Tarea 8: regularización y selección de modelos

En este ejemplo hacemos *análisis de sentimiento*, intentanto predecir si reseñas de películas son positivas o negativas a partir del texto de las reseñas. En este ejemplo veremos un enfoque relativamente simple, que consiste en considerar solamente las palabras que contienen las reseñas, sin tomar en cuenta el orden (el modelo de bolsa de palabras o *bag of words*).

Usaremos regresión logística regularizada.

Feature engineering básico

Hay muchas maneras de preprocesar los datos para obtener variables numéricas a partir del texto. En este caso simplemente tomamos las palabras que ocurren más frecuentemente.

- Encontramos las 3000 palabras más frecuentes sobre todos los textos, por ejemplo. Estas palabras son nuestro **vocabulario**.
- Registramos en qué documentos ocurre cada una de esas palabras.
- Cada palabra es una columna de nuestros datos, el valor es 1 si la palabra ocurre en documento y 0 si no ocurre.

Por ejemplo, para el texto "Un gato blanco, un gato negro", "un perro juega", "un astronauta juega" quedarían los datos:

texto_id	un	gato	negro	blanco	perro	juega
texto_1	1	1	1	1	0	0
texto_2	1	0	0	0	1	0
texto_3	1	0	0	0	0	1

Nótese que la palabra *astronauta* no está en nuestro vocabulario para este ejemplo.

Hay varias opciones para tener mejores variables, que pueden o no ayudar en este problema (no las exploramos en este ejercicio):

- Usar conteos de frecuencias de ocurrencia de palabras en cada documento, o usar log(1+ conteo), en lugar de 0-1's
- Usar palabras frecuentes, pero quitar las que son *stopwords*, como son preposiciones y artículos entre otras, pues no tienen significado: en inglés, por ejemplo, *so, is, then, the, a*, etc.
- Lematizar palabras: por ejemplo, contar en la misma categoría *movie* y *movies*, o *funny* y *funniest*, etc.
- Usar indicadores binarios si la palabra ocurre o no en lugar de la frecuencia
- Usar frecuencias ponderadas por qué tan rara es una palabra sobre todos los documentos (frecuencia inversa sobre documentos)
- Usar pares de palabras en lugar de palabras sueltas: por ejemplo: juntar "not" con la palabra que sigue (en lugar de usar *not* y *bad* por separado, juntar en una palabra *not_bad*),
- Usar técnicas de reducción de dimensionalidad que considera la co-ocurrencia de palabras (veremos más adelante en el curso).
- Muchas otras

Datos y preprocesamiento

Los textos originales los puedes encontrarlos en la carpeta *datos/sentiment*. Están en archivos individuales que tenemos que leer. Podemos hacer lo que sigue:

```
library(tidyverse)
library(tidymodels)
# puedes necesitar el siguiente paquete
# install.packages("textrecipes")
# install.packages("stopwords")
nombres_neg <- list.files("sentiment/neg", full.names = TRUE)
nombres_pos <- list.files("sentiment/pos", full.names = TRUE)
# positivo
textos_pos <- tibble(texto = map_chr(nombres_pos, read_file), polaridad = "pos")
textos_neg <- tibble(texto = map_chr(nombres_neg, read_file), polaridad = "neg")
textos <- bind_rows(textos_pos, textos_neg) %>%
  mutate(polaridad = factor(polaridad, levels = c("pos", "neg")))
nrow(textos)
```

```
## [1] 2000
```

```
table(textos$polaridad)
```

```
##
##  pos  neg
## 1000 1000
```

