Modelagem XGBOOST para a classificação de avaliações de livros

Ana Luzielma Dias Campos
Jaylhane Veloso Nunes
Raianny da Silva Soares

Introdução

Quando se está procurando uma nova leitura, uma das coisas que pode ser observada é a avaliação do livro. De acordo com ela, pode-se ter uma ideia inicial se o livro é bom ou se ele segue uma dinâmica que se está habituado, já que muitas pessoas deram uma nota alta de avaliação. Pensando sobre isso, levantamos o questionamento: seria possível predizer se um livro é bom sem ter acesso a nota da avaliação? Dessa forma pensamos em realizar um modelo para classificar a avaliação de um livro. Para esta tarefa utilizaremos o seguinte conjunto de dados: Goodreads-books| Kaggle e como inspiração para construção do modelo utilizaremos o seguinte guia: Tune xgboost models with early stopping to predict shelter animal status| Julia Silge.

Assim, uma das possibilidades é fazer uma categorização das avaliações dos livros em "Ruim", "Bom" e ''Ótimo", considerando respectivamente os intervalos das notas como, [0, 3.5), [3.5, 4] e (4, 5], e a partir daí predizer a avaliação dos livros utilizando o XGboost. O critério de intervalo para as categorias das notas foi definido subjetivamente ao acaso entre as participantes do grupo, tentando balancear a quantidade de observações que ficariam em cada grupo.

Além disso, como o objetivo é classificar os livros sem olhar as avaliações, as notas não farão parte do modelo, elas serão utilizadas apenas para criar as categorias e estamos supondo que de alguma forma as variáveis como número de páginas, idade do livro, editora, quantidade de notas de avaliações e quantidade de avaliações escritas estão relacionadas com a avaliação do livro.

Análise exploratória

- Limpeza dos dados

```
library(knitr)
opts_chunk$set(message=FALSE,
               warning=FALSE,
               echo = TRUE,
               cache = TRUE,
               dev = "png",
               dpi = 500)

#Pacotes necessários
library(tidyverse)
library(tidymodels)
library(lubridate)
library(vip)
library(GGally)
```

```
theme_set(theme_light(base_family = "IBMPlexSans"))

# Carregando os dados

livros <- read.csv("./Conjunto de Dados/books.csv",
                   encoding = "UTF-8",
                   header = TRUE) %>%
  select(-bookID,
         -title,
         -isbn,
         -isbn13) %>%
  na.omit()
```

- Verificando a língua

Análisando o conjunto de dados percebemos que algumas variáveis nos códigos da linguagem estavam em formato de númeração e precisaram ser removidas.

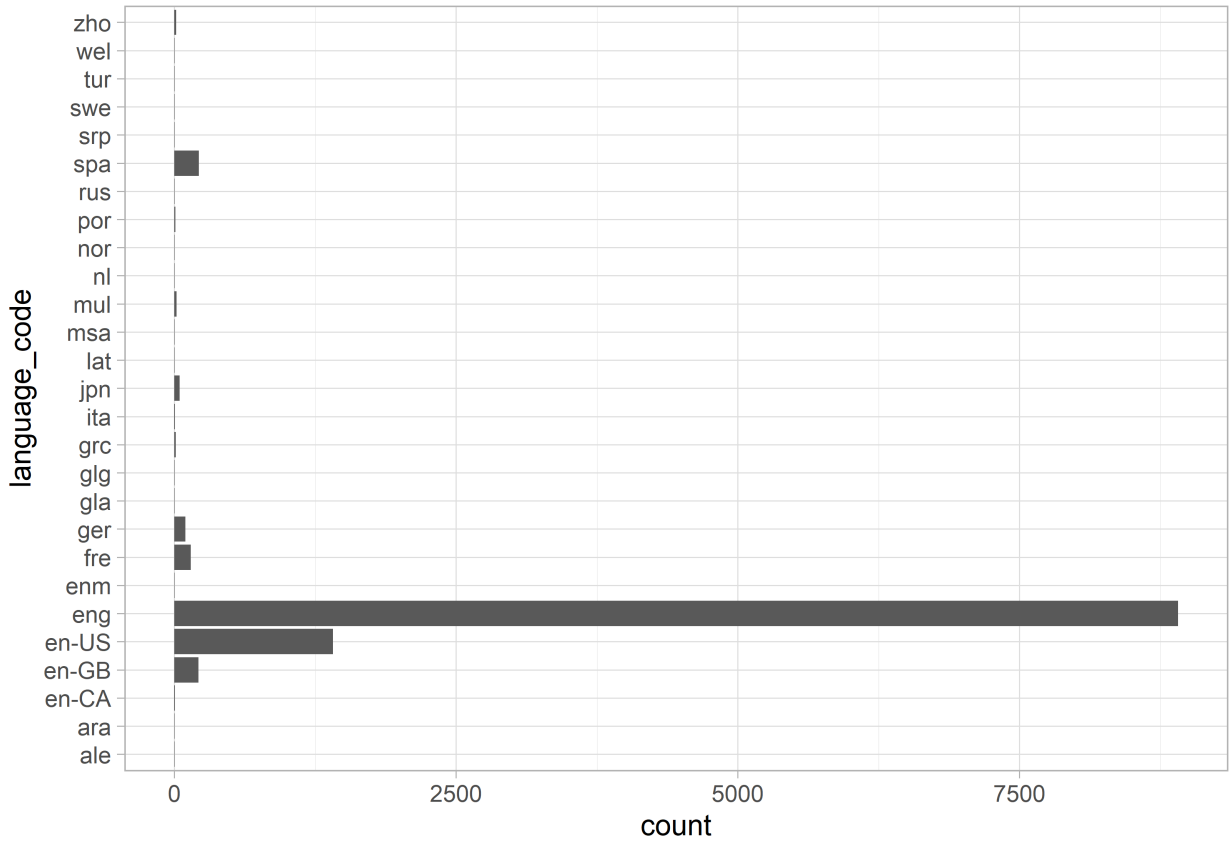
```
summary(as.factor(livros$language_code))
```

```
## 9780674842113 9780851742717 9781563841552 9781593600112           ale
##             1             1             1             1             1
##           ara         en-CA         en-GB         en-US           eng
##             1             7           214          1408          8908
##           enm           fre           ger           gla           glg
##             3           144            99             1             1
##           grc           ita           jpn           lat           msa
##            11             5            46             3             1
##           mul            nl           nor           por           rus
##            19             1             1            10             2
##           spa           srp           swe           tur           wel
##           218             1             2             1             1
##           zho
##            14
```

```
linhas_invalidas_de_language_code<- livros %>%
  filter(language_code=="9780674842113"|
           language_code=="9780851742717"|
           language_code=="9781563841552"|
           language_code=="9781593600112"|
           language_code=="")

livros <- livros %>%
  filter(!(language_code%in%linhas_invalidas_de_language_code$language_code))

rm(linhas_invalidas_de_language_code)
```

Fazendo o gráfico de barras para ver a variação dessas linguagens, obtemos o seguinte resultado:

```
livros %>%
  ggplot(aes(language_code))+
  geom_bar()+
  coord_flip()
```



Como há muito pouca variação linguistica comparado ao grupo inglês, dividiremos a categoria de `language_code` em duas: inglês e outros.

```
livros <- livros %>%
  mutate(publication_date = mdy(publication_date),
         average_rating = as.double(average_rating),
         num_pages = as.integer(num_pages),
         book_age = year(today())-year(publication_date),
         month_publication = as.factor(month(publication_date)),
         year_publication = as.factor(year(publication_date)),
         language_code = factor(
           ifelse(language_code %in% c("enm",
                                       "eng",
                                       "en-US",
                                       "en-GB",
                                       "en-CA"),
                  "English","Other")
         )
  ) %>%
  select(-authors, -publisher) %>%
  na.omit()
```

