Projet apprentissage statistique

Ndongo Seye;

2022-08-17

1 Chargement et préparation de données réelles

Nous nommerons le jeu de données de cancer du sein METABRIC téléchargé depuis kaggle par *datacancer* dans ce projet.

1.1 Chargement, description et préparation du jeu de données:

Le jeu de données compte 1904 obervations et 693 variables.

On voit que les patientes décédées du cancer ou non sont codés dans une variable catégorielle *death_from_cancer* à 3 niveaux:

- *Living*: si la patiente est en vie,
- *Died of Disease*: si la patiente est morte du cancer du sein,
- *Died of Other Causes*: si la patiente est morte d'une cause différente du cancer du sein

Pour chaque patiente, on attribue un identifiant unique *patient_id*. On donne l'âge de la patiente au moment où elle a été diagnostiquée porteuse du cancer (*age_at_dignosis*), si ellle suit une chimiothérapie ou pas (*chemotherapy*), le diagnostic du nombre de mutation du cancer (*mutation_count*), la taille de la tumeur (*tumor_size*) et son état de développement (*tumor_stage*).

Nous allons coder la variable *death from cancer* en bianaire où le niveau *Died of Disease*=1 et 0 pour les autres en:

a. créant une variable Y binaire qui vaut 1 pour les patientes décédées du cancer et 0 pour les autres,
b. ne gardant que les variables *age_at_dignosis*, *tumor_size*, *mutation_count*, *death_from_cancer* et celles allant de la colonne 32 à 693,

```
datacancer <- datacancer %>%
  select( age_at_diagnosis, tumor_size, mutation_count, Y=death_from_cancer, all_of(32:
  mutate( Y= if_else(Y=="Died of Disease",1,0) ) # pour le recodage en binaire
```

Le jeu de données compte maintenant 1904 observations et 666 colonnes.

c. modifiant les données de mutations en variables binaires: 1 pour chaque mutation, quelque soit le code qui la décrive.

Les données de mutations se terminent par *_mut*. Elles vont de la colonne 494 à la fin du tableau.

```
datacancer[494:666] <- if_else(datacancer[494:666]==0,0,1) # si pas mutation: 0 sinon 1

datacancer %>%
  filter(is.na(mutation_count) | is.na(tumor_size))# 65 données manquantes pour l'ensem

datacancer <- datacancer %>%
  drop_na(tumor_size,mutation_count)
any(is.na(datacancer)) # pour vérifier qu'il ne reste rien de données manquantes
```

Le jeu de données *datacancer* est en fin préparée pour les parties 2 et 3. J'ai préféré supprimer les données manquantes puisqu'il n'y a qu'en tout 65 au total.

2 Prédiction

2.1 Choisir trois méthodes de prédiction vues en cours

- Méthodes de prédiction avec forêt aléatoire:

  Tout d'abord un arbre de décision est un organigramme qui permet de classifier des données d'entrées ou de prédire la valeur de sortie de ces données. Il est facile à interprété et à visualiser et est péfèré des méthodes à noyau [1]. En particulier, les forêts aléatoires sont robustes et pratiques en apprentissage et impliquent plusieurs arbres de décision et assemblent leurs sorties [1]. Ils permettent aussi de diminuer le risque de sur-apprentissage.

- Méthode de prédiction SVM:

  Elle est simple à utiliser car nécessitant peu de paramètres. SVM est un algorithme de classification qui fonctionne en trouvant des «frontières de décision» séparant deux classes.

