PROJET MRR : Prédire la popularité d'un animé

Arnaud GRASSIAN et Vithuson VAITHILINGAM

Décembre 2025

1 Introduction

Ce rapport final est construit sous la forme d'une **concaténation de trois documents Markdown**, chacun correspondant à une étape distincte du projet. Ce choix permet de **séparer clairement** (i) la préparation des données, (ii) la modélisation prédictive de la note d'un animé et (iii) la recommandation d'un animé.

1.1 Sommaire

Anime

2025-11-22

Construction des 2 tableaux

Nous avons choisi de traiter un par un les fichiers csv pour ensuite les concaténer en 2 tableaux afin de répondre aux problématiques.

Fonctions utiles pour la suite

Voici des fonctions que nous avons ensuite utilisé pour traiter plus simplement les csv.

```
suppressPackageStartupMessages({
  library(data.table)
  library(dplyr)
  library(tidyr)
  library(stringr)
  library(lubridate)
  library(purrr)
})


# parse_list_col : nettoie une colonne contenant des listes sous forme de
# chaînes de caractères (ex"['a','b']").
# et renvoie une liste de vecteurs propres.
parse_list_col <- function(x) {
  x <- as.character(x)
  x[is.na(x)] <- "[]"
  x <- str_replace_all(x, "\\[|\\]|\\'", "")
  x <- str_trim(x)
  out <- str_split(x, ",")
  out <- lapply(out, function(v) {
    v <- str_trim(v)
    v[v != ""]
  })
  out
}


# multihot_topk :
# - df : data frame
# - id_col : nom de la colonne identifiant
# - list_col : colonne contenant des listes
# - prefix : préfixe des nouvelles colonnes
# - top_k : garde seulement les k éléments les plus fréquents
# Produit une matrice multihot.
```

```r
multihot_topk <- function(df, id_col, list_col, prefix, top_k = NULL) {
  lst <- parse_list_col(df[[list_col]])
  all_levels <- unlist(lst)
  all_levels <- all_levels[all_levels != ""]
  if (length(all_levels) == 0) return(df[, ..id_col, drop = FALSE])

  freq <- sort(table(all_levels), decreasing = TRUE)
  levels_keep <- names(freq)
  if (!is.null(top_k)) levels_keep <- levels_keep[1:min(top_k, length(levels_keep))]

  bin_mat <- sapply(levels_keep, function(level) {
    as.integer(map_lgl(lst, ~ level %in% .x))
  })
  bin_df <- as.data.frame(bin_mat)
  names(bin_df) <- paste0(prefix, "_", make.names(levels_keep))
  bind_cols(df[, ..id_col, drop = FALSE], bin_df)
}



# onehot_topk :
# - col : colonne catégorielle simple
# Transforme en one-hot encoding, réduit aux catégories les plus fréquentes.
onehot_topk <- function(df, col, prefix, top_k = NULL) {
  v <- as.character(df[[col]])
  if (!is.null(top_k)) {
    tab  <- sort(table(v), decreasing = TRUE)
    keep <- names(tab)[seq_len(min(top_k, length(tab)))]
    v2   <- ifelse(v %in% keep, v, "Other")
  } else {
    # top_k = NULL : on garde toutes les modalités originales
    v2 <- v
  }
  mm <- model.matrix(~ v2 - 1)
  mm <- as.data.frame(mm)
  names(mm) <- paste0(prefix, "_", sub("^v2", "", names(mm)))
  dplyr::bind_cols(df %>% dplyr::select(-all_of(col)), mm)
}


# to_num : conversion numérique silencieuse (NA si impossible)
to_num <- function(x) suppressWarnings(as.numeric(x))
```

On traite ensuite 1 par 1 les fichiers csv :

character_anime_works.csv

```r
char_works <- fread("character_anime_works.csv")
# Remplace les points dans les noms de colonnes par des underscores
names(char_works) <- str_replace_all(names(char_works), "\\.", "_")

char_features <- char_works %>%
  group_by(anime_mal_id) %>%
```

```
  summarise(
    num_characters = n_distinct(character_mal_id),
    num_main_characters = n_distinct(character_mal_id[role == "Main"]),
    num_supporting_characters = n_distinct(character_mal_id[role == "Supporting"]),
    .groups = "drop"
  )

head(char_works)
```

```
##    anime_mal_id character_mal_id character_name       role
##           <int>            <int>          <char>     <char>
## 1:         2928             5781           Atoli       Main
## 2:         2928               33           Haseo       Main
## 3:         2928               32            Ovan       Main
## 4:         2928               34           Shino       Main
## 5:         2928             5785            Aina Supporting
## 6:         2928             5784        Endrance Supporting
```

```
head(char_features)
```

```
## # A tibble: 6 x 4
##   anime_mal_id num_characters num_main_characters num_supporting_characters
##          <int>          <int>               <int>                     <int>
## 1            1            126                   4                       122
## 2            5             42                   6                        36
## 3            6             97                   4                        93
## 4            7             53                   6                        47
## 5            8             10                   5                         5
## 6           15             64                   5                        59
```

Avec le tableau char_features, on obtient pour un anime le nombre de personnages, ainsi que le nombre de personnages principaux et secondaires.

character_nicknames.csv

```
char_nicks <- fread("character_nicknames.csv")
names(char_nicks) <- str_replace_all(names(char_nicks), "\\.", "_")

char_nick_feat <- char_nicks %>%
  mutate(character_mal_id = as.character(character_mal_id)) %>%
  group_by(character_mal_id) %>%
  summarise(num_nicknames = n(), .groups = "drop")
head(char_nick_feat)
```

```
## # A tibble: 6 x 2
##   character_mal_id num_nicknames
##   <chr>                    <int>
## 1 100023                       4
## 2 100053                       1
## 3 10006                        1
## 4 10007                        1
## 5 100073                       1
## 6 10010                        2
```

Pour personnage, on obtient son nombre de nicknames.

person_alternate_names.csv

```
person_alt_names <- fread("person_alternate_names.csv")
names(person_alt_names) <- str_replace_all(names(person_alt_names), "\\.", "_")
```

On n'a pas utilisé ce csv par la suite.

recommendations.csv

```
reco <- fread("recommendations.csv")
names(reco) <- str_replace_all(names(reco), "\\.", "_")

reco_features <- reco %>%
  group_by(mal_id) %>%
  summarise(
    num_recommendations_made = n(),
    .groups = "drop"
  )

reco_features <- reco_features %>% rename(anime_mal_id = mal_id)

head(reco_features)
```

```
## # A tibble: 6 x 2
##   anime_mal_id num_recommendations_made
##          <int>                    <int>
## 1            1                      165
## 2            5                       10
## 3            6                      120
## 4            7                       42
## 5            8                       10
## 6           15                       32
```

On obtient pour chaque anime le nombre de recommandations faites par cet anime.

stats.csv

```
stats <- fread("stats.csv")
names(stats) <- str_replace_all(names(stats), "\\.", "_")

stats <- stats %>% rename(anime_mal_id = mal_id)

stats_features_anime <- stats %>%
  select(
    anime_mal_id,
    watching,
    completed,
    on_hold,
    dropped,
    plan_to_watch,
    total
  )

head(stats_features_anime)
```

```
##    anime_mal_id watching completed on_hold dropped plan_to_watch total
##           <int>    <int>     <int>   <int>   <int>         <int> <int>
## 1:        59356        7       146       4      20            20   197
## 2:        56036       21       770       8      29           113   941
## 3:         2928      451     14953     302     349          6472 22527
## 4:         3269      726     22790     452     537          9762 34267
## 5:         4469      241      6918     182     266          3528 11135
## 6:          454      325     15421     261     423          5027 21457
```

On a choisi de garder pour chaque anime les colonnes suivantes : - watching, le nombre d'utilisateurs actuellement en train de regarder - completed, le nombre d'utilisateurs ayant terminé - on_hold, le nombre d'utilisateurs ayant mis en pause - dropped, le nombre d'utilisateurs ayant abandonné - plan_to_watch, le nombre d'utilisateurs prévoyant de regarder - total, total global

person_voice_works.csv

```
p_voice <- fread("person_voice_works.csv")
names(p_voice) <- str_replace_all(names(p_voice), "\\.", "_")

p_voice[, person_mal_id    := as.character(person_mal_id)]
p_voice[, character_mal_id := as.character(character_mal_id)]
p_voice[, anime_mal_id     := as.integer(anime_mal_id)]

# Supprime les doublons
p_voice <- p_voice %>% distinct()

voice_features <- p_voice %>%
  group_by(anime_mal_id) %>%
  summarise(
    num_doubleurs         = n_distinct(person_mal_id),
    num_main_doubleurs    = n_distinct(person_mal_id[role == "Main"]),
    num_support_doubleurs  = n_distinct(person_mal_id[role == "Supporting"]),
    .groups = "drop"
  )
head(voice_features)
```

```
## # A tibble: 6 x 4
##   anime_mal_id num_doubleurs num_main_doubleurs num_support_doubleurs
##          <int>         <int>              <int>                 <int>
## 1            1           418                 44                   377
## 2            5           160                 45                   116
## 3            6           302                 42                   264
## 4            7           108                 29                    81
## 5            8            18                 10                     9
## 6           15           189                 31                   163
```

On a pour chaque anime le nombre de doubleurs ainsi que le nombre de doubleurs de personnages principaux et secondaires.

profiles.csv

```
profiles <- fread("profiles.csv")
names(profiles) <- str_replace_all(names(profiles), "\\.", "_")
```

```r
profiles <- profiles %>%
  mutate(
    gender = tolower(gender),
    gender = ifelse(gender %in% c("male","female","non-binary"), gender, "unknown")
  )

birth_year <- str_extract(profiles$birthday, "[0-9]{4}")
birth_year_num <- to_num(birth_year)

profiles <- profiles %>%
  mutate(
    birth_year = birth_year_num,
    age = ifelse(!is.na(birth_year), year(Sys.Date()) - birth_year, NA_real_),
    age = ifelse(age >= 8 & age <= 100, age, NA_real_)
  )

profiles_user <- profiles %>%
  transmute(
    username,
    age,
    gender,
    prof_on_hold       = to_num(on_hold),
    prof_dropped       = to_num(dropped),
    prof_plan_to_watch = to_num(plan_to_watch)
  ) %>%
  onehot_topk("gender", "gender", top_k = NULL)
head(profiles_user)
```

```
##       username   age prof_on_hold prof_dropped prof_plan_to_watch gender_female
##         <char> <num>        <num>        <num>               <num>         <num>
## 1:   ishikawas    NA           NA           NA                  NA             0
## 2:        CKK2    NA           15            0                 405             0
## 3: --------788    NA            0            0                   1             1
## 4:  potatoaris    NA            0            0                   4             0
## 5:   Rinrintan    NA           40           16                  34             0
## 6:  Karincakes    NA           22           17                 508             0
##    gender_male gender_non-binary gender_unknown
##          <num>              <num>          <num>
## 1:           0                  0              1
## 2:           0                  0              1
## 3:           0                  0              0
## 4:           0                  0              1
## 5:           0                  0              1
## 6:           0                  0              1
```

On obtient un tableau des utilisateurs avec une colonne pour l'âge (estimé à partir de l'année de naissance), plusieurs indicateurs sur le comportement de l'utilisateur (nombre d'animes mis en pause, abandonnés ou prévus), ainsi que le genre. Ce dernier est traité par un encodage one-hot, c'est-à-dire qu'une colonne binaire est créée pour chaque valeur possible du genre (male, female, non_binary, unknown). Pour chaque utilisateur, la colonne correspondant à son genre vaut 1 et les autres valent 0.

person_anime_works.csv

```
p_anime <- fread("person_anime_works.csv")
names(p_anime) <- str_replace_all(names(p_anime), "\\.", "_")
p_anime <- p_anime %>%
  mutate(
    person_mal_id = as.character(person_mal_id),
    anime_mal_id  = as.integer(anime_mal_id)
  )

# Extraction des features liées au staff musical par anime
person_anime_features <- p_anime %>%
  mutate(
    position_clean = str_to_lower(position),
    is_op = str_detect(position_clean, "\\bop\\b|opening|theme song performance \\(op\\)|tv op"),
  is_ed = str_detect(position_clean, "\\bed\\b|ending|theme song performance \\(ed\\)"),
    is_insert  = str_detect(position_clean, "inserted song")
  ) %>%
  group_by(anime_mal_id) %>%
  summarise(
    num_music_people   = n_distinct(person_mal_id),
    num_music_entries  = n(),
    num_op_entries     = sum(is_op, na.rm = TRUE),
    num_ed_entries     = sum(is_ed, na.rm = TRUE),
    num_insert_entries = sum(is_insert, na.rm = TRUE),
    .groups = "drop"
  )
head(person_anime_features)
```

```
## # A tibble: 6 x 6
##   anime_mal_id num_music_people num_music_entries num_op_entries num_ed_entries
##          <int>            <int>              <int>          <int>          <int>
## 1            1              120                120              7              1
## 2            5              124                124              5              2
## 3            6               69                 69              2              0
## 4            7               66                 66             10              0
## 5            8               30                 30              0              0
## 6           15              266                266              3              0
## # i 1 more variable: num_insert_entries <int>
```

