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 :

```
# 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
```

```
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
```

```
# 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
```

We see, according to the p-values, that only the standard deviation and the slope seem to have an impact on the class of fat for a given cookie, as they are around 0.05. The other features are not significant.

Let's see the confusion matrix and the accuracy of our model with a k-fold:

```
n <- nrow(cookies_features_data)
ind <- sample(n)
tab <- cookies_features_data[ind, ]
```

```
K <- 4
lblock <- trunc(n/K)
predictions <- c()
true_values <- tab$YBin
for (k in (1:K)) {
  indk <- ((k-1) * lblock + 1):(k * lblock);
  tabTrain <- tab[-indk, ];
  modTrain <- glm(YBin ~ . , data = tabTrain, family = "binomial");
  testk <- predict.glm(modTrain, newdata = tab[indk, ], type = "response");
  predk <- ifelse(testk > 0.5, 1, 0);
  predictions <- c(predictions, predk);
}
```

```
## Warning: glm.fit: l'algorithme n'a pas convergé
```

```
## Warning: glm.fit: des probabilités ont été ajustées numériquement à 0 ou 1
```

```
table(true_values, predictions)
```

```
##            predictions
## true_values  0  1
##           0 12  4
##           1  4 12
```

```
## [1] "Accuracy: 0.75"
```

```
## [1] "Error rate: 0.25"
```

To conclude, we can say that our model is relatively accurate, with an accuracy close to 0.8. It means that our logistic model can explain or predict if the fat will be higher or lower than the median of all the fat, given the previous computed features. However, because we only have 32 observations, we can't be sure that our model is really accurate: we would need more observations to be sure of that.

Let's see the correlation between the features :

```
cor(cookies_features_data[, -1])
```

```
##            mean     stDev       min       max     slope
## mean  1.0000000 0.8376438 0.9388982 0.6952187 0.8132177
## stDev 0.8376438 1.0000000 0.6802539 0.9654640 0.9977856
## min   0.9388982 0.6802539 1.0000000 0.5328707 0.6505255
## max   0.6952187 0.9654640 0.5328707 1.0000000 0.9693959
## slope 0.8132177 0.9977856 0.6505255 0.9693959 1.0000000
```

We can also try to study the odd-values, but it wouldn't be relevant, as the covariables (features) seem dependent of each other.

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 $p \gg n$ 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
```

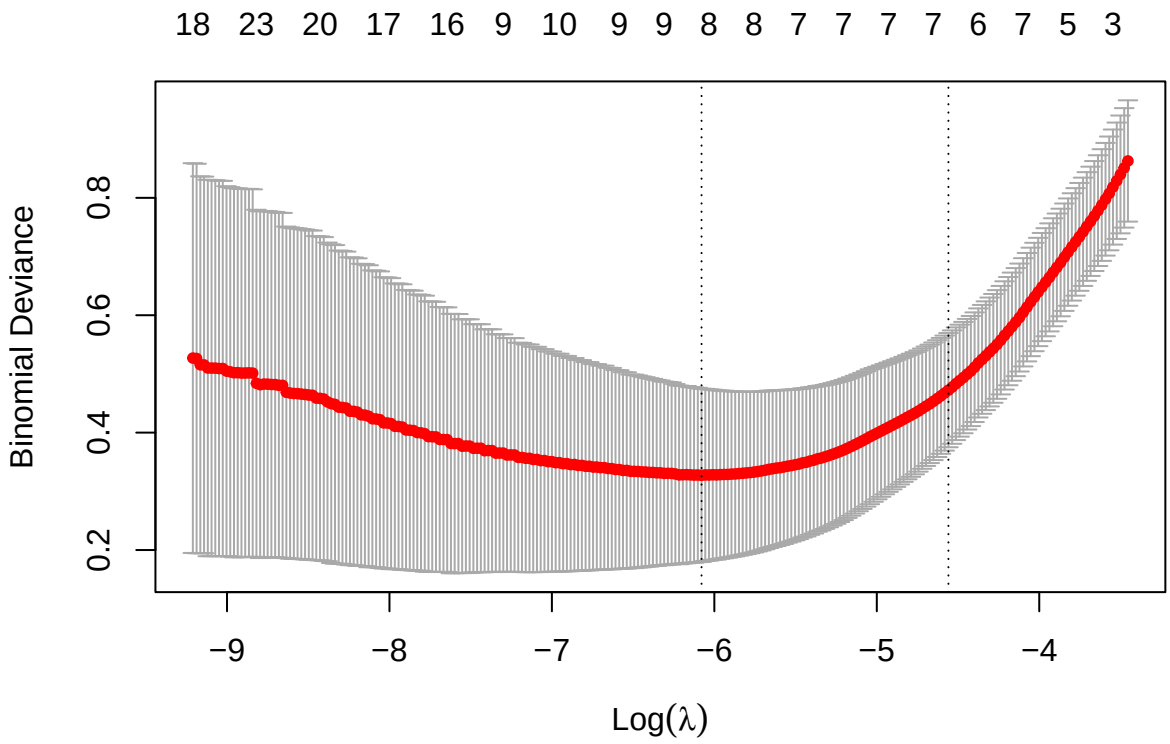
```r
# Convert predictors and response to matrix format
X <- as.matrix(spectra_predictors[, -ncol(spectra_predictors)])  # Exclude the YBin column
y <- spectra_predictors$YBin

