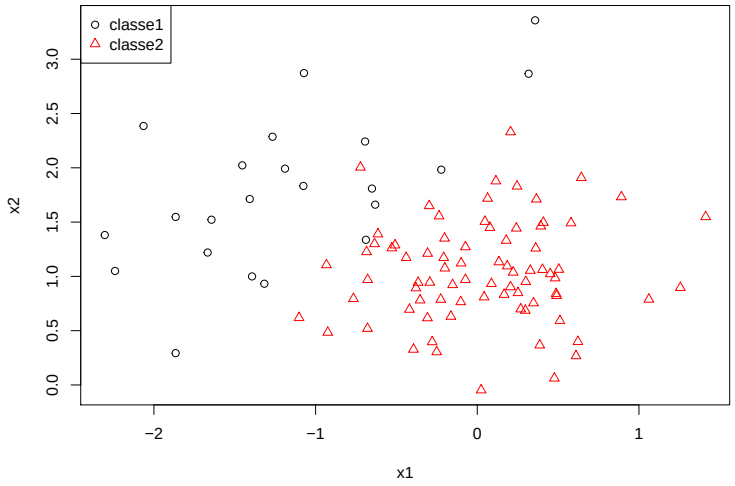
Master 2 MSS
Apprentissage supervisé

TP4 : régression logistique

Exercice 1. Les objectifs :
- Découvrir la régression logistique.
- Découvrir la fonction `glm`.

On utilise dans cet exercice les données synth_train.txt et synth_test.txt.



1. En régression logistique la variable de sortie Y prend ses valeurs dans $\{0, 1\}$. Ici, la variable Y prends ses valeurs dans $\{1, 2\}$. On recode d'abord les données pour avoir le codage suivant : ¨0=classe 1¨ et ¨1=classe 2.
2. En régression logistique on fait également l'hypothèse suivante :

$$\mathbb{P}(Y = 1|X = x) = \frac{\exp(\beta_0 + \sum_{j=1}^{p} \beta_j x_j)}{1 + \exp(\beta_0 + \sum_{j=1}^{p} \beta_j x_j)}$$

   Estimer sur les données d'apprentissage le vecteur de paramètres inconnu $(\beta_0, \beta_1, \ldots, \beta_p)$ en utilisant la fonction `glm`.

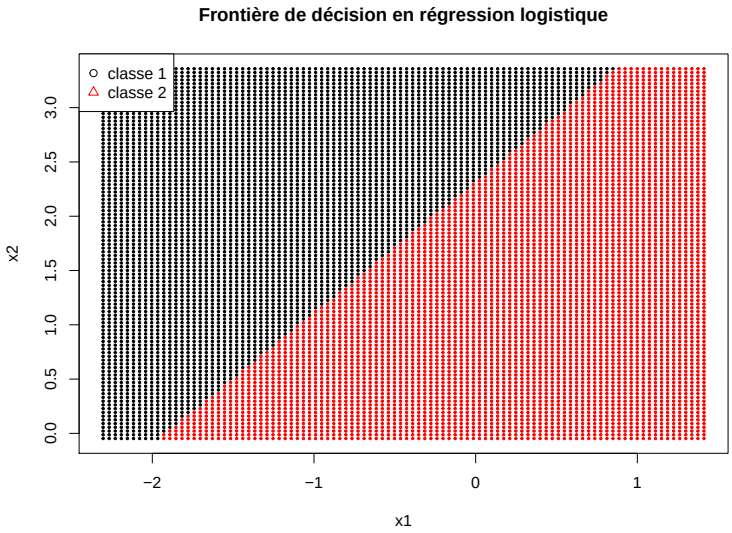
3. Interpréter les coefficients β^1 et β^2 ainsi estimés. Vous pouvez vous aider du document "complements_logistique.pdf" pour cela.

4. On veut maintenant prédire la classe de la nouvelle observation $x = (0, 1)$ avec le modèle de régression logistique. Pour cela on calcule le score logit (score linéaire)

$$\beta_0 + \sum_{j=1}^{p} \beta_j x_j$$

   et on affecte x à la classe $Y = 1$ si ce score est supérieur à 0 et à la classe $Y = 0$ sinon. Vérifier que x est bien affecté à la classe 2.

