TP3 MRR

Noah KWA MOUTOME - Victor TAN

2023-11-11

II. Cookies Study

##Logistic regression model using features

Let's begin with processing our dataset :

```r
# Load the dataset
cookies_data <- read.csv(file = "cookies.csv", header = TRUE)
# Create the binary target variable YBin
YBin <- as.numeric(cookies_data$fat > median(cookies_data$fat))

# Computation (mean, standard deviation, minimum and maximum)
cookies_data$mean <- rowMeans(cookies_data[, 2:701])
cookies_data$stDev <- apply(cookies_data[, 2:701], 1, sd)
cookies_data$min <- apply(cookies_data[, 2:701], 1, min)
cookies_data$max <- apply(cookies_data[, 2:701], 1, max)

# Computation (slope)
# Function: compute_slope
# @param: spectrum_values of a cookie (here, column 2 to 701)
# @return: slope of the spectrum curve for a cookie
compute_slope <- function(spectrum_values) {
  pos <- 1:length(spectrum_values)
  lm_model <- lm(spectrum_values ~ pos)
  slope <- coef(lm_model)[2]
  return(slope)
}

cookies_data$slope <- apply(cookies_data[, 2:701], 1, compute_slope)

# Create a data frame with features and target variable
cookies_features_data <- data.frame(
  YBin = YBin,
  mean = cookies_data$mean,
  stDev = cookies_data$stDev,
  min = cookies_data$min,
  max = cookies_data$max,
  slope = cookies_data$slope
)

head(cookies_features_data)
```

```
##   YBin      mean     stDev      min     max       slope
## 1    0 0.9851499 0.4111868 0.259270 1.73946 0.001914311
## 2    1 1.0355417 0.4123933 0.266864 1.66273 0.001898164
## 3    0 1.0010620 0.4025158 0.251654 1.60960 0.001860203
## 4    0 1.0280481 0.4040351 0.277777 1.63881 0.001861782
## 5    1 1.0655011 0.4158252 0.288328 1.70320 0.001910926
## 6    0 1.0840236 0.4262425 0.284625 1.74356 0.001967228
```

```r
print(YBin)
```

```
##  [1] 0 1 0 0 1 0 0 1 1 0 1 0 1 1 1 0 1 0 0 0 0 1 1 1 1 1 0 1 0 1 0 0
```

```r
# Fit logistic regression model
model <- glm(YBin ~ ., data = cookies_features_data, family = "binomial")

# Display the summary of the model
summary(model)
```

```
##
## Call:
## glm(formula = YBin ~ ., family = "binomial", data = cookies_features_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.80802  -0.65396  -0.04465   0.67651   1.91688
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.565e+01  2.440e+01  -1.051   0.2931
## mean        -4.440e+01  6.208e+01  -0.715   0.4745
## stDev        1.129e+03  6.271e+02   1.801   0.0717 .
## min          1.371e+02  1.288e+02   1.065   0.2870
## max          1.894e+00  2.294e+01   0.083   0.9342
## slope       -2.284e+05  1.186e+05  -1.925   0.0542 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 44.361  on 31  degrees of freedom
## Residual deviance: 26.927  on 26  degrees of freedom
## AIC: 38.927
##
## Number of Fisher Scoring iterations: 6
```

And here are the results of the regression :

```r
p1 <- predict.glm(model, type = "response")
res <- as.numeric(p1 > 0.5)

print(res)
```

```
##  [1] 0 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0
```

```r
print(cookies_features_data$YBin)
```

```
##  [1] 0 1 0 0 1 0 0 1 1 0 1 0 1 1 1 0 1 0 0 0 0 1 1 1 1 1 0 1 0 1 0 0
```

We see, according to the p-values, that only the standard deviation seems to have an impact on the class of fat for a given cookie. However, the threshold is relatively higher than usual (0.1 here) so it might be difficult to conclude.

Logistic regression model using the spectra

```r
# Load the dataset
cookies_data <- read.csv(file = "cookies.csv", header = TRUE)

# Create the binary target variable YBin
YBin <- as.numeric(cookies_data$fat > median(cookies_data$fat))

# Spectra as predictors (without fat column)
spectra_predictors <- cookies_data[, 2:701]
# Add the YBin column to the predictors
spectra_predictors$YBin <- YBin

head(spectra_predictors$YBin)
```

```
## [1] 0 1 0 0 1 0
```

Let's add some penalization to the model.

Because we have a lot of covariables and few observations, we see that $p \ll n$ and so we need to select some of them to make the model simpler.