# Fit regularized logistic regression model (L1 penalty)
lambdas <- 10^seq(-4, -1.5, 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: 0.002290868
```

```r
lasso_model <- glmnet(X, y, family = "binomial", alpha = 1, lambda = best_lambda_lasso)
print(lasso_model)
```

```
##
## Call:  glmnet(x = X, y = y, family = "binomial", alpha = 1, lambda = best_lambda_lasso)
##
##    Df  %Dev   Lambda
## 1   8 93.46 0.002291
```

```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 0.002291   115  0.3272 0.14754       9
## 1se 0.010471    49  0.4724 0.09726       7
```

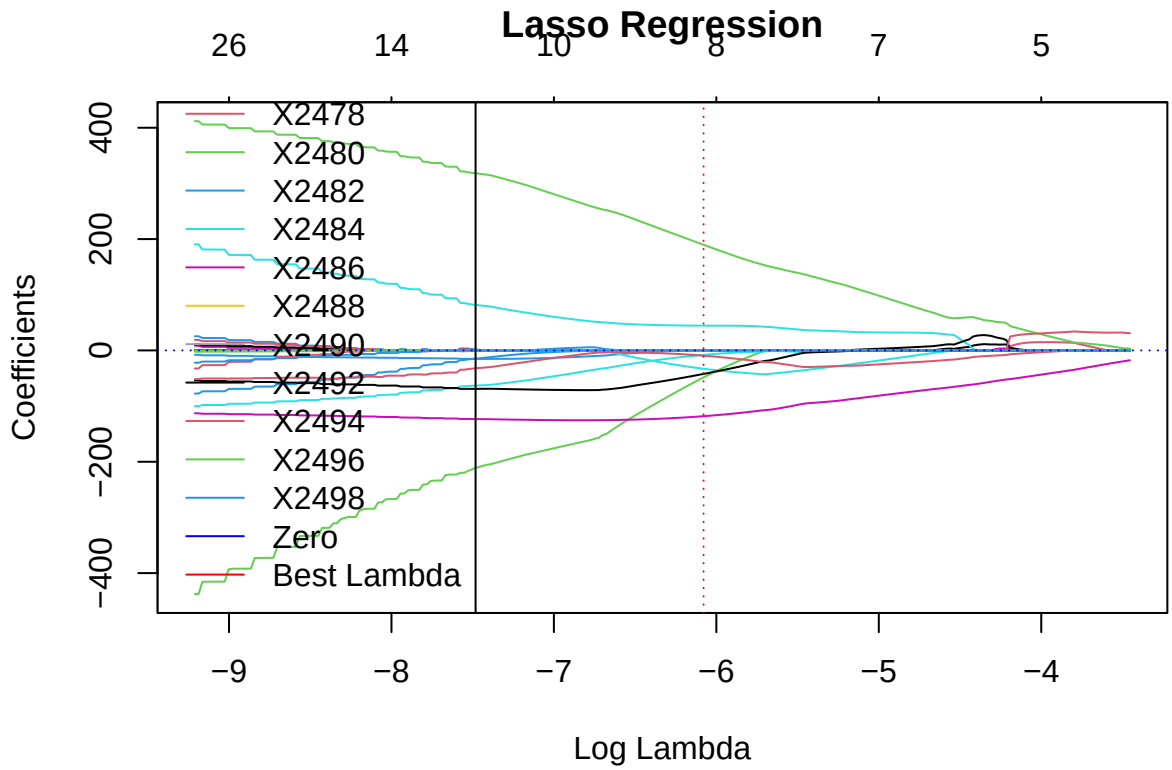
```r
non_zero_coefficients <- (lasso_model$beta[which(lasso_model$beta != 0)])
non_zero_spectra <- (colnames(X)[which(lasso_model$beta != 0)])
data.frame(non_zero_spectra, non_zero_coefficients)
```