5. On peut également calculer le score $\mathbb{P}[Y = 1|X = x]$ (probabilité à posteriori d'être affecté à la classe 2) et affecter x à la classe $Y = 1$ si ce score est supérieur à 0.5 et à la classe $Y = 0$ sinon. Calculer cette probabilité et vérifier qu'elle est bien supérieure à 0.5.
6. Utiliser la fonction `predict.glm` (méthode `predict` de la classe `glm`) pour retouver les résultats obtenus précédemment. Cette fonction permet-elle de prédire directement la classe de $x = (0, 1)$?
7. Prédire les données d'apprentissage et calculer le taux d'erreur d'apprentissage.
8. Représenter la frontière de décision de la méthode de régression logistique à partir de la grille de points du TP1.



9. Charger le jeu de données test dans R et calculer le taux d'erreur test. Comparer au taux d'erreur d'apprentissage.
10. Représenter la courbe ROC et calculer le critère AUC à partir des données test.

Exercice 2. Les objectifs :

— Appliquer la régression logistique à des données réelles.
— Régulariser la régression logistique : régression logistique ridge et lasso (pour sélectionner des variables).
— Choisir une règle de classification : comparer les performances de plusieurs méthodes.

On reprend le jeu de données complet [1] du TP2 où les exploitations agricoles sont décrites par $p = 22$ critères économiques et financiers.

```
## [1] 1260   23
##  [1] "DIFF" "R1"   "R2"   "R3"   "R4"   "R5"   "R6"   "R7"   "R8"   "R11"
## [11] "R12"  "R14"  "R17"  "R18"  "R19"  "R21"  "R22"  "R24"  "R28"  "R30"
## [21] "R32"  "R36"  "R37"
```

La variable qualitative à expliquer est toujours la variable difficulté de paiement (0=sain et 1=défaillant) avec ici
— prédire 1 (défaillant)=positif,
— prédire 0 (sain)=négatif.

1. http ://publications-sfds.fr/index.php/csbigs/article/view/351/331

On cherche à modéliser la probabilité à posteriori d'être défaillante notée $p = \mathbb{P}(Y = 1|X = x)$ par un modèle de régression logistique.

1. Estimer par maximum de vraissemblance les coefficients $(\beta_0, \beta_1, \ldots, \beta_p)$ du modèle de régression logistique

$$\mathbb{P}(Y = 1|X = x) = \frac{\exp(\beta_0 + \beta^T x)}{1 + \exp(\beta_0 + \beta^T x)}$$

où $\beta = (\beta_1, \ldots, \beta_p)$.

```
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept)  -11.693     4.3190 -2.7074 6.782e-03
## R1            24.537     8.3017  2.9557 3.120e-03
## R2             2.321     3.5047  0.6622 5.078e-01
## R3             4.374     1.8713  2.3372 1.943e-02
## R4           -10.124     7.7739 -1.3023 1.928e-01
## R5           -13.419     8.4799 -1.5824 1.135e-01
## R6            -1.165     1.6788 -0.6942 4.875e-01
## R7             2.082     1.9851  1.0487 2.943e-01
## R8            -2.675     2.0528 -1.3032 1.925e-01
## R11           -1.082     2.3500 -0.4603 6.453e-01
## R12            1.549     1.6158  0.9584 3.379e-01
## R14            2.330     0.7781  2.9941 2.753e-03
## R17           39.088     9.9563  3.9260 8.639e-05
## R18          -10.029    17.5203 -0.5724 5.670e-01
## R19            2.309    10.8064  0.2136 8.308e-01
## R21            1.111     3.9905  0.2784 7.807e-01
## R22            1.259     1.8532  0.6796 4.967e-01
## R24            2.284     5.6556  0.4039 6.863e-01
## R28          -12.125    11.6916 -1.0371 2.997e-01
## R30            7.860    11.1415  0.7055 4.805e-01
## R32           -1.834    13.9176 -0.1318 8.952e-01
## R36            1.448     0.4830  2.9975 2.722e-03
## R37           -2.144     1.7772 -1.2063 2.277e-01
```

Toutes les variables explicatives sont-elles utiles ?

2. Vous avez observé le warning suivant : "glm.fit : des probabilités ont été ajustées numériquement à 0 ou 1". Caculer les scores (logit(p) et p) des données et faire l'histogramme de ces deux scores.

3. On veut maintenant estimer le modèle de régression logistique avec le package `glmnet`. Ce package permet d'estimer le modèle de régression logistique par maximisation de la vraissemblance pénalisée par une pénalité de type elastic-net. Le problème d'optimisation dans `glmnet` est :

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} -\frac{1}{n}\ell(\beta_0, \beta) + \lambda \left((1 - \alpha)\frac{\|\beta\|_2^2}{2} + \alpha \|\beta\|_1 \right)$$

La pénalité élastic-net est une combinaison des pénalités de type ridge et lasso

— Si $\alpha = 1$, la pénalité elastic-net est une pénalité de type ridge qui utilise $\|\beta\|_2^2$ (carré de la norme L_2 de β) pour pénaliser les grandes valeurs de $\hat{\beta}$ et contrôler la variance de cet estimateur. Cette pénalité ne permet pas de sélectionner des variables.