details.csv

```
details <- fread("details.csv")
names(details) <- str_replace_all(names(details), "\\.", "_")

details <- details %>% rename(anime_mal_id = mal_id)


details_base <- details %>%
  transmute(
    anime_mal_id,
    type, status, source, rating, season, year, episodes,
    score = to_num(score),
    scored_by = to_num(scored_by),
```

```
    rank = to_num(rank),
    members = to_num(members),
    favorites = to_num(favorites),
    genres, studios, themes, demographics, producers,
    explicit_genres, licensors, streaming
  )

head(details_base)
```

```
##    anime_mal_id    type          status   source                   rating
##           <int>  <char>          <char>   <char>                   <char>
## 1:        59356   Movie Finished Airing Original              G - All Ages
## 2:        56036   Music Finished Airing Original PG-13 - Teens 13 or older
## 3:         2928     OVA Finished Airing     Game PG-13 - Teens 13 or older
## 4:         3269   Movie Finished Airing     Game PG-13 - Teens 13 or older
## 5:         4469 Special Finished Airing     Game PG-13 - Teens 13 or older
## 6:          454     OVA Finished Airing Original          R+ - Mild Nudity
##    season  year episodes score scored_by  rank members favorites
##    <char> <num>    <num> <num>     <num> <num>   <num>     <num>
## 1:           NA        1    NA        NA 17086     195         0
## 2:           NA        1  6.53       503    NA     941         2
## 3:           NA        1  6.65      9745  6366   22525        31
## 4:           NA        1  7.06     15373  4194   34264       104
## 5:           NA        1  6.35      4317  8182   11135        10
## 6:           NA        1  6.09     10021  9692   21458        20
##                               genres           studios                  themes
##                               <char>            <char>                  <char>
## 1:                        ['Comedy']                []                      []
## 2:        ['Horror', 'Supernatural']   ['Flat Studio']               ['Music']
## 3: ['Adventure', 'Drama', 'Fantasy']     ['Bee Train']          ['Video Game']
## 4:             ['Action', 'Fantasy'] ['CyberConnect2']          ['Video Game']
## 5:   ['Comedy', 'Fantasy', 'Sci-Fi']                [] ['Parody', 'Video Game']
## 6:             ['Comedy', 'Fantasy']     ['Bee Train']          ['Video Game']
##    demographics                          producers explicit_genres
##          <char>                             <char>          <char>
## 1:           []        ['Nagoya Zokei University']              []
## 2:           []                                 []              []
## 3:           [] ['Bandai Visual', 'CyberConnect2']              []
## 4:           []                  ['Bandai Visual']              []
## 5:           []                  ['Bandai Visual']              []
## 6:           []                  ['CyberConnect2']              []
##                                licensors streaming
##                                   <char>    <char>
## 1:                                    []        []
## 2:                                    []        []
## 3:                                    []        []
## 4: ['Funimation', 'Bandai Entertainment']        []
## 5:                                    []        []
## 6:              ['Bandai Entertainment']        []
```

```
details_numeric <- details_base %>%
  mutate(
    episodes = to_num(episodes),
    year = to_num(year),
```

```
  ) %>%
  select(anime_mal_id, episodes, year)

head(details_numeric)
```

```
##    anime_mal_id episodes  year
##           <int>    <num> <num>
## 1:        59356        1    NA
## 2:        56036        1    NA
## 3:         2928        1    NA
## 4:         3269        1    NA
## 5:         4469        1    NA
## 6:          454        1    NA
```

```
details_cat <- details_base %>%
  select(anime_mal_id, type, status, source, rating, season) %>%
  onehot_topk("type", "type", top_k = 10) %>%
  onehot_topk("status", "status", top_k = NULL) %>%
  onehot_topk("source", "source", top_k = 15) %>%
  onehot_topk("rating", "rating", top_k = NULL) %>%
  onehot_topk("season", "season", top_k = NULL)

head(details_cat)
```

```
##    anime_mal_id type_ type_CM type_Movie type_Music type_ONA type_OVA type_PV
##           <int> <num>   <num>      <num>      <num>    <num>    <num>   <num>
## 1:        59356     0       0          1          0        0        0       0
## 2:        56036     0       0          0          1        0        0       0
## 3:         2928     0       0          0          0        0        1       0
## 4:         3269     0       0          1          0        0        0       0
## 5:         4469     0       0          0          0        0        0       0
## 6:          454     0       0          0          0        0        1       0
##    type_Special type_TV type_TV Special status_Currently Airing
##           <num>   <num>           <num>                    <num>
## 1:            0       0               0                        0
## 2:            0       0               0                        0
## 3:            0       0               0                        0
## 4:            0       0               0                        0
## 5:            1       0               0                        0
## 6:            0       0               0                        0
##    status_Finished Airing status_Not yet aired source_4-koma manga source_Book
##                     <num>                <num>               <num>       <num>
## 1:                      1                    0                   0           0
## 2:                      1                    0                   0           0
## 3:                      1                    0                   0           0
## 4:                      1                    0                   0           0
## 5:                      1                    0                   0           0
## 6:                      1                    0                   0           0
##    source_Game source_Light novel source_Manga source_Mixed media source_Music
##          <num>              <num>        <num>              <num>        <num>
## 1:           0                  0            0                  0            0
## 2:           0                  0            0                  0            0
## 3:           1                  0            0                  0            0
## 4:           1                  0            0                  0            0
```

```
## 5:           1                  0           0                  0           0
## 6:           0                  0           0                  0           0
##    source_Novel source_Original source_Other source_Picture book source_Unknown
##           <num>           <num>        <num>                <num>          <num>
## 1:            0               1            0                    0              0
## 2:            0               1            0                    0              0
## 3:            0               0            0                    0              0
## 4:            0               0            0                    0              0
## 5:            0               0            0                    0              0
## 6:            0               1            0                    0              0
##    source_Visual novel source_Web manga source_Web novel rating_
##                  <num>            <num>            <num>   <num>
## 1:                   0                0                0       0
## 2:                   0                0                0       0
## 3:                   0                0                0       0
## 4:                   0                0                0       0
## 5:                   0                0                0       0
## 6:                   0                0                0       0
##    rating_G - All Ages rating_PG - Children rating_PG-13 - Teens 13 or older
##                  <num>                <num>                            <num>
## 1:                   1                    0                                0
## 2:                   0                    0                                1
## 3:                   0                    0                                1
## 4:                   0                    0                                1
## 5:                   0                    0                                1
## 6:                   0                    0                                0
##    rating_R - 17+ (violence & profanity) rating_R+ - Mild Nudity
##                                    <num>                  <num>
## 1:                                     0                      0
## 2:                                     0                      0
## 3:                                     0                      0
## 4:                                     0                      0
## 5:                                     0                      0
## 6:                                     0                      1
##    rating_Rx - Hentai season_ season_fall season_spring season_summer
##                 <num>   <num>       <num>         <num>         <num>
## 1:                  0       1           0             0             0
## 2:                  0       1           0             0             0
## 3:                  0       1           0             0             0
## 4:                  0       1           0             0             0
## 5:                  0       1           0             0             0
## 6:                  0       1           0             0             0
##    season_winter
##            <num>
## 1:             0
## 2:             0
## 3:             0
## 4:             0
## 5:             0
## 6:             0
```

```r
genres_mh <- multihot_topk(details_base, "anime_mal_id", "genres", "genre", top_k = 30)
themes_mh <- multihot_topk(details_base, "anime_mal_id", "themes", "theme", top_k = 25)
demo_mh   <- multihot_topk(details_base, "anime_mal_id", "demographics", "demo", top_k = NULL)
```

```r
stud_mh   <- multihot_topk(details_base, "anime_mal_id", "studios", "studio", top_k = 30)
prod_mh   <- multihot_topk(details_base, "anime_mal_id", "producers", "producer", top_k = 30)
lic_mh    <- multihot_topk(details_base, "anime_mal_id", "licensors", "licensor", top_k = 20)
stream_mh <- multihot_topk(details_base, "anime_mal_id", "streaming", "stream", top_k = 20)

details_features <- details_numeric %>%
  left_join(details_cat, by = "anime_mal_id") %>%
  left_join(genres_mh, by = "anime_mal_id") %>%
  left_join(themes_mh, by = "anime_mal_id") %>%
  left_join(demo_mh, by = "anime_mal_id") %>%
  left_join(stud_mh, by = "anime_mal_id") %>%
  left_join(prod_mh, by = "anime_mal_id") %>%
  left_join(lic_mh, by = "anime_mal_id") %>%
  left_join(stream_mh, by = "anime_mal_id")

head(details_features)
```

```
##    anime_mal_id episodes  year type_ type_CM type_Movie type_Music type_ONA
##           <int>    <num> <num> <num>   <num>      <num>      <num>    <num>
## 1:        59356        1    NA     0       0          1          0        0
## 2:        56036        1    NA     0       0          0          1        0
## 3:         2928        1    NA     0       0          0          0        0
## 4:         3269        1    NA     0       0          1          0        0
## 5:         4469        1    NA     0       0          0          0        0
## 6:          454        1    NA     0       0          0          0        0
##    type_OVA type_PV type_Special type_TV type_TV Special
##       <num>   <num>        <num>   <num>           <num>
## 1:        0       0            0       0               0
## 2:        0       0            0       0               0
## 3:        1       0            0       0               0
## 4:        0       0            0       0               0
## 5:        0       0            1       0               0
## 6:        1       0            0       0               0
##    status_Currently Airing status_Finished Airing status_Not yet aired
##                       <num>                  <num>                <num>
## 1:                       0                      1                    0
## 2:                       0                      1                    0
## 3:                       0                      1                    0
## 4:                       0                      1                    0
## 5:                       0                      1                    0
## 6:                       0                      1                    0
##    source_4-koma manga source_Book source_Game source_Light novel source_Manga
##                  <num>       <num>       <num>              <num>        <num>
## 1:                   0           0           0                  0            0
## 2:                   0           0           0                  0            0
## 3:                   0           0           1                  0            0
## 4:                   0           0           1                  0            0
## 5:                   0           0           1                  0            0
## 6:                   0           0           0                  0            0
##    source_Mixed media source_Music source_Novel source_Original source_Other
##                  <num>        <num>        <num>           <num>        <num>
## 1:                   0            0            0               1            0
## 2:                   0            0            0               1            0
## 3:                   0            0            0               0            0
```

```
## 4:                  0           0           0              0           0
## 5:                  0           0           0              0           0
## 6:                  0           0           0              1           0
##    source_Picture book source_Unknown source_Visual novel source_Web manga
##                  <num>          <num>               <num>            <num>
## 1:                   0              0                   0                0
## 2:                   0              0                   0                0
## 3:                   0              0                   0                0
## 4:                   0              0                   0                0
## 5:                   0              0                   0                0
## 6:                   0              0                   0                0
##    source_Web novel rating_ rating_G - All Ages rating_PG - Children
##               <num>   <num>               <num>                <num>
## 1:                0       0                   1                    0
## 2:                0       0                   0                    0
## 3:                0       0                   0                    0
## 4:                0       0                   0                    0
## 5:                0       0                   0                    0
## 6:                0       0                   0                    0
##    rating_PG-13 - Teens 13 or older rating_R - 17+ (violence & profanity)
##                               <num>                                 <num>
## 1:                                0                                     0
## 2:                                1                                     0
## 3:                                1                                     0
## 4:                                1                                     0
## 5:                                1                                     0
## 6:                                0                                     0
##    rating_R+ - Mild Nudity rating_Rx - Hentai season_ season_fall season_spring
##                      <num>              <num>   <num>       <num>         <num>
## 1:                       0                  0       1           0             0
## 2:                       0                  0       1           0             0
## 3:                       0                  0       1           0             0
## 4:                       0                  0       1           0             0
## 5:                       0                  0       1           0             0
## 6:                       1                  0       1           0             0
##    season_summer season_winter genre_Comedy genre_Fantasy genre_Action
##            <num>         <num>        <int>         <int>        <int>
## 1:             0             0            1             0            0
## 2:             0             0            0             0            0
## 3:             0             0            0             1            0
## 4:             0             0            0             1            1
## 5:             0             0            1             1            0
## 6:             0             0            1             1            0
##    genre_Adventure genre_Sci.Fi genre_Drama genre_Romance genre_Slice.of.Life
##              <int>        <int>       <int>         <int>               <int>
## 1:               0            0           0             0                   0
## 2:               0            0           0             0                   0
## 3:               1            0           1             0                   0
## 4:               0            0           0             0                   0
## 5:               0            1           0             0                   0
## 6:               0            0           0             0                   0
##    genre_Hentai genre_Supernatural genre_Avant.Garde genre_Mystery genre_Ecchi
##           <int>              <int>             <int>        <int>       <int>
## 1:            0                  0                 0            0           0
```

```
## 2:            0                 1                0            0           0
## 3:            0                 0                0            0           0
## 4:            0                 0                0            0           0
## 5:            0                 0                0            0           0
## 6:            0                 0                0            0           0
##    genre_Sports genre_Horror genre_Suspense genre_Award.Winning genre_Gourmet
##           <int>        <int>          <int>               <int>         <int>
## 1:            0            0              0                   0             0
## 2:            0            1              0                   0             0
## 3:            0            0              0                   0             0
## 4:            0            0              0                   0             0
## 5:            0            0              0                   0             0
## 6:            0            0              0                   0             0
##    genre_Boys.Love genre_Girls.Love genre_Erotica theme_Music theme_School
##              <int>            <int>         <int>       <int>        <int>
## 1:               0                0             0           0            0
## 2:               0                0             0           1            0
## 3:               0                0             0           0            0
## 4:               0                0             0           0            0
## 5:               0                0             0           0            0
## 6:               0                0             0           0            0
##    theme_Historical theme_Mecha theme_Anthropomorphic theme_Parody
##               <int>       <int>                <int>        <int>
## 1:                0           0                    0            0
## 2:                0           0                    0            0
## 3:                0           0                    0            0
## 4:                0           0                    0            0
## 5:                0           0                    0            1
## 6:                0           0                    0            0
##    theme_Adult.Cast theme_Military theme_Super.Power theme_Martial.Arts
##               <int>          <int>             <int>              <int>
## 1:                0              0                 0                  0
## 2:                0              0                 0                  0
## 3:                0              0                 0                  0
## 4:                0              0                 0                  0
## 5:                0              0                 0                  0
## 6:                0              0                 0                  0
##    theme_Space theme_Mythology theme_Harem theme_Isekai theme_Psychological
##          <int>           <int>       <int>        <int>               <int>
## 1:           0               0           0            0                   0
## 2:           0               0           0            0                   0
## 3:           0               0           0            0                   0
## 4:           0               0           0            0                   0
## 5:           0               0           0            0                   0
## 6:           0               0           0            0                   0
##    theme_Idols..Female. theme_Mahou.Shoujo theme_Strategy.Game
##                   <int>              <int>               <int>
## 1:                    0                  0                   0
## 2:                    0                  0                   0
## 3:                    0                  0                   0
## 4:                    0                  0                   0
## 5:                    0                  0                   0
## 6:                    0                  0                   0
##    theme_Educational theme_Team.Sports theme_Detective theme_Gag.Humor
```

```
##                <int>             <int>           <int>           <int>
## 1:                 0                 0               0               0
## 2:                 0                 0               0               0
## 3:                 0                 0               0               0
## 4:                 0                 0               0               0
## 5:                 0                 0               0               0
## 6:                 0                 0               0               0
##    theme_Racing theme_Gore theme_CGDCT demo_Kids demo_Shounen demo_Seinen
##           <int>      <int>       <int>     <int>        <int>       <int>
## 1:            0          0           0         0            0           0
## 2:            0          0           0         0            0           0
## 3:            0          0           0         0            0           0
## 4:            0          0           0         0            0           0
## 5:            0          0           0         0            0           0
## 6:            0          0           0         0            0           0
##    demo_Shoujo demo_Josei studio_Toei.Animation studio_Sunrise studio_J.C.Staff
##          <int>      <int>                 <int>          <int>            <int>
## 1:           0          0                     0              0                0
## 2:           0          0                     0              0                0
## 3:           0          0                     0              0                0
## 4:           0          0                     0              0                0
## 5:           0          0                     0              0                0
## 6:           0          0                     0              0                0
##    studio_TMS.Entertainment studio_Madhouse studio_Production.I.G
##                       <int>           <int>                <int>
## 1:                        0               0                    0
## 2:                        0               0                    0
## 3:                        0               0                    0
## 4:                        0               0                    0
## 5:                        0               0                    0
## 6:                        0               0                    0
##    studio_Shanghai.Animation.Film.Studio studio_Studio.Deen studio_OLM
##                                    <int>              <int>      <int>
## 1:                                     0                  0          0
## 2:                                     0                  0          0
## 3:                                     0                  0          0
## 4:                                     0                  0          0
## 5:                                     0                  0          0
## 6:                                     0                  0          0
##    studio_Studio.Pierrot studio_Shin.Ei.Animation studio_A.1.Pictures
##                    <int>                   <int>               <int>
## 1:                     0                       0                   0
## 2:                     0                       0                   0
## 3:                     0                       0                   0
## 4:                     0                       0                   0
## 5:                     0                       0                   0
## 6:                     0                       0                   0
##    studio_DLE studio_Nippon.Animation studio_AIC studio_Tatsunoko.Production
##         <int>                   <int>      <int>                       <int>
## 1:          0                       0          0                           0
## 2:          0                       0          0                           0
## 3:          0                       0          0                           0
## 4:          0                       0          0                           0
## 5:          0                       0          0                           0
```

```
## 6:          0                       0         0                          0
##    studio_Bones studio_Xebec studio_Shaft studio_T.Rex studio_Gonzo
##           <int>        <int>        <int>        <int>        <int>
## 1:            0            0            0            0            0
## 2:            0            0            0            0            0
## 3:            0            0            0            0            0
## 4:            0            0            0            0            0
## 5:            0            0            0            0            0
## 6:            0            0            0            0            0
##    studio_Kyoto.Animation studio_SILVER.LINK. studio_LIDENFILMS
##                     <int>               <int>             <int>
## 1:                      0                   0                 0
## 2:                      0                   0                 0
## 3:                      0                   0                 0
## 4:                      0                   0                 0
## 5:                      0                   0                 0
## 6:                      0                   0                 0
##    studio_AQUA.ARIS studio_Satelight studio_X..Brains.Base.. studio_Arms
##               <int>            <int>                   <int>       <int>
## 1:                0                0                       0           0
## 2:                0                0                       0           0
## 3:                0                0                       0           0
## 4:                0                0                       0           0
## 5:                0                0                       0           0
## 6:                0                0                       0           0
##    studio_MAPPA studio_Seven producer_NHK producer_TV.Tokyo producer_Aniplex
##           <int>        <int>        <int>             <int>            <int>
## 1:            0            0            0                 0                0
## 2:            0            0            0                 0                0
## 3:            0            0            0                 0                0
## 4:            0            0            0                 0                0
## 5:            0            0            0                 0                0
## 6:            0            0            0                 0                0
##    producer_Lantis producer_Movic producer_Kadokawa producer_AT.X
##              <int>          <int>             <int>         <int>
## 1:               0              0                 0             0
## 2:               0              0                 0             0
## 3:               0              0                 0             0
## 4:               0              0                 0             0
## 5:               0              0                 0             0
## 6:               0              0                 0             0
##    producer_Pony.Canyon producer_Bandai.Visual producer_Fuji.TV producer_Dentsu
##                   <int>                 <int>            <int>           <int>
## 1:                    0                     0                0               0
## 2:                    0                     0                0               0
## 3:                    0                     1                0               0
## 4:                    0                     1                0               0
## 5:                    0                     1                0               0
## 6:                    0                     0                0               0
##    producer_KlockWorx producer_Kodansha producer_Shueisha producer_Sotsu
##                 <int>             <int>             <int>          <int>
## 1:                  0                 0                 0              0
## 2:                  0                 0                 0              0
## 3:                  0                 0                 0              0
```

```
## 4:                  0                0                0             0
## 5:                  0                0                0             0
## 6:                  0                0                0             0
##    producer_Mainichi.Broadcasting.System producer_Pink.Pineapple
##                                    <int>                   <int>
## 1:                                     0                       0
## 2:                                     0                       0
## 3:                                     0                       0
## 4:                                     0                       0
## 5:                                     0                       0
## 6:                                     0                       0
##    producer_bilibili producer_TBS producer_Tokyo.MX producer_DAX.Production
##                <int>        <int>             <int>                   <int>
## 1:                 0            0                 0                       0
## 2:                 0            0                 0                       0
## 3:                 0            0                 0                       0
## 4:                 0            0                 0                       0
## 5:                 0            0                 0                       0
## 6:                 0            0                 0                       0
##    producer_Tencent.Video producer_BS11 producer_Sanrio producer_TV.Asahi
##                     <int>         <int>           <int>             <int>
## 1:                      0             0               0                 0
## 2:                      0             0               0                 0
## 3:                      0             0               0                 0
## 4:                      0             0               0                 0
## 5:                      0             0               0                 0
## 6:                      0             0               0                 0
##    producer_Kadokawa.Shoten producer_Nippon.Television.Network producer_Genco
##                       <int>                             <int>          <int>
## 1:                        0                                 0              0
## 2:                        0                                 0              0
## 3:                        0                                 0              0
## 4:                        0                                 0              0
## 5:                        0                                 0              0
## 6:                        0                                 0              0
##    producer_Magic.Capsule producer_Crunchyroll licensor_Funimation
##                     <int>                <int>               <int>
## 1:                      0                    0                   0
## 2:                      0                    0                   0
## 3:                      0                    0                   0
## 4:                      0                    0                   1
## 5:                      0                    0                   0
## 6:                      0                    0                   0
##    licensor_Sentai.Filmworks licensor_Discotek.Media licensor_ADV.Films
##                        <int>                   <int>              <int>
## 1:                         0                       0                  0
## 2:                         0                       0                  0
## 3:                         0                       0                  0
## 4:                         0                       0                  0
## 5:                         0                       0                  0
## 6:                         0                       0                  0
##    licensor_Media.Blasters licensor_Aniplex.of.America licensor_VIZ.Media
##                      <int>                       <int>              <int>
## 1:                       0                           0                  0
```

```
## 2:                        0                           0                 0
## 3:                        0                           0                 0
## 4:                        0                           0                 0
## 5:                        0                           0                 0
## 6:                        0                           0                 0
##    licensor_Bandai.Entertainment licensor_Geneon.Entertainment.USA
##                            <int>                             <int>
## 1:                             0                                 0
## 2:                             0                                 0
## 3:                             0                                 0
## 4:                             1                                 0
## 5:                             0                                 0
## 6:                             1                                 0
##    licensor_Crunchyroll licensor_Nozomi.Entertainment
##                   <int>                         <int>
## 1:                    0                             0
## 2:                    0                             0
## 3:                    0                             0
## 4:                    0                             0
## 5:                    0                             0
## 6:                    0                             0
##    licensor_Central.Park.Media licensor_Kitty.Media licensor_GKIDS
##                          <int>                <int>          <int>
## 1:                           0                    0              0
## 2:                           0                    0              0
## 3:                           0                    0              0
## 4:                           0                    0              0
## 5:                           0                    0              0
## 6:                           0                    0              0
##    licensor_Critical.Mass.Video licensor_NuTech.Digital licensor_Maiden.Japan
##                           <int>                  <int>                 <int>
## 1:                            0                      0                     0
## 2:                            0                      0                     0
## 3:                            0                      0                     0
## 4:                            0                      0                     0
## 5:                            0                      0                     0
## 6:                            0                      0                     0
##    licensor_Manga.Entertainment licensor_NYAV.Post licensor_NIS.America
##                           <int>              <int>                <int>
## 1:                            0                  0                    0
## 2:                            0                  0                    0
## 3:                            0                  0                    0
## 4:                            0                  0                    0
## 5:                            0                  0                    0
## 6:                            0                  0                    0
##    stream_Crunchyroll stream_Netflix stream_HIDIVE stream_Bahamut.Anime.Crazy
##                 <int>          <int>         <int>                      <int>
## 1:                  0              0             0                          0
## 2:                  0              0             0                          0
## 3:                  0              0             0                          0
## 4:                  0              0             0                          0
## 5:                  0              0             0                          0
## 6:                  0              0             0                          0
##    stream_Bilibili.Global stream_Aniplus.TV stream_Laftel stream_Ani.One.Asia
```

```
##                     <int>             <int>         <int>                <int>
## 1:                      0                 0             0                    0
## 2:                      0                 0             0                    0
## 3:                      0                 0             0                    0
## 4:                      0                 0             0                    0
## 5:                      0                 0             0                    0
## 6:                      0                 0             0                    0
##    stream_Bilibili stream_Muse.Asia stream_Anime.Digital.Network
##              <int>            <int>                        <int>
## 1:               0                0                            0
## 2:               0                0                            0
## 3:               0                0                            0
## 4:               0                0                            0
## 5:               0                0                            0
## 6:               0                0                            0
##    stream_CatchPlay stream_MeWatch stream_iQIYI stream_Aniplus.Asia
##               <int>          <int>        <int>                <int>
## 1:                0              0            0                    0
## 2:                0              0            0                    0
## 3:                0              0            0                    0
## 4:                0              0            0                    0
## 5:                0              0            0                    0
## 6:                0              0            0                    0
##    stream_Shahid stream_Sushiroll stream_Hulu stream_Animax.Korea
##            <int>            <int>       <int>                <int>
## 1:             0                0           0                    0
## 2:             0                0           0                    0
## 3:             0                0           0                    0
## 4:             0                0           0                    0
## 5:             0                0           0                    0
## 6:             0                0           0                    0
##    stream_Animekey
##              <int>
## 1:               0
## 2:               0
## 3:               0
## 4:               0
## 5:               0
## 6:               0
```

```
anime_targets <- details_base %>%
  select(anime_mal_id, score) %>%
  rename(
    Y_score_global = score
  )

head(anime_targets)
```

```
##    anime_mal_id Y_score_global
##           <int>          <num>
## 1:        59356             NA
## 2:        56036           6.53
## 3:         2928           6.65
## 4:         3269           7.06
## 5:         4469           6.35
```

```
## 6:          454           6.09
```