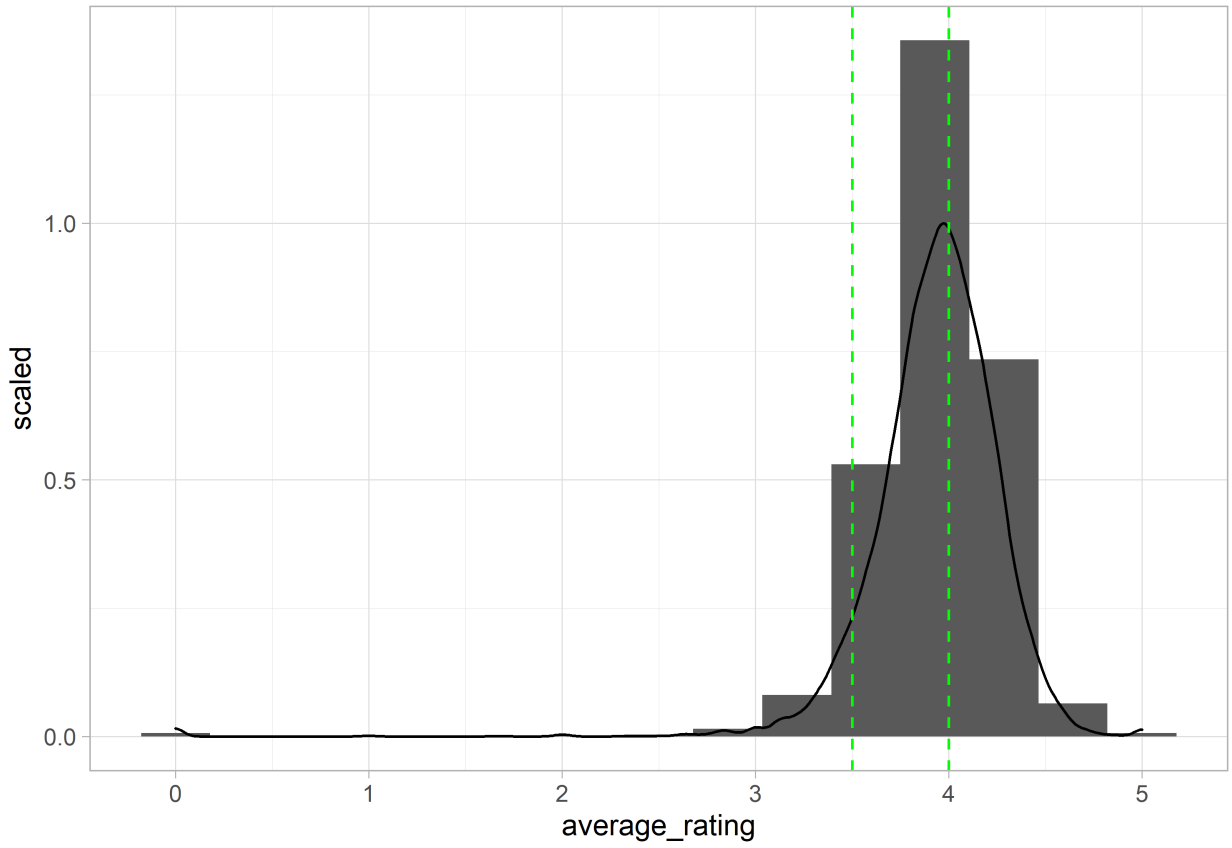
```
summary(livros)
```

```
##  average_rating  language_code     num_pages      ratings_count
##  Min.   :0.000   English:10539   Min.   :   0.0   Min.   :      0
##  1st Qu.:3.770   Other   :  582   1st Qu.: 192.0   1st Qu.:    104
##  Median :3.960                   Median : 299.0   Median :    745
##  Mean   :3.934                   Mean   : 336.3   Mean   :  17945
##  3rd Qu.:4.140                   3rd Qu.: 416.0   3rd Qu.:   4996
##  Max.   :5.000                   Max.   :6576.0   Max.   :4597666
##
##  text_reviews_count publication_date        book_age      month_publication
##  Min.   :    0.0    Min.   :1900-01-01   Min.   :  2.00   9      :1278
##  1st Qu.:    9.0    1st Qu.:1998-07-17   1st Qu.: 17.00   10     :1212
##  Median :   47.0    Median :2003-03-01   Median : 19.00   1      :1057
##  Mean   :  542.1    Mean   :2000-08-29   Mean   : 21.83   4      : 991
##  3rd Qu.:  238.0    3rd Qu.:2005-10-01   3rd Qu.: 24.00   5      : 922
##  Max.   :94265.0    Max.   :2020-03-31   Max.   :122.00   6      : 879
##                                                           (Other):4782
##  year_publication
##  2006   :1700
##  2005   :1260
##  2004   :1069
##  2003   : 931
##  2002   : 798
##  2001   : 656
##  (Other):4707
```

- Separando os grupos

Temos 50% das observações estão entre [0,3.96] e o 1º Q é 3.77, que é bem próximo, mostrando que há uma concentração de avaliações, verificando o histograma dessa variável temos:

```
livros %>%
  ggplot(aes(x=average_rating, after_stat(scaled)))+
  geom_histogram(aes(y=..density..),
                 bins = 15)+
  geom_density()+
  geom_vline(xintercept = c(3.5,4), color = "green", lty=2)
```



E conferindo a quantidade de observações menores de 3 temos:

```
livros %>%
  filter(average_rating<3.5) %>%
  count()
```

```
##     n
## 1 733
```

```
summary(livros$average_rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   3.770   3.960   3.934   4.140   5.000
```

```
quantile(livros$average_rating,.67)
```

```
##  67%
## 4.07
```

```
livros %>%
  filter(average_rating>4) %>%
  count()
```

```
##      n
## 1 4734
```

Dessa forma, trabalharemos apenas com três categorias, “Ruim”, “Bom” e ’’Ótimo”, considerando respectivamente os intervalos de nota de [0, 3.5), de [3.5, 4] e de (4, 5], uma vez que pelo histograma é notado a distribuição nas avaliações de 3 a 5.

Sendo assim, nosso conjunto de dados final é composto por três categorias: “Ruim”, “Bom” e ’’Ótimo”. Aplicando no conjunto de dados:

```
livros <- livros %>%
  mutate(
    book_rating =
      case_when(average_rating<3.5 ~ "Ruim",
                average_rating<=4 ~ "Bom",
                TRUE ~ "Ótimo")
  ) %>%
  select(-average_rating)

livros %>%
  group_by(book_rating) %>%
  count()
```

```
## # A tibble: 3 x 2
## # Groups:   book_rating [3]
##   book_rating     n
##   <chr>       <int>
## 1 Bom          5654
## 2 Ótimo        4734
## 3 Ruim          733
```

```
# Salvando os dados atuais

write.csv(livros,
          "./Conjunto de Dados/books_t.csv",
          fileEncoding = "UTF-8",
          row.names = FALSE)
```

Análise Descritiva e Exploratória

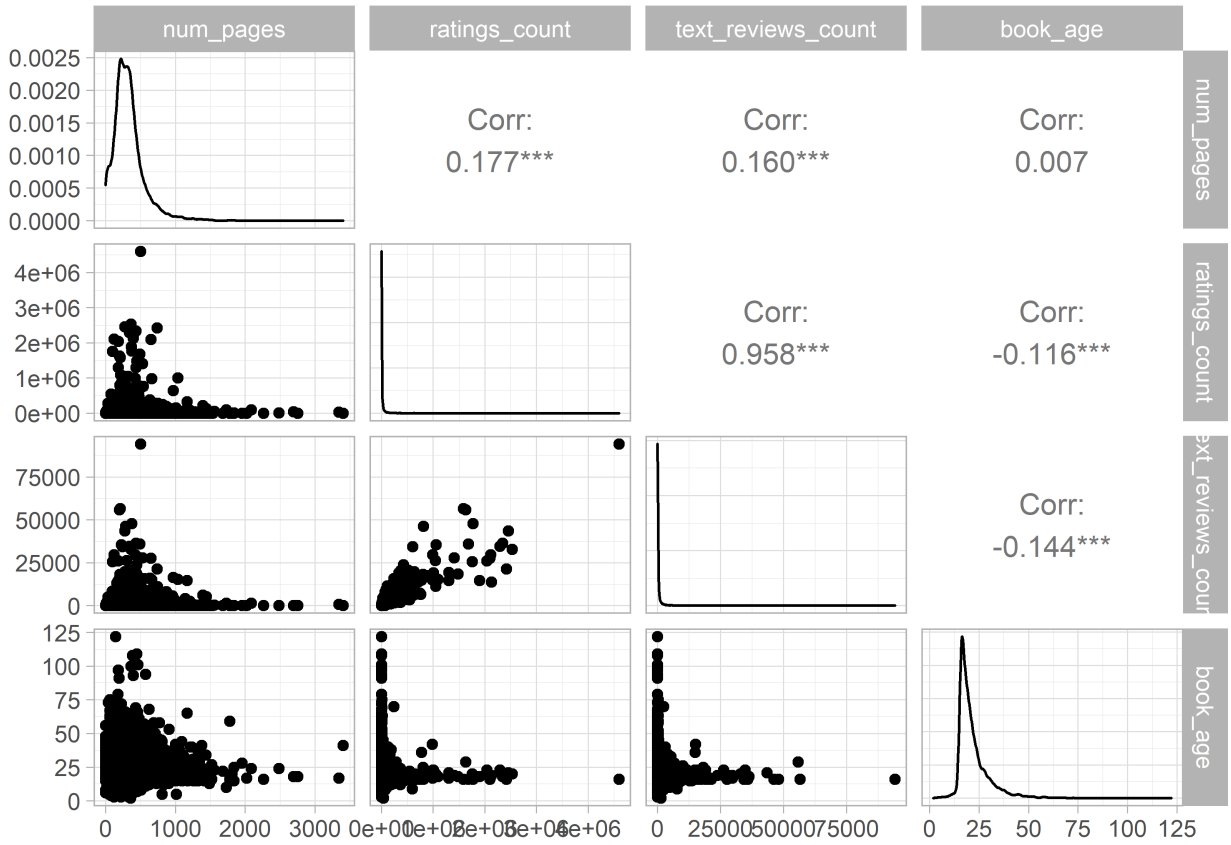
```
# Carregando conjunto de dados após a limpeza

livros <- read.csv("./Conjunto de Dados/books_t.csv",
                   encoding = "UTF-8") %>%
  mutate_if(is.character,factor) %>%
  mutate(month_publication=factor(month_publication),
         year_publication=factor(year_publication),
         book_rating=factor(book_rating,
                            levels = c("Ótimo","Bom","Ruim")))
```