Y un fragmento del primer texto:

```
str_sub(textos$texto[[5]], 1, 300)
```

```
## [1] "moviemaking is a lot like being the general manager of an nfl team in the post-salary cap era -
```

```
set.seed(83224)
polaridad_particion <- initial_split(textos, 0.7)
textos_ent <- training(polaridad_particion)
textos_pr <- testing(polaridad_particion)
nrow(textos_ent)
```

```
## [1] 1400
```

```
# install.packages("textrecipes")
# install.packages("stopwords")
library(textrecipes)
receta_polaridad <- recipe(polaridad ~ ., textos_ent) %>%
  step_relevel(polaridad, ref_level = "neg", skip = TRUE) %>%
  step_mutate(texto = str_remove_all(texto, "[_()]")) %>%
  step_mutate(texto = str_remove_all(texto, "[0-9]*")) %>%
  step_tokenize(texto) %>% # separar por palabras
  step_stopwords(texto) %>%
  step_tokenfilter(texto, max_tokens = 6000) %>%
  step_tf(texto, weight_scheme = "raw count")

# en el prep se separa en palabras, se eliminan stopwords,
# se filtran los de menor frecuencia y se crean las variables
# 0 - 1 que discutimos arriba, todo con los textos de entrenamiento
receta_prep <- receta_polaridad %>% prep()
```

Los términos seleccionados (el vocabulario) están aquí (una muestra)

```
receta_prep$term_info %>% sample_n(30)
```

```
## # A tibble: 30 x 4
##    variable                 type    role      source
##    <chr>                    <chr>   <chr>     <chr>
##  1 tf_texto_seriousness     numeric predictor derived
##  2 tf_texto_magnificent     numeric predictor derived
##  3 tf_texto_straightforward numeric predictor derived
##  4 tf_texto_capture         numeric predictor derived
##  5 tf_texto_suffers         numeric predictor derived
##  6 tf_texto_classics        numeric predictor derived
##  7 tf_texto_placement       numeric predictor derived
##  8 tf_texto_president       numeric predictor derived
##  9 tf_texto_resembles       numeric predictor derived
## 10 tf_texto_mate            numeric predictor derived
## # ... with 20 more rows
```

El tamaño de la matriz que usaremos para regresión logística tiene 1600 renglones (textos) por 3000 columnas de términos:

```
mat_textos_entrena <- juice(receta_prep)
dim(mat_textos_entrena)
```

```
## [1] 1400 6001
```

```
head(mat_textos_entrena)
```

```
## # A tibble: 6 x 6,001
##   polaridad tf_texto_aaron tf_texto_abandon tf_texto_abando~ tf_texto_abilit~
##   <fct>              <dbl>            <dbl>            <dbl>            <dbl>
## 1 pos                    0                0                0                0
## 2 pos                    0                0                0                0
## 3 pos                    0                0                0                0
## 4 pos                    0                0                0                0
## 5 pos                    0                0                0                0
## 6 pos                    0                0                0                0
## # ... with 5,996 more variables: tf_texto_ability <dbl>, tf_texto_able <dbl>,
## #   tf_texto_aboard <dbl>, tf_texto_absence <dbl>, tf_texto_absent <dbl>,
## #   tf_texto_absolute <dbl>, tf_texto_absolutely <dbl>,
## #   tf_texto_absorbed <dbl>, tf_texto_absorbing <dbl>, tf_texto_absurd <dbl>,
## #   tf_texto_abuse <dbl>, tf_texto_abusive <dbl>, tf_texto_abyss <dbl>,
## #   tf_texto_academy <dbl>, tf_texto_accent <dbl>, tf_texto_accents <dbl>,
## #   tf_texto_accept <dbl>, tf_texto_acceptable <dbl>, tf_texto_accepted <dbl>,
## #   tf_texto_accepting <dbl>, tf_texto_accepts <dbl>, tf_texto_access <dbl>,
## #   tf_texto_accident <dbl>, tf_texto_accidentally <dbl>,
## #   tf_texto_acclaimed <dbl>, tf_texto_accompanied <dbl>,
## #   tf_texto_accompanying <dbl>, tf_texto_accomplish <dbl>,
## #   tf_texto_accomplished <dbl>, tf_texto_accomplishment <dbl>,
## #   tf_texto_according <dbl>, tf_texto_account <dbl>, tf_texto_accurate <dbl>,
## #   tf_texto_accused <dbl>, tf_texto_ace <dbl>, tf_texto_achieve <dbl>,
## #   tf_texto_achieved <dbl>, tf_texto_achievement <dbl>,
## #   tf_texto_achieves <dbl>, tf_texto_acid <dbl>, tf_texto_across <dbl>,
## #   tf_texto_act <dbl>, tf_texto_acted <dbl>, tf_texto_acting <dbl>,
## #   tf_texto_action <dbl>, tf_texto_actions <dbl>, tf_texto_active <dbl>,
## #   tf_texto_activities <dbl>, tf_texto_activity <dbl>, tf_texto_actor <dbl>,
```

```
## #   `tf_texto_actor's` <dbl>, tf_texto_actors <dbl>, tf_texto_actress <dbl>,
## #   tf_texto_actresses <dbl>, tf_texto_acts <dbl>, tf_texto_actual <dbl>,
## #   tf_texto_actually <dbl>, tf_texto_ad <dbl>, tf_texto_adam <dbl>,
## #   tf_texto_adams <dbl>, tf_texto_adaptation <dbl>,
## #   tf_texto_adaptations <dbl>, tf_texto_adapted <dbl>, tf_texto_add <dbl>,
## #   tf_texto_added <dbl>, tf_texto_addict <dbl>, tf_texto_addicted <dbl>,
## #   tf_texto_adding <dbl>, tf_texto_addition <dbl>, tf_texto_addresses <dbl>,
## #   tf_texto_adds <dbl>, tf_texto_adequate <dbl>, tf_texto_admirable <dbl>,
## #   tf_texto_admirably <dbl>, tf_texto_admiration <dbl>, tf_texto_admire <dbl>,
## #   tf_texto_admission <dbl>, tf_texto_admit <dbl>, tf_texto_admits <dbl>,
## #   tf_texto_admittedly <dbl>, tf_texto_adolescent <dbl>,
## #   tf_texto_adopts <dbl>, tf_texto_adorable <dbl>, tf_texto_ads <dbl>,
## #   tf_texto_adult <dbl>, tf_texto_adults <dbl>, tf_texto_advance <dbl>,
## #   tf_texto_advanced <dbl>, tf_texto_advances <dbl>, tf_texto_advantage <dbl>,
## #   tf_texto_adventure <dbl>, tf_texto_adventures <dbl>,
## #   tf_texto_advertised <dbl>, tf_texto_advice <dbl>, tf_texto_affair <dbl>,
## #   tf_texto_affairs <dbl>, tf_texto_affect <dbl>, tf_texto_affected <dbl>,
## #   tf_texto_affecting <dbl>, tf_texto_affection <dbl>, ...
```

