Proyecto Final
Análisis de Sentimientos Amazon

Francisco Madrigal González 191874 Héctor Núñez Vega 156253
Diego Mata Mateos 150548 Daniel Castañón 157016

28/5/2021

Proyecto Final: Economía Computacional

Amazon Reviews

Instalamos la paqueterías que utilizaremos

```
library(tidytext)
library(tidyverse)
library(tm)
library(wordcloud)
library(dplyr)
library(reshape2)
library(rsample)
library(questionr)
library(fastDummies)
library(gamlr)
library(glmnet)
library(broom)
library(janitor)
library(ranger)
library(yardstick)
library(syuzhet)
library(stargazer)
library(kableExtra)
library(magrittr)
library(ggpubr)
library(xgboost)
library(purrr)
library(caret)
library(pROC)
```

Cargamos la base de datos

```
file_path <-  "D://francisco_madrigal//Desktop//ITAM//Semestre 4//economia_compu//Proyecto Final//data/
data <- read_delim(file_path, delim = " ", quote = "\"",
                   escape_backslash = T,
                   escape_double = F)
```

Tenemos una variable de tiempo (*Time*) que convertiremos en clase *date* y generamos **ID** ya que la base está a nivel reseña.

```
data$Time <- as.Date(data$Time)
data <- mutate(data, ID_review = row_number())
```

Análisis exploratorio

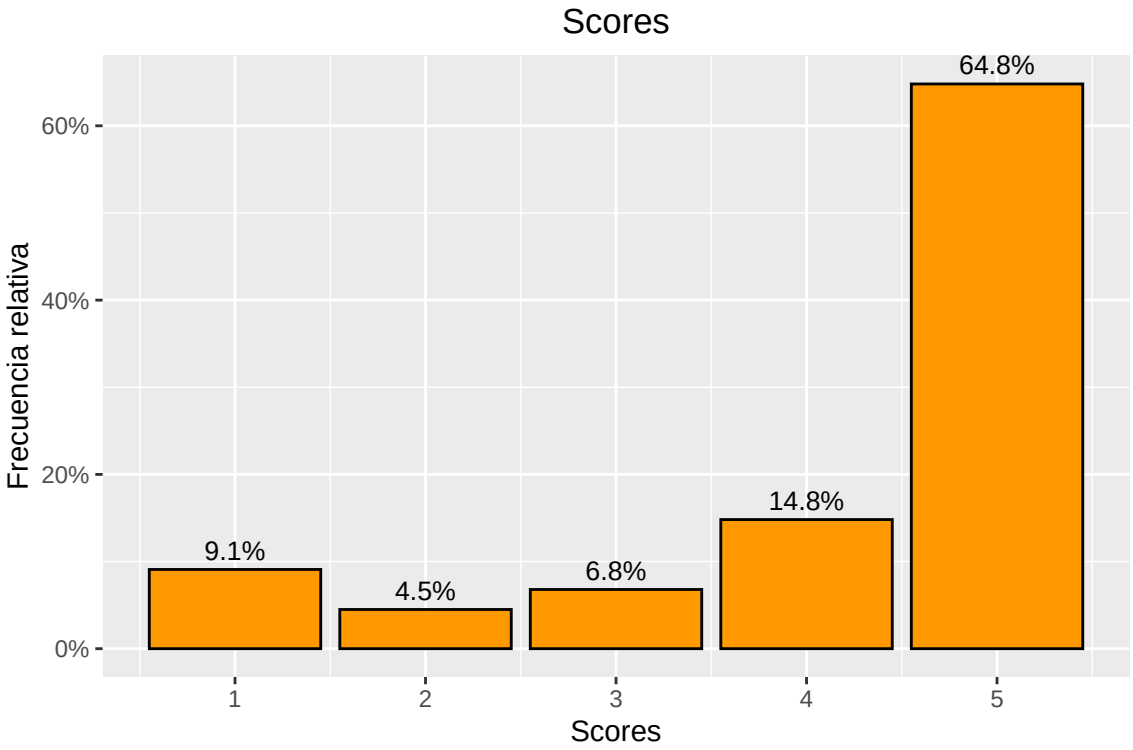
Primero damos un vistazo a la base. Vemos que tiene 13,319 observaciones y 9 columnas.

- **ProductId**: Indica el id de cada producto con alguna reseña en Amazon;
- **UserId**: Indica el id de usuario único en el portal de Amazon;
- **Score**: El score que dio el usuario al producto adquirido;
- **Time**: La fecha en que el usuario realizó la reseña;
- **Summary**: El texto de la reseña que realizó el usuario;
- **Nrev**: El número de reseñas que ha realizado el usuario;
- **Length**: El conteo de palabras de la reseña;
- **Prod_Category**: Categoría del producto, y
- **Prod_Group**: El grupo al que pertenece el producto.

```
glimpse(data)
```

Analicemos la frecuencia de los *Scores* que asignaron los usuario en la base:

```
data %>%
  count(Score) %>%
  mutate(perc = n / nrow(data)) -> data2
ggplot(data2, aes(x = Score, y = perc)) +
  geom_bar(stat = "identity",fill = "#FF9900",
           color="black") +
  xlab("Scores") +
  ylab("Frecuencia relativa") +
  ggtitle("Scores") +
  scale_y_continuous(labels=scales::percent) +
  geom_text(aes(label=scales::percent(perc)), vjust = -.5,
            color="black", size=3.5) +
  theme(plot.title = element_text(hjust = 0.5))
```



Como se observa en el histograma la frecuencia de *Scores* es:

- **Score 1:** Presenta una frecuencia relativa de 9.1% del total de reseñas.
- **Score 2:** Presenta una frecuencia relativa de 4.5% del total de reseñas.
- **Score 3:** Presenta una frecuencia relativa de 6.8% del total de reseñas.
- **Score 4:** Presenta una frecuencia relativa de 14.8% del total de reseñas.
- **Score 5:** Presenta una frecuencia relativa de 64.8% del total de reseñas.

De la misma forma, es relevante conocer la distribución del *Score* promedio dependiendo de los grupos a los que pertenece el producto:

```
product_gped <-
  data %>%
  group_by(Prod_Group) %>%
  summarise(promedio = mean(Score),
  frecuencia_relativa = (n()/nrow(data)))
product_gped$promedio <- round(product_gped$promedio ,3)
product_gped$frecuencia_relativa<- round(product_gped$frecuencia_relativa ,3)
product_gped.tabla <-
  product_gped[order(product_gped$promedio, decreasing = T),]
```

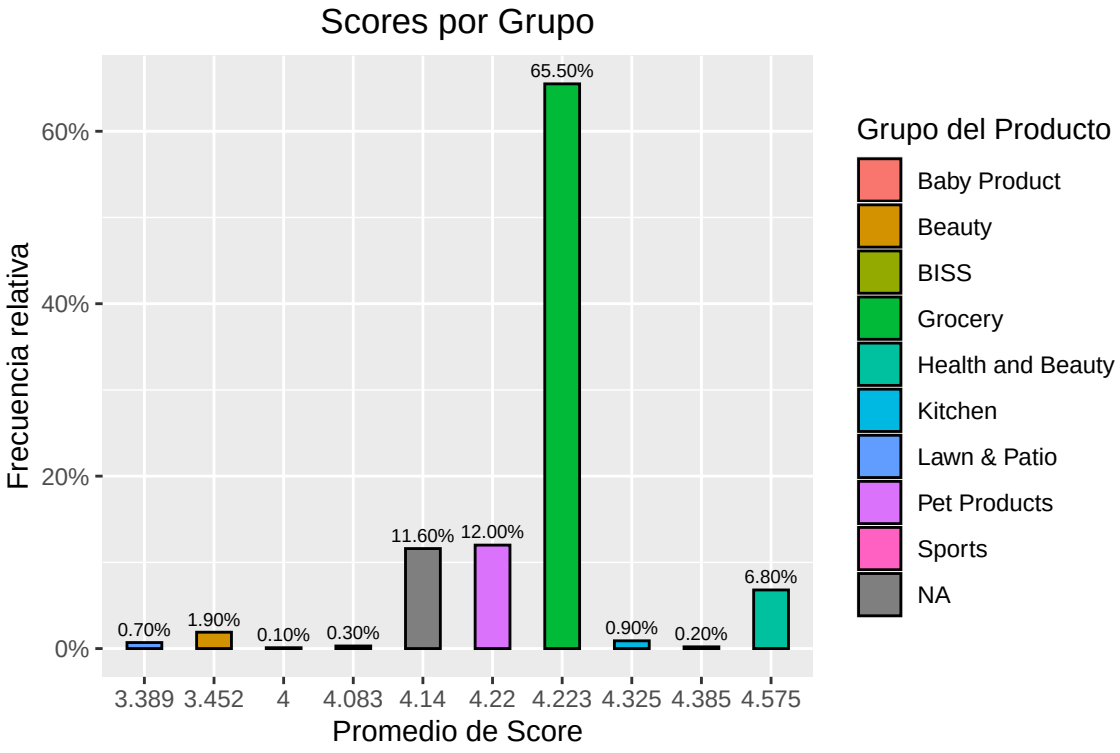
```
kable(product_gped.tabla,digits = 2,
      col.names = c("Grupo del Producto","Promedio","Frecuencia Relativa"),
      caption = "Scores por Grupo de Producto",booktabs = T)
```

```
ggplot(product_gped.tabla,aes(x=factor(promedio),y=frecuencia_relativa,
                              fill = Prod_Group)) +
```