- Separando em Treino e Teste

```
set.seed(1904, kind = "Mersenne-Twister", normal.kind = "Inversion")

livros_split <- initial_split(livros, prop = .75, strata = book_rating)

livros_treino <- training(livros_split)

livros_teste <- testing(livros_split)
```

- Verificando a correlação das variáveis quantitativas

```
livros_treino %>%
  select(where(is.numeric)) %>%
  ggpairs(upper = list(continuous = wrap("cor", method = "spearman")))
```



Dado que identificamos alta correlação entre as variáveis `text_reviews_counte` `rating_count` a variável text será removida pois não necessariamente todo mundo que dá uma nota de avaliação também deixa uma avaliação escrita, o que inclusive explica a forte correlação entre essas variáveis, pois certamente todos que deixaram avaliação escrita também deixaram nota, no entanto, consideramos essa medida importante para avaliar se o livro é ótimo ou ruim, supondo que quando um livro for uma dessas duas opções as pessoas façam mais questão de comentar.

Sendo assim, criaremos uma variável proporção:

```
livros_treino <- livros_treino %>%
  mutate(prop_text_reviews = text_reviews_count / ratings_count) %>%
```

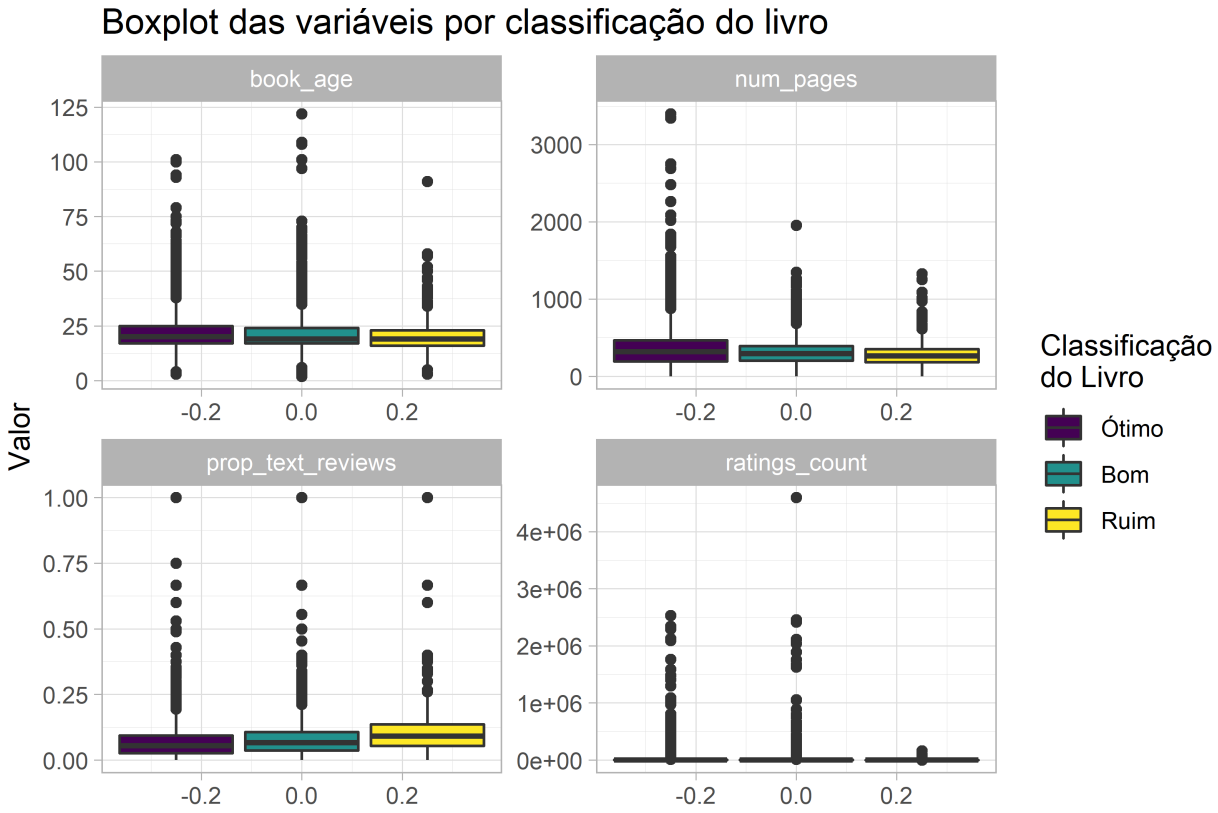
```
  select(-text_reviews_count)

cor(livros_treino$prop_text_reviews,livros_treino$ratings_count,
    use = "complete", method = "spearman")
```

```
## [1] -0.3605444
```

Com essa nova variável tivemos uma baixa correlação, assim evitamos a multicolineariedade.

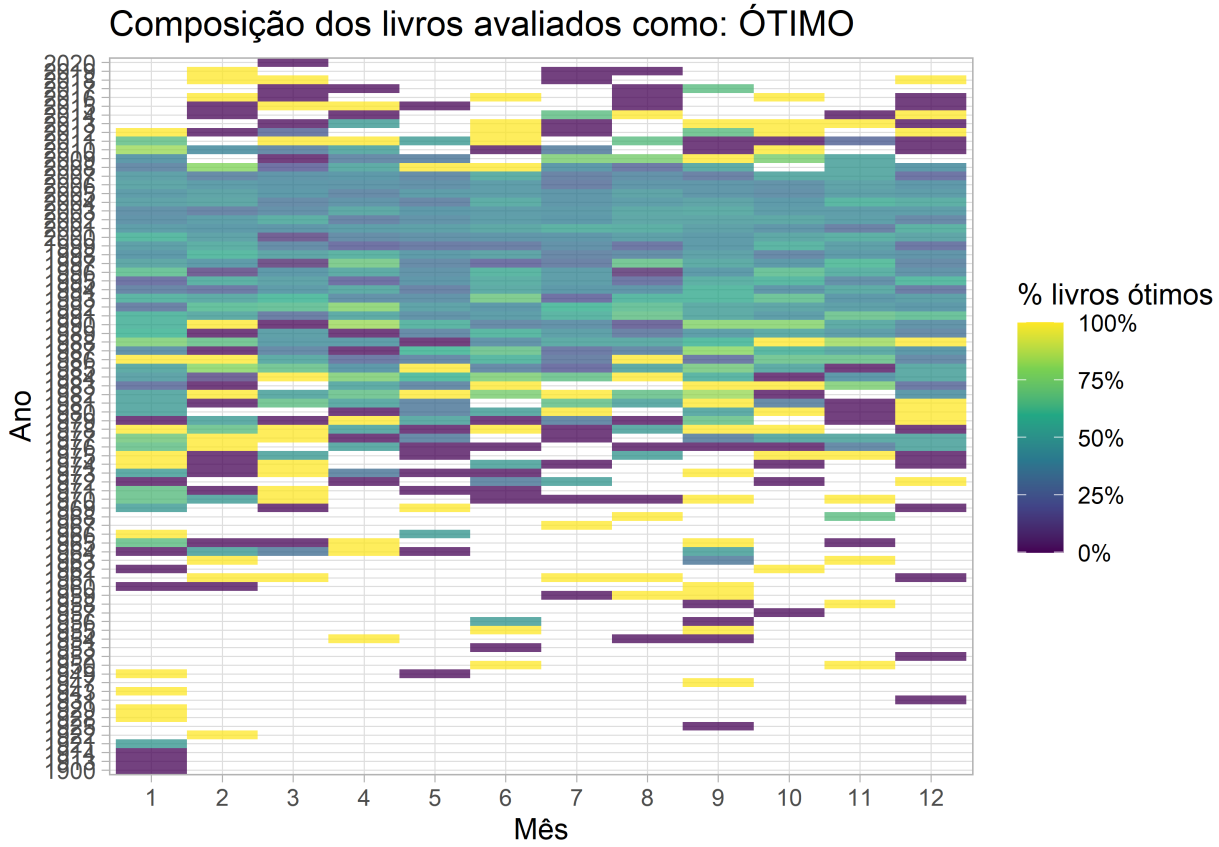
- Análise das variávies

```
livros_treino %>%
  select(where(is.numeric),book_rating) %>%
  pivot_longer(-book_rating) %>%
  ggplot(.,aes(fill = book_rating)) +
  geom_boxplot(aes(y=value)) +
  facet_wrap(~ name, scales = "free") +
  labs(x="",
       y="Valor",
       fill = "Classificação\ndo Livro",
       title = "Boxplot das variáveis por classificação do livro")+
  scale_fill_viridis_d()
```



