fur et à mesure que les valeurs prédites augmentent (*Trumpet shape*). Cela témoigne d'une grande hétéroscédasticité, autrement dit d'une importante variance des résidus. C'est une des raisons pour lesquelles le modèle linéaire est mis en défaut car il présuppose une distribution gaussienne des erreurs.

Si on s'intéresse aux nuages de points des valeurs prédites et observées dans la Figure 4.2, on se rend compte qu'aucune des méthodes ne réussit à prédire les cas où il n'y a pas de pluie. La variable de sortie étant quantitative et continue, cela pouvait se prévoir. La méthode des réseaux de neurones fournissant des valeurs négatives, on teste la transformation des valeurs prédites en classes en assignant la classe `no_rain` aux valeurs négatives mais le résultat est également mauvais.

En fin de compte, les méthodes de régression sont totalement inadaptées au

Régression linéaire avec pénalisation Lasso	Régression à vecteurs de support
0,304	0,289
k-plus proches voisins	Réseau de neurones
0,197	0,251
Arbre CART	Forêt aléatoire
0,189	0,310

TABLEAU 4.1 – Coefficients R^2

Régression linéaire avec pénalisation Lasso	Régression à vecteurs de support
0,634	0,647
k-plus proches voisins	Réseau de neurones
0,731	0,682
Arbre CART	Forêt aléatoire
0,739	0,629

TABLEAU 4.2 – Erreurs moyennes quadratiques (MSE)

problème étudié. En témoignent les coefficients R^2 dans le Tableau 4.1 qui sont très éloignés de la valeur optimale 1. Les prédictions les plus mauvaises sont réalisées par les méthodes des k-plus proches voisins et de l'Arbre de régression, ce qui apparait également dans le Tableau 4.2 des erreurs moyennes quadratiques.

4.2 Problème de classification

L'erreur de généralisation peut être quantifiée dans le cas d'un problème de classification multiclasses en se basant sur les indicateurs suivants :

— la matrice de confusion, qui résume le nombre de bonnes et de mauvaises prédictions en fonction des K modalités $\{m_1, \ldots, m_K\}$. Il s'agit d'un tableau à double entrée de la forme suivante :

Observation / Prédiction	m_1	$\cdots$	m_l	$\cdots$	m_K
m_1					
$\vdots$					
m_k			$\sharp\{i \in I_{test}, \widehat{Y}_i = m_k, Y_i = m_l\}$		
$\vdots$					
m_K					

— le taux de mal classés, donné par

$$\frac{\sharp\{i \in I_{test}, \widehat{Y}_i \neq Y_i\}}{\sharp\{i \in I_{test}\}}.$$