Table 1: Scores por Grupo de Producto

Grupo del Producto	Promedio	Frecuencia Relativa
Health and Beauty	4.58	0.07
Baby Product	4.38	0.00
Kitchen	4.32	0.01
Grocery	4.22	0.66
Pet Products	4.22	0.12
NA	4.14	0.12
Sports	4.08	0.00
BISS	4.00	0.00
Beauty	3.45	0.02
Lawn & Patio	3.39	0.01

```
geom_bar(stat = "identity",color = "black",width = .5) +
xlab("Promedio de Score") +
ylab("Frecuencia relativa") +
ggtitle("Scores por Grupo") +
scale_y_continuous(labels=scales::percent) +
geom_text(aes(label=scales::percent(frecuencia_relativa)), vjust = -.5,
          color="black", size=2.5) +
theme(plot.title = element_text(hjust = 0.5)) +
guides(fill=guide_legend(title="Grupo del Producto"))
```



De los nueve productos disponibles, el grupo de **Health and Beauty** presenta el mejor score promedio con 4.57 y representa el 6.8% de las reseñas, seguido por **Baby Product** y **Kitchen** con un score promedio

de 4.38 y 4.32, respectivamente. Sin embargo, estos productos representan menos del 8% del total de las reseñas. Por otro lado, el grupo **Grocery** que cuenta con un score promedio de 4.22, representa más del 65.5% de las reseñas.

Es importante comentar que para el 11.63% de los productos no se cuenta con el grupo al que pertenecen. Por lo tanto es relevante llevar a cabo un análisis de `NA's`

```
na <- questionr::freq.na(data)
stargazer(na,summary = F, header = F,
          type = "text", title = "Frecuencia NA's",
          covariate.labels = c("","Frecuencia Absoluta",
                               "Frecuencia Relativa (%)"))
```

Frecuencia NA's

Frecuencia Absoluta Frecuencia Relativa (%)

Prod_Category 1,649 12
Prod_Group 1,549 12
ProductId 0 0
UserId 0 0
Score 0 0
Time 0 0
Summary 0 0
Nrev 0 0
Length 0 0
ID_review 0 0

Sólo las variables de *Prod_Category* y *Prod_Group* tienen valores `NA's`, sin embargo, al analizar de forma detenida la estructura de la base de datos caímos en cuenta que muchas de las reseñas se repetían para productos de los que sí se conoce la categoría y el grupo. Por ejemplo, para la categoría **magnesio** que pertenece al grupo *Health and Beauty* se tienen exactamente las mismas reseñas, en algunos casos con `NA` y en otros casos no. La manera de corregirlo fue reemplzar los valores faltantes con su correspondiente valor asociado, vía la reseña.

```
data.nas <- data
data.nas$Summary <- gsub("[[:punct:]]", " ", data.nas$Summary)
data.nas$Summary <- as.character(data.nas$Summary)
base1 <-
  data.nas %>%
  na.omit(Prod_Group) %>%
  filter(!duplicated(Summary))
base2 <-
  data.nas %>%
  filter(is.na(Prod_Group))
base3 <-
  data.nas %>%
  filter(is.na(Prod_Category))
base4 <-
  data.nas %>%
  na.omit(Prod_Category) %>%
  filter(!duplicated(Summary))
```

```
basejoin <- left_join(select(base2,-"Prod_Group"), select(base1,c("Summary","Prod_Group")), by = "Summa
basejoin <- left_join(select(basejoin,-"Prod_Category"), select(base4,c("Summary","Prod_Category")), by
basejoin$Prod_Group <- replace_na(basejoin$Prod_Group, "sin_grupo")
basejoin$Prod_Category <- replace_na(basejoin$Prod_Category, "sin_categoria")
data <- na.omit(data)
data <- rbind(data, basejoin)
```

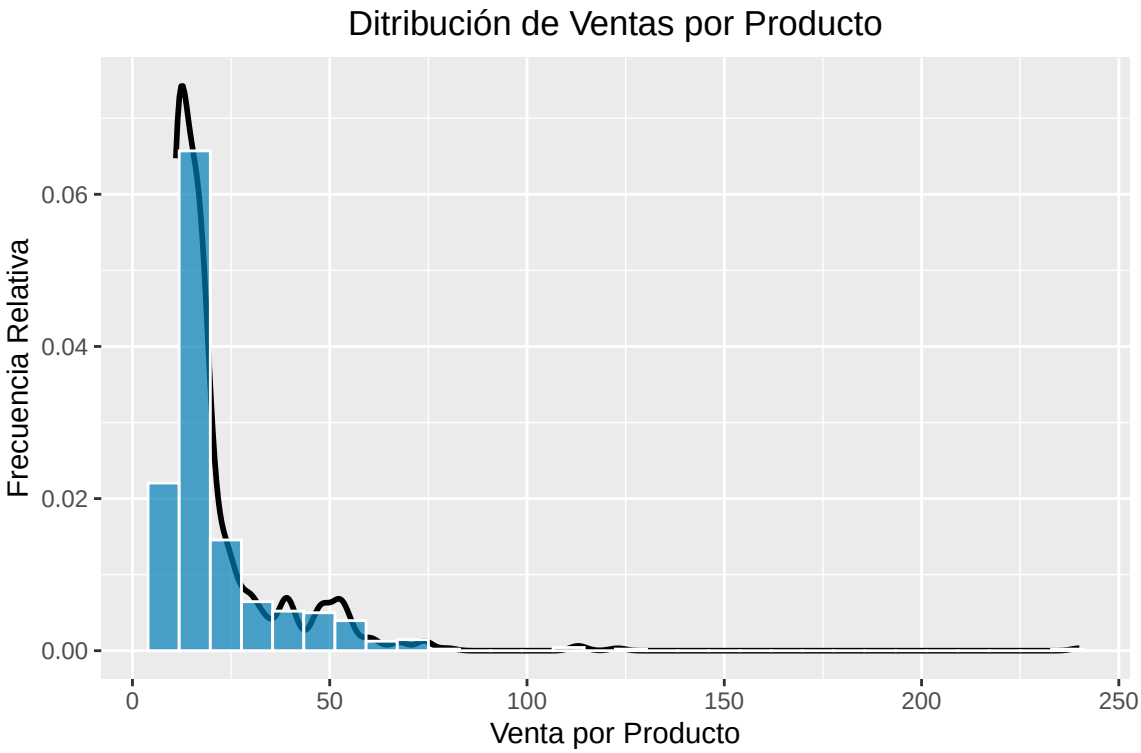
En relación a la variable *Prod_Category*, se cuentan con 142 categorías diferentes de productos y un total de 609 productos distintos.

```
length(unique(data$Prod_Category))
length(unique(data$ProductId))
```

También es relevante analizar la distribución de ventas por productos y de los *Scores*:

```
productID_gped <-
  data %>%
  group_by(ProductId) %>%
  summarise(
    frecuencia_absoluta = n(),
    frecuencia_relativa = (n()/nrow(data))*100
  )
productID_gped[order(productID_gped$frecuencia_relativa, decreasing = T),]
```

```
ggplot(productID_gped, aes(x = frecuencia_absoluta)) +
  stat_density(geom = 'line',size = 1) +
  geom_histogram(aes(y = ..density..),color = 'white', fill = '#007eb9', alpha = 0.7) +
  ggtitle("Ditribución de Ventas por Producto") +
  xlab("Venta por Producto") + ylab("Frecuencia Relativa") +
  theme(plot.title = element_text(hjust = 0.5))
```



Como se observa en la distribución anterior, aproximadamente el 20% de los productos presentan 11 ventas, 65% de los productos se vendieron 22 veces y el más vendido presenta 240 ventas.