— Si $\alpha = 0$, la pénalité elastic-net est une pénalité de type lasso qui utilise $\|\beta\|_1$ (norme L_1 de β) pour mettre à 0 certains coefficients de $\hat{\beta}$ et sélectionner des variables.

Le paramètre λ est appellé paramètre de régularisation et :

— Si $\lambda = 0$ on fait de la régression logistique par maximum de vraissemblance.

— Plus λ augmente, plus on pénalise la vraissemblance.

  — et si $\alpha = 0$, λ est la paramètre de régularisation de la régression logistique ridge,
  — et si $\alpha = 1$, λ est la paramètre de régularisation de la régression logistique lasso.

(a) On utilise d'abord la fonction `glmnet` pour estimer les coefficients du modèle de régression logistique par maximum vraissemblance non pénalisée. Retrouve-t-on exactement les résultats obtenus avec la fonction `glm` ?

(b) On utilise ensuite la fonction `glmnet` pour effectuer une régression logistique ridge.

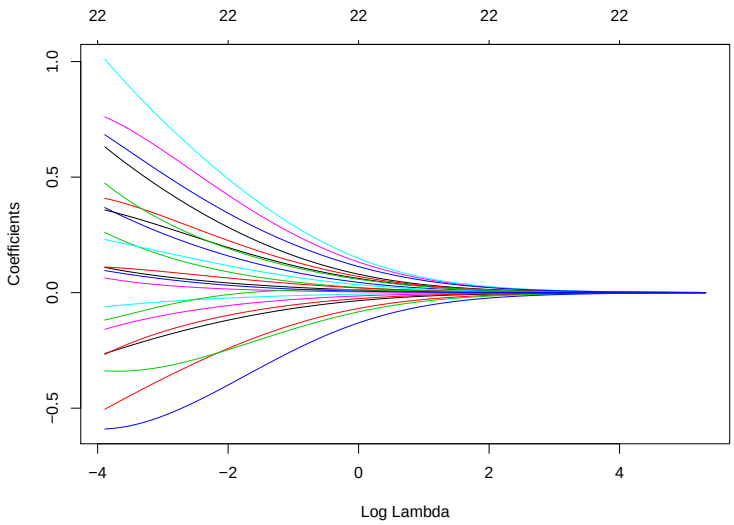
i. Le code ci-dessous permet d'estimer les coefficients du modèle de régression logistique avec la méthode ridge pour une grille de valeurs du paramètre de régularisation λ.

```
g <- glmnet(as.matrix(X), as.factor(Y), family="binomial",
            alpha=0,
            standardize = FALSE)
g$lambda
```

Comment est définie cette grille de valeurs de λ ?

ii. A l'aide de la méthode `coef` (associée aux objects de classe `glmnet`), comparer les vecteurs de coefficients obtenus avec une grande valeur de λ ($\lambda = 200$) et une petite valeur de λ ($\lambda = 0.02$). Vérifier que plus λ est grand plus $\|\beta\|_2$ est petite.

iii. Faire le plot des coefficients de $p = 22$ variables en fonction du paramètre de régularisation λ.



Ce graphique confirme-t-il les résultats de la question précédente ? Que signifient les chiffres (ici 22) sur l'axe supérieur du graphique ?

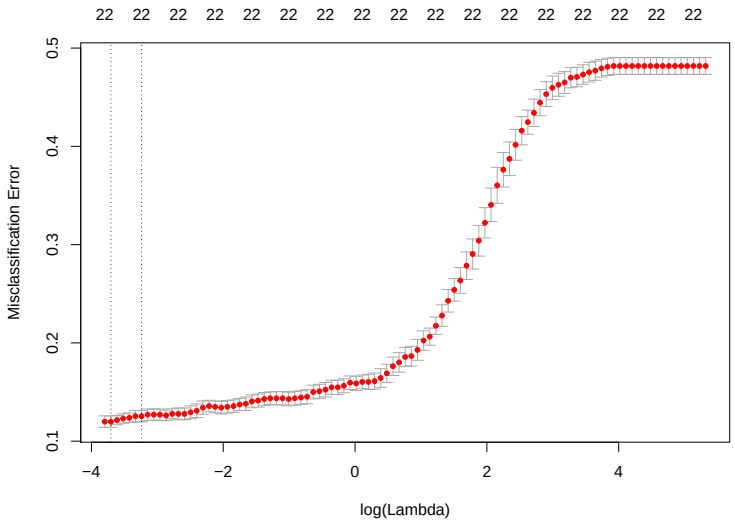
iv. On veut maintenant choisir automatiquement la valeur du paramètre λ (pour faire de la régression ridge ici). On utilise pour cela la fonction `glmnet.cv` pour choisir λ par validation croisée 10-fold. Deux valeurs particulièrtre du paramètre λ sont fournies en sortie de la méthode :