Pelos box-plots é notado que as distribuições das classificações de acordo com as variáveis há poucas diferenças entre si.

```
grafico_otimos_mes_ano <- livros_treino %>%
  mutate(book_rating = book_rating == "Ótimo") %>%
  group_by(
    mes = month_publication,
    ano = year_publication
  ) %>%
  summarise(book_rating = mean(book_rating)) %>%
  ggplot(aes(mes,ano, fill = book_rating)) +
  geom_tile(alpha = .75) +
  scale_fill_viridis_c(labels = scales::percent) +
  labs(fill = "% livros ótimos" , x="Mês", y="Ano",
       title = "Composição dos livros avaliados como: ÓTIMO")+
  theme(legend.position = "right");grafico_otimos_mes_ano
```

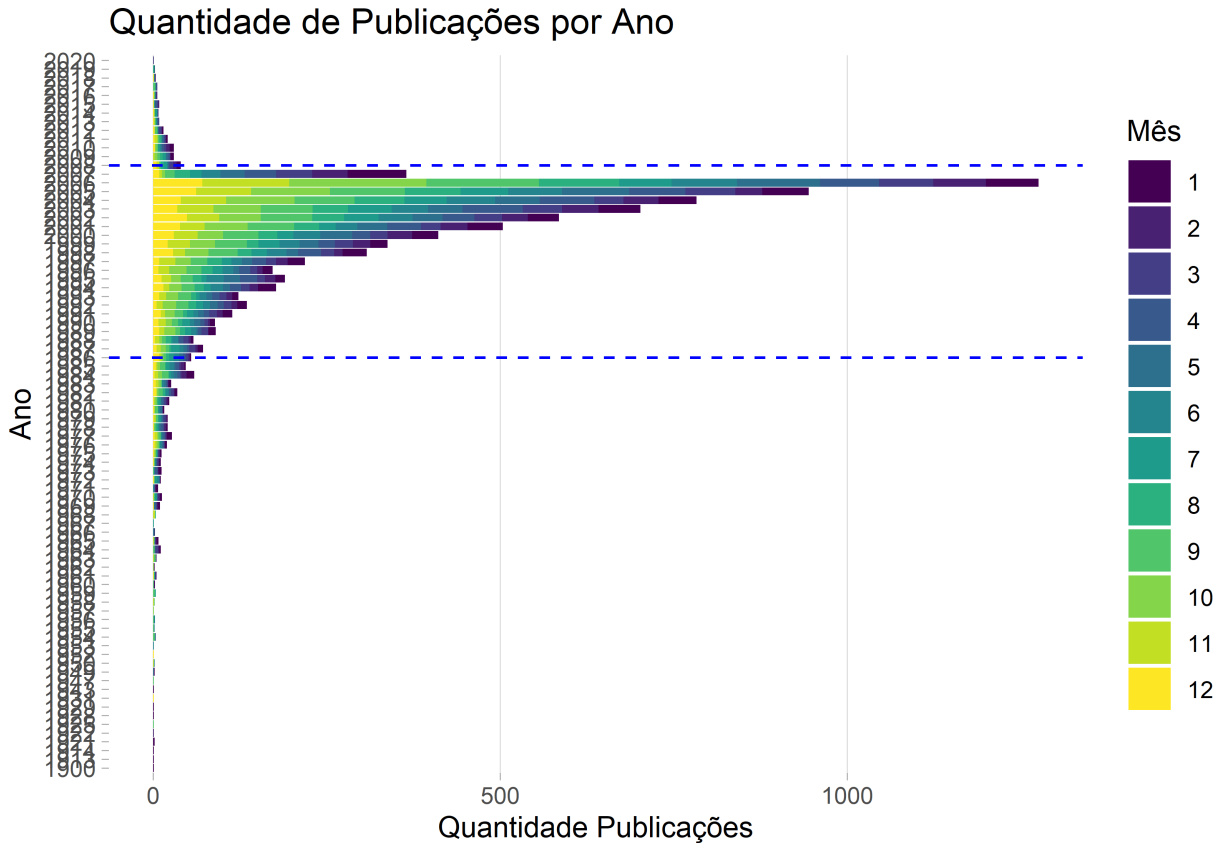


É observado que há uma maior avaliação de livros a partir dos anos 80 até em torno do ano de 2012. Nesses anos tiveram muitas avaliações de livros e a porcentagem de avaliação para ótimo está em torno de 25% a 75% em sua maioria.

Verificando a distribuição de livros publicados ao longo dos anos temos:

```
livros_treino %>%
  group_by(
    mes = month_publication,
    ano = year_publication
  ) %>%
  count() %>%
```

```
ggplot(aes(n,ano, fill=mes))+
geom_col()+
geom_hline(yintercept = "1986", color = "blue", lty=2)+
geom_hline(yintercept = "2008", color = "blue", lty=2)+
theme(panel.border = element_blank(),
      panel.grid.major.y = element_blank(),
      panel.grid.minor = element_blank())+
labs(x = "Quantidade Publicações",
     y = "Ano",
     fill = "Mês",
     title = "Quantidade de Publicações por Ano")+
scale_fill_viridis_d()
```

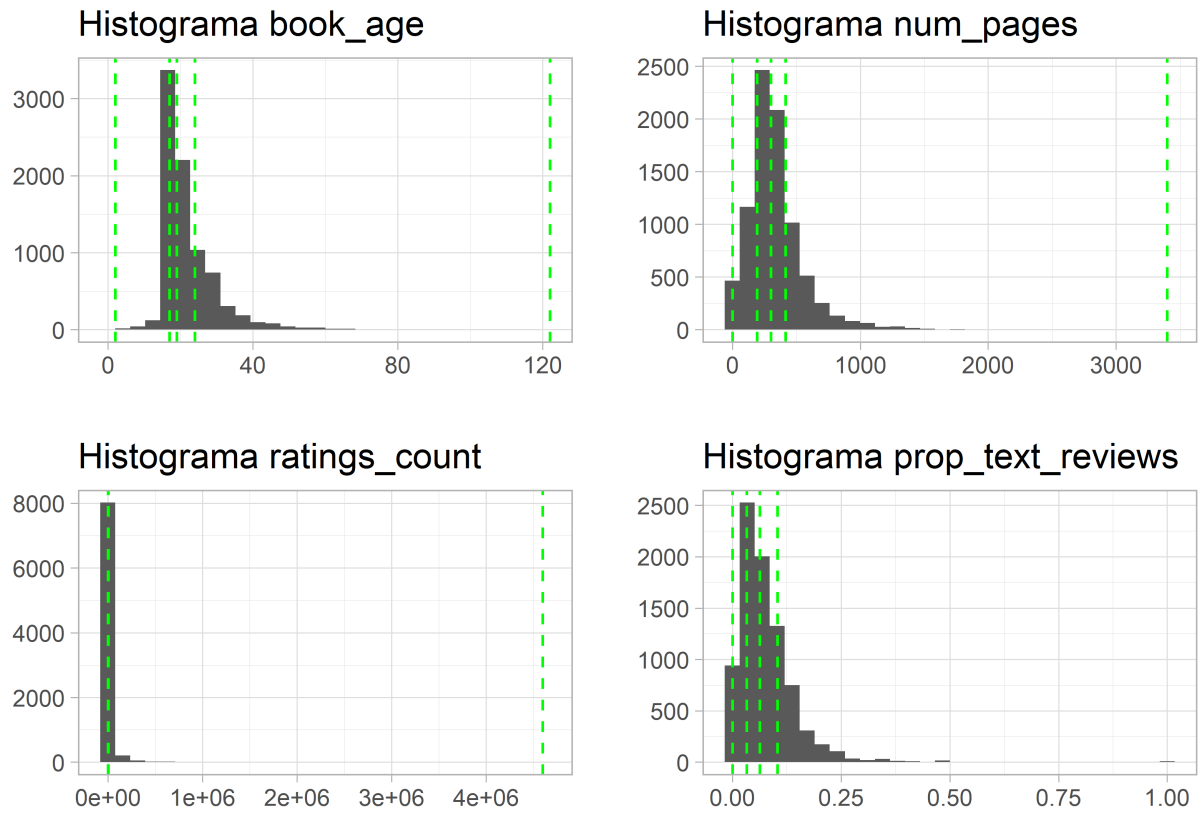


O aumento das publicações dos livros foram bem proporcionais entre 1986 e 2007, não é perceptível algum mês que se destaque entre os demais. Inclusive até mesmo no ano 2008 em que teve uma redução, ainda assim também foi proporcional entre os meses.