- Méthode de prédiction avec pénalité:

  La régression pénalisée permet de gérer les corrélations entre covariables ou encore la sélection des variables. Deux types de pénalités: Lasso pour la sélection de variable et Ridge pour la corrélation entre variables. La pénalité Elastic-net est un bon compromis entre les deux autres types de pénalités.

a. aucune réduction de dimension

1. Méthodes de prédiction avec forêt aléatoire

   - Séparation du jeu d'apprentissage et mise en facteur la variable Y

```
# On met la variable Y en facteur labellisée en (0="no" for Others cases) et ( 1="yes"
datacancer <- datacancer %>%
  mutate(Y=as.factor(if_else(Y==0,"no","yes") ) )

#  Séparation du jeu de données en jeu de test et d'apprentissage
inTrain <- createDataPartition(
  y <- datacancer$Y,
  p=.75,
  list = FALSE
)

# jeu d'apprentissage
cancerTraining <- datacancer %>%
  slice(n=inTrain)
# jeu de test
cancerTest <- datacancer %>%
  slice(n=-inTrain)

# crtl est un paramètre de contrôle avec validation croisée qu'on utilisera dans train
ctrl <- trainControl(
  #method = "boot",
  method="repeatedcv",
  number=10,
  repeats = 3,
  classProbs = TRUE,
  summaryFunction = twoClassSummary
)

# Ajustement
rfFit <- train(
  Y ~.,
  data = cancerTraining,
```

```
  method = "rf",
  tuneLength = 13,
  trControl = ctrl,
  metric = "ROC"
)
```

```
knitr::include_graphics("pictures/ROC_cross_validation.png", auto_pdf = T)
```

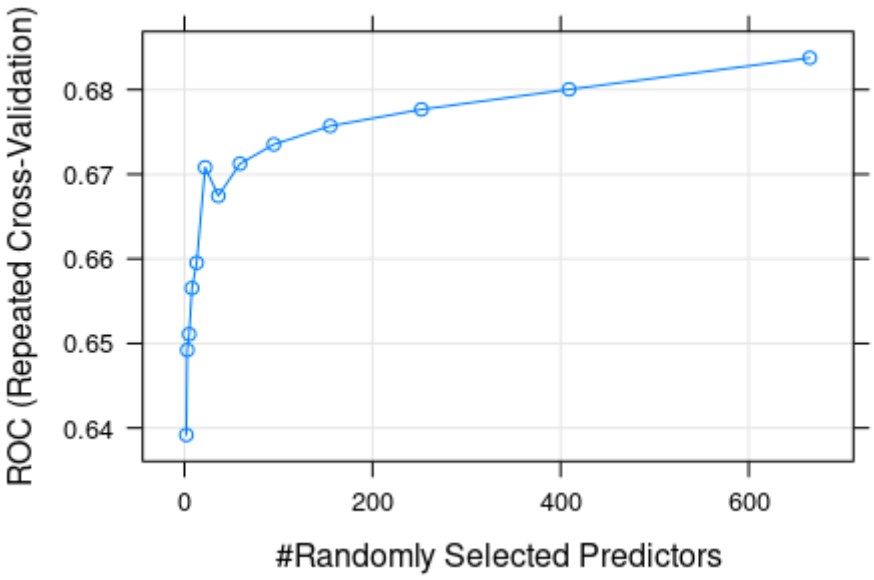

Figure 1: Courbe ROC avec validation croisée

Le modèle ayant la meilleure prédiction par validation croisée est alors le modèle final en sortie. On regardera ensuite les performances de l'arbre apris sur le jeu test

```
decision <- rfFit$finalModel

#  Prédiction du modèle
prediction <- predict(decision,cancerTest)
# la matrice de confusion
confMat <- confusionMatrix(prediction,cancerTest$Y)
save(confMat, file = "data/confMat.RData")
```

```
load("data/confMat.RData")
confMat
```

l'Accuracy indique la proportion de bien classé (le pourcentage de nombres d'individus bien classés sur le jeu text). Il indique 70%. la sensitivité qui indique la proportion de vraies décédées du cancer est de 96%. On a 7% qui sont survécues ou qont mortes d'une autre façon différente non causée par le cancer du sein.