— "lambda.min" qui minimise ici l'erreur (moyenne) de validation croisée,
— "lambda.1se" qui est la plus grande valeur de λ dont l'erreur (moyenne) de validation croisée est inférieure à l'erreur optimale (de "lambda.min") plus une fois son écart-type.

```
g <- cv.glmnet(as.matrix(X), as.factor(Y), family="binomial",
               alpha=0,
               standardize=FALSE,
               type.measure="class")
g$lambda.min
g$lambda.1se
```

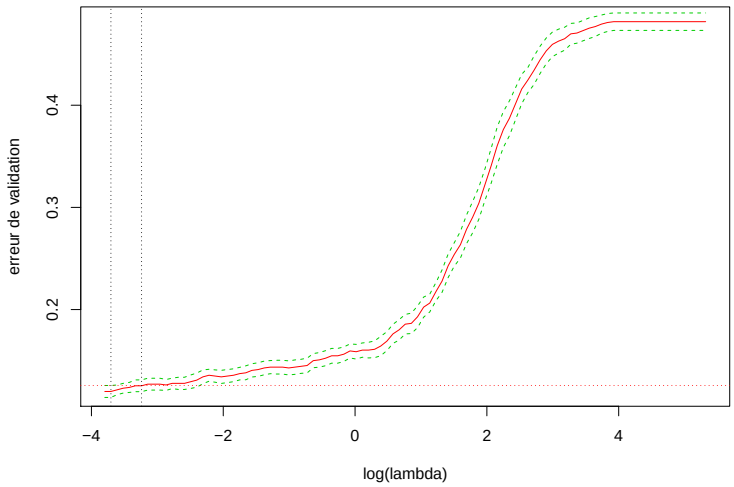
Expliquer à quoi correspondent, pour une valeur de λ donnée, l'erreur moyenne de validation croisée et l'écart-type associé. Avec les sorties `cvm` et `cvsd` :
— trouver l'erreur (moyenne) de validation croisée minimum et retrouver "lambda.min",
— calculer l'erreur (moyenne) de validation croisée minimum + 1 fois l'écart-type associé. Ce seuil sera utilisé pour trouver "lambda.1se".

v. Pour mieux comprendre la "règle du 1-se", on visulalise le graphique du taux d'erreur de validation en fonction du paramètre de régularisation avec la méthode `plot` de la classe `glmnet`. Les valeurs "lambda.min" et "lambda.1se" y sont représentées par deux lignes pointillées verticales.



Expliquer avec ce graphique la "règle du 1-se".

vi. Faire "à la main" ce graphique en ajoutant une ligne verticale correspondant au seuil permettant de trouver "lambda.1se".



vii. On choisit finalement λ qui minimise l'erreur de validation. Prédire alors avec la méthode `predict` de la classe `glmnet` la classe de la première exploitation agricole, puis sa probabilité d'être défaillante.

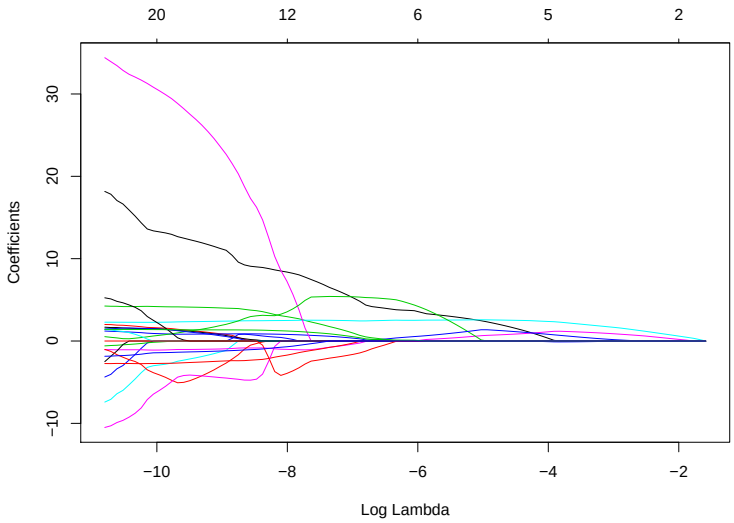
(c) La régression ridge ne permet pas de sélectionner des variables. On utilise donc maintenant la fonction `glmnet` pour effectuer une régression logistique lasso.