To do so, we're going to use a l_1-regularization :

Lasso :

```r
# Load the glmnet package
library(glmnet)
```

```
## Le chargement a nécessité le package : Matrix

## Loaded glmnet 4.1-8
```

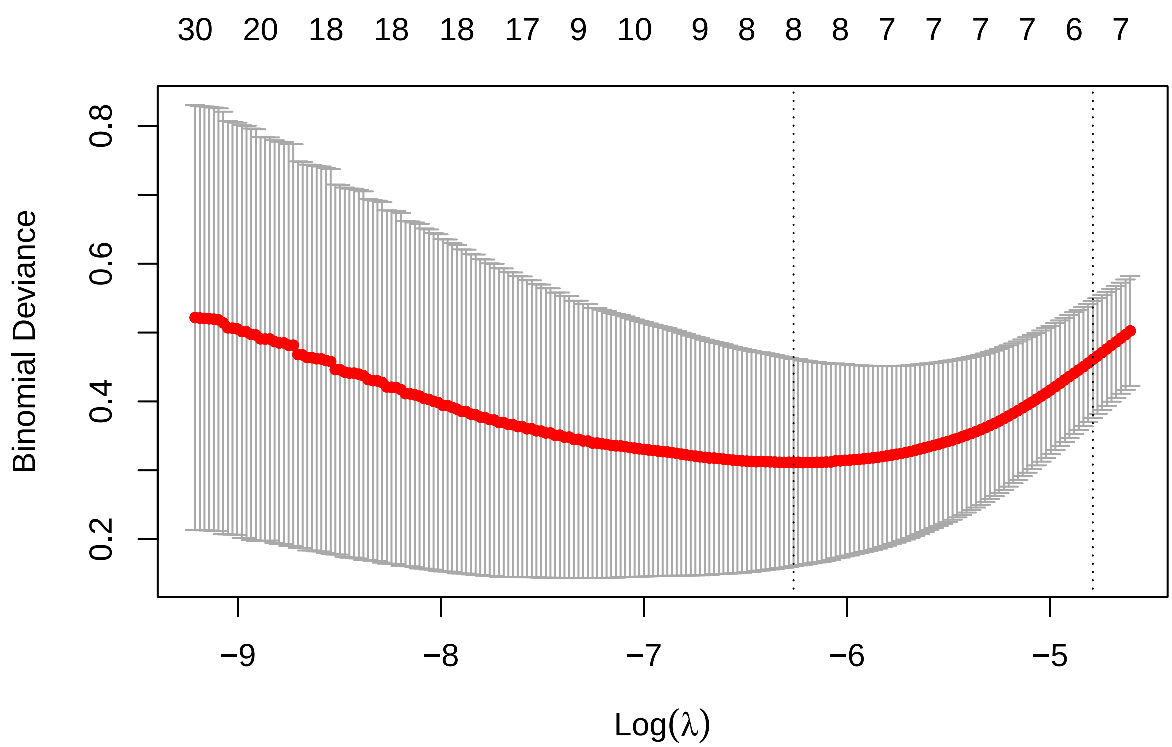
```r
# Convert predictors and response to matrix format
X <- as.matrix(spectra_predictors[, -ncol(spectra_predictors)])  # Exclude the YBin column
y <- spectra_predictors$YBin

# Fit regularized logistic regression model (L1 penalty)
lambdas <- 10^seq(-4, -2, 0.01)
cv_model_lasso <- cv.glmnet(X, y, family = "binomial", alpha = 1, lambda = lambdas)

# Display the optimal lambda
best_lambda_lasso <- cv_model_lasso$lambda.min
cat("Optimal lambda for Lasso:", best_lambda_lasso, "\n")
```

```
## Optimal lambda for Lasso: 0.001905461
```

```r
plot(cv_model_lasso)
```