characters.csv

```
characters <- fread("characters.csv")
names(characters) <- str_replace_all(names(characters), "\\.", "_")

characters <- characters %>%
  mutate(character_mal_id = as.character(character_mal_id))

char_dict <- characters %>%
  transmute(
    character_mal_id,
    character_favorites = to_num(favorites),
    about_length = ifelse(is.na(about), 0, nchar(about))
  ) %>%
  left_join(char_nick_feat, by = "character_mal_id") %>%
  distinct(character_mal_id, .keep_all = TRUE)

char_works <- char_works %>%
  mutate(character_mal_id = as.character(character_mal_id))

char_to_anime <- char_works %>%
  select(anime_mal_id, character_mal_id, role) %>%
  left_join(char_dict, by = "character_mal_id") %>%
  group_by(anime_mal_id) %>%
  summarise(
    sum_character_favorites = sum(character_favorites, na.rm = TRUE),
    avg_character_favorites = mean(character_favorites, na.rm = TRUE),
    max_character_favorites = ifelse(
      all(is.na(character_favorites)),
      NA_real_,
      max(character_favorites, na.rm = TRUE)
    ),
    avg_nicknames_per_character = mean(num_nicknames, na.rm = TRUE),
    .groups = "drop"
  )
head(char_to_anime)
```

```
## # A tibble: 6 x 5
##   anime_mal_id sum_character_favorites avg_character_favorites
##          <int>                   <dbl>                   <dbl>
## 1            1                   69689                   553.
## 2            5                   68925                  1641.
## 3            6                   22655                   234.
## 4            7                     489                    9.23
## 5            8                      35                    3.5
## 6           15                    5936                   92.8
## # i 2 more variables: max_character_favorites <dbl>,
## #   avg_nicknames_per_character <dbl>
```