i. Le code ci-dessous permet d'estimer les coefficients du modèle de régression logistique avec la méthode lasso pour une grille de valeurs du paramètre de régularisation λ.

```
g <- glmnet(as.matrix(X), as.factor(Y), family="binomial",
            alpha=1,
            standardize = FALSE)
g$lambda
```

ii. A l'aide de la méthode `coef` (associée aux objects de classe `glmnet`), comparer les vecteurs de coefficients obtenus avec une grande valeur de λ ($\lambda = 0.2$) et une petite valeur de λ ($\lambda = 0.002$). Vérifier que plus λ est grand plus le nombre de variables sélectionnées est petit.

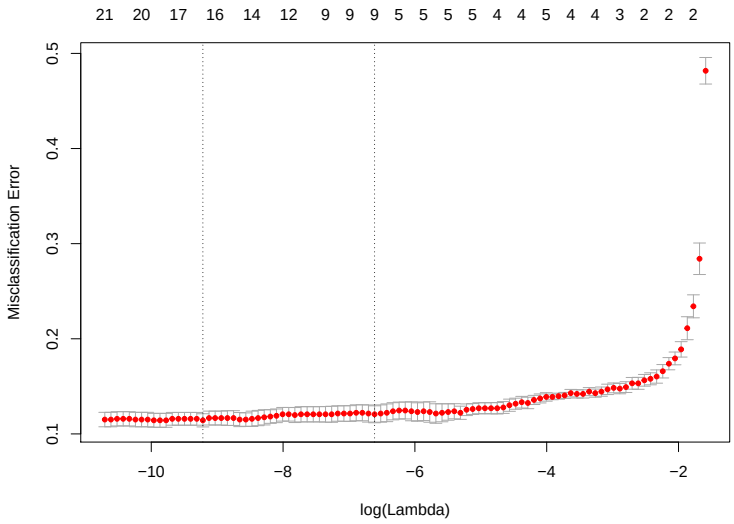
iii. A l'aide de la méthode `plot`, faire le plot des coefficients des $p = 22$ variables en fonction du paramètre de régularisation λ.



Ce graphique confirme-t-il les résultats de la question précédente ? Que signifient les chiffres sur l'axe supérieur du graphique ?

iv. Utiliser la fonction `glmnet.cv` pour choisir la valeur du paramètre de régularisation λ par validation croisée 10-fold. Quelles sont les valeurs "lambda.min" et "lambda.1se" trouvées ?

v. Afin de choisir parmi ces deux valeurs de λ on visualise le graphique du taux d'erreur de validation en fonction du paramètre de régularisation. Le modèle obtenu avec "lambda.1se" est-il beaucoup plus parcimonieux que le modèle obtenu avec "lambda.min" ?



vi. On choisit finalement λ avec la "règle du 1-se". Quelles variables sont sélectionnés ?

vii. Prédire la classe de la première exploitation agricole puis sa probabilité d'être défaillante.

4. On veut maintenant comparer la régression logistique avec l'analyse discriminante linéaire. On veut aussi sélectionner les variables les plus importantes sans trop détériorer les prédictions. Retrouver le graphique des boxplots des taux d'erreurs test afin de comparer la régression logistique avec et sans sélection de variables et l'analyse discriminante linéaire avec et sans sélection de variables.

   Ces boxplots ont été obtenus avec la méthodologie suivante :

   — Créez B découpages aléatoires des données en 945 observations d'apprentissage et 315 observations test.
   — Pour chaque découpage :
      — Estimer sur les données d'apprentissage :
         — les paramètres d'analyse discriminant sur les 22 variables (fonction lda),
         — les paramètres d'analyse discriminante sur une selection de variable (fonction greedy.wilks),
         — les coefficient de la régression logistique sur les 22 variables (fonction glm),
         — les coefficients de la régression logistique avec pénalité lasso et les variables sélectionnées avec *lambda.min*.
      — Prédire les données test avec les 4 modèles ainsi estimés et calculer les taux d'erreurs test de chaque méthode.
   — Faire un boxplot des erreurs test des différentes méthodes.