2. Méthode de prédiction SVM

    - Sur le jeu d'apprentissage créé précédemment, on apllique les deux méthodes de SVM: linéaire et à noyaux.

```
linearsvm <-  svm(formula = Y~.,
                  data = cancerTraining,
                  type = 'C-classification',
                  kernel = 'linear'
                  )


gaussiansvm <- svm(formula = Y~.,
                   data = cancerTraining,
                   type = 'C-classification',
                   kernel = 'radial'
                   )
```

- Prédiction sur le jeu test

```
linearpred   <- predict(linearsvm, newdata = cancerTest)
gaussianpred <- predict(gaussiansvm, newdata = cancerTest)

confMatSVM_lin <- confusionMatrix(data=linearpred,cancerTest$Y)
confMatSVM_gau <- confusionMatrix(data=gaussianpred,cancerTest$Y)
save(confMatSVM_lin,file="data/confMatSVM_lin.RData")
save(confMatSVM_gau,file="data/confMatSVM_gau.RData")
```

```
load(file="data/confMatSVM_lin.RData")
load(file="data/confMatSVM_gau.RData")
print(confMatSVM_lin)
```

```
## $positive
## [1] "no"
##
## $table
##           Reference
## Prediction  no yes
```

```
##        no  213  89
##        yes  94  63
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.6013072      0.1073930      0.5548900      0.6464103      0.6688453
## AccuracyPValue  McnemarPValue
##      0.9989714      0.7674680
##
## $byClass
##          Sensitivity          Specificity       Pos Pred Value
##            0.6938111            0.4144737            0.7052980
##       Neg Pred Value            Precision               Recall
##            0.4012739            0.7052980            0.6938111
##                   F1           Prevalence       Detection Rate
##            0.6995074            0.6688453            0.4640523
## Detection Prevalence    Balanced Accuracy
##            0.6579521            0.5541424
##
## $mode
## [1] "sens_spec"
##
## $dots
## list()
##
## attr(,"class")
## [1] "confusionMatrix"
```

```
print(confMatSVM_gau)
```

```
## $positive
## [1] "no"
##
## $table
##           Reference
## Prediction  no yes
##        no  287 129
##        yes  20  23
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   6.753813e-01   1.052163e-01   6.304286e-01   7.180558e-01   6.688453e-01
## AccuracyPValue  McnemarPValue
##   4.041379e-01   8.933764e-19
```

```
##
## $byClass
##          Sensitivity          Specificity       Pos Pred Value
##            0.9348534            0.1513158            0.6899038
##       Neg Pred Value            Precision               Recall
##            0.5348837            0.6899038            0.9348534
##                   F1           Prevalence       Detection Rate
##            0.7939142            0.6688453            0.6252723
## Detection Prevalence    Balanced Accuracy
##            0.9063181            0.5430846
##
## $mode
## [1] "sens_spec"
##
## $dots
## list()
##
## attr(,"class")
## [1] "confusionMatrix"
```

3. Méthode de prédiction avec pénalité

Nous disposons de trois méthodes pour la régression pénalisée: Lasso, Ridge et Elastic

```
#jeu d'apprentissage
y.train <- cancerTraining %>%
  select(Y) %>%
  as.matrix()

x.train <- cancerTraining %>%
  select(-Y) %>%
  as.matrix()
```

- Pénalité Ridge

```
model.ridge <- glmnet(x.train,y.train,family = "binomial",alpha=0)
plot(model.ridge)
```

```
knitr::include_graphics("pictures/ridge.png")
```