Análisis de texto

Para el análisis de sentimientos, así como para el modelo de predicción, transformamos la base con las reseñas a un DataFrame que contenga las palabras *tokenizadas*, quitando los *stopwords* y homolgando todo a minúsculas.

```
data_txt <-
  data %>%
  unnest_tokens(output = word, input = Summary) %>%
  anti_join(get_stopwords())
# Removemos números
numeros <-
  data_txt %>%
  filter(str_detect(word, "^[0-9]")) %>%
  select(word) %>%
  unique()
data_txt <-
  data_txt %>%
  anti_join(numeros, by = "word")
# Por último, quitamos palabras con menor frecuencia
data_txt <-
  data_txt %>%
  group_by(word) %>%
  filter(n() > 10) %>%
  ungroup
data_txt$word <- gsub("'", "", data_txt$word)
```

Nube de palabras

Tranformamos la base de datos y hacemos *ngrams* de una palabra para graficar nuestro *word cloud*.

```
data_txt %>%
  count(word, sort = TRUE) %>%
  with(wordcloud(word, n, max.words = 200,
                 colors = c("#FF9900","#007eb9","#131A22","#232f3e")))
```

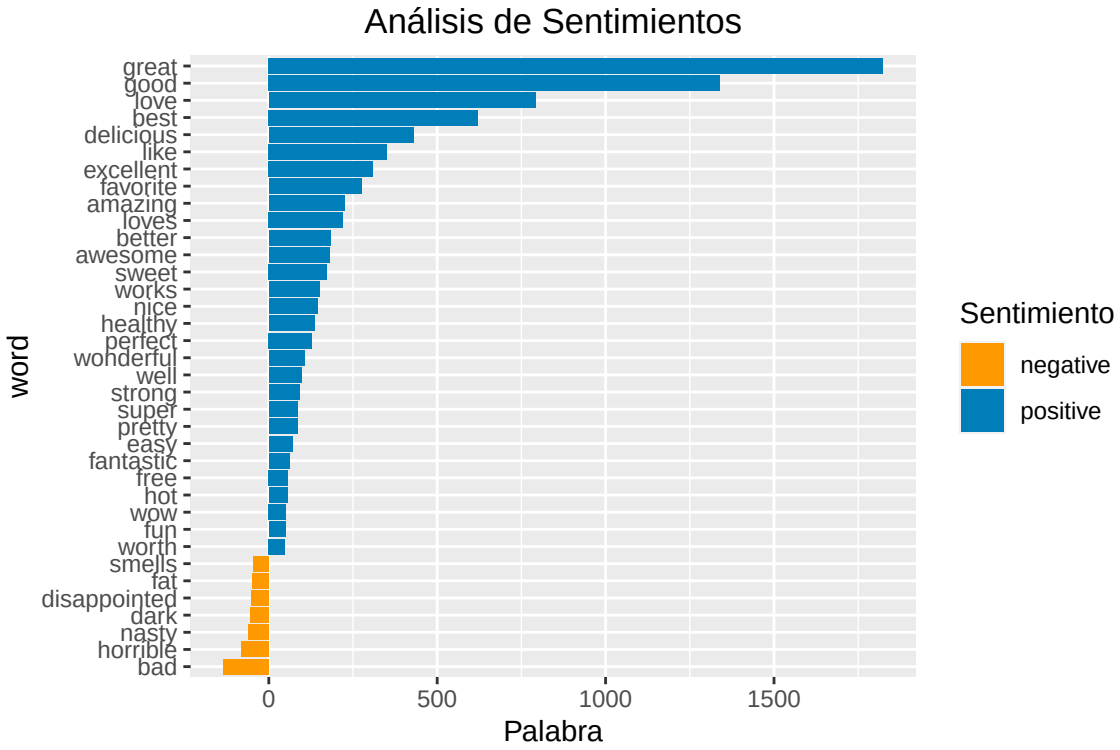
Como podemos ver, palabras como *great*, *good* y *love* son de las palabras con mayor frecuencia usadas en las reseñas.

Con bigramas vemos algunas frases que también nos dan una idea del sentimiento de la reseña. No obstante, para efectos de las estimaciones y los modelos de predicción usaremos solo palabras.

```
data_txt_bi <- data %>%
  unnest_tokens(output = word, input = Summary, token = "ngrams", n = 2) %>%
  anti_join(get_stopwords())
data_txt_bi %>%
  count(word, sort = TRUE) %>%
  with(wordcloud(word, n, max.words = 200,colors = c("#FF9900","#007eb9","#131A22","#232f3e")))
```

Análisis de sentimientos

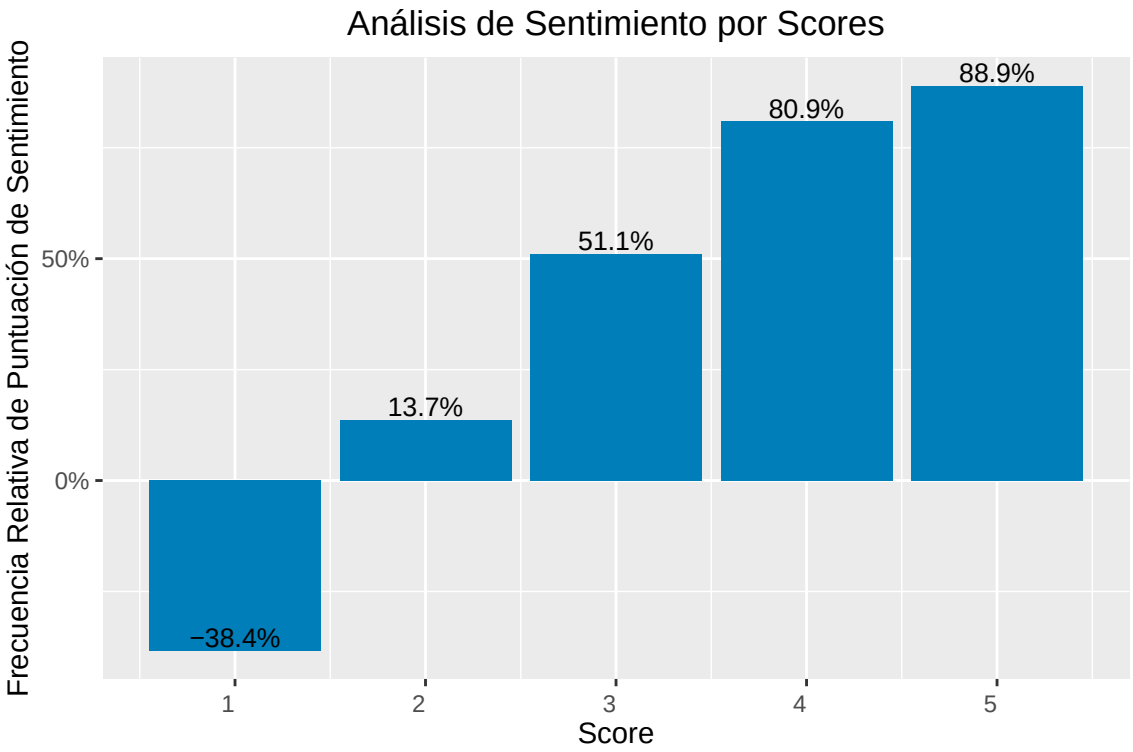
```
sentiments <- get_sentiments("bing")
sentiments_counts <-
  data_txt %>%
  inner_join(sentiments)  %>%
  count(word, sentiment, sort = TRUE)
sentiments_counts %>%
  filter(n > 45) %>%
  mutate(n = ifelse(sentiment == "negative", -n, n)) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  scale_fill_manual(values = c("#FF9900","#007EB9")) +
  geom_col() +
  coord_flip() +
  labs(y = "Conteo de palabras") +
  guides(fill=guide_legend(title="Sentimiento")) +
  ggtitle("Análisis de Sentimientos") + ylab("Palabra") +
  theme(plot.title = element_text(hjust = 0.5))
```



Como se aprecia en la figura anterior, en el conteo de sentimientos predominan las palabras asociadas a un sentimiento positivo, siendo *great* la palabra con mayor frecuencia dentro de este grupo y *bad* la palabra con mayor frecuencia dentro del grupo de palabras asociadas un sentimiento negativo.