Como o boxplot apresentou muitos outliers e percebemos uma concentração nos dados iremos realizar uns filtros para melhorar a modelagem:

```
gridExtra::grid.arrange(ncol=2,
                        livros_treino %>%
  ggplot(aes(x=book_age)) +
  geom_histogram(bins=30)+
  geom_vline(xintercept = quantile(livros_treino$book_age),
             color="green", lty=2)+
```

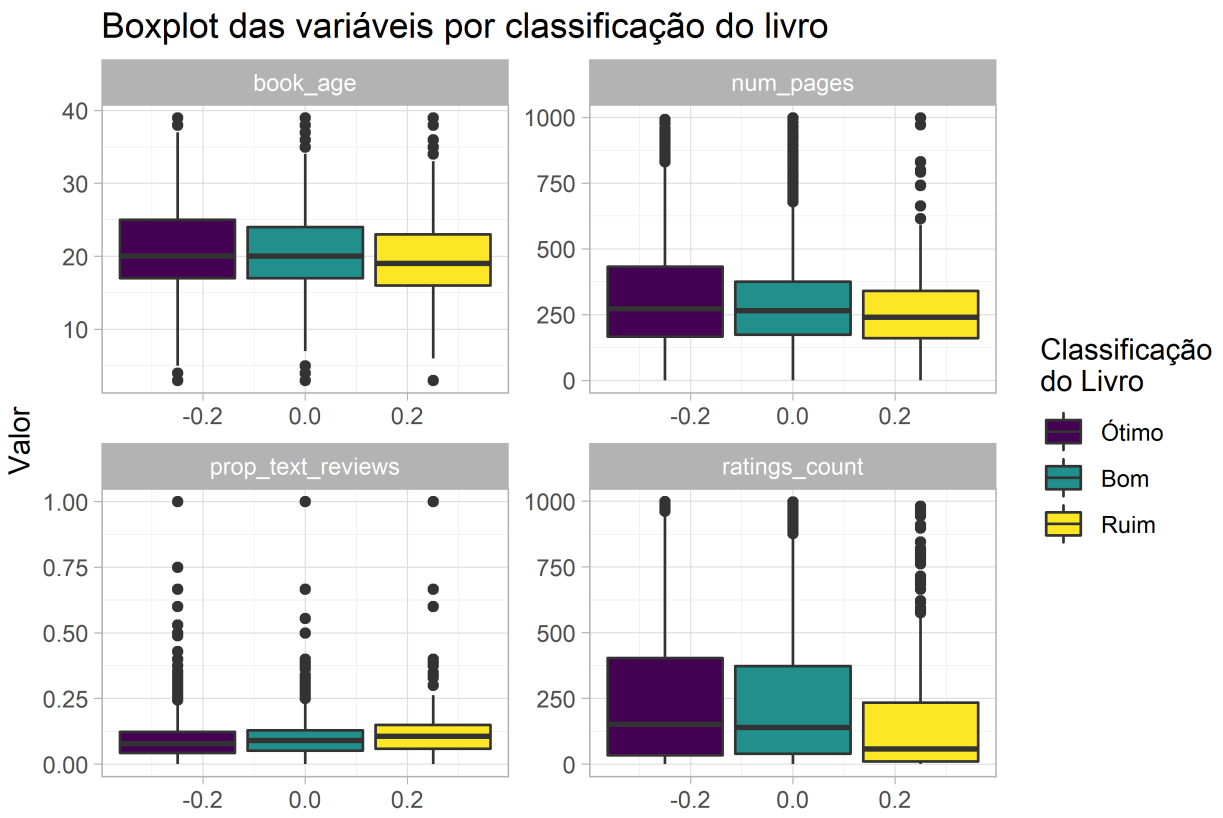
```
  labs(title = "Histograma book_age",
       x="",
       y="")
,
livros_treino %>%
  ggplot(aes(x=num_pages)) +
  geom_histogram(bins=30)+
  geom_vline(xintercept = quantile(livros_treino$num_pages),
             color="green", lty=2)+
  labs(title = "Histograma num_pages",
       x="",
       y="")
,
livros_treino %>%
  ggplot(aes(x=ratings_count)) +
  geom_histogram(bins=30)+
  geom_vline(xintercept = quantile(livros_treino$ratings_count),
             color="green", lty=2)+
  labs(title = "Histograma ratings_count",
       x="",
       y="")
,
livros_treino %>%
  ggplot(aes(x=prop_text_reviews)) +
  geom_histogram(bins=30)+
  geom_vline(xintercept = quantile(livros_treino$prop_text_reviews, na.rm = TRUE),
             color="green", lty=2)+
  labs(title = "Histograma prop_text_reviews",
       x="",
       y="")
)
```



Com os histogramas das variáveis númericas percebemos que a maioria tem uma assimetria a direita, então com isso decidimos aplicar filtros nessas variáveis para diminuir os outliers dos nossos dados. A única variável em que o filtro não será aplicado por não ter a assimetria será a `ratings_count`.

E agora o novo boxplot com os filtros aplicados:

```
livros_treino %>%
  filter(book_age<40) %>%
  filter(num_pages<1000) %>%
  filter(ratings_count<1000) %>%
  select(where(is.numeric),book_rating) %>%
  pivot_longer(-book_rating) %>%
  ggplot(.,aes(fill = book_rating)) +
  geom_boxplot(aes(y=value)) +
  facet_wrap(~ name, scales = "free") +
  labs(x="",
       y="Valor",
       fill = "Classificação\ndo Livro",
       title = "Boxplot das variáveis por classificação do livro")+
  scale_fill_viridis_d()
```



Com as mudanças feitas, houve uma diferença notável nas distribuições da idade do livro, número de páginas e avaliações, as amplitudes e variações em comparação aos box-plots anteriores. As contagens de avaliações, na classificação ótimas a amplitude e a variação é maior que as demais, se assemelha a classificação "Bom", "Ruim" é a que possui mais outliers mas sua amplitude e vaiação é a menor.

Modelagem

Considerando as alterações no conjunto de dados, após a análise exploratória, será necessário carregar novamente o conjunto de dados e gerar novo conjunto de treino e teste:

```
livros <- read.csv("./Conjunto de Dados/books_t.csv",
                   encoding = "UTF-8") %>%
  mutate(publication_date=as.Date(publication_date),
         prop_text_reviews = text_reviews_count / ratings_count,
         prop_text_reviews = ifelse(prop_text_reviews %in% c(NaN,Inf), 0, prop_text_reviews),
         book_rating=factor(book_rating,
                            levels = c("Ótimo","Bom","Ruim"))) %>%
  select(-month_publication, -year_publication, -text_reviews_count) %>%
  filter(book_age<40) %>%
  filter(num_pages<1000) %>%
  filter(ratings_count<1000)

set.seed(1904, kind = "Mersenne-Twister", normal.kind = "Inversion")
livros_split <- initial_split(livros, prop = .75, strata = book_rating)
```

```
livros_treino <- training(livros_split)

livros_teste <- testing(livros_split)

#####Criando Folds#####

set.seed(1989)
(livros_folds <- vfold_cv(livros_treino, strata = book_rating, v=10))
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 2
##    splits             id
##    <list>             <chr>
##  1 <split [3764/420]> Fold01
##  2 <split [3765/419]> Fold02
##  3 <split [3765/419]> Fold03
##  4 <split [3766/418]> Fold04
##  5 <split [3766/418]> Fold05
##  6 <split [3766/418]> Fold06
##  7 <split [3766/418]> Fold07
##  8 <split [3766/418]> Fold08
##  9 <split [3766/418]> Fold09
## 10 <split [3766/418]> Fold10
```

```
#####Criando Métricas#####

(livros_metricas <- metric_set(accuracy, roc_auc, mn_log_loss))
```

```
## # A tibble: 3 x 3
##   metric      class        direction
##   <chr>       <chr>        <chr>
## 1 accuracy    class_metric maximize
## 2 roc_auc     prob_metric  maximize
## 3 mn_log_loss prob_metric  minimize
```

Para *tunar* o modelo, além de criar os folds é necessário também estabelecer as métricas, isso porque enquanto um método gradiente ele também busca maximizar o desempenho do algoritmo e apresentar maior acerto na predição.

Para isso, é necessário acompanhar as métricas de acordo com a capacidade de classificação do modelo (com a acurácia), a probabilidade de identificar verdadeiros positivos e negativos (com acurva ROC), e a taxa de aprendizagem do algoritmo (acompanhada pelo *log_loss*).