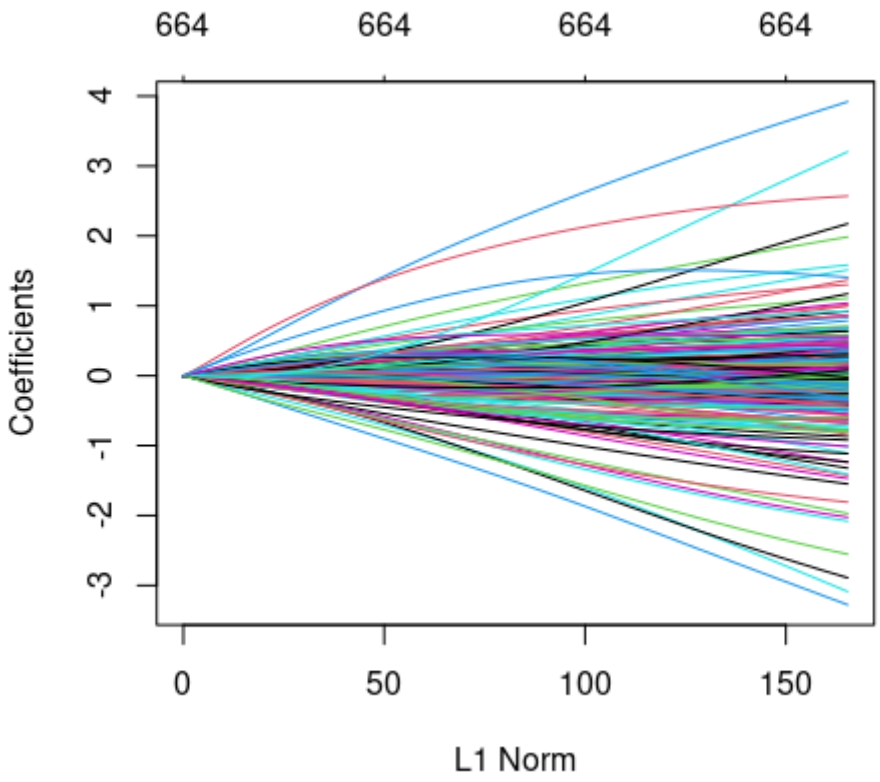

Figure 2: Modèle ridge

- Lasso

```
model.lasso <- glmnet(x.train,y.train,family = "binomial",alpha=1)
```

```
knitr::include_graphics("pictures/lasso.png")
```

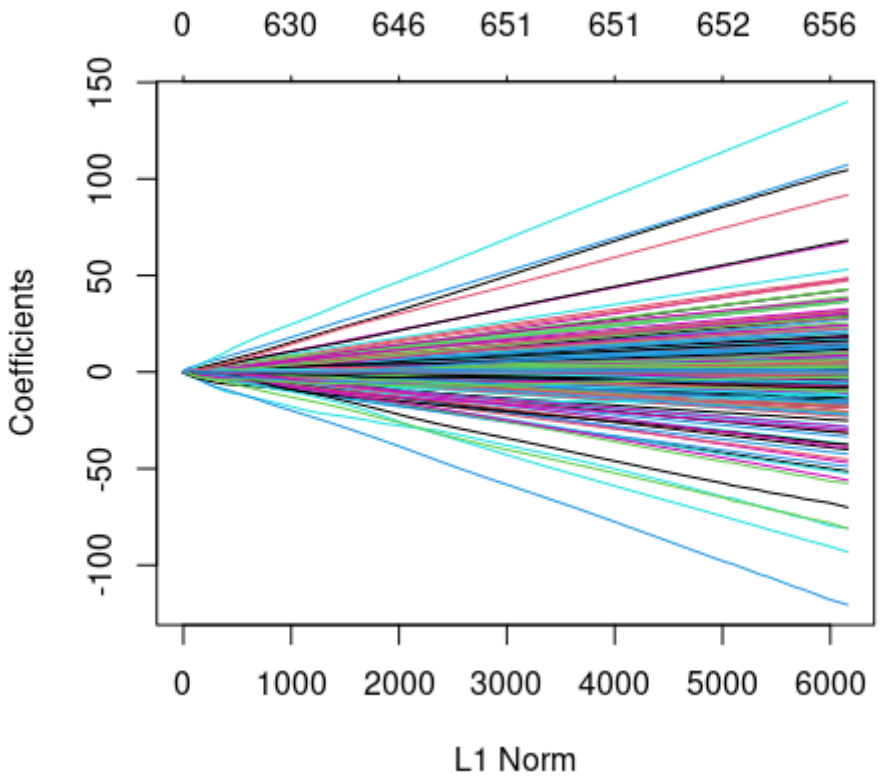

Figure 3: Modèle Lasso

- Elastic net

```
model.elastic_net <- glmnet(x.train,y.train,family = "binomial",alpha=.5)
```

```
knitr::include_graphics("pictures/elastic_net.png")
```

Nous choisirons la méthode *elastic net* en procédant par validation croisée. Car c'est un bon compromis entre les deux.