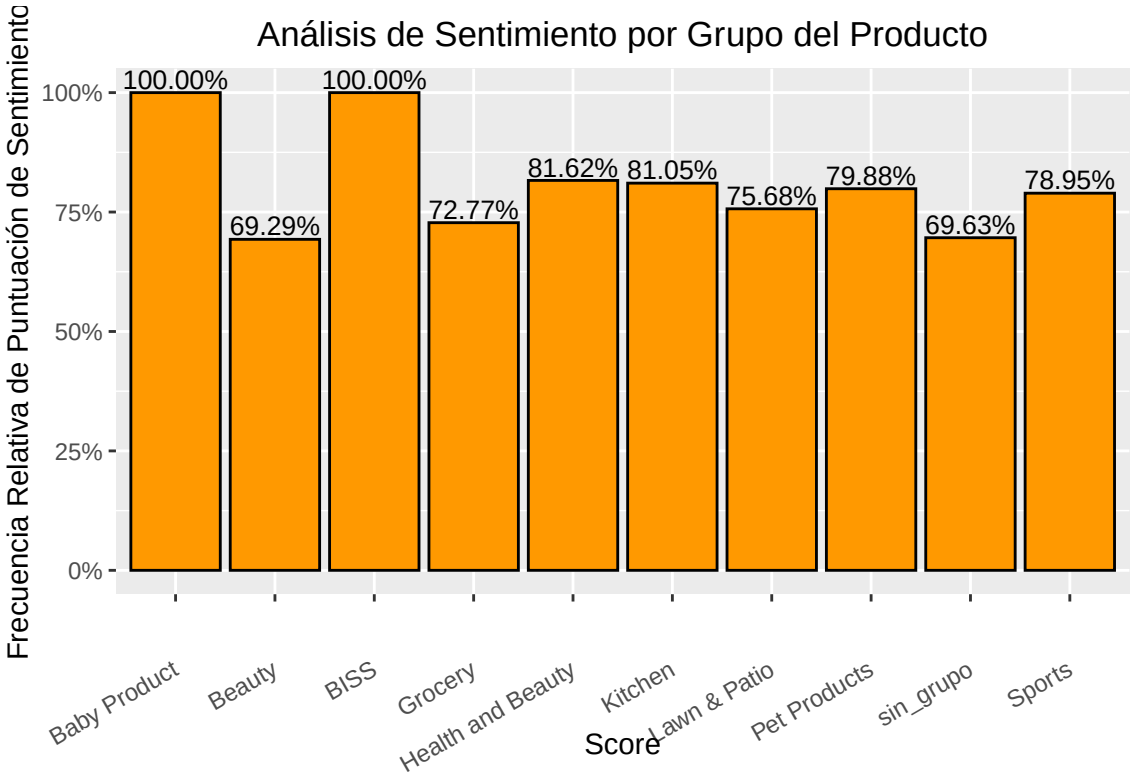
```
data_txt %>%
  inner_join(sentiments) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("#FF9900","#007EB9"),
                   max.words = 100)
```


```
por_score <-
  data_txt %>%
  inner_join(sentiments) %>%
  count(Score, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
por_score$sentiment_rel <- por_score$sentiment/(por_score$positive + por_score$negative)
ggplot(por_score, aes(x=Score, y=sentiment_rel)) +
  geom_bar(stat = "identity", fill = "#007eb9") +
  xlab("Score") +
  ylab("Frecuencia Relativa de Puntuación de Sentimiento") +
  ggtitle("Análisis de Sentimiento por Scores")  +
  scale_y_continuous(labels=scales::percent) +
  geom_text(aes(label=scales::percent(sentiment_rel)), vjust = -.2,
            color="black", size=3.5) +
  theme(axis.text.x = element_text( vjust = 0.5, hjust=1),
        plot.title = element_text(hjust = 0.5))
```



Se puede observar que la frecuencia relativa del *Score 1* esta asociada con un sentimiento negativo, a media que el *Score* mejora la frecuencia relativa asociada al sentimiento también lo hace de manera paulatina hasta el *Score 5* que refleja lo opuesto que la menor valuación; con una frecuencia relativa al sentimiento de 88.9%

```
por_grupo <-
  data_txt %>%
  inner_join(sentiments) %>%
  count(Prod_Group, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
por_grupo$sentiment_rel <- por_grupo$sentiment/(por_grupo$positive + por_grupo$negative)
ggplot(por_grupo, aes(x=Prod_Group, y=sentiment_rel)) +
  geom_bar(stat = "identity", fill = "#FF9900",
           color ="black") +
  xlab("Score") +
  ylab("Frecuencia Relativa de Puntuación de Sentimiento") +
  ggtitle("Análisis de Sentimiento por Grupo del Producto") +
  scale_y_continuous(labels=scales::percent) +
  geom_text(aes(label=scales::percent(sentiment_rel)), vjust = -.2,
            color="black", size=3.5) +
  theme(axis.text.x = element_text(angle = 30, vjust = 0.5, hjust=1),
        plot.title =  element_text(hjust = 0.5))
```



Considerando la suma de los valores absolutos y calculando una frecuencia relativa sobre estos totales se observa que los grupos *Baby Product* y *Business, Industrial, and Scientific Supplies* (BISS) representan la mayor puntuación de sentimiento, mientras que *Beauty* y los productos sin grupos asignado representan la menor frecuencia en términos de dicha puntuación.

Score para sentimientos

Usamos la librería *syuzhet* para aplicar un score de sentimiento a cada reseña. Estos datos lo vamos a utilizar también en los modelos de predicción de los *Score* ya que *bing* asocia a sentimientos positivo o negativo, mientras que esta libreria lo hace usando una escala de valores.

```
data$summary_score <- get_sentiment(data$Summary, method="afinn")
summary_score_gped <-
  data %>%
  group_by(Score) %>%
  summarise(sentiment_score = mean(summary_score))
stargazer(summary_score_gped,header = F,
          summary = F, type = "latex",
          title = "Sentimiento por Score",
          covariate.labels = c("","Score","Sentimiento"))
```

De este análisis, vemos que los *scores*, con el método *afinn*, se ajustan a los niveles de Scores de cada producto.

Scores promedio por producto de cada usuario

Una dato interesante que podría servir para predecir los *scores* es si el usuario ha comprado el producto con anterioridad.

Table 2: Sentimiento por Score

	Score	Sentimiento
1	1	-0.559121621621622
2	2	0.313659359190556
3	3	1.09655937846837
4	4	2.07026476578411
5	5	2.24122653608488

```
# Ordenamos por tiempo
data <- data[order(as.Date(data$Time, format="%Y/%m/%d")),]
data$unit <- 1
# Creamos una columna en donde combinaos el usuarios y el producto id
data$usu_prod <- with(data, paste0(UserId, ProductId))
# Creamos columna para saber cuántas veces ha comprado el producto
data$buy_before <- ave(data$unit, data$usu_prod, FUN=cumsum)
# Y creamos dummy para saber si el usuario, cuando hizo la reseña, habia comprado antes el producto
data$buy_before[data$buy_before == 1] <- 0
data$buy_before[data$buy_before > 1] <- 1
data <- select(data, -c('usu_prod', 'unit'))
data %>%
  group_by(buy_before) %>%
  summarise(score_user_mean = mean(Score)) -> antes.desp
antes.desp <- round(antes.desp,2)
stargazer(antes.desp, summary = F,
          header = F, title = "Score Cliente Frecuente",
          type = "latex",
          covariate.labels = c("","¿Compró antes?","Score Promedio"))
```

Table 3: Score Cliente Frecuente

	¿Compró antes?	Score Promedio
1	0	4.22
2	1	4.7

Como es de esperarse, los clientes frecuentes suelen otorgar puntajes más altos, mientras que los clientes que compran por primera ocasión no suelen otorgar *scores* altos, en promedio.

Data Transformation

Convertimos las variables categóricas de la base original en dummies, asimismo, filtramos algunas observaciones. Ya que pudimos haber tirados varias reseñas cuando filtramos data_txt, al quitar palabras con pocas frecuencia y números. Lo anterior será importante más adelante al generar la base de entrenamiento.

```
# Susituimmos los espacios del texto del grupo y categoría de productos, con el fin de no tener problem
data$Prod_Group <- gsub(" ", "_", data$Prod_Group)
data$Prod_Category <- gsub(" ", "_", data$Prod_Category)
data$Prod_Group <- gsub("-", "_", data$Prod_Group)
data$Prod_Category <- gsub("-", "_", data$Prod_Category)
```

```
data$Prod_Group <- gsub(",", "", data$Prod_Group)
data$Prod_Category <- gsub(",", "", data$Prod_Category)
data$Prod_Group <- gsub("&", "y", data$Prod_Group)
data$Prod_Category <- gsub("&", "y", data$Prod_Category)
# Pasamos a dummies Prod_Group y Prod_Category
data <- dummy_columns(data,
                      select_columns = c('Prod_Group', 'Prod_Category'),
                      remove_first_dummy = FALSE,
                      ignore_na = FALSE,
                      remove_selected_columns = TRUE)
unique_ID_review <-
  data_txt %>%
  count(ID_review) %>%
  select(ID_review)
data <-
  data %>%
  inner_join(unique_ID_review, by = c('ID_review'))
```

Para las predicciones, se tomó la decisión de separar los scores en 2 rangos. Lo anterior, ya que hay muy pocas observaciones para scores menores o iguales a 4. Y un *oversampling* crearía muchas observaciones sintéticas y un *undersampling* dejaría la base con muy pocas observaciones y perderíamos grados de libertad. Por lo anterior, se optó por hacer la variable *dummy* de *Score*, en donde:

$$exclente = 1 \cdot \{score > 4\} regular/malo = 1 \cdot \{score \leq 4\}$$

```
data$Score[data$Score < 4] <- 0
data$Score[data$Score >= 4] <- 1
freq <- freq(data$Score,digits = 2,total = T,valid = F)
stargazer(freq, summary = F, header = F,
          type = "latex", title = "Frecuencia Scores")
```

Table 4: Frecuencia Scores

	n	%
0	2,386	19.120
1	10,091	80.880
Total	12,477	100

Las clases presentan un desbalance considerable, por lo tanto, usaremos un *SMOTE* para tratar de balancearlas. La razón por la haremos *oversampling* y no *undersampling* es que nos quedaríamos con muy pocas observaciones y, por lo tanto, se perdería poder predictivo y eficiencia.