favs.csv

```
favs <- fread("favs.csv")
names(favs) <- str_replace_all(names(favs), "\\.", "_")

favs_user <- favs %>%
  group_by(username, fav_type) %>%
  summarise(n = n(), .groups = "drop") %>%
  pivot_wider(names_from = fav_type, values_from = n, values_fill = 0) %>%
  rename_with(~ paste0("nb_fav_", .x), -username)

head(favs_user)
```

```
## # A tibble: 6 x 5
##   username        nb_fav_anime nb_fav_character nb_fav_people nb_fav_company
##   <chr>                  <int>            <int>         <int>          <int>
## 1 ""                         4                0             0              0
## 2 "-----9-----"             10                0             0              0
## 3 "-----Ren-----"           10               10            10              0
## 4 "----Haku----"            10               10             4              0
## 5 "----phoebelyn"            5               10             5              0
## 6 "----shia"                 0                5             2              0
```

```
fav_anime <- favs %>%
  filter(fav_type == "anime") %>%
  rename(anime_mal_id = id) %>%
  mutate(anime_mal_id = as.integer(anime_mal_id))

fav_anime_genres <- fav_anime %>%
  left_join(genres_mh, by = "anime_mal_id") %>%
  group_by(username) %>%
  summarise(across(starts_with("genre_"), \(x) mean(x, na.rm = TRUE)),
            .groups="drop") %>%
  rename_with(~ paste0("pref_", .x), starts_with("genre_"))
head(fav_anime_genres)
```

```
## # A tibble: 6 x 22
##   username        pref_genre_Comedy pref_genre_Fantasy pref_genre_Action
##   <chr>                       <dbl>              <dbl>             <dbl>
## 1 ""                          0                  1                 1
## 2 "-----9-----"               0.5                0.3               0.5
## 3 "-----Ren-----"             0.222              0.222             0.444
## 4 "----Haku----"              0.111              0.111             0.111
## 5 "----phoebelyn"             0.4                0.2               0.2
## 6 "---123---"                 0.3                0.2               0.1
## # i 18 more variables: pref_genre_Adventure <dbl>, pref_genre_Sci.Fi <dbl>,
## #   pref_genre_Drama <dbl>, pref_genre_Romance <dbl>,
## #   pref_genre_Slice.of.Life <dbl>, pref_genre_Hentai <dbl>,
## #   pref_genre_Supernatural <dbl>, pref_genre_Avant.Garde <dbl>,
## #   pref_genre_Mystery <dbl>, pref_genre_Ecchi <dbl>, pref_genre_Sports <dbl>,
## #   pref_genre_Horror <dbl>, pref_genre_Suspense <dbl>,
## #   pref_genre_Award.Winning <dbl>, pref_genre_Gourmet <dbl>, ...
```

ratings.csv

```
ratings <- fread("ratings.csv")
names(ratings) <- str_replace_all(names(ratings), "\\.", "_")

ratings <- ratings %>%
  transmute(
    username,
    anime_id = to_num(anime_id),
    status = tolower(status),
    score = to_num(score),
    is_rewatching = to_num(is_rewatching),
    num_watched_episodes = to_num(num_watched_episodes)
  )

ratings_scored <- ratings %>%
  filter(!is.na(score) & score >= 1 & score <= 10) %>%
  mutate(
    status = ifelse(status %in% c("completed","watching","dropped","on_hold","plan_to_watch"),
                    status, "other"),
    status_completed     = as.integer(status == "completed"),
    status_watching      = as.integer(status == "watching"),
    status_dropped       = as.integer(status == "dropped"),
    status_on_hold       = as.integer(status == "on_hold"),
    status_plan_to_watch = as.integer(status == "plan_to_watch")
  ) %>%
  select(-status)

user_activity <- ratings_scored %>%
  group_by(username) %>%
  summarise(
    user_mean_score_from_ratings = mean(score, na.rm = TRUE),
    user_nb_rated = n(),
    user_nb_completed = sum(status_completed == 1, na.rm = TRUE),
    user_nb_watching = sum(status_watching == 1, na.rm = TRUE),
    user_rewatch_rate = mean(is_rewatching > 0, na.rm = TRUE),
    user_mean_watched_eps = mean(num_watched_episodes, na.rm = TRUE),
    .groups = "drop"
  )
head(user_activity)
```

```
## # A tibble: 6 x 7
##   username        user_mean_score_from_ratings user_nb_rated user_nb_completed
##   <chr>                                  <dbl>         <int>             <int>
## 1 ""                                      8                7                 2
## 2 "--------788"                           8.98            64                63
## 3 "-----9-----"                           6.89           593               590
## 4 "-----Ren-----"                         8.21           143               141
## 5 "----Blank----"                         7.08           158               146
## 6 "----Haku----"                          8.31           509               432
## # i 3 more variables: user_nb_watching <int>, user_rewatch_rate <dbl>,
## #   user_mean_watched_eps <dbl>
```

Construction du tableau anime-level

```
anime_level <- details_features %>%
  left_join(char_features,         by = "anime_mal_id") %>%
  left_join(char_to_anime,         by = "anime_mal_id") %>%
  left_join(voice_features,        by = "anime_mal_id") %>%
  left_join(person_anime_features, by = "anime_mal_id") %>%
  left_join(reco_features,         by = "anime_mal_id") %>%
  left_join(anime_targets,         by = "anime_mal_id")

anime_level_train <- anime_level %>%
  filter(!is.na(Y_score_global))

fwrite(anime_level_train, "anime_level.csv")
```

Construction du tableau user–anime

```
anime_reco_num <- details_base %>%
  transmute(
    anime_mal_id = as.integer(anime_mal_id),
    episodes     = to_num(episodes),
    year         = to_num(year)
  )

anime_reco_cat <- details_base %>%
  select(anime_mal_id, type, rating, status, season, source) %>%   # + source ici
  mutate(across(c(type, rating, status, season, source), as.character)) %>%  # + source ici aussi
  onehot_topk("type",   "type",   top_k = 10) %>%    # types les 10 plus fréquents
  onehot_topk("rating", "rating", top_k = NULL) %>%
  onehot_topk("status", "status", top_k = NULL) %>%
  onehot_topk("season", "season", top_k = NULL) %>%
  onehot_topk("source", "source", top_k = 15)        # 15 sources les plus fréquentes


genres_mh_small <- multihot_topk(details_base, "anime_mal_id", "genres", "genre", top_k = 30)

anime_reco <- anime_reco_num %>%
  left_join(anime_reco_cat,  by = "anime_mal_id") %>%
  left_join(genres_mh_small, by = "anime_mal_id") %>%
  rename(mal_id = anime_mal_id)


setDT(anime_reco)
anime_reco[, mal_id := as.integer(mal_id)]
setkey(anime_reco, mal_id)

setDT(ratings_scored)
# Échantillonnage pour limiter l'usage RAM (2 millions de lignes)
ratings_scored_sample <- ratings_scored[sample(.N, 2e6)]  # 2 million

ratings_keep <- ratings_scored_sample[, .(
  username,
```

```
  mal_id = as.integer(anime_id),
  score,
  is_rewatching,
  num_watched_episodes,
  status_completed, status_watching, status_dropped,
  status_on_hold, status_plan_to_watch
)]

user_anime_step1 <- anime_reco[ratings_keep, on = "mal_id"]

user_anime_step1 <- as_tibble(user_anime_step1)

user_anime_level <- user_anime_step1 %>%
  left_join(profiles_user, by = "username") %>%
  left_join(user_activity, by = "username") %>%
  left_join(favs_user, by = "username") %>%
  left_join(fav_anime_genres, by = "username") %>%
  rename(Y_user_score = score)

fwrite(user_anime_level, "user_anime_level.csv")
```

Prédiction de la note d'un animé

```r
# On charge les librairies nécessaires
suppressPackageStartupMessages({
    library(data.table)
    library(dplyr)
    library(glmnet)
    library(caret)

})
# On charge le dataset 'anime_level.csv'
df <- fread("anime_level.csv")

# On vérifie les dimensions
dim(df)
```

```
## [1] 18882   211
```

Traitement des données

```r
# On supprime les colonnes avec plus de 50% des valeurs manquantes
seuil_col <- 0.5 * nrow(df)
cols_to_keep <- colSums(is.na(df)) <= seuil_col
df_clean <- df[, ..cols_to_keep]
cat("Après suppression colonnes vides :", ncol(df_clean), "colonnes.\n")
```

```
## Après suppression colonnes vides : 208 colonnes.
```

```r
# On supprime les lignes avec plus de 50% des valeurs manquantes
row_na_percent <- rowMeans(is.na(df_clean))
df_clean <- df_clean[row_na_percent <= 0.5, ]
cat("Après suppression lignes vides :", nrow(df_clean), "lignes.\n")
```

```
## Après suppression lignes vides : 18882 lignes.
```

```r
# On supprime les lignes restantes contenant des NA .
df_final <- na.omit(df_clean)
```

Parititonnement des données pour la modélisation

```r
# On définit la variable cible que l'on cherche à prédire
target_col <- "Y_score_global"

# On liste les variables explicatives en excluant la cible et l'identifiant
features_cols <- setdiff(names(df_final), c(target_col, "anime_mal_id"))

# On fixe la graine aléatoire
set.seed(2025)

# On génère les indices pour sélectionner 80% des données de manière
# stratifiée.
train_index <- createDataPartition(df_final[[target_col]], p = 0.8, list = FALSE)

# On crée le sous-ensemble d'entraînement contenant 80% des lignes.
train_data <- df_final[train_index, ]

# On crée le sous-ensemble de test
test_data <- df_final[-train_index, ]

# On convertit les variables explicatives d'entraînement en matrice numérique
X_train <- as.matrix(train_data[, ..features_cols])

# On extrait le vecteur des notes cibles pour l'entraînement.
y_train <- train_data[[target_col]]

# On convertit les variables explicatives de test en matrice numérique.
X_test <- as.matrix(test_data[, ..features_cols])

# On extrait le vecteur des notes cibles pour l'évaluation
y_test <- test_data[[target_col]]
```

Modèle OLS

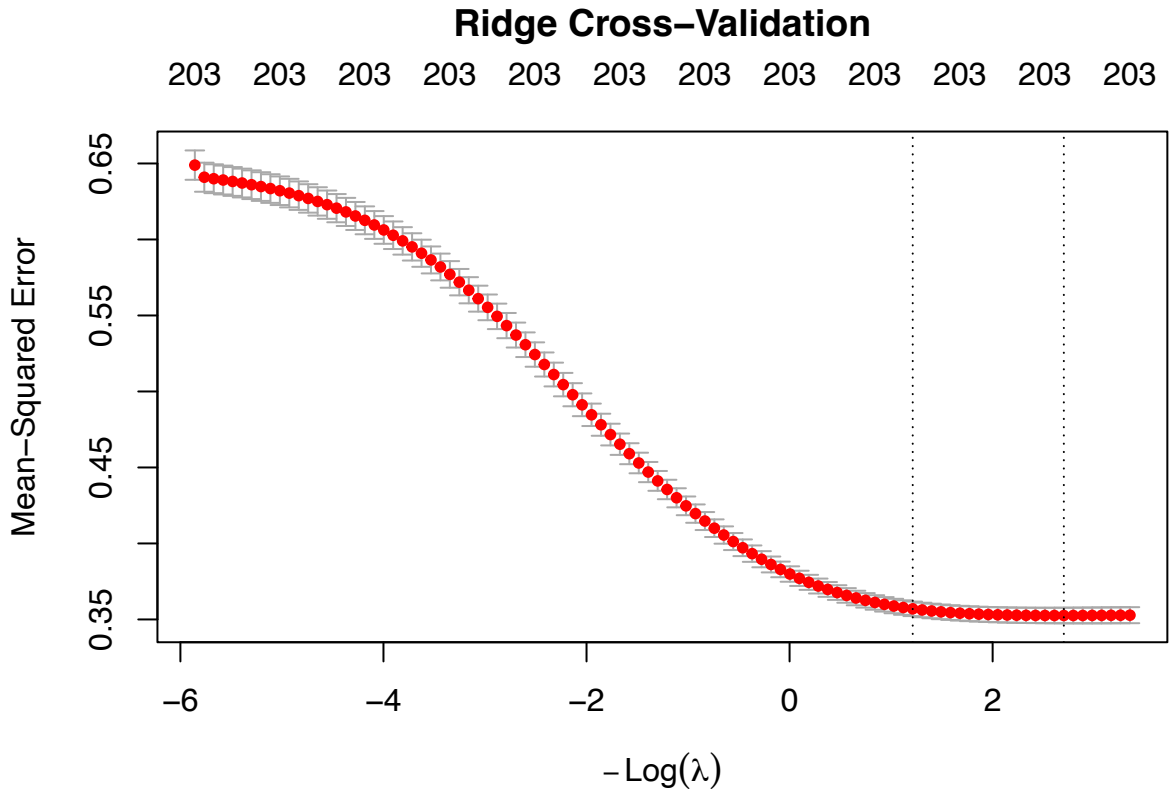
```r
# On prépare les données pour la fonction lm() qui attend un dataframe (et non
# une matrice).  On ne garde que la cible et les features (pas l'ID).
data_ols_train <- train_data[, c(target_col, features_cols), with = FALSE]

# On définit la formule
formule <- as.formula(paste(target_col, "~ ."))

# On entraîne le modèle linéaire
model_ols <- lm(formule, data = data_ols_train)

# On affiche un résumé rapide
cat("R2 ajusté (Train) :", summary(model_ols)$adj.r.squared, "\n")
```

```
## R2 ajusté (Train) : 0.4716315
```