Pre-processamento dos Dados

```
livros_rec <- recipe(book_rating ~ ., data = livros_treino) %>%
  themis::step_downsample(book_rating) %>%
  step_date(publication_date, features = c("month"),
            keep_original_cols = FALSE) %>%
  step_dummy(all_nominal_predictors(), one_hot = TRUE) %>%
  step_zv(all_predictors()) %>%
```

```
  prep()

head(prep(livros_rec) %>%
       bake(new_data =NULL))
```

```
## # A tibble: 6 x 19
##   num_pages ratings_count book_age prop_text_revie~ book_rating language_code_E~
##       <int>         <int>    <int>            <dbl> <fct>                   <dbl>
## 1       203           997       27           0.0140 Ótimo                       1
## 2       368             0       34           0      Ótimo                       1
## 3       384             3       22           0      Ótimo                       0
## 4       544           901       22           0.0400 Ótimo                       1
## 5       256           273       16           0.0366 Ótimo                       1
## 6         0            45       35           0.0222 Ótimo                       1
## # ... with 13 more variables: language_code_Other <dbl>,
## #   publication_date_month_jan <dbl>, publication_date_month_fev <dbl>,
## #   publication_date_month_mar <dbl>, publication_date_month_abr <dbl>,
## #   publication_date_month_mai <dbl>, publication_date_month_jun <dbl>,
## #   publication_date_month_jul <dbl>, publication_date_month_ago <dbl>,
## #   publication_date_month_set <dbl>, publication_date_month_out <dbl>,
## #   publication_date_month_nov <dbl>, publication_date_month_dez <dbl>
```

O pré-processamento de dados desempenha papel importante no *XGboost*, mas necessita de menos atenção do que outros algoritmos de classificação, isso porque ele desempenha melhor com dados esparsos do que outros, sendo inclusive capaz de desempenhar mesmo com dados faltantes e sem normalizar os dados.

No entanto, assim como outros métodos de aprendizagem supervisionado, ele apresenta melhor desempenho quando as observações estão balanceadas, ou seja, com a mesma quantidade de observações por categoria, motivo pelo qual optamos por incluir o `step_downsample()` nessa etapa, e dado que temos muitas variáveis *dummy*, devido escolha de usarmos o mês como uma variável preditora, o `step_zv()` ajudou a "limpar" as variáveis que contém apenas um único valor.

Grid de Procura, Tune e Parada antecipada

```
stopping_spec <-
  boost_tree(
    trees = 500,
    mtry = tune(),
    learn_rate = tune(),
    stop_iter = tune()
  ) %>%
  set_engine("xgboost", validation = 0.2) %>%
  set_mode("classification")

stopping_grid <-
  grid_latin_hypercube(
    mtry(range = c(5L, 18L)),
    learn_rate(range = c(-5, -1)),
    stop_iter(range = c(10L, 50L)),
    size = 10
  )
```

```
early_stop_wf <- workflow(livros_rec, stopping_spec)

doParallel::registerDoParallel()
set.seed(2022)
stopping_rs <- tune_grid(
  early_stop_wf,
  livros_folds,
  grid = stopping_grid,
  metrics = livros_metricas
)
```

Na etapa de definir o *grid de procura*, *fluxo de trabalho* e *tuning* do modelo, tivemos algumas diferenciações quanto ao conteúdo visto em sala. O primeiro deles foi a *taxa de aprendizagem*, que já vinhamos tendo que lidar com ela desde o momento de definir as métricas, e *critério de parada*.

Como o *XGboost* apresenta melhor desempenho que os demais algoritmos, como pode ser observado abaixo:

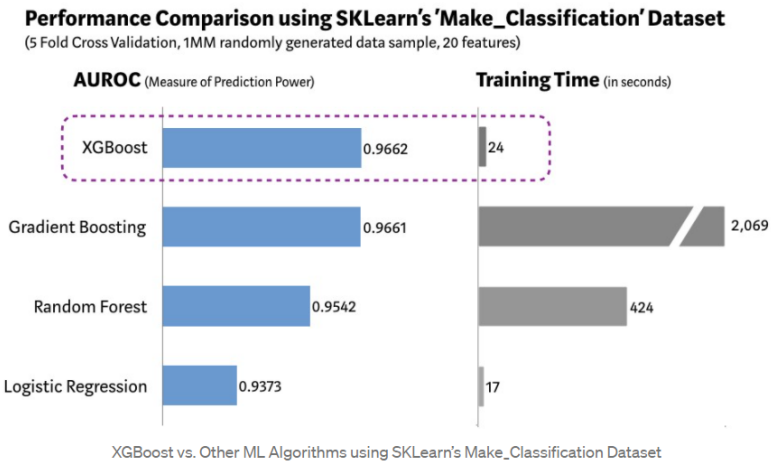


Figura 1: Vishal Morde, 2019 - XGBoost Algorithm: Long May She Reign!

Foi necessário compreender de onde vem essa eficiência, e basicamente é devido a capacidade de aprendizagem de modelo, *mas como o modelo aprende e como acompanhar esse aprendizado*?

No momento de definir os parâmetros da árvore de decisão é necessário definir a quantidade de ramificações (`trees`) que permitam que o modelo aprenda bem, optamos por 500 pois consideramos uma quantidade grande o suficiente, sem precisar tunar este parâmetro.

No entanto, é necessário definir os parametros de taxa de aprendizagem, usualmente bons parametros de aprendizagem ficam abaixo de 0.1, de forma que escolhemos 5 niveis para turnar este parametro, assim como valores de 10 a 50 para definir a quantidade de interações antes do critério de parada, essas interações são justamente as interações durante o processo de "ramificação" da árvore de decisão (não é a quantidade de ramos, mas quantas interações ele realiza na escolha dos ramos, dado que como o algoritmo atua em *paralelização*, ele pode tomar mais de uma decisão quanto a "*árvore*" ao mesmo tempo).

Com relação ao grid de procura, ao invés de utilizarmos um grid regular, conforme visto na disciplina, optamos por um irregular que tentasse "varrer" de forma mais eficiente e mesclada as possibilidade de parâmetro, definindo 10 níveis, uma vez que estipulamos 10 folds, assim conseguiriamos para cada fold testar um nível da busca dos parametros. Como resultado teriamos aproximadamente 50 modelos (5 níveis de taxa de aprendizado x 10 folds) para testar nossos parametros.

Nosso `workflow` carregou a receita e nossas especificações da árvore de decisão, com todos os passos anteriores culminando no `tune` do modelo com todas as especificações, busca de parametros e critérios de parada, que são necessários pois do contrário o *XGBoost* ficará indefinidamente buscando o melhor modelo, sendo que a partir do momento em que a capacidade de aprendizagem diminui (a taxa de `learn_rate` aumenta), ele pode parar.

Ele consegue verificar se está escolhendo modelos melhores ou piores comparando cada modelo gerado com o seu anterior, isso porque na definição da `boost_tree` ao definirmos `set_engine("xgboost", validation = 0.2)`estamos reservando 20% do conjunto de treino em cada fold para conferir os acertos, é como se fosse estivessemos definindo um novo conjunto de teste dentro do conjunto de treino,

Dessa forma conseguimos responder:

1. *Como o modelo aprende* ? Verificando a capacidade de acerto comparado ao modelo anterior,
2. *Como acompanhar esse aprendizado* ? Por meio da métricas de *mn_log_loss* , enquanto o valor estiver diminuindo o algoritmo estará aprendendo, mas quando o valor de *mn_log_loss* aumentar com relação ao modelo anterior, então a capacidade de aprendizagem estará diminuindo.

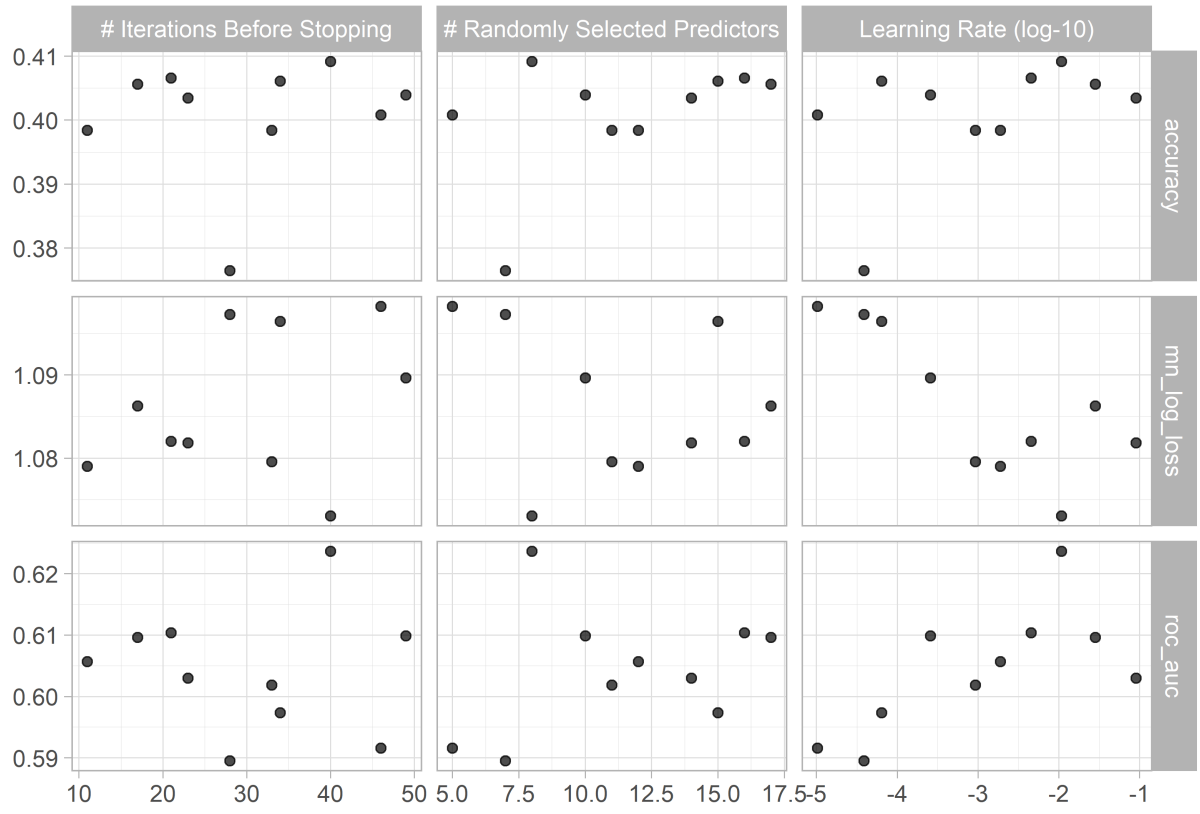
Sendo assim, podemos definir que o melhor modelo será aquele que apresentar a combinação das três métricas, com as maiores médias de *roc_auc* e *accuracy* e a menor taxa de *mn_log_loss*.