Primero dividimos el *DataFrame* en el set de entrenamiento (75%) y validación (25%)

```
split <-
  data %>%
  select(ID_review) %>%
  initial_split()
train_set <- training(split)
test_set <- testing(split)
```

```
# Colapsamos la base para saber sólo si en la reseña se escribe o no la palabra
data_clps <-
  data_txt %>%
  count(ID_review, word)
data_clps$word <- gsub("'", "", data_clps$word)
# Nos interesa saber si la palabra se usó en la reseña, por lo que homolgamos a 1
data_clps$n[data_clps$n > 1] <- 1
# Lo convertimos a matrix y después lo pasamos a dataframe para que todas las columnas de palabras se c
matrix_words <- data_clps %>%
  cast_sparse(ID_review,word, n)
df_words <- as.data.frame(as.matrix(matrix_words))
data <- cbind(data, df_words)
```

Trabajamos con nuestra base de entrenamiento.

```
# Elegimos nuestra base de entrenamiento
train_data <-
  data %>%
  inner_join(train_set, by = c('ID_review'))
X_train <-
  train_data %>%
  select(-c(Score, ProductId, UserId,Summary,Nrev,
                                    Length, ID_review))
X_train$Time <- as.integer(X_train$Time)
X_train<-clean_names(X_train)
y_train<- train_data$Score
X_train<-data.frame(X_train)
X_train[is.na(X_train)] = 0
```

Oversampling

Como se mencionó anteriormente las clases están desbalancedas, por lo tanto, hacemos *oversampling* vía *SMOTE* para lograr un mejor balance y que el modelo no sólo sea bueno prediciendo buenos *Scores.*

```
smoted <- smotefamily::SMOTE(X=X_train,
               target=y_train,
               dup_size = 3, K=5)
smoted_data <- smoted$data
colnames(smoted_data)[length(names(smoted_data))] = "Score"
y_train<- as.numeric(smoted_data$Score)
X_train<-select(smoted_data, -c(Score))
```

Volvemos a revisar el balance de nuestra base de datos y observamos que el balance mejora respecto al inicial (sin oversampling)

```
freq <- freq(y_train,digits = 2,total = T,valid = F)
colnames(freq) <- c("Número de observaciones","Porcentaje de observaciones")
stargazer(freq, summary = F, header = F,
          title = "Balance Después de SMOTE",type = "latex")
```

Convertimos nuestra información en una *sparse matrix* para poder correrla en nuestro modelo Lasso.

Table 5: Balance Después de SMOTE

	Número de observaciones	Porcentaje de observaciones
0	7,148	48.560
1	7,571	51.440
Total	14,719	100

```
X_train<-sparse.model.matrix(~ . + 0, data = X_train)
dim(X_train)
```

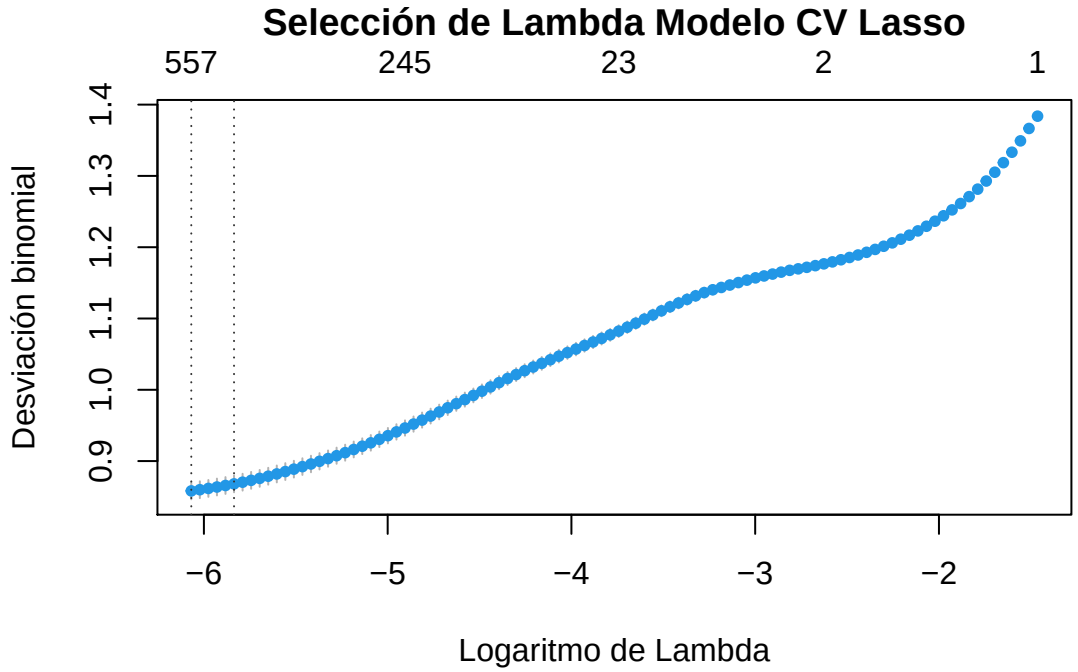
Machine learning models

Lasso

```
lasso <-  cv.gamlr(x=X_train,y=y_train,verb = T,family="binomial",
                   standarize=T,nfold=5)
```

```
## fold 1,2,3,4,5,done.
```

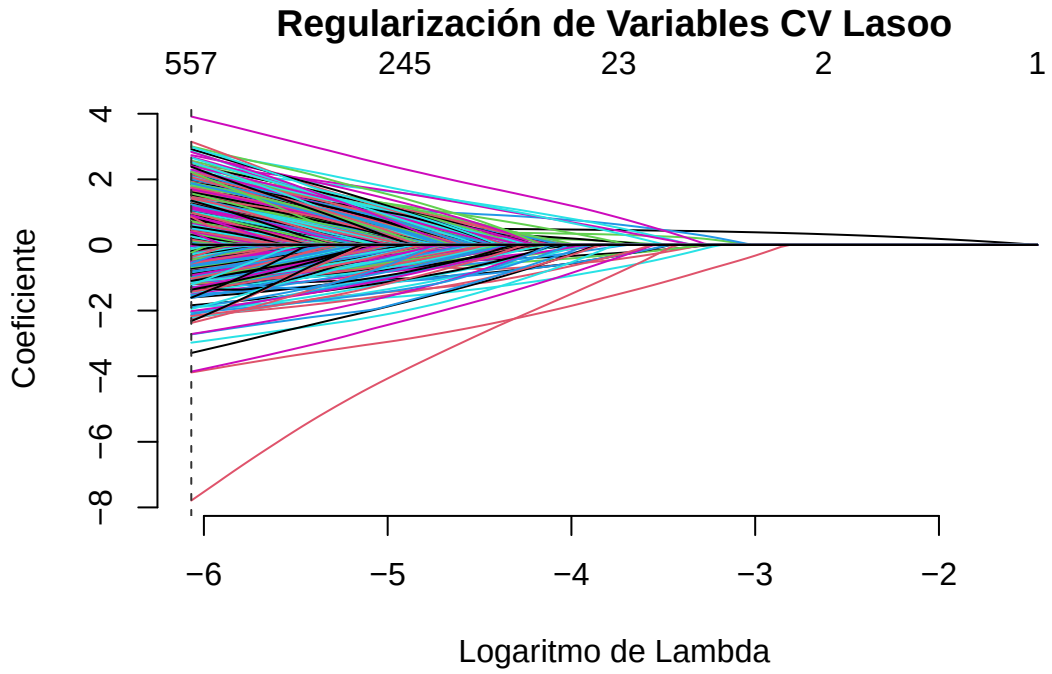
```
plot(lasso, xlab = "Logaritmo de Lambda", ylab="Desviación binomial",
     main = "Selección de Lambda Modelo CV Lasso")
```



```
best1 <- coef(lasso,select = "min")
best1_df <- as.data.frame(best1[order(abs(best1), decreasing=TRUE),])
sum(best1_df != 0)
```

Como se observa en la gráfica anterior $\lambda_{min} \approx 0.0023$ minimiza el *deviance* y regulariza 287, dejando 532 variables para la predicción.

```
plot(lasso$gamlr, xlab = "Logaritmo de Lambda", ylab="Coeficiente",
     main = "Regularización de Variables CV Lasoo")
```