Modèles intermédiaires

Ridge

```
# On lance la validation croisée pour identifier le paramètre de pénalité
# optimal lambda.
cv_ridge <- cv.glmnet(X_train, y_train, alpha = 0, family = "gaussian")

# On visualise l'évolution de l'erreur quadratique moyenne (MSE) en fonction de
# lambda pour repérer le minimum.
plot(cv_ridge)
title("Ridge Cross-Validation", line = 2.5)
```



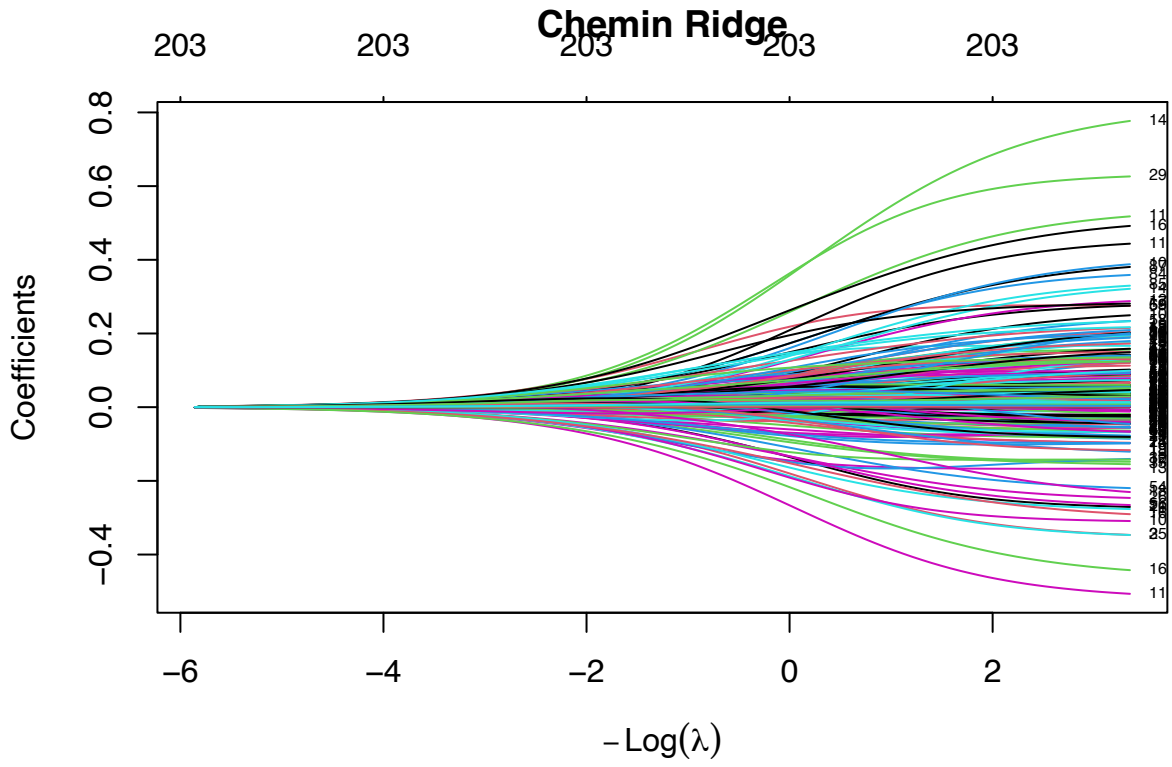

```
# On construit le modèle en utilisant le lambda qui a minimisé l'erreur
ridge_min <- glmnet(X_train, y_train, alpha = 0, lambda = cv_ridge$lambda.min)

# On construit le modèle en utilisant la règle du '1 Standard Error'
# (lambda.1se).
ridge_1se <- glmnet(X_train, y_train, alpha = 0, lambda = cv_ridge$lambda.1se)

# On affiche les valeurs des lambda sélectionnés
cat("Ridge Lambda min:", cv_ridge$lambda.min, "\n")
```

```
## Ridge Lambda min: 0.0671192
```

```r
cat("Ridge Lambda 1se:", cv_ridge$lambda.1se, "\n")
```

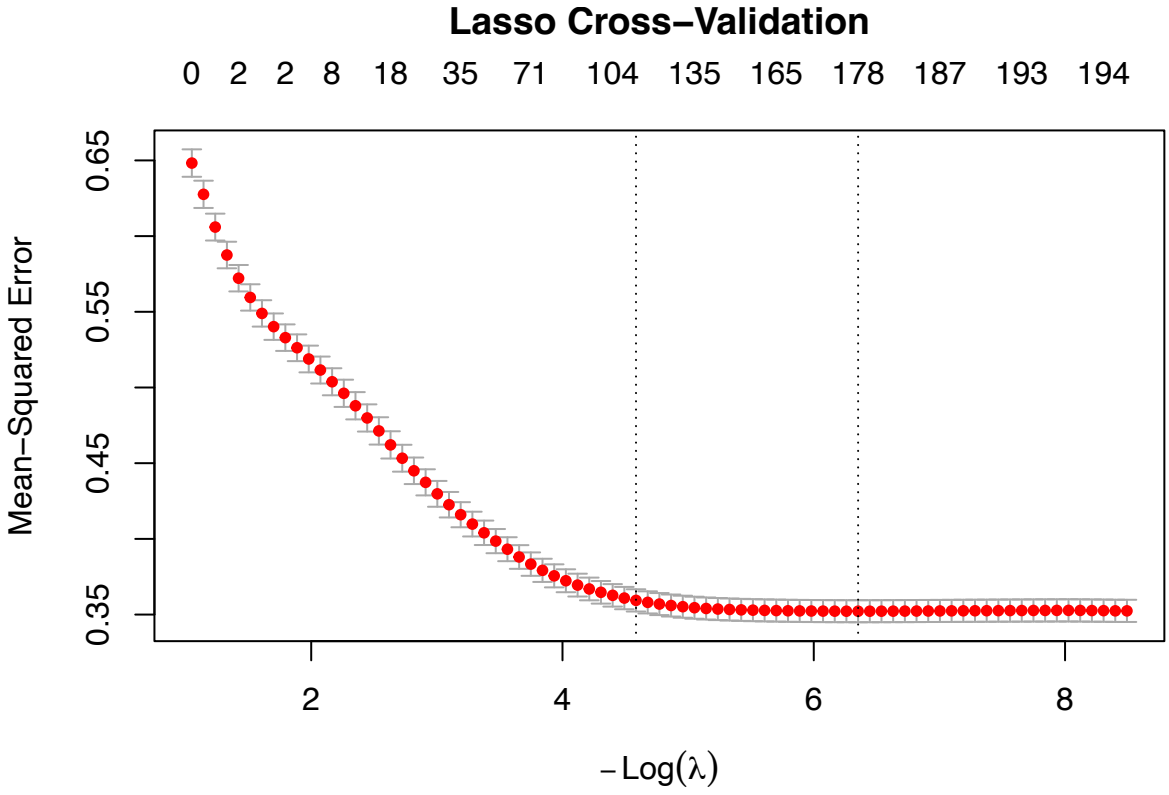
```
## Ridge Lambda 1se: 0.2973798
```

```r
# On trace le chemin de régularisation pour observer l'évolution des
# coefficients
fit_ridge_plot <- glmnet(X_train, y_train, alpha = 0)
plot(fit_ridge_plot, xvar = "lambda", label = TRUE, main = "Chemin Ridge")
```



Lasso

```r
# On lance la validation croisée pour identifier le paramètre de pénalité
# optimal lambda.
cv_lasso <- cv.glmnet(X_train, y_train, alpha = 1, family = "gaussian")

# On visualise l'évolution de l'erreur quadratique moyenne (MSE) en fonction de
# lambda pour repérer le minimum.
plot(cv_lasso)
title("Lasso Cross-Validation", line = 2.5)
```



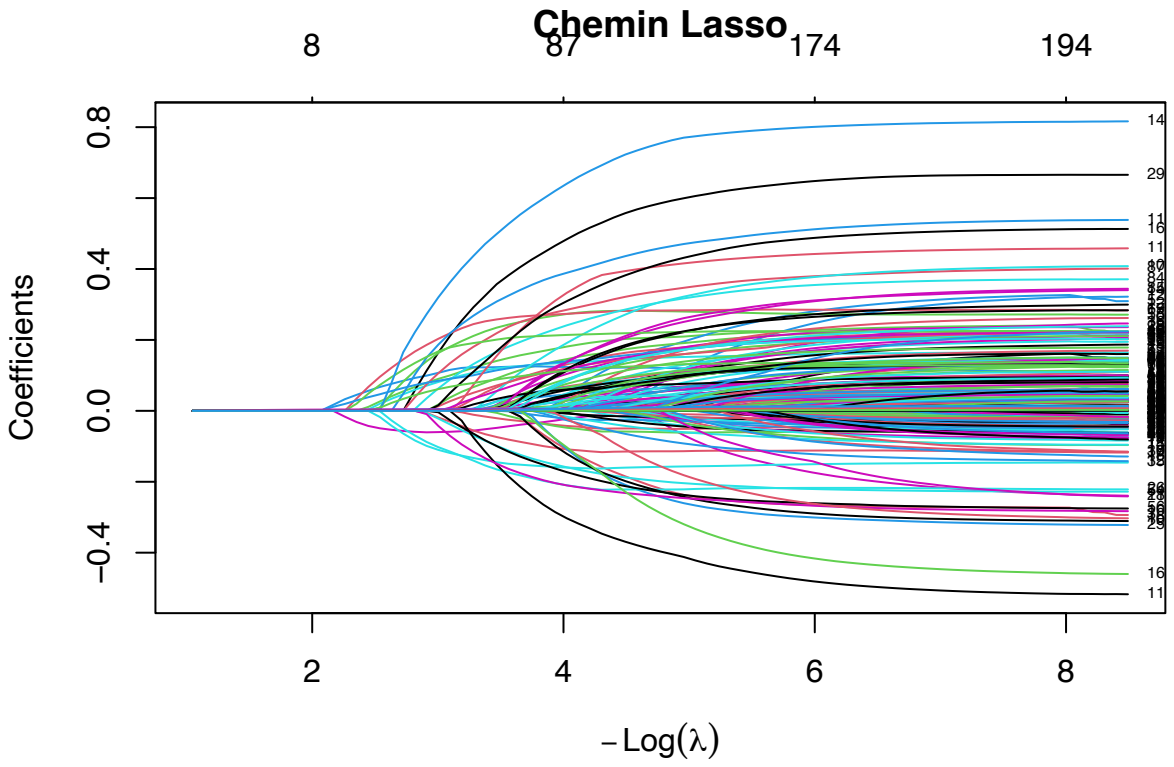

```r
# On construit le modèle en utilisant le lambda qui a minimisé l'erreur
lasso_min <- glmnet(X_train, y_train, alpha = 1, lambda = cv_lasso$lambda.min)

# On construit le modèle en utilisant la règle du '1 Standard Error'
# (lambda.1se).
lasso_1se <- glmnet(X_train, y_train, alpha = 1, lambda = cv_lasso$lambda.1se)

# On affiche les valeurs exactes des lambda sélectionnés
cat("Lasso Lambda min:", cv_lasso$lambda.min, "\n")
```

```
## Lasso Lambda min: 0.00174176
```

```r
cat("Lasso Lambda 1se:", cv_lasso$lambda.1se, "\n")
```

```
## Lasso Lambda 1se: 0.01020152
```

```r
# ON affiche les variables sélectionnées par le Lasso (non nulles)
coef_l <- coef(lasso_1se)

# On trace le chemin de régularisation pour observer l'évolution des
# coefficients
fit_lasso_plot <- glmnet(X_train, y_train, alpha = 1)
plot(fit_lasso_plot, xvar = "lambda", label = TRUE, main = "Chemin Lasso")
```



Elastic Net

```r
# On définit une grille de valeurs pour alpha (de 0.1 à 0.9) afin d'explorer
# les mélanges intermédiaires entre Ridge et Lasso.
list_alpha <- seq(0.1, 0.9, by = 0.1)

# On initialise un tableau pour enregistrer l'erreur minimale et le lambda
# associé à chaque configuration.
results_en <- data.frame(alpha = list_alpha, cv_error = NA, lambda = NA)

# On lance une boucle pour évaluer systématiquement la performance de chaque
# nuance de mélange
for (i in 1:length(list_alpha)) {
    a <- list_alpha[i]
    # On exécute la validation croisée pour l'alpha courant afin de mesurer sa
    # capacité prédictive.
    cv_temp <- cv.glmnet(X_train, y_train, alpha = a, family = "gaussian")

    # On mémorise l'erreur minimale obtenue pour pouvoir comparer cet alpha aux
    # autres.
    results_en$cv_error[i] <- min(cv_temp$cvm)
    results_en$lambda[i] <- cv_temp$lambda.min
}
```

```r
# On sélectionne l'alpha qui a offert la plus faible erreur moyenne sur
# l'ensemble des tests.
best_res <- results_en[which.min(results_en$cv_error), ]
best_alpha <- best_res$alpha

# On affiche l'alpha optimal retenu par l'algorithme .
cat("Meilleur Alpha Elastic Net trouvé :", best_alpha, "\n")
```

```
## Meilleur Alpha Elastic Net trouvé : 0.7
```

```r
# On relance une validation croisée finale fixée sur cet alpha optimal
cv_enet <- cv.glmnet(X_train, y_train, alpha = best_alpha, family = "gaussian")

# On génère le modèle combinant l'alpha optimal et la sélection de variables
# pour maximiser la parcimonie.
enet_1se <- glmnet(X_train, y_train, alpha = best_alpha, lambda = cv_enet$lambda.1se)

# On génère le modèle final avec les mêmes hyperparamètres optimisés.
enet_min <- glmnet(X_train, y_train, alpha = best_alpha, lambda = cv_enet$lambda.min)

# On affiche les valeurs des lambda
cat("Elastic net Lambda min:", cv_enet$lambda.min, "\n")
```

```
## Elastic net Lambda min: 0.002488228
```

```r
cat("Elastic net Lambda 1se:", cv_enet$lambda.1se, "\n")
```

```
## Elastic net Lambda 1se: 0.01755396
```