Além disso, a vantagem dos passos descritos anteriormente e o que se torna um diferencial do *XGboost* é que a combinação dos passos descritos anteriormente fazem com que o algoritmo evite o super ajuste (*overfiting*) do modelo ao conjunto de treino.

Avaliação do Modelo

Com o plot das métricas do modelo, focamos apenas na `mn_log_loss` onde nos mostra o comportamento de apredizagem do modelo classificando o momento de parada, pois assim que era percebe que após testar o modelo não há mais avanços na aprendizagem ele para.

```
autoplot(stopping_rs)
```




```
show_best(stopping_rs, metric = "mn_log_loss")
```

```
## # A tibble: 5 x 9
##    mtry learn_rate stop_iter .metric     .estimator  mean     n std_err .config
##   <int>      <dbl>     <int> <chr>       <chr>      <dbl> <int>   <dbl> <chr>
## 1     8   0.0108          40 mn_log_loss multiclass  1.07    10 0.00451 Preproc~
## 2    12   0.00189         11 mn_log_loss multiclass  1.08    10 0.00407 Preproc~
## 3    11   0.000933        33 mn_log_loss multiclass  1.08    10 0.00281 Preproc~
## 4    14   0.0910          23 mn_log_loss multiclass  1.08    10 0.00344 Preproc~
## 5    16   0.00455         21 mn_log_loss multiclass  1.08    10 0.00481 Preproc~
```

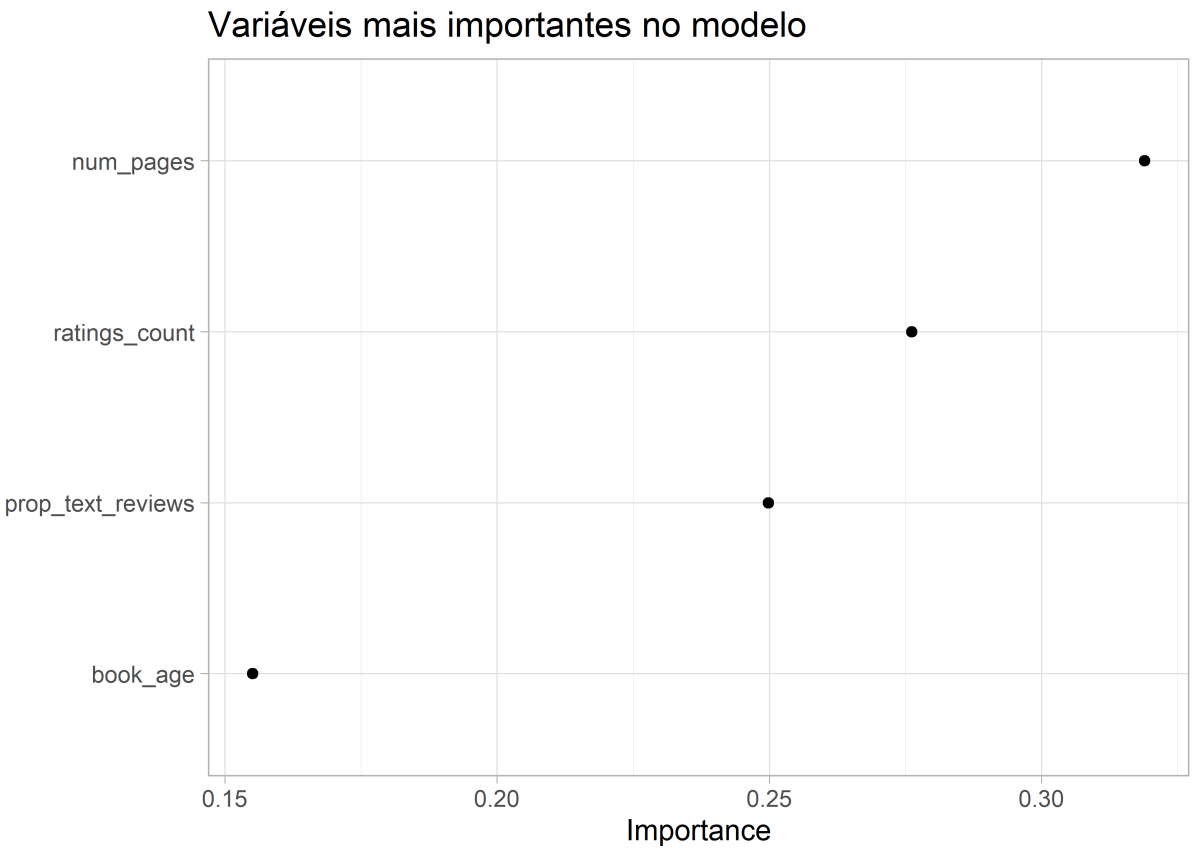
```
stopping_fit <- early_stop_wf %>%
  finalize_workflow(select_best(stopping_rs, "roc_auc")) %>%
  last_fit(livros_split)

stopping_fit
```

```
## # Resampling results
## # Manual resampling
## # A tibble: 1 x 6
##   splits              id              .metrics  .notes   .predictions .workflow
##   <list>              <chr>           <list>    <list>   <list>       <list>
## 1 <split [4184/1396]> train/test split <tibble ~ <tibble~ <tibble [1,~ <workflo~
```

```
collect_metrics(stopping_fit)
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy multiclass     0.418 Preprocessor1_Model1
## 2 roc_auc  hand_till      0.602 Preprocessor1_Model1
```

```
extract_workflow(stopping_fit) %>%
  extract_fit_parsnip() %>%
  vip(num_features = 15, geom = "point")+
  ggtitle("Variáveis mais importantes no modelo")
```



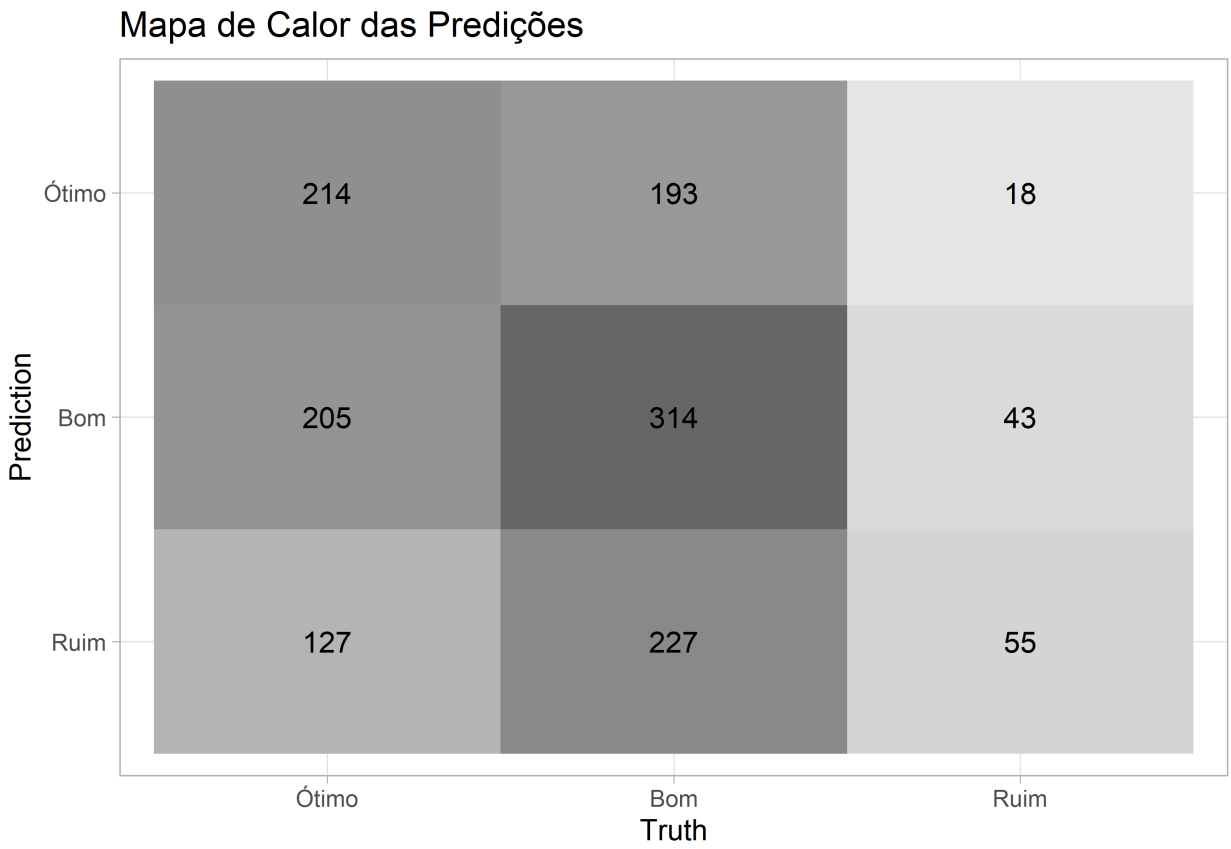
As variáveis mais importantes para o modelo são os número de páginas, contagem de avaliações, comentários de textos e a idade do livro. O que faz sentido, pois a junção dessas variáveis para fazer uma classificação é o esperado.