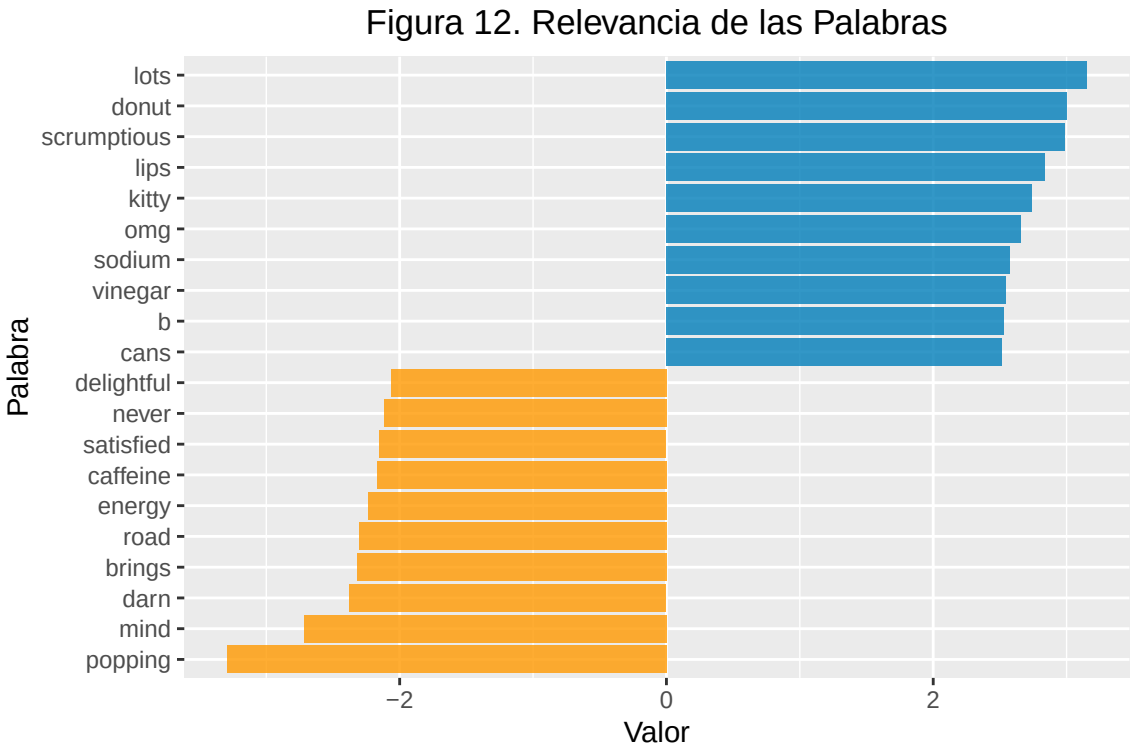


En la figura anterior se observa la forma en la que el *Lasso* regulariza las variables.

Ahora veamos cuáles de los coeficicentes tienen mayor probabilidad de tener un efecto en el *Score.*

```
coefs <-
  coef(lasso, select=c("min")) %>%
  tidy()
coefs_mal_words <-
  coefs %>%
  filter(!str_detect(row,"prod")) %>%
  filter(!str_detect(row,"intercept")) %>%
  filter(!str_detect(row,"per"))

coefs_mal_words %>%
  group_by(value > 0) %>%
  top_n(10, abs(value)) %>%
  ungroup() %>%
  ggplot(aes(fct_reorder(row, value), value,fill = value > 0)) +
  geom_col(alpha = 0.8, show.legend = FALSE) +
```

```
  coord_flip() +
  xlab("Palabra")+
  ylab("Valor")+
  labs(title = "Figura 12. Relevancia de las Palabras") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("#FF9900","#007EB9"))
```



Las palabras que tienen mayor relevancia para predecir el *Score* son:

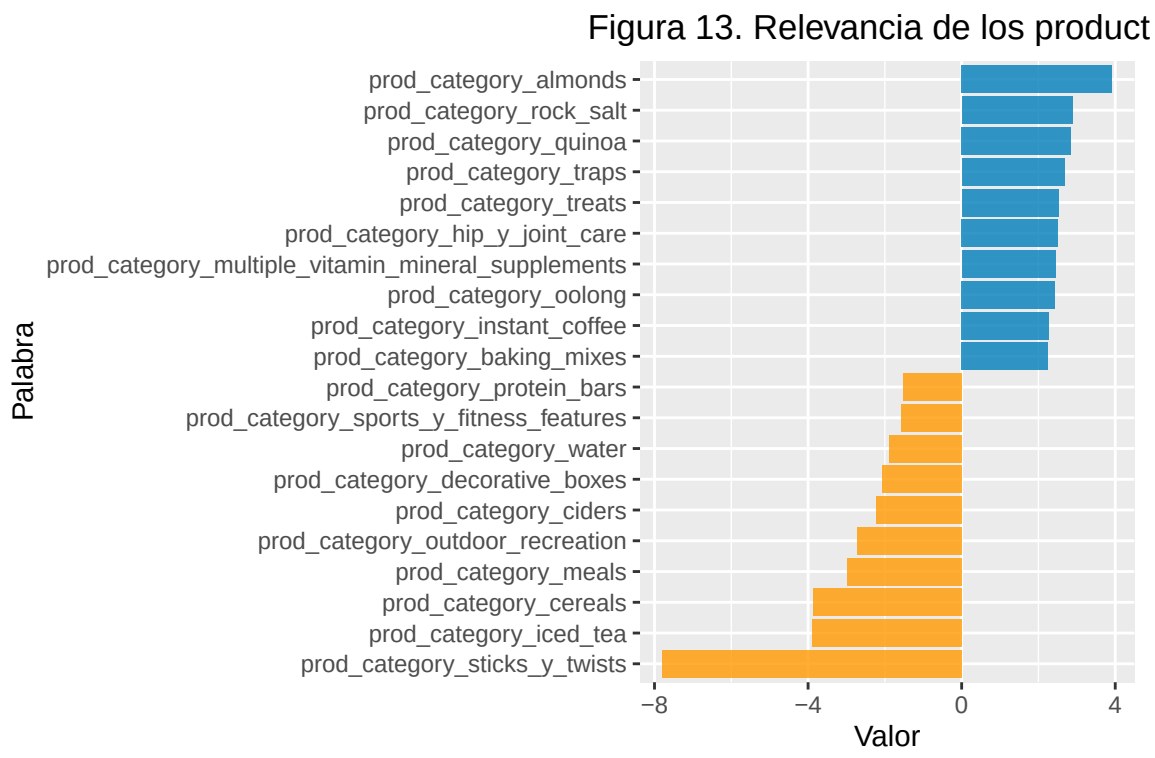
Probabilidad de obtener un alto puntaje positivo: - **lots** - **outstanding** - **kitty**

Probabilidad de obtener un alto puntaje negativo: - **energy** - **caffeine** - **popping**

Es interesante que estas palabras no coinciden con las asociadas al análisis de sentimientos, esto porque estas palabras no tienen una "carga" de sentimientos y las perdíamos; sin embargo, aparecen frecuentemente a lo largo de las reseñas y el *Lasso* considera todas las palabras, dándole más importancia a las que tienen mayor poder predictivo.

```
coefs_mal_prod <-
  coefs %>%
  filter(str_detect(row, "prod"))
coefs_mal_prod %>%
  group_by(value > 0) %>%
  top_n(10, abs(value)) %>%
  ungroup() %>%
  ggplot(aes(fct_reorder(row, value), value, fill = value > 0)) +
  geom_col(alpha = 0.8, show.legend = FALSE) +
  coord_flip() +
  xlab("Palabra")+
  ylab("Valor")+
```

```
  labs(title = "Figura 13. Relevancia de los productos") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("#FF9900","#007EB9"))
```



Las categorías que tienen mayor relevancia para predecir el *Score* son:

Probabilidad de obtener un alto puntaje positivo: - **traps** - **almonds** - **rock salt**

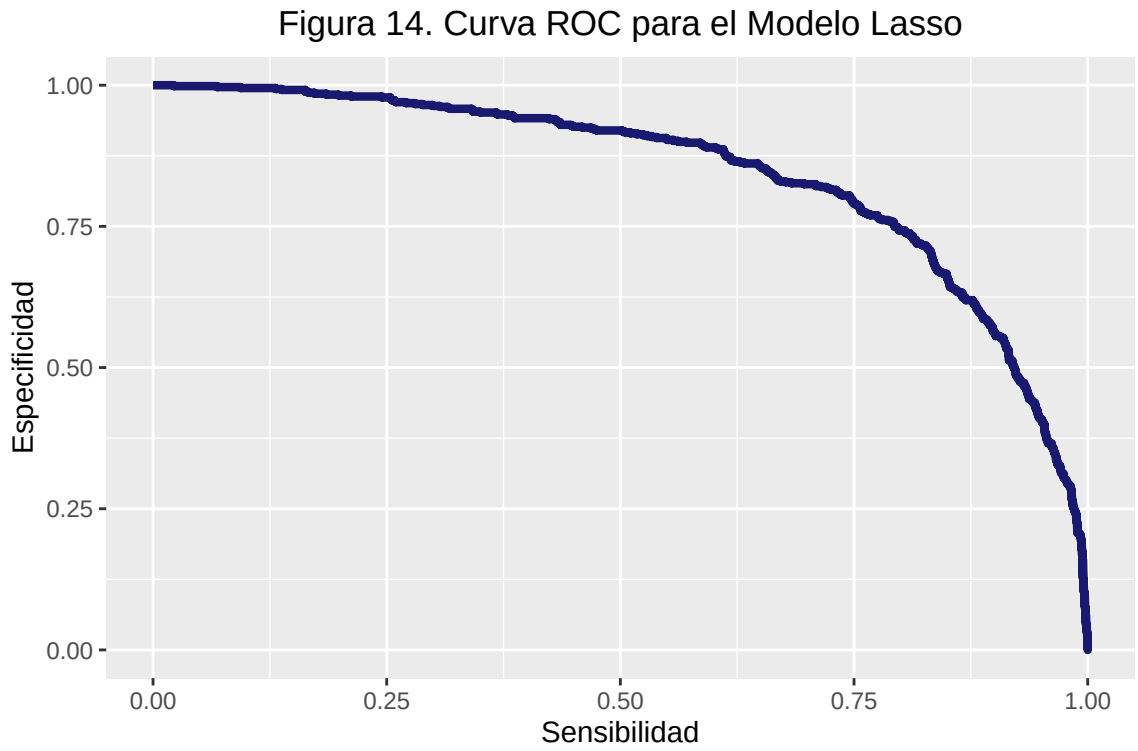
Probabilidad de obtener un alto puntaje negativo: - **sticks and twists** - **iced tea** - **cereals**

Llama la atención que la mayoría de las variables son categorías y no grupos (sólo hay un grupo), esto sucede porque las categorías tienen un mayor nivel de desagregación y dado que hay mayor varianza y se generan más señales, el modelo capta más información de ellas.