Comparaison et Evaluation

```r
# On définit une fonction pour calculer le RMSE et le R^2 afin de rendre les
# modèles comparables.
eval_model <- function(model, x_data, y_data, type = "glmnet") {
    if (type == "lm") {
        preds <- predict(model, newdata = as.data.frame(x_data))
    } else {
        preds <- predict(model, newx = x_data)
    }

    # Calculs métriques
    rmse <- sqrt(mean((y_data - preds)^2))
    sst <- sum((y_data - mean(y_data))^2)
    sse <- sum((y_data - preds)^2)
    r2 <- 1 - (sse/sst)

    return(c(RMSE = round(rmse, 4), R2 = round(r2, 4)))
}
```

```r
# On applique cette évaluation uniforme sur l'ensemble de nos 7 modèles
# candidats (Ridge, Lasso, Elastic Net) en utilisant le jeu de test.
res_ols <- eval_model(model_ols, X_test, y_test, type = "lm")
```

```
## Warning in predict.lm(model, newdata = as.data.frame(x_data)): prediction from
## a rank-deficient fit may be misleading
```

```r
res_ridge_min <- eval_model(ridge_min, X_test, y_test)
res_ridge_1se <- eval_model(ridge_1se, X_test, y_test)
res_lasso_min <- eval_model(lasso_min, X_test, y_test)
res_lasso_1se <- eval_model(lasso_1se, X_test, y_test)
res_enet_min <- eval_model(enet_min, X_test, y_test)
res_enet_1se <- eval_model(enet_1se, X_test, y_test)

# On rassemble l'ensemble des résultats dans un tableau récapitulatif unique
# pour faciliter l'analyse.
final_table <- rbind(`OLS (Standard)` = res_ols, `Ridge (min)` = res_ridge_min, `Ridge (1se)` = res_rid
    `Lasso (min)` = res_lasso_min, `Lasso (1se)` = res_lasso_1se, `Elastic Net (min)` = res_enet_min,
    `Elastic Net (1se)` = res_enet_1se)

# On affiche le classement final trié par RMSE croissant pour identifier
# objectivement le modèle offrant la meilleure précision prédictive.
print(final_table[order(final_table[, "RMSE"]), ])
```

```
##                     RMSE     R2
## Ridge (min)       0.6205 0.4629
## Lasso (min)       0.6205 0.4629
## Elastic Net (min) 0.6205 0.4629
## OLS (Standard)    0.6210 0.4621
## Ridge (1se)       0.6265 0.4525
## Lasso (1se)       0.6291 0.4480
## Elastic Net (1se) 0.6323 0.4423
```

Analyse de la corrélation entre les variables

```r
# On ne garde que les colonnes numériques pour le calcul
cols_num <- sapply(df_final, is.numeric)
df_corr_input <- df_final[, ..cols_num]

# On retire la cible et les identifiants
vars_a_exclure <- c("Y_score_global", "anime_mal_id", "mal_id", "rank")
# On identifie les variables à conserver en soustrayant cette liste d'exclusion
# des noms de colonnes disponibles.
vars_restantes <- setdiff(names(df_corr_input), vars_a_exclure)
# On réduit le dataframe en ne sélectionnant que ces colonnes pertinentes pour
# préparer le calcul matriciel.
df_corr_input <- df_corr_input[, ..vars_restantes]


# On calcule la matrice de corrélation .
cor_mat <- cor(df_corr_input, use = "pairwise.complete.obs")
```

```
## Warning in cor(df_corr_input, use = "pairwise.complete.obs"): the standard
## deviation is zero
```

```
# On neutralise la diagonale et le triangle inférieur de la matrice pour éviter
# les redondances et les auto-corrélations.
cor_mat[lower.tri(cor_mat, diag = TRUE)] <- NA
# On transforme la matrice carrée en liste linéaire de corrélations.
cor_list <- as.data.frame(as.table(cor_mat))
# On filtre cette liste en supprimant les lignes devenues vides (NA) pour ne
# conserver que les corrélations uniques.
cor_list <- na.omit(cor_list)

# On trie la liste par valeur absolue décroissante afin de faire remonter les
# paires de variables les plus fortement liées (positivement ou négativement).
cor_list_sorted <- cor_list[order(-abs(cor_list$Freq)), ]

# On affiche les 20 corrélations les plus fortes
cat("\n--- TOP 20 DES VARIABLES LES PLUS CORRÉLÉES ---\n")
```

```
##
## --- TOP 20 DES VARIABLES LES PLUS CORRÉLÉES ---
```

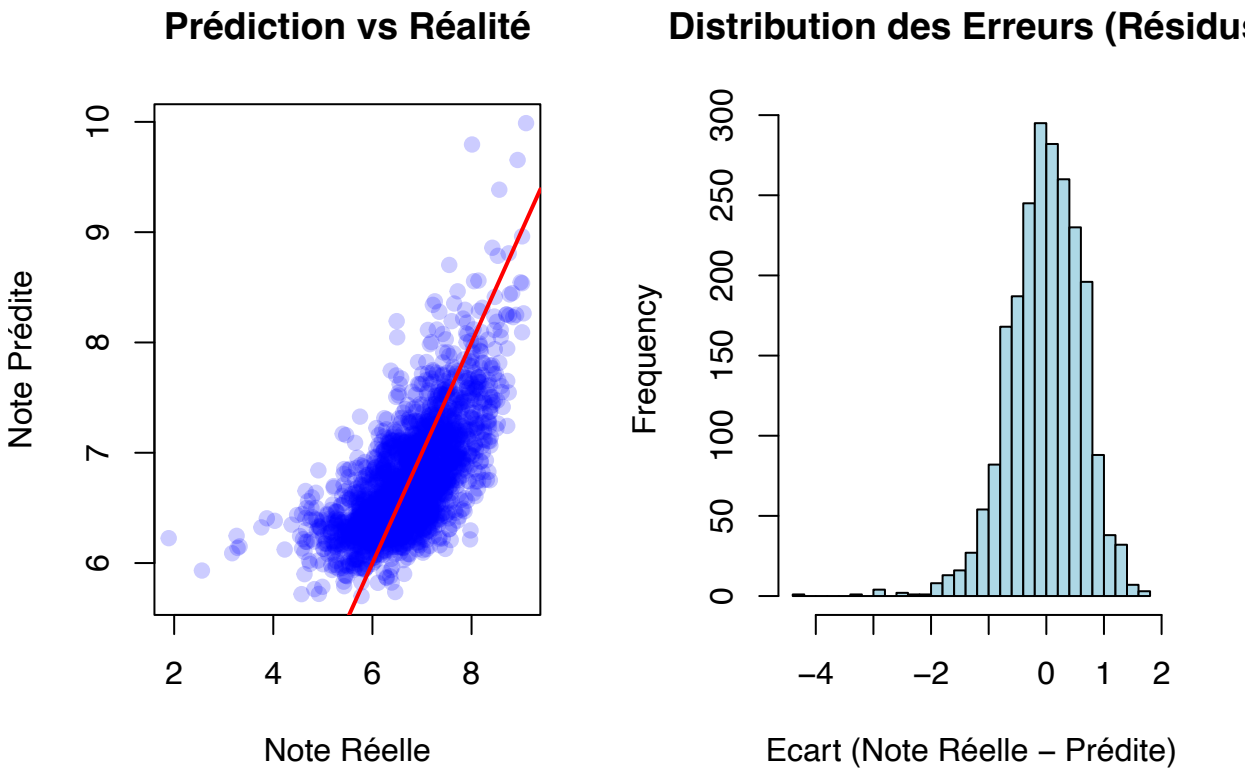
```
print(head(cor_list_sorted, 20))
```

```
##                            Var1                       Var2       Freq
## 2484    status_Currently Airing    status_Finished Airing -1.0000000
## 41814          num_music_people         num_music_entries  1.0000000
## 7426                      type_TV                   season_ -0.9992695
## 10130        rating_Rx - Hentai              genre_Hentai  0.9964617
## 40157           num_characters num_supporting_characters  0.9933705
## 41399            num_doubleurs     num_support_doubleurs  0.9905294
## 38088         stream_CatchPlay            stream_MeWatch  0.9432780
## 40778   sum_character_favorites   max_character_favorites  0.9389519
## 41395 num_supporting_characters     num_support_doubleurs  0.7807730
## 41393           num_characters     num_support_doubleurs  0.7721742
## 40981           num_characters             num_doubleurs  0.7618204
## 40983 num_supporting_characters             num_doubleurs  0.7614305
## 40779   avg_character_favorites   max_character_favorites  0.7183909
## 38913           stream_MeWatch          stream_Sushiroll  0.6922041
## 40572   sum_character_favorites   avg_character_favorites  0.6749554
## 35998 licensor_Sentai.Filmworks             stream_HIDIVE  0.6646772
## 16741             genre_Sports         theme_Team.Sports  0.6588682
## 41193            num_doubleurs        num_main_doubleurs  0.6540145
## 38912         stream_CatchPlay          stream_Sushiroll  0.6465250
## 15713           genre_Suspense       theme_Psychological  0.6378297
```

Analayse des résidus et diagnostic

```
# On génère les prédictions finales sur le jeu de test en utilisant le modèle
# Elastic Net optimisé.
```

```r
best_preds <- predict(cv_enet, newx = X_test)
# On calcule les résidus
residuals <- y_test - best_preds

# On configure la fenêtre graphique
par(mfrow = c(1, 2))
# On trace le nuage de points comparant réalité et prédiction pour visualiser
# la qualité de l'ajustement.
plot(y_test, best_preds, main = "Prédiction vs Réalité", xlab = "Note Réelle",
    ylab = "Note Prédite", pch = 19, col = rgb(0, 0, 1, 0.2))
# On ajoute la droite d'identité (y=x) servant de référence pour une prédiction
# parfaite.
abline(0, 1, col = "red", lwd = 2)

# On trace l'histogramme des erreurs pour vérifier leur centrage sur zéro et
# l'absence de biais systématique.
hist(residuals, main = "Distribution des Erreurs (Résidus)", xlab = "Ecart (Note Réelle - Prédite)",
    col = "lightblue", breaks = 30)
```



Comparaison détaillée : Réalité vs Prédiction

```r
# On construit un dataframe de synthèse regroupant la note réelle, la
# prédiction et l'écart pour chaque observation du test.
```

```r
df_comparaison <- data.frame(Note_Reelle = as.numeric(y_test), Note_Predite = round(as.numeric(best_pred
    2), Ecart = round(as.numeric(y_test - best_preds), 2))

# On affiche les 20 premières lignes .
cat("\n--- Tableau Comparatif : Note Réelle vs Prédite  ---\n")
```

```
##
## --- Tableau Comparatif : Note Réelle vs Prédite  ---
```

```r
print(head(df_comparaison, 20))
```

```
##    Note_Reelle Note_Predite Ecart
## 1         5.43         6.39 -0.96
## 2         6.26         6.94 -0.68
## 3         5.43         6.55 -1.12
## 4         5.74         6.49 -0.75
## 5         6.52         7.14 -0.62
## 6         6.74         6.76 -0.02
## 7         6.11         6.79 -0.68
## 8         8.90         8.25  0.65
## 9         4.68         6.19 -1.51
## 10        7.64         6.91  0.73
## 11        6.21         6.35 -0.14
## 12        6.86         7.09 -0.23
## 13        7.67         6.88  0.79
## 14        6.68         6.41  0.27
## 15        5.57         6.04 -0.47
## 16        6.13         6.17 -0.04
## 17        8.01         7.63  0.38
## 18        5.53         6.68 -1.15
## 19        6.81         6.57  0.24
## 20        6.36         6.45 -0.09
```

```r
# On calcule les statistiques descriptives des erreurs (Moyenne, Médiane,
# Quartiles) pour quantifier la dispersion globale du modèle.
cat("\n--- Statistiques des erreurs ---\n")
```

```
##
## --- Statistiques des erreurs ---
```

```r
print(summary(df_comparaison$Ecart))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4.34000 -0.41000  0.01000 -0.01431  0.43000  1.76000
```

Explication du succès d'un animé

```r
# On extrait les coefficients bruts du modèle Lasso parcimonieux (`1se`) qui a
# effectué la sélection de variables.
coefs_raw <- coef(lasso_1se)

# On convertit cette matrice creuse en un dataframe standard pour faciliter la
# manipulation.
df_coefs <- data.frame(Variable = rownames(coefs_raw), Poids = as.vector(coefs_raw))

# On nettoie le tableau en supprimant l'intercept (constante) et les variables
# dont le coefficient a été réduit à zéro (non sélectionnées).
df_coefs <- df_coefs[df_coefs$Poids != 0 & df_coefs$Variable != "(Intercept)", ]

# On trie les variables par ordre décroissant de valeur absolue pour identifier
# les facteurs ayant le plus fort impact (positif ou négatif).
df_coefs <- df_coefs[order(-abs(df_coefs$Poids)), ]

cat("\n--- TOP 20 DES FACTEURS D'INFLUENCE ---\n")
```

```
##
## --- TOP 20 DES FACTEURS D'INFLUENCE ---
```

```r
print(head(df_coefs, 20))
```

```
##                         Variable      Poids
## 145      producer_Tencent.Video  0.7342013
## 30             source_Web novel  0.5679236
## 115      studio_Kyoto.Animation  0.4445294
## 113                studio_T.Rex  0.3992725
## 167              licensor_GKIDS  0.3937417
## 118            studio_AQUA.ARIS -0.3789994
## 88                  theme_CGDCT  0.3279412
## 85               theme_Gag.Humor  0.2931293
## 104    studio_Shin.Ei.Animation  0.2848401
## 70              theme_Adult.Cast  0.2825250
## 59           genre_Award.Winning  0.2818516
## 169     licensor_NuTech.Digital -0.2503349
## 165 licensor_Central.Park.Media -0.2350481
## 141           producer_bilibili  0.2230227
## 27               source_Unknown -0.2216377
## 26          source_Picture book -0.2213733
## 126            producer_Aniplex  0.2196355
## 57                 genre_Horror -0.2184453
## 86                 theme_Racing  0.2159882
## 106                  studio_DLE -0.2085160
```

```r
# On prend le Top 15 pour que ce soit lisible
top_15 <- head(df_coefs, 15)
```