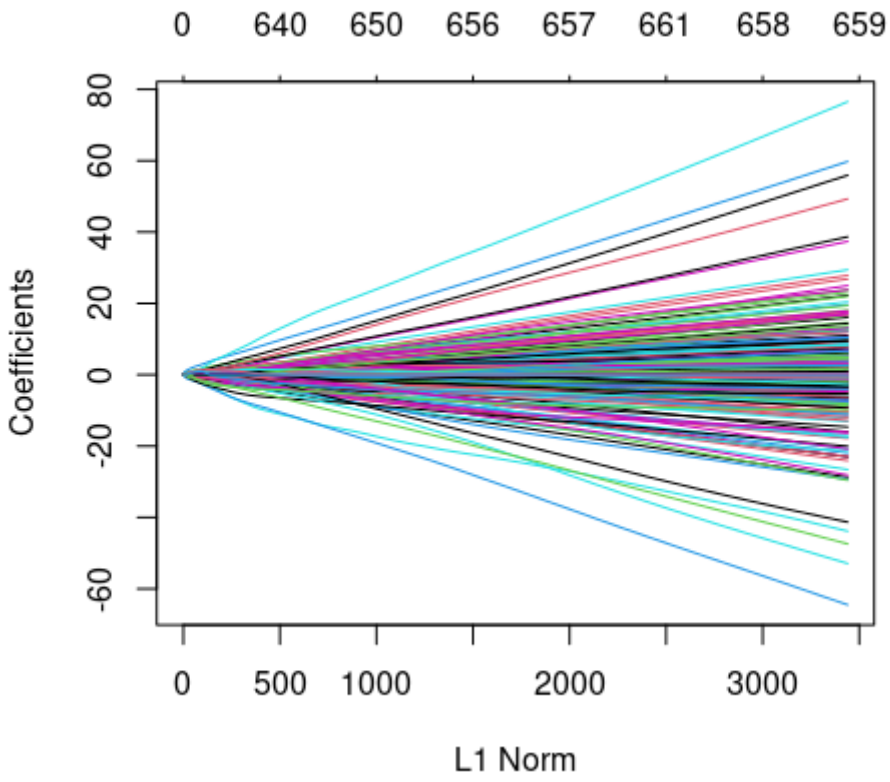

Figure 4: Modèle Elastic net

```
cancer.cv <- cv.glmnet(x.train,y.train,nfolds=20,family="binomial")
cancer.cv$lambda.min
model.EN.cv <- glmnet(x.train,y.train,family="binomial",alpha=0.5,nlambda=1,lambda=cance
```

- Prédiction du modèle sur le jeu test

```
cancerpredict.EN.cv <- predict(model.EN.cv,x.train,type="response",family="binomial")
results <- tibble(proba=cancerpredict.EN.cv,y.pred=round(cancerpredict.EN.cv),y.truth=(y
results
cfMat.EN <- confusionMatrix(data = as_factor(results$y.pred),as_factor(results$y.truth))
save(cfMat.EN,file = "pictures/cfMat_acp.RData")
```

```
load(file = "pictures/cfMat_acp.RData")
cfMat.EN
```

La matrice de confusion donne un bon taux de classement (78%) sur le jeu test avec une bonne sensitivité de 93% et une spécificité assez faible de 47%.

b. par ACP

- Réduction de dimension par ACP

```
acp <- datacancer %>%
  select(-Y) %>% as.matrix() %>%
  prcomp()

# selection des variables avec 70%
nvar <- which(cumsum(acp[["sdev"]])/sum(acp[["sdev"]])>.7)

cancerTraining_ACP <- as_tibble(acp$x[,1:nvar]) %>%
                    mutate(Y=datacancer$Y,
                    .before="PC1"
                      ) %>%
                    slice(n=inTrain)

cancerTest_ACP <- as_tibble(acp$x[,1:nvar]) %>%
                    mutate(Y=datacancer$Y,
                    .before="PC1"
                      ) %>%
                    slice(n=-inTrain)
```