Clasificador de textos

Ahora hacemos regresión logística con regularización ridge/lasso. La penalización es de la forma

$$\lambda((1-\alpha)\sum_j \beta_j^2 + \alpha \sum_j |\beta_j|)$$

de manera que combina ventajas de ridge (encoger juntos parámetros de variables correlacionadas) y lasso (eliminar variables que aportan poco a la predicción).

Seleccionaremos los parámetros con validación cruzada

Preguntas: 1. Calcula la devianza validación cruzada y el area bajo la curva ROC para un modelo poco regularizado (por ejemplo, $\log(\lambda) = -12$).

```
textos_pr <- testing(polaridad_particion)

modelo_baja_reg <- logistic_reg(mixture = 0.5, penalty = exp(-12)) %>%
  set_engine("glmnet") %>%
  set_args(lambda.min.ratio = 0)

flujo_textos <- workflow() %>%
  add_recipe(receta_polaridad) %>%
  add_model(modelo_baja_reg) %>%
  fit(textos_ent)

preds_baja_reg <- predict(flujo_textos, textos_pr, type = "prob") %>%
  bind_cols(textos_pr %>% select(polaridad))

preds_baja_reg %>%
  mn_log_loss(factor(polaridad), .pred_pos)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 mn_log_loss binary         0.585
```

```
predict(flujo_textos, textos_ent, type = "prob") %>%
  bind_cols(textos_ent %>% select(polaridad)) %>%
  mn_log_loss(factor(polaridad), .pred_pos)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 mn_log_loss binary      0.000627
```

```
## Cálculo ROC AUC
roc_auc(preds_baja_reg, polaridad, .pred_pos)
```