Vérification du surapprentissage

```r
# On crée une liste contenant tous nos modèles pour pouvoir boucler dessus.
tous_modeles <- list(`OLS (Standard)` = model_ols, `Ridge (min)` = ridge_min, `Ridge (1se)` = ridge_1se
    `Lasso (min)` = lasso_min, `Lasso (1se)` = lasso_1se, `Elastic Net (min)` = enet_min,
    `Elastic Net (1se)` = enet_1se)

# On prépare un tableau vide pour stocker les résultats comparatifs.
tab_overfit <- data.frame(Modele = character(), R2_Train = numeric(), R2_Test = numeric(),
    Ecart = numeric(), stringsAsFactors = FALSE)

# On boucle sur chaque modèle pour calculer son score sur le Train et sur le
# Test.
for (nom in names(tous_modeles)) {
    mod <- tous_modeles[[nom]]
    type_mod <- if (inherits(mod, "lm"))
        "lm" else "glmnet"

    # Calcul sur le TRAIN
    res_train <- eval_model(mod, X_train, y_train, type = type_mod)
    r2_train <- res_train["R2"]

    # Calcul sur le TEST
    res_test <- eval_model(mod, X_test, y_test, type = type_mod)
    r2_test <- res_test["R2"]

    # On ajoute la ligne au tableau (Ecart = Train - Test)
    tab_overfit[nrow(tab_overfit) + 1, ] <- list(nom, r2_train, r2_test, r2_train -
        r2_test)
}
```

```
## Warning in predict.lm(model, newdata = as.data.frame(x_data)): prediction from
## a rank-deficient fit may be misleading
## Warning in predict.lm(model, newdata = as.data.frame(x_data)): prediction from
## a rank-deficient fit may be misleading
```

```r
# On affiche le tableau final complet
print(tab_overfit[order(tab_overfit$Ecart), ])
```

```
##              Modele R2_Train R2_Test  Ecart
## 7 Elastic Net (1se)   0.4547  0.4423 0.0124
## 5       Lasso (1se)   0.4612  0.4480 0.0132
## 3       Ridge (1se)   0.4673  0.4525 0.0148
## 2       Ridge (min)   0.4798  0.4629 0.0169
## 4       Lasso (min)   0.4802  0.4629 0.0173
## 6 Elastic Net (min)   0.4802  0.4629 0.0173
## 1    OLS (Standard)   0.4832  0.4621 0.0211
```

```r
# On affiche le tableau
print(tab_overfit)
```

```
##              Modele R2_Train R2_Test  Ecart
## 1    OLS (Standard)   0.4832  0.4621 0.0211
## 2       Ridge (min)   0.4798  0.4629 0.0169
## 3       Ridge (1se)   0.4673  0.4525 0.0148
## 4       Lasso (min)   0.4802  0.4629 0.0173
## 5       Lasso (1se)   0.4612  0.4480 0.0132
## 6 Elastic Net (min)   0.4802  0.4629 0.0173
## 7 Elastic Net (1se)   0.4547  0.4423 0.0124
```

modeleuser

2025-12-06

Chargement, Nettoyage et Création de la Cible

```r
# On charge les librairies nécessaires
suppressPackageStartupMessages({
    library(data.table)
    library(dplyr)
    library(glmnet)
    library(caret)

})

# On charge les datasets 'anime_level.csv' et 'user_anime_level.csv'
df_anime_source <- fread("anime_level.csv")
df_user <- fread("user_anime_level.csv")

# On réduit la taille de l'échantillon à 100 000 lignes pour limiter le temps
# de calcul.
if (nrow(df_user) > 1e+05) {
    set.seed(2025)
    df_user <- df_user[sample(.N, 1e+05)]
}

# On transforme la note continue (1-10) en classe binaire : 1 si l'utilisateur
# a aimé (Note >= 7), sinon 0.
df_user$Liked <- ifelse(df_user$Y_user_score >= 7, 1, 0)

# On convertit la variable cible en facteur
df_user$Liked <- as.factor(df_user$Liked)

# On définit le seuil de tolérance des valeurs manquantes à 50% du nombre total
# d'observations.
seuil_col <- 0.5 * nrow(df_user)
# On identifie les colonnes dont le cumul de valeurs manquantes reste inférieur
# ou égal à ce seuil.
cols_keep <- colSums(is.na(df_user)) <= seuil_col
# On filtre le dataframe pour ne conserver que les colonnes avec moins de 50%
# de valeurs manquantes
df_user <- df_user[, ..cols_keep]

# On définit le seuil de tolérance des valeurs manquantes à 50% du nombre total
# d'observations.
row_na_pct <- rowMeans(is.na(df_user))
# On ne conserve que les lignes ayant au maximum 50 % de données manquantes
# pour garantir la fiabilité des données.
```

```r
df_user <- df_user[row_na_pct <= 0.5, ]


# On supprime les lignes restantes contenant des NA .
df_user_final <- na.omit(df_user)
df_anime_source <- na.omit(df_anime_source)

# On liste les colonnes techniques à exclure : l'ancien score continu, les
# identifiants et les pseudos.
cols_to_remove <- c("Y_user_score", "username", "mal_id", "anime_mal_id")
# On vérifie l'intersection pour ne retirer que les colonnes qui existent
# encore après le nettoyage.
cols_to_remove <- intersect(names(df_user_final), cols_to_remove)
# On définit la liste finale des features en excluant la cible 'Liked' et les
# variables techniques.
features_cols <- setdiff(names(df_user_final), c("Liked", cols_to_remove))

# On affiche les dimensions
cat("Dimensions finales :", dim(df_user_final), "\n")
```

```
## Dimensions finales : 53120 112
```

```r
# On affiche la répartition des classes (0/1)
print(table(df_user_final$Liked))
```

```
##
##     0     1
## 13649 39471
```

Partitionnement des données

```r
# On fixe la graine aléatoire
set.seed(2025)

# On génère les indices pour sélectionner 80% des données de manière
# stratifiée.
train_idx <- createDataPartition(df_user_final$Liked, p = 0.8, list = FALSE)

# On crée le sous-ensemble d'entraînement contenant 80% des lignes.
train_u <- df_user_final
# On crée le sous-ensemble de test
test_u <- df_user_final[-train_idx, ]

# On convertit les variables explicatives d'entraînement en matrice numérique
X_train <- data.matrix(train_u[, ..features_cols])
# On isole le vecteur cible contenant les labels (0 ou 1) pour la phase
# d'apprentissage.
y_train <- train_u$Liked
# On convertit les variables explicatives d'entraînement en matrice numérique
X_test <- data.matrix(test_u[, ..features_cols])
# On isole le vecteur cible (labels réels) du jeu de test pour la future
# évaluation des performances.
y_test <- test_u$Liked
```

Modèles intermédiaires

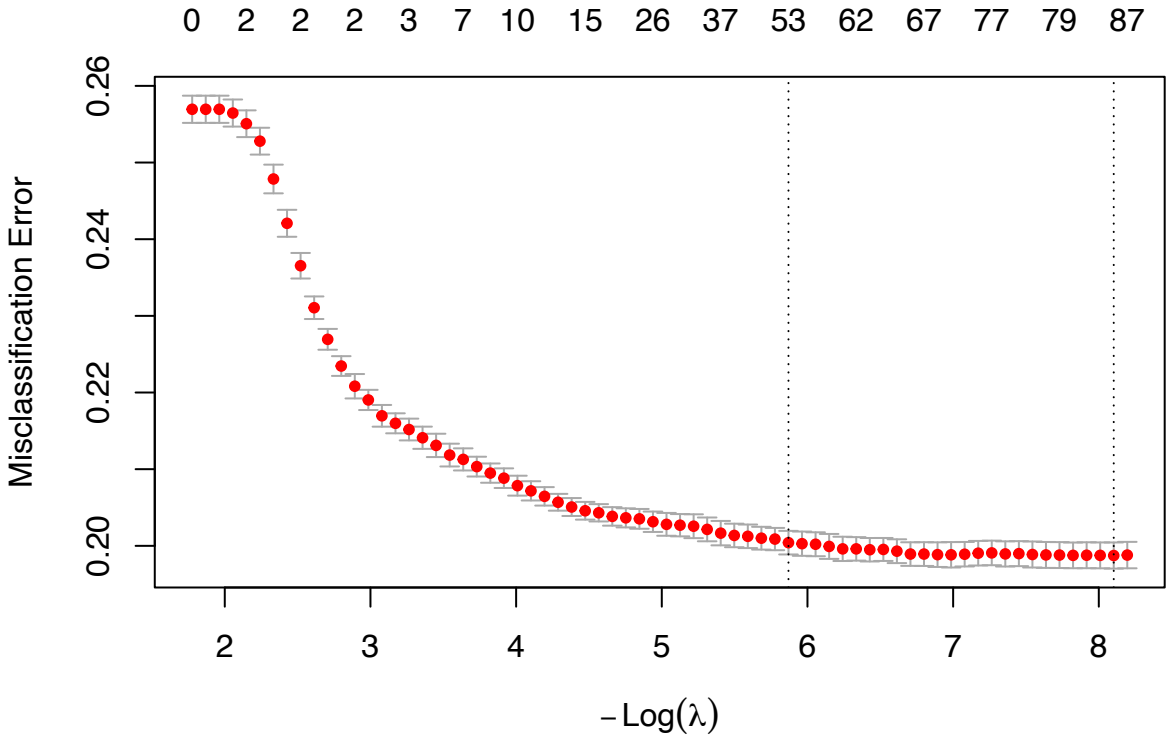
#Modèle logistique classique

```
# Pour le modèle standard (glm), il faut un dataframe et non une matrice.  On
# reconstitue le jeu d'entraînement.
df_train_std <- as.data.frame(X_train)
df_train_std$Liked <- y_train

# On entraîne le modèle logistique classique (équivalent OLS pour la
# classification).  'family = binomial' indique qu'on travaille sur du 0/1.
std_model <- glm(Liked ~ ., data = df_train_std, family = "binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

#Lasso

```
# On lance la validation croisée avec alpha = 1 pour forcer la sélection de
# variables (Lasso).  On spécifie `family = 'binomial'` pour la classification
cv_lasso <- cv.glmnet(X_train, y_train, family = "binomial", alpha = 1, type.measure = "class")
# On visualise la courbe d'erreur de classification en fonction de la pénalité
# pour identifier le point de bascule.
plot(cv_lasso)
```




```
# On instancie le modèle Lasso optimal qui offre le taux d'erreur le plus
# faible sur la validation croisée.
lasso_model_min <- glmnet(X_train, y_train, family = "binomial", alpha = 1, lambda = cv_lasso$lambda.mi
# On instancie le modèle Lasso le plus parcimonieux qui simplifie le modèle
# tout en restant dans la marge d'erreur standard.
lasso_model_1se <- glmnet(X_train, y_train, family = "binomial", alpha = 1, lambda = cv_lasso$lambda.1s

# On affiche les valeurs exactes des lambda sélectionnés
cat("Lasso Lambda min:", cv_lasso$lambda.min, "\n")
```

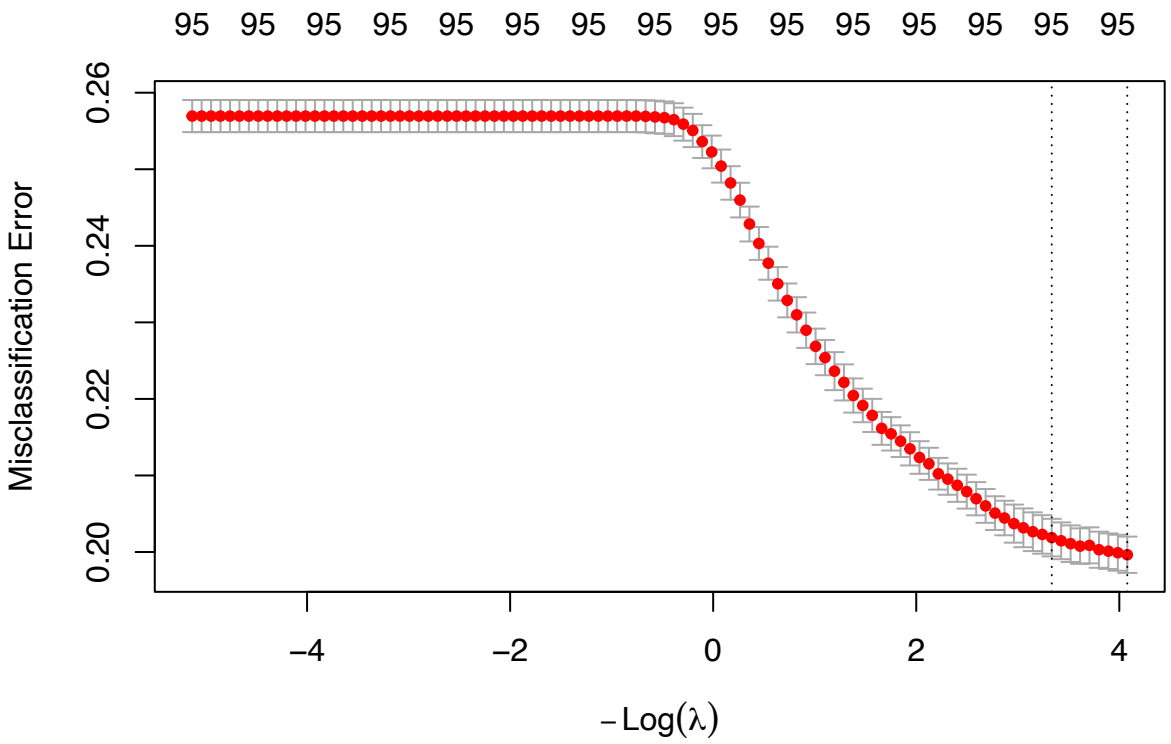
```
## Lasso Lambda min: 0.0003027936
```