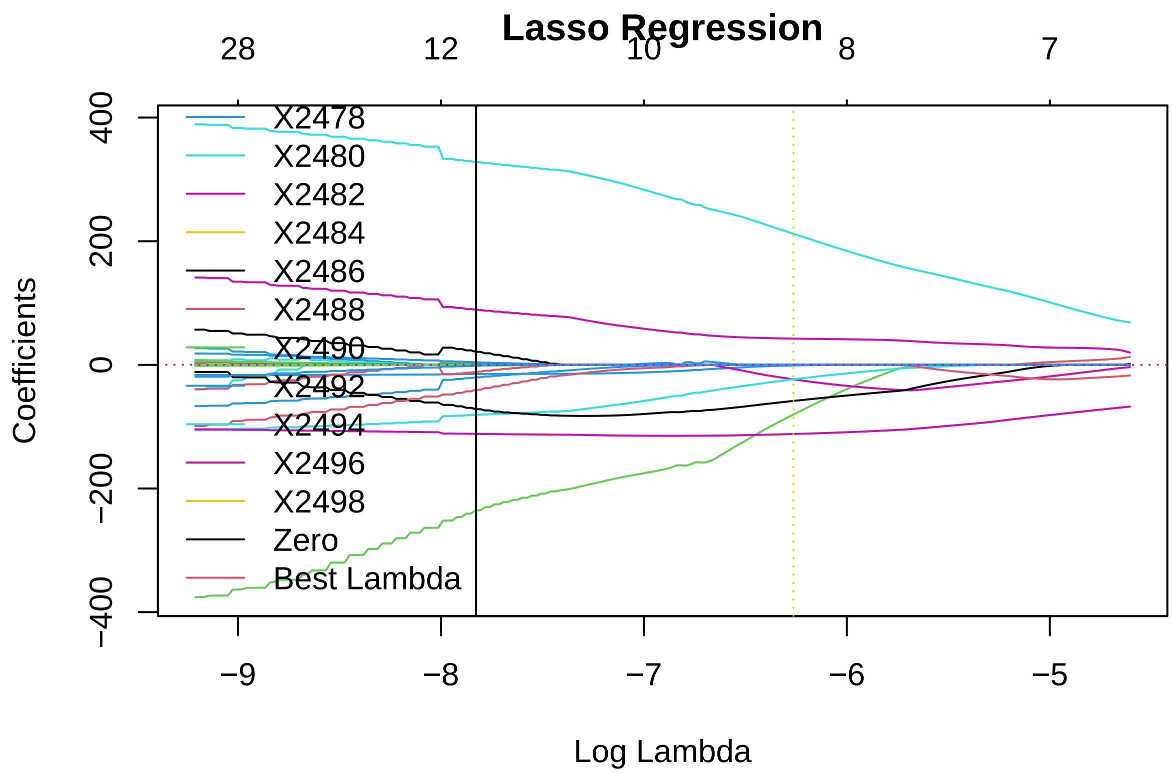



```r
print(cv_model_lasso)
```

```
##
## Call:  cv.glmnet(x = X, y = y, lambda = lambdas, family = "binomial",      alpha = 1)
##
## Measure: Binomial Deviance
##
##        Lambda Index Measure     SE Nonzero
## min 0.001905    73  0.3111 0.1504       8
## 1se 0.008318     9  0.4609 0.0847       6
```

Here's the regulariztions path :

```r
plot(cv_model_lasso$glmnet.fit, xvar = "lambda", main="Lasso Regression")
abline(h = 0, col = 6, lty = 3)
abline(v = log(best_lambda_lasso), col = 7, lty = 3)
legend("bottomleft", legend = c(colnames(X), "Zero", "Best Lambda"), col = 1:7, lty = 1)
```



Since with the best λ we get 9 not null coefficients over the 700 initial covariables, against 6 for the 1 Standard Error λ, we choose to use the λ_{min}.

Here's the Lasso regression with the best λ :

```
##
## Call:  glmnet(x = X, y = y, family = "binomial", alpha = 1, lambda = best_lambda_lasso)
##
##   Df  %Dev   Lambda
## 1  8 94.41 0.001905
```

Let's now test its perfomance by using a K-fold :

```
# We split into 2 dataframe randomly
indice_train <- sample(1:nrow(spectra_predictors), 0.8 * nrow(spectra_predictors))
train_data <- spectra_predictors[indice_train, ] # use to train the model
test_data <- spectra_predictors[-indice_train, ] # use to test the model

# We define X & y for both dataframe
y_train <- train_data$YBin
X_train <- train_data[, -ncol(train_data)]
# ---
y_test <- test_data$YBin
X_test <- test_data[, -ncol(test_data)]

# We train the model with another cross-validation
```

```
# in order to get the best value for lambda
cv_lasso_model <- cv.glmnet(as.matrix(X_train), y_train, alpha = 1, family = "binomial", grouped = FALSE
# We choose the best lambda
best_lambda <- cv_lasso_model$lambda.min
# Using the best lambda, we define our model
lasso_model <- glmnet(as.matrix(X_train), y_train, lambda = best_lambda, alpha = 1, family = "binomial")

# We make prediction on the X_test
predictions_lasso <- predict.glmnet(lasso_model, s = best_lambda, newx = as.matrix(X_test), type = "resp

# We compute the class for our prediction using the threshold 0.5
predictions <- as.numeric(predictions_lasso > 0.5)

# Compute the performance
performance <- mean(predictions == y_test)
print(paste("Performance globale de la validation croisée (accuracy):", round(performance, 2)))
```

```
## [1] "Performance globale de la validation croisée (accuracy): 1"
```

```
# Confusion matrix
print(table(test_data$YBin, predictions))
```

```
##    predictions
##     0 1
##   0 4 0
##   1 0 3
```

We can conclude that our model using Lasso regression is pretty accurate, and so with only 9 co-variables instead of 700 !