É difícil ter muitos livros com muitas páginas, conseguir seguir uma linha de raciocínio e uma trama na qual prenda o leitor. Além disso, quanto mais páginas provavelmente mais caro será o livro.

Principalmente, atualmente, a questão de um livro está sendo muito avaliado, muito divulgado nas redes, faz com que mais pessoas queiram consumir eles, tanto pela curiosidade de saber por que ele é tão bem avaliado e descutido.

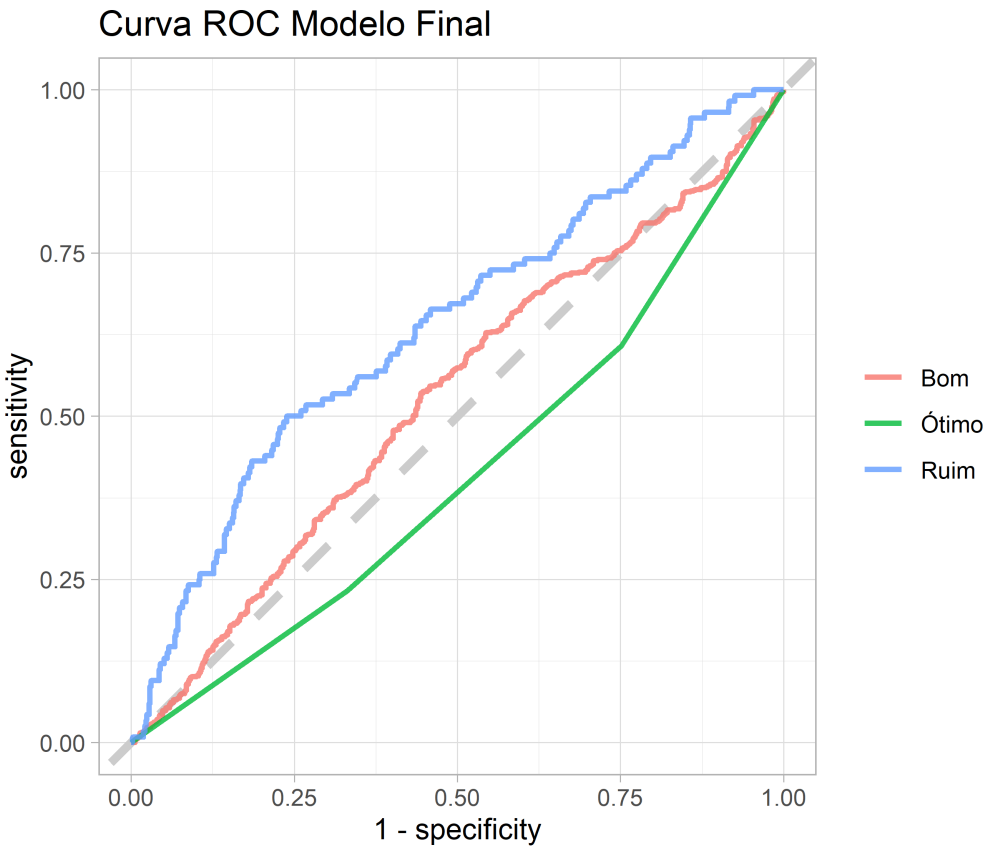
Em relação a idade do livro é um fator interessante, há os livros que se tornam clássicos, os que são deixados de lado e os que é possível fazer sucesso mesmo com um certo tempo de publicação.

```
collect_predictions(stopping_fit) %>%
  conf_mat(book_rating, .pred_class) %>%
  autoplot(type = "heatmap")+
  ggtitle("Mapa de Calor das Predições")
```



O modelo não foi o melhor, principalmente para avaliar "Ótimo" e "Bom". Porque dos livros classificados como "Ótimo", o modelo classificou 175 como "Bom" e 152 como "Ruim". Dos livros classificados como "Bom", o modelo classificou 211 como "Ótimo" e 256 como "Ruim".

```
collect_predictions(stopping_fit, summarize = FALSE) %>%
  roc_curve(book_rating, .pred_class:.pred_Ruim) %>%
  ggplot(aes(1 - specificity, sensitivity, color = .level)) +
  geom_abline(lty = 2, color = "gray80", size = 1.5) +
  geom_path(alpha = 0.8, size = 1) +
  coord_equal() +
  labs(color = NULL,
       title =  "Curva ROC Modelo Final")
```



O que percebemos olhando a curva ROC é que o modelo não está predizendo bem e está cometendo erros em falsos positvos e negativos. Tanto a categoria "Bom" e "Ruim" estão com curvas próximas, mostrando que ele está acertando mais que errando essas classificações, mas a sensibilidade está em torno de 0.5 a 0.6. A categoria "Ótimo" pela curva mostra que a probabilidade do modelo errar é maior que ele acertar, por isso a curva está abaixo.

```
## Verdadeiro positivo
collect_predictions(stopping_fit) %>%
  sens(book_rating, .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    macro          0.431
```

```
## Verdadeiro negativo
collect_predictions(stopping_fit) %>%
  spec(book_rating, .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    macro          0.700
```

```
collect_predictions(stopping_fit, summarize = FALSE) %>%
  roc_curve(book_rating, .pred_class:.pred_Ruim) %>%
  group_by(.level) %>%
  summarise(mean_sens=mean(sensitivity),
            mean_spec=mean(specificity))
```

```
## # A tibble: 3 x 3
##   .level mean_sens mean_spec
##   <chr>      <dbl>     <dbl>
## 1 Bom        0.514     0.515
## 2 Ótimo      0.568     0.383
## 3 Ruim       0.628     0.511
```

Pelas métricas observamos que a sensibilidade para ótimo está em 0.43, o que mostra que o modelo está acertando a categoria "ótimo" em 43% dos casos, que é baixo. Pela sensibilidade que é 0.7, o modelo está classificando como "ótimo" sendo que na verdade é "bom" e "ruim" em 70% dos casos, o que é um valor bem alto.

Conclusões

O modelo não gerou os resultados esperados, mas visto que o a parte da engenharia dos dados é um fator importante, provavelmente isso prejudicou o desempenho, já que foi a parte mais trabalhosa.

Sendo assim, algumas mudanças poderiam melhorar o modelo, tais como:

1. uma melhora na limpeza dos dados, removendo ainda mais outliers;
2. melhor definição nos níveis e categorias das variáveis preditoras;
3. variáveis preditoras mais informativas e com mais distinção entre os níveis.

Por fim, não consideramos satisfatório a capacidade preditora do modelo e concluímos que a melhor forma de saber se um livro é ótimo, bom ou ruim, é lendo-o.

Referências:

- MORDE, Vishal. **XGBoost Algorithm: Long May She Reign!** - Abril, 2019. Publicado em *Towards Data Science*. Disponível em: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d
- SILGE, Julia. **Tune xgboost models with early stopping to predict shelter animal status** - Agosto, 2021. Publicado em Julia Silge. Disponível em: https://juliasilge.com/blog/shelter-animals/
- R Core Team (2021). **R:*A language and environment for statistical computing*** . R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
- SOUMIK . **Goodreads-books *comprehensive list of books listed in goodreads*** - Maio, 2019. Publicado em Kagle. Disponível em: https://www.kaggle.com/jealousleopard/goodreadsbooks