```
cat("Lasso Lambda 1se:", cv_lasso$lambda.1se, "\n")
```

```
## Lasso Lambda 1se: 0.002823863
```

#Ridge

```
# On lance la validation croisée avec alpha = 0 pour conserver toutes les
# variables tout en réduisant leur impact
cv_ridge <- cv.glmnet(X_train, y_train, family = "binomial", alpha = 0, type.measure = "class")
# On visualise l'erreur de classification pour vérifier la stabilité du modèle
# face à la régularisation.
plot(cv_ridge)
```




```
# On construit le modèle en utilisant le lambda qui a minimisé l'erreur
ridge_model_min <- glmnet(X_train, y_train, family = "binomial", alpha = 0, lambda = cv_ridge$lambda.mi
# On construit le modèle en utilisant la règle du '1 Standard Error'
# (lambda.1se).
ridge_model_1se <- glmnet(X_train, y_train, family = "binomial", alpha = 0, lambda = cv_ridge$lambda.1s

# On affiche les valeurs des lambda sélectionnés
cat("Ridge Lambda min:", cv_ridge$lambda.min, "\n")
```

```
## Ridge Lambda min: 0.01692862
```

```
cat("Ridge Lambda 1se:", cv_ridge$lambda.1se, "\n")
```

```
## Ridge Lambda 1se: 0.03563312
```

Elastic Net

```r
# On définit une grille de valeurs pour alpha (de 0.1 à 0.9) afin d'explorer
# les mélanges intermédiaires entre Ridge et Lasso.
list_alpha <- seq(0.1, 0.9, by = 0.1)
# On initialise un tableau pour enregistrer l'erreur minimale et le lambda
# associé à chaque configuration.
results_en <- data.frame(alpha = list_alpha, cv_error = NA, lambda = NA)

# On lance une boucle pour évaluer systématiquement la performance de chaque
# nuance de mélange
for (i in 1:length(list_alpha)) {
    a <- list_alpha[i]
    # On exécute la validation croisée pour l'alpha courant afin de mesurer sa
    # capacité prédictive.
    cv_temp <- cv.glmnet(X_train, y_train, alpha = a, family = "binomial", type.measure = "class")
    # On mémorise l'erreur minimale obtenue pour pouvoir comparer cet alpha aux
    # autres.
    results_en$cv_error[i] <- min(cv_temp$cvm)
    results_en$lambda[i] <- cv_temp$lambda.min
}

# On sélectionne l'alpha qui a offert la plus faible erreur moyenne sur
# l'ensemble des tests.
best_res <- results_en[which.min(results_en$cv_error), ]
best_alpha <- best_res$alpha

# On relance une validation croisée finale fixée sur cet alpha optimal
cv_final_enet <- cv.glmnet(X_train, y_train, alpha = best_alpha, family = "binomial",
    type.measure = "class")

# On instancie le modèle Elastic Net final de performance pure (`lambda.min`)
# sur l'ensemble d'entraînement.
enet_model_min <- glmnet(X_train, y_train, alpha = best_alpha, family = "binomial",
    lambda = cv_final_enet$lambda.min)
# On instancie le modèle Elastic Net final robuste (`lambda.1se`) favorisant la
# parcimonie.
enet_model_1se <- glmnet(X_train, y_train, alpha = best_alpha, family = "binomial",
    lambda = cv_final_enet$lambda.1se)

# On affiche les valeurs des lambda
cat("Elastic Net Lambda min:", cv_final_enet$lambda.min, "\n")
```

```
## Elastic Net Lambda min: 0.0004189743
```

```r
cat("Elastic Net Lambda 1se:", cv_final_enet$lambda.1se, "\n")
```

```
## Elastic Net Lambda 1se: 0.003243966
```

Évaluation et Comparaison des Performances

```r
# On définit une fonction Accuracy pour mesurer la qualité de la
# classification.
eval_classif <- function(model, x_data, y_true, nom_modele = "Modèle") {
```

```r
    # On génère les probabilités d'appartenance à la classe 1 ('Aimé') pour les
    # nouvelles données.
    probs <- predict(model, newx = x_data, type = "response")
    # On applique un seuil de décision à 50% : si probabilité > 0.5, on prédit
    # 1, sinon 0.
    preds_class <- ifelse(probs > 0.5, 1, 0)

    # On construit la matrice de confusion

    cm <- table(Réalité = y_true, Prédiction = preds_class)

    print(cm)

    # On calcule l'Accuracy globale : proportion de prédictions correctes sur
    # le total des observations.
    accuracy <- sum(diag(cm))/sum(cm)
    cat("Accuracy :", round(accuracy, 4), "\n")

    return(accuracy)
}

# On calcule les scores de précision (Accuracy) pour les variantes Lasso sur le
# jeu de test.
acc_lasso_min <- eval_classif(lasso_model_min, X_test, y_test, "Lasso (Min)")
```

```
##         Prédiction
## Réalité    0    1
##       0 1047 1682
##       1  443 7451
## Accuracy : 0.8
```

```r
acc_lasso_1se <- eval_classif(lasso_model_1se, X_test, y_test, "Lasso (1se)")
```

```
##         Prédiction
## Réalité    0    1
##       0  998 1731
##       1  397 7497
## Accuracy : 0.7997
```

```r
# On calcule les scores de précision (Accuracy) pour les variantes Ridge sur le
# jeu de test
acc_ridge_min <- eval_classif(ridge_model_min, X_test, y_test, "Ridge (Min)")
```

```
##         Prédiction
## Réalité    0    1
##       0  948 1781
##       1  348 7546
## Accuracy : 0.7996
```

```r
acc_ridge_1se <- eval_classif(ridge_model_1se, X_test, y_test, "Ridge (1se)")
```

```
##         Prédiction
## Réalité    0    1
##       0  865 1864
##       1  277 7617
## Accuracy : 0.7985
```

```r
# On calcule les scores de précision (Accuracy) pour les variantes Elastic Net
# sur le jeu de test
acc_enet_min <- eval_classif(enet_model_min, X_test, y_test, "ELASTIC NET (Min)")
```

```
##        Prédiction
## Réalité    0    1
##       0 1048 1681
##       1  443 7451
## Accuracy : 0.8001
```

```r
acc_enet_1se <- eval_classif(enet_model_1se, X_test, y_test, "ELASTIC NET (1se)")
```

```
##        Prédiction
## Réalité    0    1
##       0 1008 1721
##       1  404 7490
## Accuracy : 0.8
```

```r
# On rassemble l'ensemble des résultats dans un tableau récapitulatif pour
# comparer les performances.
final_tab <- data.frame(Modele = c("Lasso (min)", "Lasso (1se)", "Ridge (min)", "Ridge (1se)",
    "Elastic Net (min)", "Elastic Net (1se)"), Accuracy = c(acc_lasso_min, acc_lasso_1se,
    acc_ridge_min, acc_ridge_1se, acc_enet_min, acc_enet_1se))

# On trie et affiche le tableau par ordre décroissant d'Accuracy pour
# identifier le meilleur modèle prédictif.
print(final_tab[order(-final_tab$Accuracy), ])
```

```
##              Modele  Accuracy
## 5 Elastic Net (min) 0.8000565
## 1       Lasso (min) 0.7999623
## 6 Elastic Net (1se) 0.7999623
## 2       Lasso (1se) 0.7996799
## 3       Ridge (min) 0.7995858
## 4       Ridge (1se) 0.7984562
```

Recommandation ciblée

```r
# On sélectionne un anime spécifique (ici la ligne 890) pour lequel on souhaite
# trouver l'audience idéale.
target_anime <- df_anime_source[890, ]
# On affiche le titre de l'animé
cat("Simulation pour l'anime :", target_anime$title, "\n")
```

```
## Simulation pour l'anime :
```

```r
# On extrait une liste unique d'utilisateurs pour éviter les doublons dans la
# recommandation.
prediction_set <- df_user[!duplicated(username), ]

# On identifie les colonnes communes entre l'anime et l'historique utilisateur
# (Genres, Studio, etc.).
cols_communes <- intersect(names(target_anime), names(prediction_set))

# On exclut les variables propres à l'utilisateur (Pseudo, Note moyenne) pour
```

```r
# ne modifier que les infos de l'anime.
cols_a_modifier <- setdiff(cols_communes, c("username", "Y_user_score", "Liked",
    "user_mean_score_from_ratings"))

# On remplace les données existantes par les caractéristiques de l'anime cible
# pour *tous* les utilisateurs.
for (col in cols_a_modifier) {
    valeur_cible <- target_anime[[col]][1]
    set(prediction_set, j = col, value = valeur_cible)
}

# On convertit ce jeu de données simulé en matrice numérique compatible avec le
# modèle Lasso entraîné.
cols_valides <- intersect(features_cols, names(prediction_set))
X_target <- data.matrix(prediction_set[, ..cols_valides])

# On utilise le modèle Lasso (1se) pour prédire la probabilité ('type =
# response') que chaque utilisateur aime l'anime.
scores_prob <- predict(lasso_model_1se, newx = X_target, type = "response")

# On structure les résultats en convertissant la probabilité en un 'Score de
# Compatibilité' (%).
resultats <- data.frame(Utilisateur = prediction_set$username, Score_Compatibilite = round(as.numeric(s
    100, 2))

# On extrait le Top 30 des utilisateurs les plus susceptibles d'aimer l'œuvre.
top_30 <- head(resultats[order(-resultats$Score_Compatibilite), ], 30)

print(top_30)
```

```
##            Utilisateur Score_Compatibilite
## 937         ReMightyRon               99.85
## 936              8angel               99.69
## 205           PerfectGod               99.65
## 14294        THE_HIDDEN               99.61
## 5087       SasakiMichie               99.44
## 14500       60189134403               99.32
## 25658            qOskuu               99.29
## 34238          BENBOURY               99.22
## 3482              bog20               99.17
## 71           Mikura-san               99.12
## 4496              DSAMei               99.12
## 20159 HerrscherOfFlame               99.12
## 19221      WellingtonSO               99.09
## 12124    anime_king1000               99.07
## 27602           -Kirito               99.07
## 43079     Twintail-Sama               99.07
## 61378      airimizuno27               99.07
## 27254             s1k1b               99.06
## 1587     Lelouch_Caesar               99.05
## 6904    joy_chakrabarty               99.05
## 18882        slickpanda               99.05
## 23774   EcchiGodMamster               99.05
## 25465      heraodiada25               99.05
```

```
## 44799        BmonAnime              99.05
## 47353          Saber_4              99.05
## 19190         Shirox_x              99.04
## 25108   JustCallMyName              99.04
## 39217        Soul-Land              99.04
## 12342        GrandSnow              99.03
## 21012          ivanjrs              99.03
```

Vérification du surapprentissage

```r
# On définit une petite fonction locale pour récupérer juste l'Accuracy.
get_acc_only <- function(model, x, y) {
    p <- predict(model, newx = x, type = "response")
    pred <- ifelse(p > 0.5, 1, 0)
    return(sum(diag(table(y, pred)))/length(y))
}

# On liste les modèles
liste_modeles <- list(`Lasso (min)` = lasso_model_min, `Lasso (1se)` = lasso_model_1se,
    `Ridge (min)` = ridge_model_min, `Ridge (1se)` = ridge_model_1se, `Elastic Net (min)` = enet_model_
    `Elastic Net (1se)` = enet_model_1se)

# On prépare le tableau
tab_overfit_classif <- data.frame(Modele = character(), Acc_Train = numeric(), Acc_Test = numeric(),
    Ecart = numeric(), stringsAsFactors = FALSE)

# On boucle
for (nom in names(liste_modeles)) {
    mod <- liste_modeles[[nom]]

    # Calcul Accuracy Train
    acc_train <- get_acc_only(mod, X_train, y_train)

    # Calcul Accuracy Test
    acc_test <- get_acc_only(mod, X_test, y_test)

    # Ajout au tableau
    tab_overfit_classif[nrow(tab_overfit_classif) + 1, ] <- list(nom, round(acc_train,
        4), round(acc_test, 4), round(acc_train - acc_test, 4))
}

# On affiche le résultat trié par le plus petit écart
print(tab_overfit_classif[order(tab_overfit_classif$Ecart), ])
```

```
##              Modele Acc_Train Acc_Test  Ecart
## 4       Ridge (1se)    0.7985   0.7985 0.0000
## 2       Lasso (1se)    0.8000   0.7997 0.0003
## 6 Elastic Net (1se)    0.8007   0.8000 0.0007
## 3       Ridge (min)    0.8009   0.7996 0.0013
## 5 Elastic Net (min)    0.8017   0.8001 0.0017
## 1       Lasso (min)    0.8018   0.8000 0.0018
```