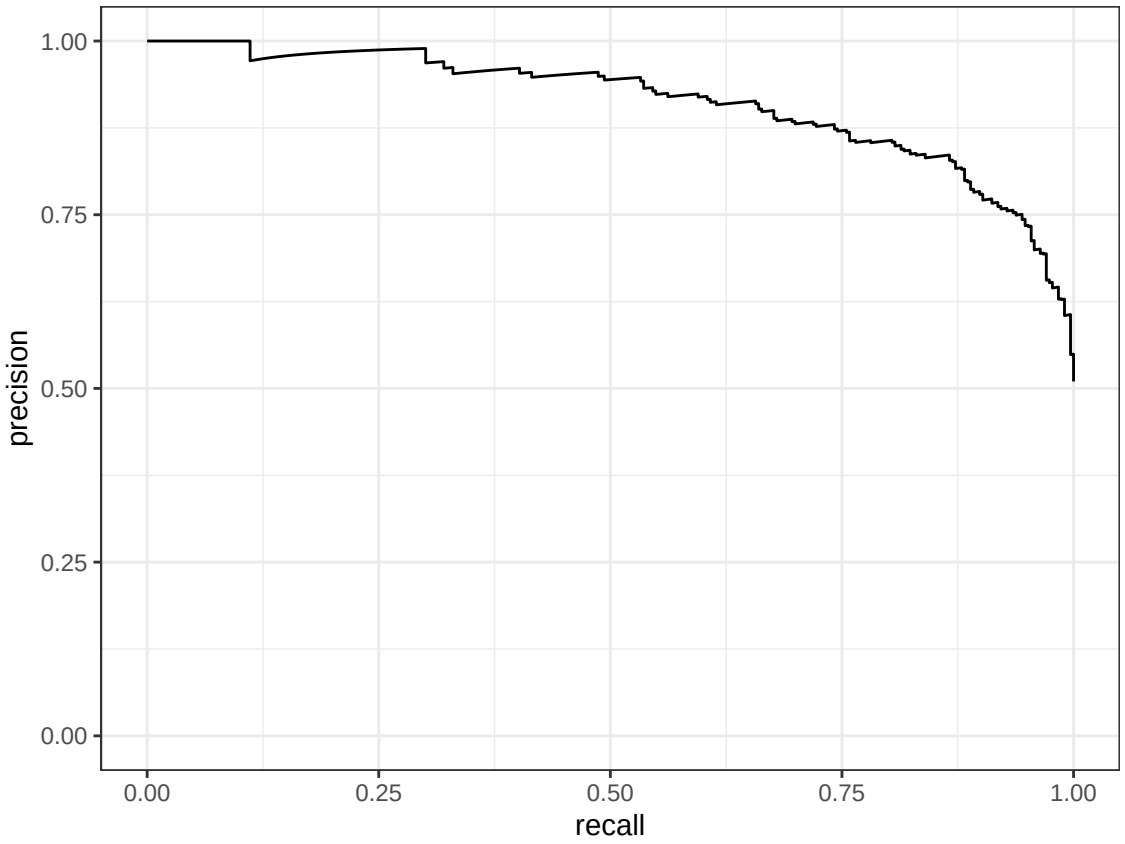
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.911
```

```
## Cálculo de devianza validación cruzada
metricas <- metric_set(mn_log_loss, roc_auc)

validacion_particion <- vfold_cv(textos_pr, v=10)

eval_vc <- flujo_textos %>%
  fit_resamples(
    resamples = validacion_particion,
    metrics = metricas
  ) %>%
  collect_metrics()

eval_vc
```

```
## # A tibble: 2 x 5
##   .metric     .estimator  mean     n std_err
##   <chr>       <chr>      <dbl> <int>   <dbl>
## 1 mn_log_loss binary     0.601    10  0.0417
## 2 roc_auc     binary     0.877    10  0.0107
```

La gráfica de precisión recall es:

```
autoplot(preds_baja_reg %>% pr_curve(polaridad, .pred_pos))
```



Pregunta: Este modelo tiene sobreajuste fuerte. ¿Por qué?

2. Selecciona un modelo con regularización mayor:

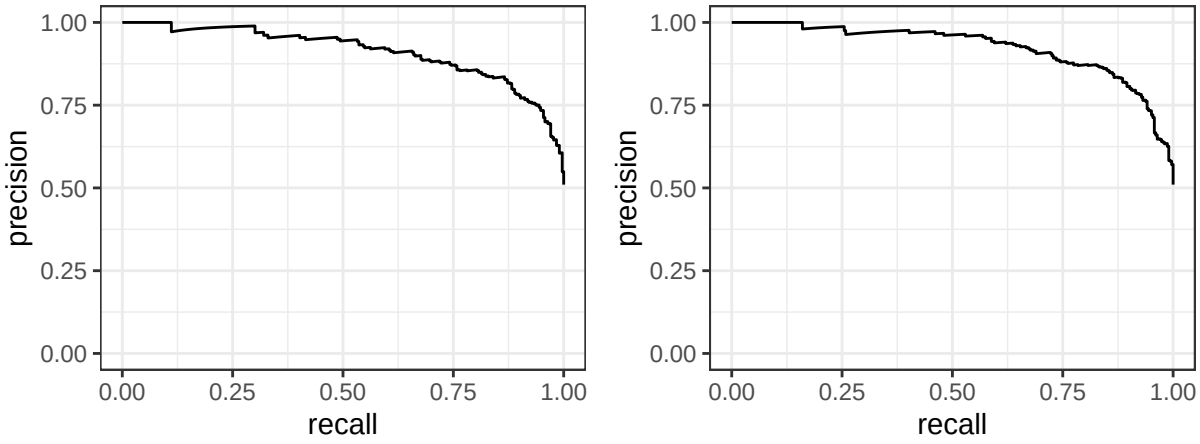
```
modelo_mas_reg <- logistic_reg(mixture = 0.5, penalty = 0.01) %>%
  set_engine("glmnet") %>%
  set_args(lambda.min.ratio = 0)

flujo_textos_alta <- workflow() %>%
  add_recipe(receta_polaridad) %>%
  add_model(modelo_mas_reg) %>%
  fit(textos_ent)

preds_alta_reg <- predict(flujo_textos_alta, textos_pr, type = "prob") %>%
  bind_cols(textos_pr %>% select(polaridad))

preds_alta_reg %>%
  mn_log_loss(polaridad, .pred_pos)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 mn_log_loss binary         0.360
```

Pregunta: 3. Grafica juntas las curvas de precisión recall. ¿Algún modelo domina a otro? compara desempeño de los dos clasificadores en pérdida logarítmica.

```
library(patchwork)
```

```
## Gráficas conjuntas de precisión-recall para los dos modelos
g1 <- autoplot(preds_baja_reg %>% pr_curve(polaridad, .pred_pos))
g2 <- autoplot(preds_alta_reg %>% pr_curve(polaridad, .pred_pos))

g1 + g2
```




```
# ## Graficar las curvas juntas
# pr_curve(preds_baja_reg, polaridad, .pred_pos)
#
# ggplot(
#
# )
```

*__Pregunta__ 4. Obtén los coeficientes de los dos modelos que comparaste arriba. Compara los coeficientes más negativos y más positivos de cada modelo.

```
# Por ejemplo:
coefs_baja <- flujo_textos %>% pull_workflow_fit() %>% tidy
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.0-2
```

```
coefs_alta <- flujo_textos_alta %>% pull_workflow_fit() %>% tidy

coefs_2mod <- bind_rows(coefs_baja, coefs_alta) %>%
  mutate(tipo = ifelse(penalty < 0.0001, "coef_reg_baja", "coef_reg_alta")) %>%
  select(term, tipo, estimate) %>%
  pivot_wider(names_from = tipo, values_from = estimate)

## Test code
# coefs_2mod %>%
  # filter(min(coef_reg_baja))
  # slice(which.min(coef_reg_alta))
  # slice(which.max(coef_reg_baja))
```

```r
## Creating table with extreme coefficients and corresponding words
tibble(
  coef_reg_baja_vals = c(min(coefs_2mod$coef_reg_baja), max(coefs_2mod$coef_reg_baja)),
  coef_reg_baja_terms = c(slice(coefs_2mod, which.min(coef_reg_baja))$term, slice(coefs_2mod, which.max
  coef_reg_alta_vals = c(min(coefs_2mod$coef_reg_alta), max(coefs_2mod$coef_reg_alta)),
  coef_reg_alta_terms = c(slice(coefs_2mod, which.min(coef_reg_alta))$term, slice(coefs_2mod, which.max
  )
```