```
##   non_zero_spectra non_zero_coefficients
## 1            X1208            -57.025997
## 2            X1210             -6.574214
## 3            X1724            -26.443515
## 4            X1902            204.907041
## 5            X1912             29.737955
## 6            X2072            -68.634028
```

```
## 7            X2074           -57.280971
## 8            X2300           -52.323934
```

Here's the regulariztions path :

```
plot(cv_model_lasso$glmnet.fit, xvar = "lambda", main="Lasso Regression")
abline(h = 0, col = "blue", lty = 3)
abline(v = log(best_lambda_lasso), col = "red", lty = 3)
legend("bottomleft", legend = c(colnames(X), "Zero", "Best Lambda"), col = c(1:700, "blue", "red"), lty
```



Since with the best λ we get 9 not null coefficients (including intercept) over the 700 initial covariables, (and 7 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 93.46 0.002291
```

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

```
n <- nrow(spectra_predictors)
ind <- sample(n)
```

```r
tab <- spectra_predictors[ind, ]
K <- 4
lblock <- trunc(n/K)
predictions <- c()
true_values <- tab$YBin
for (k in (1:K)) {
  indk <- ((k-1) * lblock + 1):(k * lblock);
  tabTrain <- tab[-indk, ];
  formula <- as.formula(paste("YBin ~ ", paste(non_zero_spectra, collapse = " + ")))
  modTrain <- glm(formula, data = tabTrain, family = "binomial", maxit = 10000);
  testk <- predict.glm(modTrain, newdata = tab[indk, ], type = "response");
  predk <- ifelse(testk > 0.5, 1, 0);
  predictions <- c(predictions, predk);
}
```

```
## Warning: glm.fit: des probabilités ont été ajustées numériquement à 0 ou 1

## Warning: glm.fit: des probabilités ont été ajustées numériquement à 0 ou 1

## Warning: glm.fit: des probabilités ont été ajustées numériquement à 0 ou 1

## Warning: glm.fit: des probabilités ont été ajustées numériquement à 0 ou 1
```

```r
table(true_values, predictions)
```

```
##            predictions
## true_values  0  1
##           0 15  1
##           1  2 14
```

```
## [1] "Accuracy: 0.91"
```

```
## [1] "Error rate: 0.09"
```

We can conclude that our model using Lasso regression is pretty accurate, and so with only less co-variables instead of 700. It means that our logistic model can explain or predict if the fat will be higher or lower than the median of all the fat, given these co-variables. However, because we only have 32 observations, we can't be sure that our model is really accurate: we would need more observations to be sure of that.