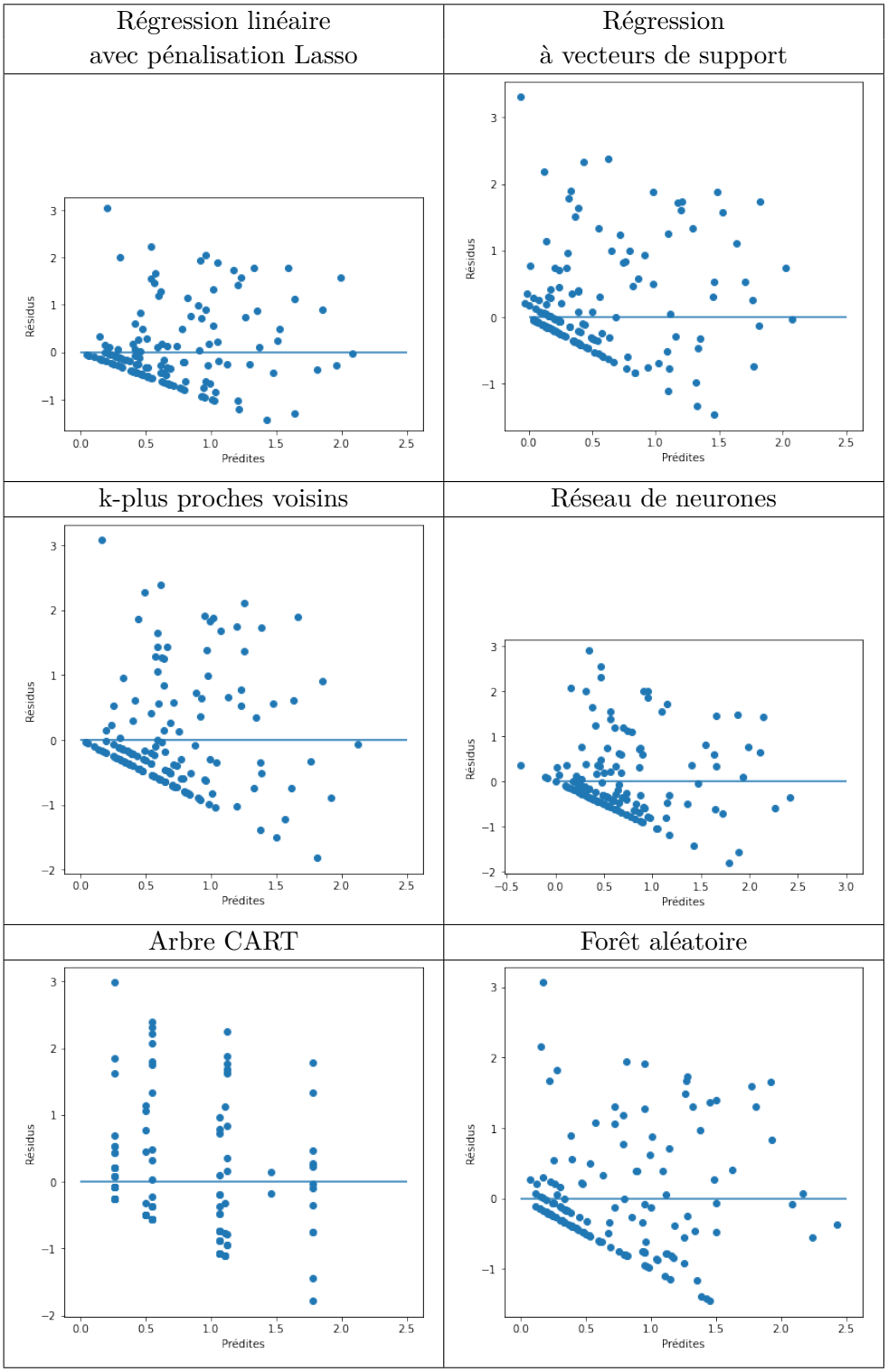

FIGURE 4.1 – Résidus entre les valeurs prédites et observées

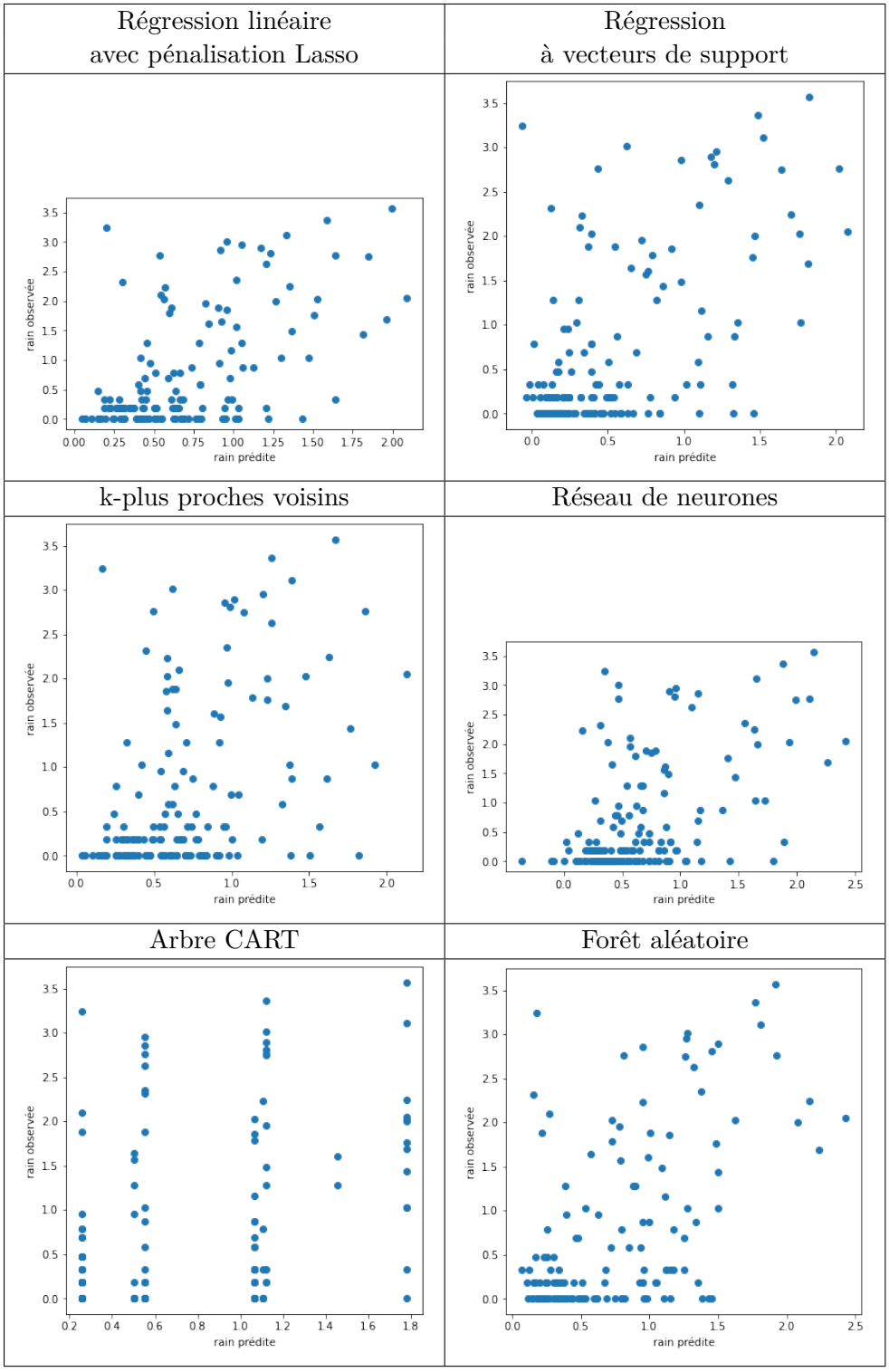

FIGURE 4.2 – Nuages de points des valeurs prédites et observées

À noter qu'on peut tracer la courbe ROC dans le cas d'un problème de classification binaire. Elle représente le taux de "vrais positifs" en fonction du taux de "faux positifs".

4.2.1 Comparaison des méthodes sur un couple d'échantillons

Une comparaison de l'efficacité des méthodes de régression est d'abord réalisée sur un premier couple d'échantillons d'apprentissage et de test.

Les matrices de confusion et les taux de mal-classés associés sont présentés dans les Tableaux 4.3 et 4.4. Les résultats sont assez décevants, avec un taux de mal-classés proche de 50%. Aucune des méthodes ne réussit finalement à discriminer correctement les trois classes de pluie, comme on l'avait pressenti lors de l'analyse de données en observant avec l'ACP le mélange des observations.