Ahora comparamos con la base de validación

```
validation_data <-
  data  %>%
  inner_join(test_set, by = c('ID_review'))
X_validation<-
  validation_data %>%
  select(-c(Score, ProductId, UserId,Summary,Nrev,Length, ID_review))
X_validation$Time <- as.numeric(X_validation$Time)
X_validation<-clean_names(X_validation)
y_validation<- validation_data$Score
X_validation <- lapply(X_validation, as.integer)
X_validation<-data.frame(X_validation)
X_validation[is.na(X_validation)] = 0
X_validation<-sparse.model.matrix(~ . + 0, data = X_validation,
                                  drop.unused.levels=TRUE)
dim(X_validation)
```

```r
pred <- drop(predict(lasso$gamlr, X_validation,
                     type = "response",
                     select = lasso$seg.min))
eval <- bind_cols(validation_data$Score,pred)
colnames(eval) <- c("Score", "Pred_lasso")
eval$Score <-  factor(eval$Score, levels = c(1, 0))
roc.lasso <- roc_curve(data = eval,
                               truth=Score, "Pred_lasso",
                               event_level= "first")
roc.lasso %>%
  ggplot(aes(x = sensitivity, y = specificity)) +
  geom_line(color = "midnightblue", size = 1.5)+
  xlab("Sensibilidad") + ylab("Especificidad")+
  labs(title = "Figura 14. Curva ROC para el Modelo Lasso") +
  theme(plot.title = element_text(hjust = 0.5))
```



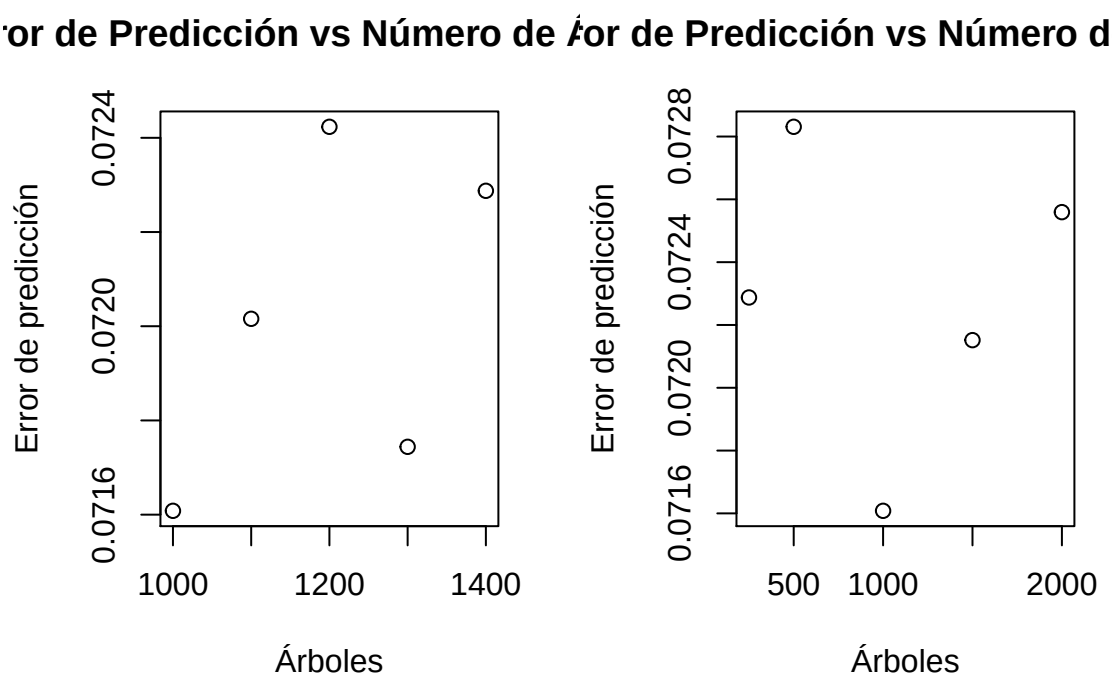
En la figura anterior se observa la curva ROC del modelo *Lasso* y el *trade-off* que existe entre sensibilidad y especificidad. Como se aprecia, el modelo tiene un mayor grado de sensibilidad que de especificidad, es decir, predice de mejor manera los verdaderos positivos que los falsos positivos, sin embargo, no tiene una mala especificidad.

Random forest

A continuación estimamos dos *Random Forest*, es importante recalcar que este modelo, si bien, tiene un mayor poder predictivo que el *Lasso* el costo de este poder es la pérdida en interpretabilidad de los resultados, es por eso que decidimos estimar ambos modelos.

```
train_data_ranger <- smoted_data
train_data_ranger$Score <- factor(train_data_ranger$Score, levels = c(1, 0))
train_data_ranger<-clean_names(train_data_ranger)
names(train_data_ranger) <- make.names(names(train_data_ranger))
train_data_ranger[is.na(train_data_ranger)] = 0
a <- Sys.time()
trees_list <- list(1000,1100,1200,1300,1400)
pred.error_list <- c()
for (tree in trees_list){

  rf <-ranger(score~., data = train_data_ranger, classification = T, num.trees = tree,
          importance = "impurity", splitrule="gini", seed=787, oob.error=T)

  pred.error_list <- c(pred.error_list, rf$prediction.error)
}
Sys.time() -a
```

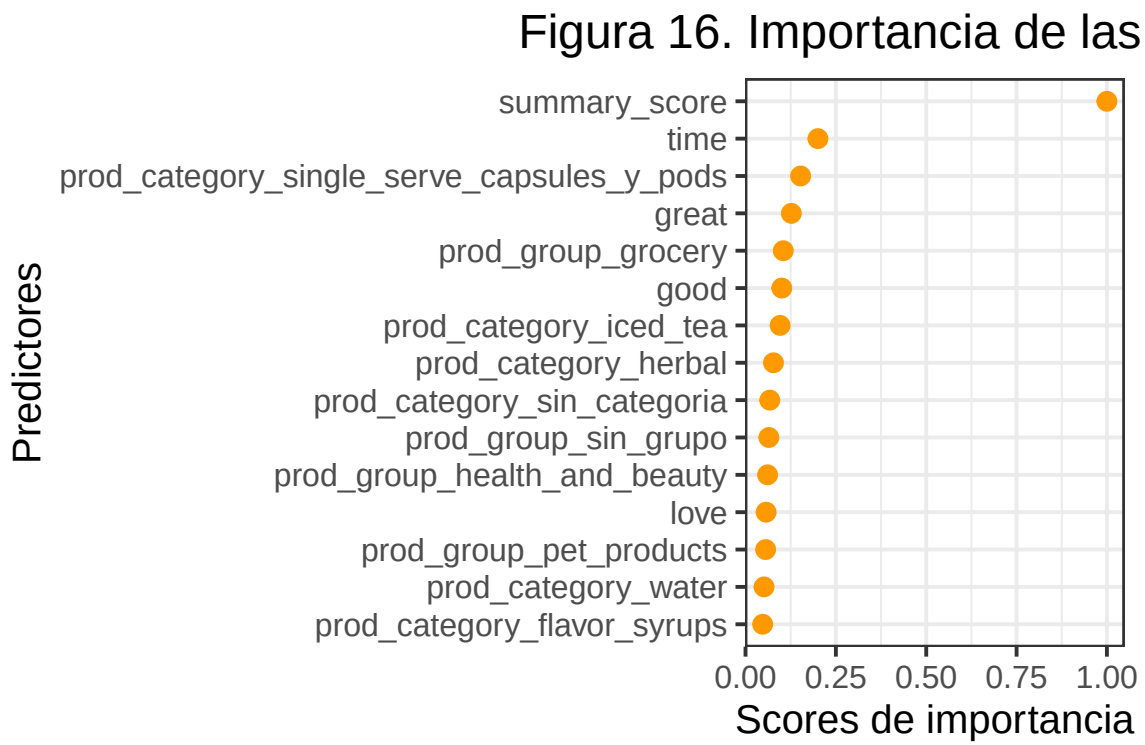
```
a <- Sys.time()
trees_list1 <- list(250,500,1000,1500,2000)
pred.error_list1 <- c()
for (tree in trees_list1){

  rf1 <-ranger(score~., data = train_data_ranger, classification = T, num.trees = tree,
          importance = "impurity", splitrule="gini", seed=787, oob.error=T)

  pred.error_list1 <- c(pred.error_list1, rf1$prediction.error)
}
Sys.time() -a
{par(mfrow=c(1,2))
plot(x=trees_list, y=pred.error_list,
     xlab="Árboles", ylab='Error de predicción') +
title('Error de Predicción vs Número de Árboles')
plot(x=trees_list1, y=pred.error_list1,
     xlab="Árboles", ylab='Error de predicción') +
title('Error de Predicción vs Número de Árboles')}
```



En la figura anterior se observan dos gráifcas asociadas a los dos modelos estimados, esto porque buscamos el número de árboles que minimizara el error de predicción, con base en esta métrica, decidimos utilizar un *Random Forest* con 1500 árboles.