1. Méthodes de prédiction avec forêt aléatoire

```
rfFit_ACP <- train(
  Y ~ .,
  data = cancerTraining_ACP,
  method = "rf",
  tuneLength = 13, #donne le nombre de valeurs à essayer pour chaque paramètre à faire
  trControl = ctrl,
  metric = "ROC"
)
plot(rfit)
```

```
knitr::include_graphics("pictures/ROC_cross_validation_acp.png", auto_pdf = TRUE)
```

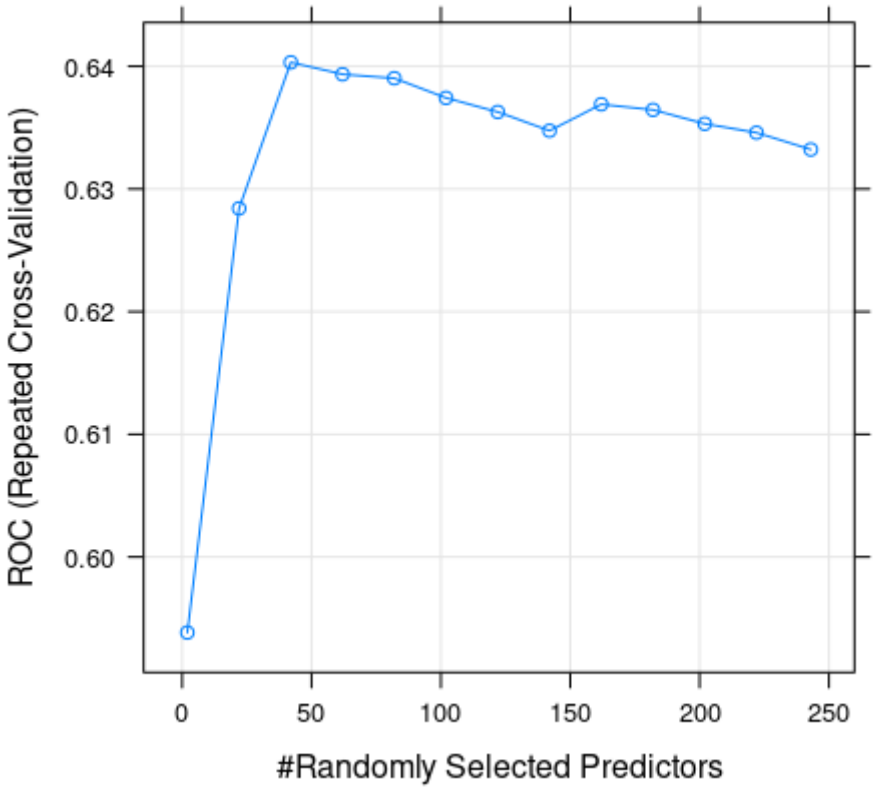

Figure 5: Courbe ROC par validation croisée/Avec réduction de dimension

```
load("data/confMat_ACP.RData")
confMat_ACP
```

```
## $positive
```

```
## [1] "no"
##
## $table
##           Reference
## Prediction  no yes
##        no  297 146
##        yes  10   6
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   6.601307e-01   8.913742e-03   6.147833e-01   7.033988e-01   6.688453e-01
## AccuracyPValue  McnemarPValue
##   6.738991e-01   3.132446e-27
##
## $byClass
##          Sensitivity          Specificity       Pos Pred Value
##           0.96742671           0.03947368           0.67042889
##       Neg Pred Value            Precision               Recall
##           0.37500000           0.67042889           0.96742671
##                   F1           Prevalence       Detection Rate
##           0.79200000           0.66884532           0.64705882
## Detection Prevalence    Balanced Accuracy
##           0.96514161           0.50345020
##
## $mode
## [1] "sens_spec"
##
## $dots
## list()
##
## attr(,"class")
## [1] "confusionMatrix"
```