Tout comme pour la régression, les méthodes des k-plus proches voisins et de l'Arbre de régression sont celles qui donnent les prédictions les plus mauvaises, avec un taux de mal-classés de 48%. Sur 52 observations où il ne pleut pas, la première méthode prévoit même une forte pluie dans 17% des cas et la deuxième dans 15% des cas ! L'inefficacité de ces méthodes n'est pas spécifique au problème étudié. En effet, la méthode des k-plus proches voisins ne tient pas compte des observations de la variable de sortie, tandis que l'algorithme CART n'est pas robuste du fait de sa variance importante.

Le Réseau de neurones ne semble pas non plus fournir une bonne prédiction sur ce couple d'échantillons, avec $46,7\%$ de mal-classés, malgré une calibration fine des paramètres du modèle. À noter qu'un perceptron à une seule couche est retenu sur ce problème par validation croisée plutôt qu'un perceptron à deux ou trois couches. Viennent ensuite les méthodes SVM et Forêt aléatoire, avec 46% de mal-classés. Cependant, cette dernière prédit une forte pluie dans le cas où il ne pleut pas dans un peu plus de 13% des cas.

Sur ce premier couple d'échantillons, la méthode de Régression logistique semble la plus performante, bien que d'une efficacité très limitée, avec un taux de mal-classés à $44,7\%$. Il est intéressant d'observer les coefficients du modèle optimal correspondant, obtenus avec la pénalisation Lasso (*cf.* Figure 4.3). La pénalisation Lasso diminue les coefficients des variables explicatives les moins significatives, jusqu'à en mettre certains à zéro. Les coefficients correspondant aux mois de mai, juin, juillet et août sont très faibles voire nuls, ce qui parait cohérent avec le fait que les précipitations sont les plus faibles sur cette période de l'année. À l'inverse, les coefficients correspondant aux mois de novembre, décembre et janvier sont importants. Les variables explicatives du jour courant ne semblent pas influentes, exceptés le taux d'humidité et la quantité de précipitation, ce qui paraît cohérent également. Quant aux variables explicatives du jour suivant, prédites par le modèle AROME, les plus influentes semblent être celles associées au vent et à la pression au niveau

Régression logistique avec pénalisation Lasso

	high_rain	low_rain	no_rain
high_rain	20	10	5
low_rain	15	43	27
no_rain	3	7	20

Machines à vecteurs de support

	high_rain	low_rain	no_rain
high_rain	19	8	3
low_rain	16	45	32
no_rain	3	7	17

k-plus proches voisins

	high_rain	low_rain	no_rain
high_rain	20	10	8
low_rain	14	43	29
no_rain	4	7	15

Réseau de neurones

	high_rain	low_rain	no_rain
high_rain	19	8	5
low_rain	14	45	31
no_rain	5	7	16

Arbre CART

	high_rain	low_rain	no_rain
high_rain	20	17	9
low_rain	13	38	23
no_rain	5	5	20

Forêt aléatoire

	high_rain	low_rain	no_rain
high_rain	22	11	7
low_rain	14	39	23
no_rain	2	10	22

TABLEAU 4.3 – Matrices de confusion

Régression logistique avec pénalisation Lasso	Machines à vecteurs de support
0,447	0,460
k-plus proches voisins	Réseau de neurones
0,480	0,467
Arbre CART	Forêt aléatoire
0,480	0,460

TABLEAU 4.4 – Taux de mal classés entre les classes prédites et les classes observées

de la mer, le taux d'humidité et la quantité de précipitation prédites n'étant pas retenus, ce qui pose question ici.

Les indices d'importance des variables donnés par la méthode des Forêts aléatoires sont mis en parallèle de ces coefficients à droite de la Figure 4.3. Les six premières variables significatives déterminées par cette méthode sont cohérentes avec celles déterminées par la méthode LASSO ; en revanche, les autres ne le sont pas, en particulier aucun mois n'est retenu comme siginificatif ici, ce qui n'est pas logique.