```
## # A tibble: 2 x 4
##   coef_reg_baja_vals coef_reg_baja_terms coef_reg_alta_vals coef_reg_alta_terms
##                <dbl> <chr>                          <dbl> <chr>
## 1              -2.69 tf_texto_atrocious            -0.985 tf_texto_fares
## 2               4.31 tf_texto_slip                  1.60  tf_texto_slip
```

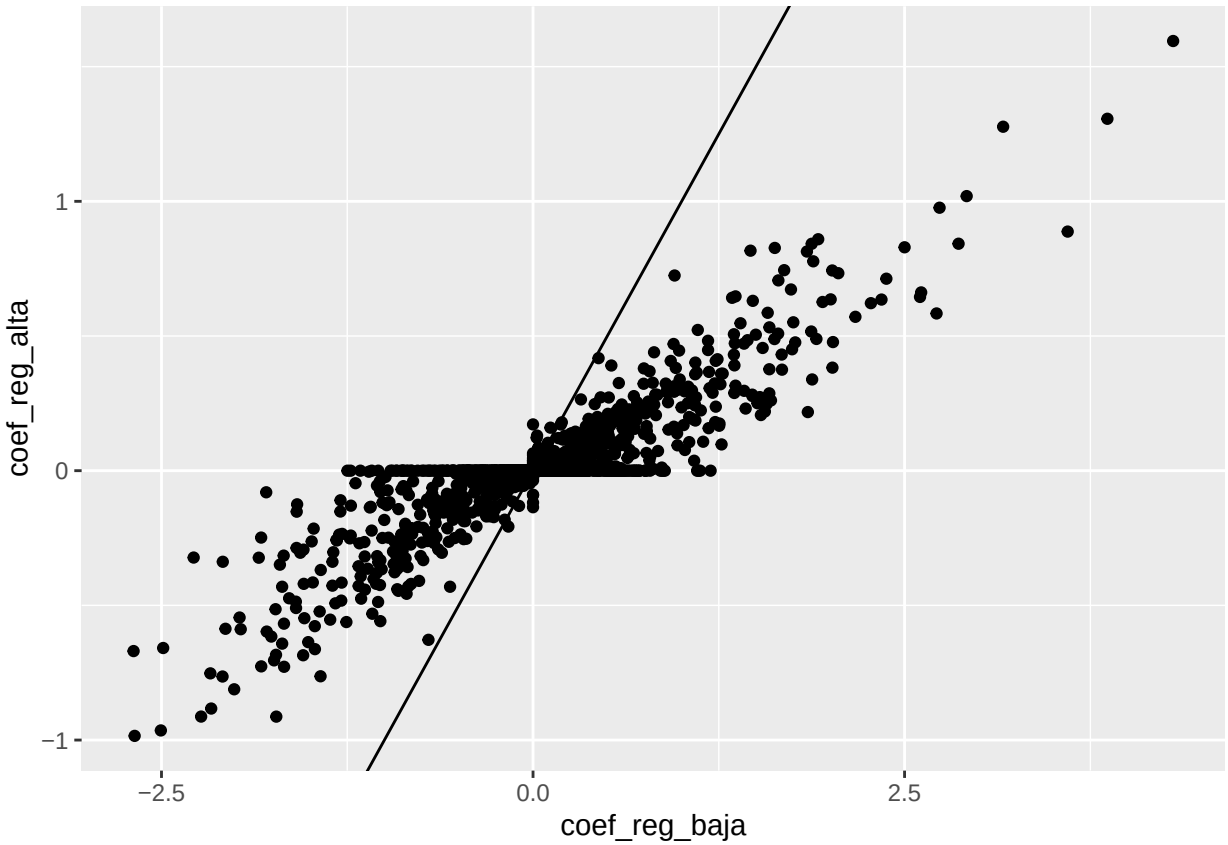
¿Cuáles tienen valores más grandes en valor absoluto?

¿Por qué?

¿Tiene sentido cuáles palabras tienen coeficiente positivo y cuáles negativo?

Pregunta : 5. Grafica coeficientes de un modelo contra los del otro modelo (agrega una recta y=x). ¿Cómo describes los patrones que ves en esa gŕafica?