66% des individus sont bien classés selon cette méthode avec un bon taux du nombre de patientes décédées (97%). Cependant la spécifité est très faible (4%)

2. Méthode de prédiction SVM

- On réutilise les jeux d'apprentissage et de test précédemment dans cette dsection (*cancderTraining_ACP* et *canccerTest_ACP*).

```
linearsvm_acp <-  svm(formula = Y~.,
                  data = cancerTraining_ACP,
                  type = 'C-classification',
```

```
                  kernel = 'linear')

gaussiansvm_acp <- svm(formula = Y~.,
                   data = cancerTraining_ACP,
                   type = 'C-classification',
                   kernel = 'radial')
```

- Prédiction sur le jeu test et matrice de confusion

```
linearpred_acp      <- predict(linearsvm_acp, newdata = cancerTest_ACP)
gaussianpred_acp    <- predict(gaussiansvm_acp, newdata = cancerTest_ACP)
confMat_svm_acp_lin <- confusionMatrix(data=linearpred_acp,cancerTest_ACP$Y)
confMat_svm_acp_gau <- confusionMatrix(data=gaussianpred_acp,cancerTest_ACP$Y)

save(confMat_svm_acp_lin, file = "data/confMat_svm_acp_lin.RData")
save(confMat_svm_acp_gau,file = "data/confMat_svm_acp_gau.RData")
```

```
load("data/confMat_svm_acp_lin.RData")
load("data/confMat_svm_acp_gau.RData")

confMat_svm_acp_lin
```

```
## $positive
## [1] "no"
##
## $table
##           Reference
## Prediction  no yes
##        no  240 110
##        yes  67  42
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##    0.614379085    0.062541108    0.568139669    0.659134549    0.668845316
## AccuracyPValue  McnemarPValue
##    0.993830902    0.001594487
##
## $byClass
##          Sensitivity          Specificity       Pos Pred Value
##            0.7817590            0.2763158            0.6857143
##       Neg Pred Value            Precision               Recall
##            0.3853211            0.6857143            0.7817590
##                   F1           Prevalence       Detection Rate
```

```
##            0.7305936            0.6688453            0.5228758
## Detection Prevalence    Balanced Accuracy
##            0.7625272            0.5290374
##
## $mode
## [1] "sens_spec"
##
## $dots
## list()
##
## attr(,"class")
## [1] "confusionMatrix"
```

```
confMat_svm_acp_gau
```

```
## $positive
## [1] "no"
##
## $table
##           Reference
## Prediction  no yes
##        no  305 145
##        yes   2   7
##
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   6.797386e-01   5.185279e-02   6.349081e-01   7.222341e-01   6.688453e-01
## AccuracyPValue  McnemarPValue
##   3.293343e-01   1.106874e-31
##
## $byClass
##          Sensitivity          Specificity       Pos Pred Value
##            0.99348534           0.04605263           0.67777778
##       Neg Pred Value            Precision               Recall
##            0.77777778           0.67777778           0.99348534
##                   F1           Prevalence       Detection Rate
##            0.80581242           0.66884532           0.66448802
## Detection Prevalence    Balanced Accuracy
##            0.98039216           0.51976899
##
## $mode
## [1] "sens_spec"
##
## $dots
```

```
## list()
##
## attr(,"class")
## [1] "confusionMatrix"
```

3. Méthode de prédiction avec pénalité

```
#jeu d'apprentissage
y.train_acp <- cancerTraining_ACP %>%
  select(Y) %>%
  as.matrix()

x.train_acp <- cancerTraining_ACP %>%
  select(-Y) %>%
  as.matrix()
```

- Pénalité Ridge

```
model.ridge_acp <- glmnet(x.train_acp,y.train_acp,family = "binomial" ,alpha=0)
plot(model.ridge_acp)
```

```
knitr::include_graphics("pictures/ridge_acp.png", auto_pdf = TRUE)
```