4.2.2 Comparaison des méthodes par validation croisée MC

L'échantillon des 688 observations fournies est de taille modeste, par conséquent la construction d'une règle de prédiction et l'estimation de l'erreur de généralisation

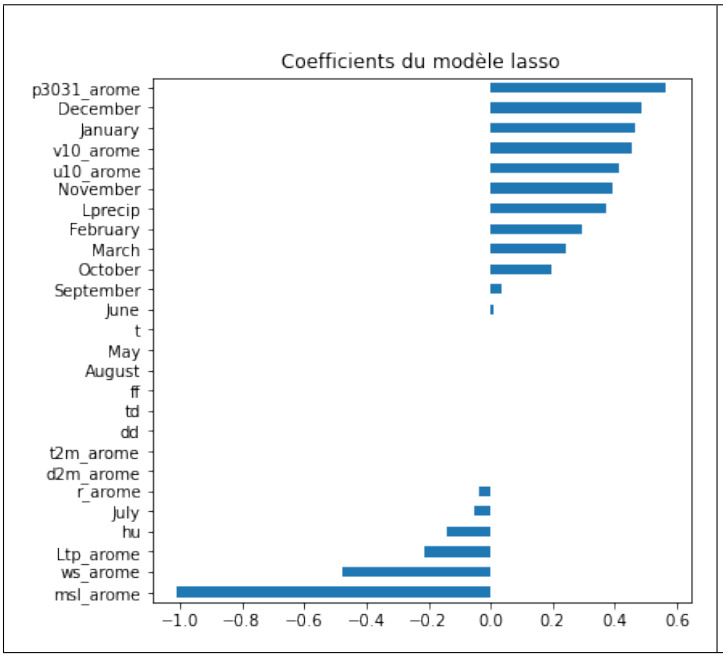

FIGURE 4.3 – Coefficients du modèle de régression logistique (à gauche), obtenus avec la pénalisation Lasso. Indices d'importance des variables donnés par la méthode des Forêts aléatoires pour la classification (à droite).

peuvent présenter une variance importante selon le couple d'échantillons d'apprentissage et de test choisis aléatoirement au départ. Par ailleurs, aucune méthode n'est performante pour le problème étudié et les erreurs de généralisation estimées sont du même ordre de grandeur, de sorte qu'il paraît compliqué de comparer les méthodes entre elles avec un seul couple d'échantillons.

Une solution est d'estimer la distribution complète de l'erreur de généralisation pour chacune des méthodes considérées en opérant une forme de validation croisée *k-folds* Monte Carlo. Plus précisément, on itère plusieurs fois le découpage de l'échantillon d'observations en échantillons d'apprentissage et de test et on applique à chaque itération et pour chacune des méthodes la procédure de construction d'une règle de prédiction optimale et d'estimation de l'erreur. On peut ensuite représenter la distribution de l'erreur associée à chaque méthode par un diagramme à moustaches et calculer ses indicateurs statistiques, permettant ainsi une comparaison plus robuste et plus précise des méthodes entre elles.

Les diagrammes à moustaches et les indicateurs statistiques des erreurs de généralisation des différentes méthodes optimisées par une procédure de validation croisée 10-*folds* Monte Carlo sont donnés respectivement par la Figure 4.4 et le Tableau 4.5. La procédure est très coûteuse en temps de calcul, et a été par conséquent réalisée sur 10 itérations uniquement. Elle offre néanmoins un bon aperçu de la distribution des erreurs.

	Moyenne	Médiane	Ecart-type	$Q_3 - Q_1$
Régression logistique	0,468	0,470	0,040	0,047
SVM	0,479	0,483	0,027	0,028
k-NN	0,505	0,513	0,025	0,032
Arbre CART	0,509	0,507	0,029	0,023
Forêt aléatoire	0,490	0,490	0,025	0,032
Réseau de neurones	0,491	0,487	0,034	0,040

TABLEAU 4.5 – Indicateurs statistiques des erreurs de généralisation

Les méthodes des k-plus proches voisins et de l'Arbre de régression sont bien les moins performantes avec des erreurs moyennes et médianes plus importantes que les autres. La méthode de Forêt aléatoire permet de diminuer légèrement les erreurs moyenne et médiane de la méthode CART, mais ne réduit pas la dispersion ici, alors que c'est normalement le cas.