```r
## Graph with both models' coefficients
ggplot(
  coefs_2mod,
  aes(x=coef_reg_baja, y=coef_reg_alta)
) +
  geom_point() +
  geom_abline()
```



Pregunta: 6. Calcula cuántas de las predicciones tienen probabilidad menor a 0.01 en el modelo sobreajustado.

```
## Número de predicciones con probabilidad menor a 0.01 en modelo con baja regulación
coefs_2mod %>%
  filter(coef_reg_baja < 0.01) %>%
  tally() %>%
  mutate(prop = n/nrow(coefs_2mod))
```

```
## # A tibble: 1 x 2
##       n  prop
##   <int> <dbl>
## 1  5361 0.893
```

```
preds_baja_reg %>%
  mutate(menor_01 = .pred_pos < 0.01) %>%
  group_by(menor_01, polaridad) %>%
  count() %>%
  group_by()
```

```
## # A tibble: 4 x 3
##   menor_01 polaridad     n
##   <lgl>    <fct>     <int>
## 1 FALSE    pos         295
## 2 FALSE    neg         129
## 3 TRUE     pos          11
## 4 TRUE     neg         165
```

Entre esas predicciones, ¿cuál es la probabilidad de que una reseña sea de hecho positiva (según la muestra de prueba)?

Describe por qué esto explica en parte que la devianza sea tan grande para el modelo no regularizado comparado con el regularizado.

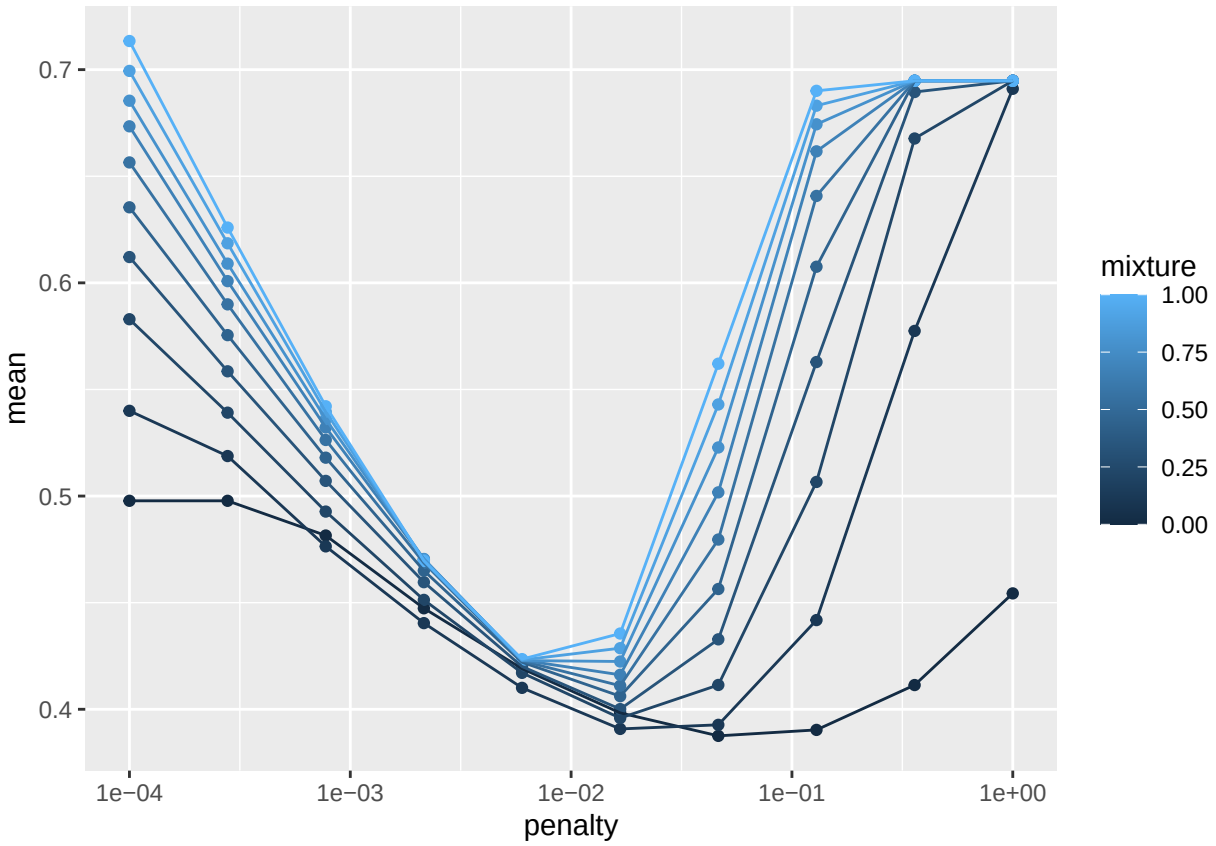
7. Afina los parámetros *mixture* y *penalty* usando validación cruzada con el siguiente código. Este proceso tarda unos minutos pues hay que ajustar varios modelos (puede paralelizarse).