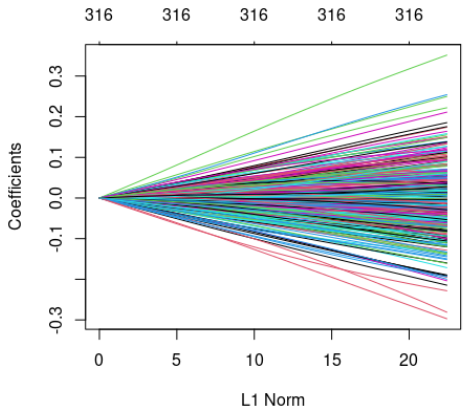

Figure 6: Méthde de Ridge

- Pénalité Lasso

```
model.lasso_acp <- glmnet(x.train_acp,y.train_acp,family = "binomial" , alpha=1)
plot(model.lasso_acp)
```

```
knitr::include_graphics("pictures/lasso_acp.png", auto_pdf = TRUE)
```

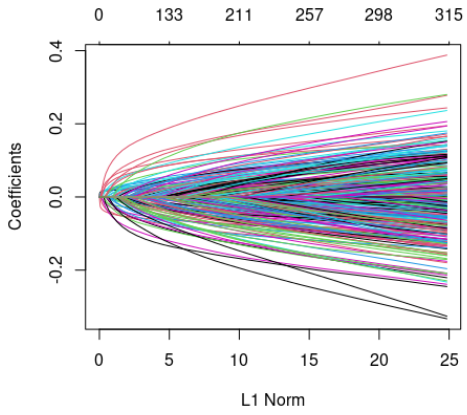

Figure 7: Pénalité Lasso

- Pénalité Elastic net

```
model.elastic_net_acp <- glmnet(x.train_acp,y.train_acp,family = "binomial" , alpha=.5)
plot(model.elastic_net_acp)
```

```
knitr::include_graphics("pictures/elastic_net_acp.png", auto_pdf = TRUE)
```

- Prédition avec elastic net

```
cancer.cv_acp <- cv.glmnet(x.train_acp,y.train_acp,nfolds=10,family="binomial" )
cancer.cv_acp$lambda.min
model.EN.cv_acp <- glmnet(x.train_acp,y.train_acp,family="binomial" ,alpha=0.5,nlambda=1

cancerpredict.EN.cv_acp <- predict(model.EN.cv_acp,x.train_acp,type="response", family="
results_acp <- tibble(proba=cancerpredict.EN.cv_acp,pred=round(cancerpredict.EN.cv_acp),
results_acp
confMat_EN_acp <- confusionMatrix(data=as_factor(results_acp$pred),as_factor(results_acp
save(confMat_EN_acp,file = "data/confMat_EN_acp.RData")
```

- Matrice de confusion

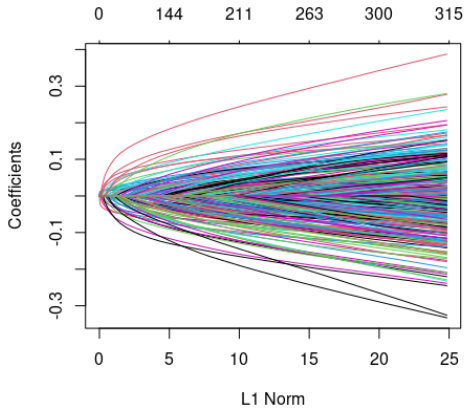

Figure 8: Elastic net

```
load(file = "data/confMat_EN_acp.RData")
confMat_EN_acp
```

c. par auto-encodeur

1. Méthodes de prédiction avec forêt aléatoire
2. Méthode de prédiction SVM
3. Méthode de prédiction avec pénalité

3 Sélection

Références

[1] CHOLLET F., KALINOWSKI T., ALLAIRE J. J. Second Edition.[s.l.] : MANNING, 2022.