Les méthodes SVM et Réseau de neurones présentent des erreurs moyennes et médianes similaires mais la dispersion est plus faible pour la méthode SVM, que ce soit en terme d'écart-type ou d'écart interquartile. La méthode de Régression logistique fournit quant à elle les erreurs moyenne et médiane les plus faibles mais avec une dispersion non négligeable.

En conséquence, la méthode qui réalise le meilleur compromis entre biais et variance sur les 10 couples d'échantillons considérés est la méthode SVM. C'est donc cette méthode qu'on retiendra ici, même si les prédictions obtenues sont loin d'être satisfaisantes.

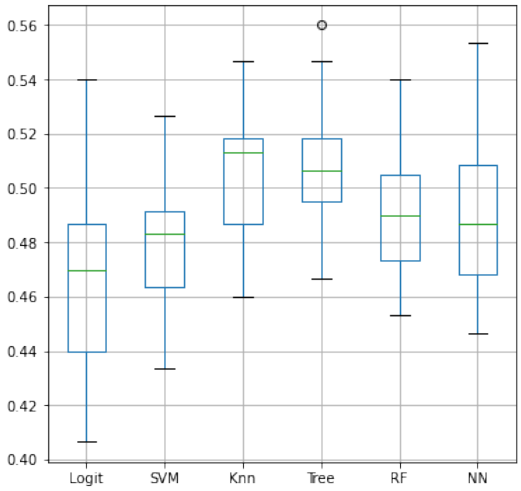

FIGURE 4.4 – Diagrammes à moustaches des erreurs de généralisation des différentes méthodes optimisées par une procédure de validation croisée 10-*folds* Monte Carlo

Chapitre 5

Conclusion

L'objectif de ce projet était de pouvoir prédire la quantité de pluie à un endroit donné le jour suivant, à partir d'un échantillon d'observations et de prédictions. Plusieurs méthodes d'apprentissage supervisé ont été mises à l'épreuve : Régressions linéaire et logistique avec pénalisation Lasso, Machine et Régression à vecteurs de support, k-plus proches voisins, Arbres CART de régression et de classification, Forêts aléatoires, Réseaux de neurones. En dépit des nombreux efforts accomplis pour construire un modèle de prédiction optimal (calibration fine des paramètres par validation croisée, optimisation par une procédure de validation croisée Monte-Carlo), aucune des méthodes testées ne s'est avérée satisfaisante pour répondre à l'objectif.

Si l'analyse des données effectuée présageait de la difficulté à discriminer les différentes classes de pluie, les résultats n'en demeurent pas moins décevants. Les méthodes de régression ont donné des résultats médiocres, avec une importante hétéroscédasticité et une incapacité à prédire les cas où la quantité de pluie était nulle. Quant aux méthodes de classification, les résultats sont meilleurs mais restent insuffisants, avec un taux de mal classés de 50% en moyenne et une variance importante selon les couples d'échantillons d'apprentissage et de test choisis. On se contentera d'une prédiction par la méthode SVM qui fournit des prévisions de pluie justes dans 53% des cas en moyenne avec une variance acceptable, réalisant le meilleur compromis biais/variance sur les couples d'échantillons étudiés.

Sur un problème de discrimination aussi complexe, relevant en fait d'un challenge IA, il paraît ainsi nécessaire de dépasser les méthodes d'apprentissage supervisé classiques, dont ce projet a été l'occasion de souligner les limites. Le recours à des méthodes plus sophistiquées, telles que celles combinant différents modèles, s'avère indispensable ici.