```
# alpha es mixture, y lambda es penalty
modelo_reg <- logistic_reg(mixture = tune(), penalty = tune()) %>%
  set_mode("classification") %>%
  set_engine("glmnet") %>%
  set_args(lambda.min.ratio = 0)
flujo_textos <- workflow() %>%
  add_recipe(receta_polaridad) %>%
  add_model(modelo_reg)
```

```
# cortes de validación cruzada
particion_vc <- vfold_cv(textos_ent, v = 5)
# crea un grid (empieza con uno chico)
glmnet_set <- parameters(penalty(range = c(-4, 0), trans = log10_trans()),
                         mixture(range = c(0, 1)))
glmnet_grid <- grid_regular(glmnet_set, levels = 10)
# afinar
glmnet_tune <- tune_grid(flujo_textos,
            resamples = particion_vc,
            grid = glmnet_grid,
            metrics = metric_set(roc_auc, mn_log_loss))
res <- collect_metrics(glmnet_tune)
res
```

```
## # A tibble: 200 x 8
##     penalty mixture .metric     .estimator  mean     n std_err .config
##       <dbl>   <dbl> <chr>       <chr>      <dbl> <int>   <dbl> <chr>
##  1 0.0001         0 mn_log_loss binary     0.498     5 0.0207  Model001
##  2 0.0001         0 roc_auc     binary     0.902     5 0.00657 Model001
##  3 0.000278       0 mn_log_loss binary     0.498     5 0.0207  Model002
##  4 0.000278       0 roc_auc     binary     0.902     5 0.00657 Model002
##  5 0.000774       0 mn_log_loss binary     0.481     5 0.0198  Model003
##  6 0.000774       0 roc_auc     binary     0.903     5 0.00656 Model003
##  7 0.00215        0 mn_log_loss binary     0.447     5 0.0168  Model004
##  8 0.00215        0 roc_auc     binary     0.904     5 0.00626 Model004
##  9 0.00599        0 mn_log_loss binary     0.419     5 0.0144  Model005
## 10 0.00599        0 roc_auc     binary     0.906     5 0.00604 Model005
## # ... with 190 more rows
```

```
ggplot(res %>% filter(.metric == "mn_log_loss"), aes(x = penalty, y = mean, colour = mixture, group = mi
  geom_point() + geom_line() + scale_x_log10()
```



Escogemos el modelo más simple a un error estándar del mejor:

```
select_by_one_std_err(glmnet_tune, metric = "mn_log_loss", desc(penalty))
```

```
## # A tibble: 1 x 10
##   penalty mixture .metric    .estimator  mean     n std_err .config .best .bound
##     <dbl>   <dbl> <chr>      <chr>      <dbl> <int>   <dbl> <chr>   <dbl>  <dbl>
## 1   0.129       0 mn_log_lo~ binary     0.390     5 0.00785 Model0~ 0.388  0.397
```

Podemos seleccionar modelos comparables con mayor penalización lasso para obtener modelos más parsimoniosos.