Encontramos de la lista el árbol con menor error de predicción y las variables con mayor relevancia en la predicción.

```
best.forest <- trees_list1[which.min(pred.error_list)][[1]]
rf_best <-
  ranger(score~., data = train_data_ranger, classification = T,
         num.trees = best.forest, importance = "impurity", splitrule="gini",
         seed=787, oob.error=T, probability=T)


todo_imps <-
  data.frame(var = names(select(train_data_ranger, c(-score))),
             imps = rf_best$variable.importance/max(rf_best$variable.importance))
todo_imps <- todo_imps[order(todo_imps$imps, decreasing = T),]
todo_imps_short <- todo_imps[1:15,]
todo_imps_short %>%
  ggplot(aes(imps, x = reorder(var, imps))) +
  geom_point(size = 3, colour = "#ff9900") +
  coord_flip() + ggtitle("Figura 16. Importancia de las Variables") +
  labs(x = "Predictores", y = "Scores de importancia") +
  theme_bw(15) + theme(plot.title = element_text(hjust = 0.5))
```



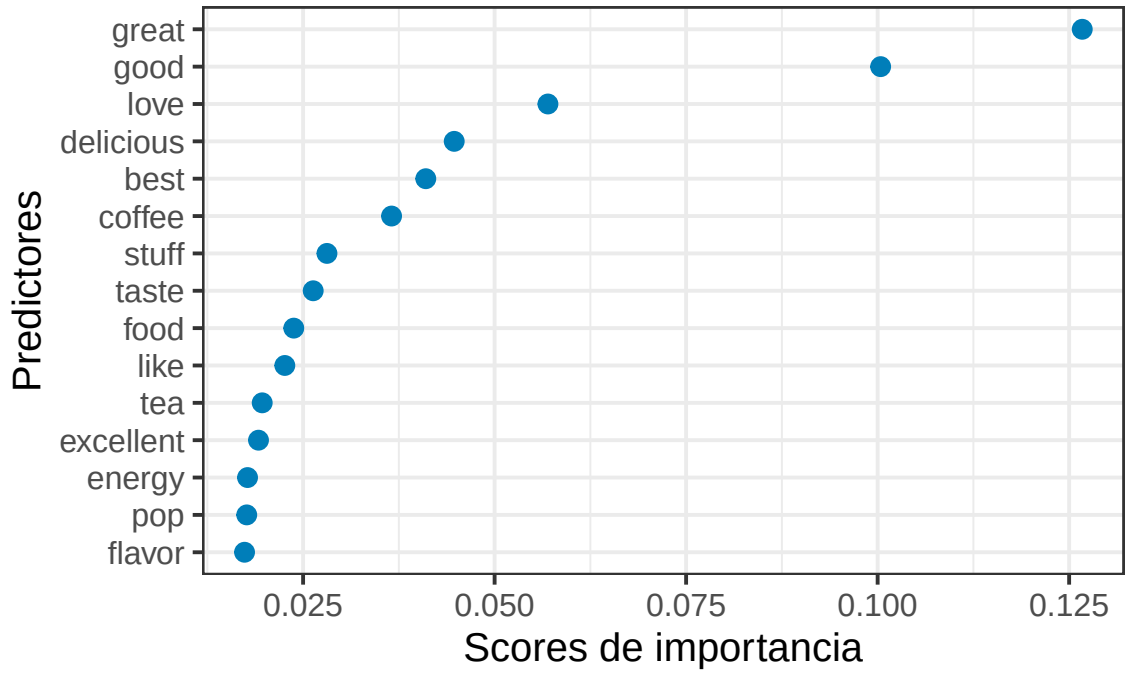
Los resultados que se presentan en la figura anterior son de suma importancia, como se observa, el *Score* de todo el *string* de las reseñas es la variable más relevante para pedecir, seguida de la fecha en la que se realizó la reseña, seguido por otros productos y palabras, como:

- **single serve capsules and pods**
- **great**
- **grocery**

También nos interesa el poder predictivo por palabra, por lo tanto, presentamos el peso en la predicción por palabra.

```
var_imps <-
  data.frame(var = names(select(train_data_ranger, c(-score))),
             imps = rf_best$variable.importance/max(rf_best$variable.importance))
var_imps <- var_imps[order(var_imps$imps, decreasing = T),]
var_imps_short <- filter(var_imps,
                         !str_detect(var, "prod"))
var_imps_short <- filter(var_imps_short,
                         !str_detect(var, "summary_score"))
var_imps_short <- filter(var_imps_short,
                         !str_detect(var, "time"))[1:15,]
var_imps_short %>%
  ggplot(aes(imps, x = reorder(var, imps))) +
  geom_point(size = 3, colour = "#007eb9") +
  coord_flip() + ggtitle("Figura 16. Importancia de las palabras") +
  labs(x = "Predictores", y = "Scores de importancia") +
  theme_bw(15) + theme(plot.title = element_text(hjust = 0.5))
```

Figura 16. Importancia de las palabras

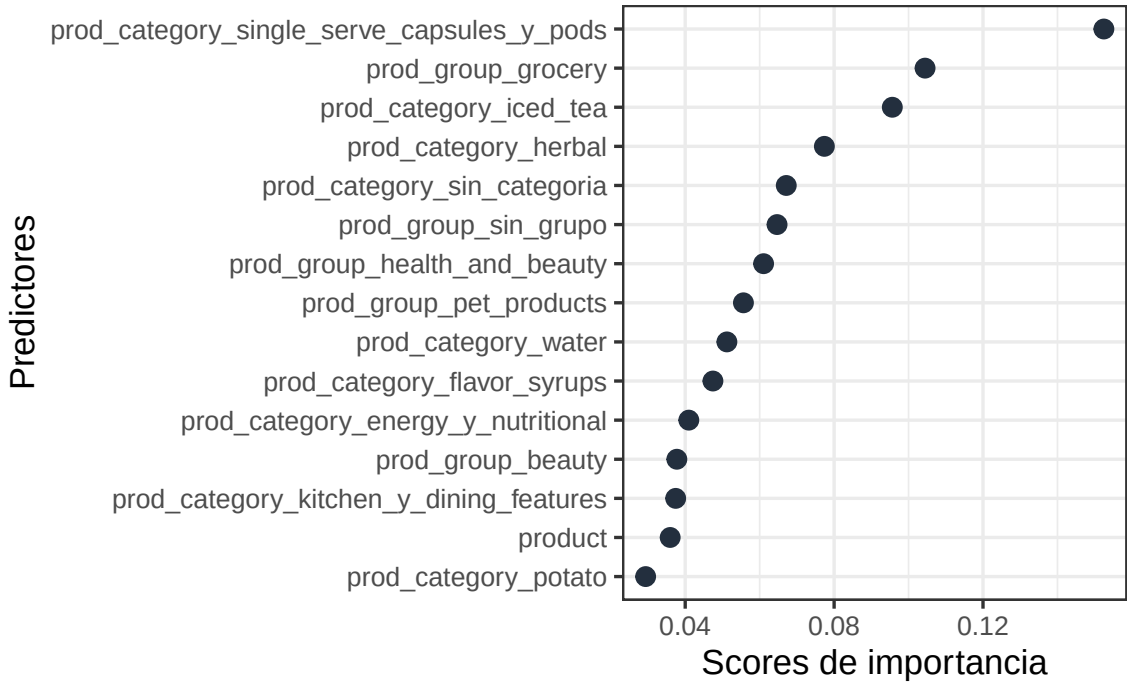


A diferencia del *Lasso*, las palabras que tienen mayor relevancia para la predicción sí coinciden con el análisis de sentimientos que se realizó, como se observa, las palabras con mayor peso tienen asociado un sentmiento positvo, como: *great*, *good* y *love*.

Por último revisamos cuales son los productos que tienen mayor relevancia para la predicción.

```
var_imps_prod <- var_imps %>%
  filter(str_detect(var, "prod"))
var_imps_prod <-  var_imps_prod[1:15,]
var_imps_prod %>%
  ggplot(aes(imps, x = reorder(var, imps))) +
  geom_point(size = 3, colour = "#232F3E") +
  coord_flip() + ggtitle("Figura 17. Importancia de los productos") +
  labs(x = "Predictores", y = "Scores de importancia") +
  theme_bw(13) + theme(plot.title = element_text(hjust = 0.5))
```

Figura 17. Importancia de los prod



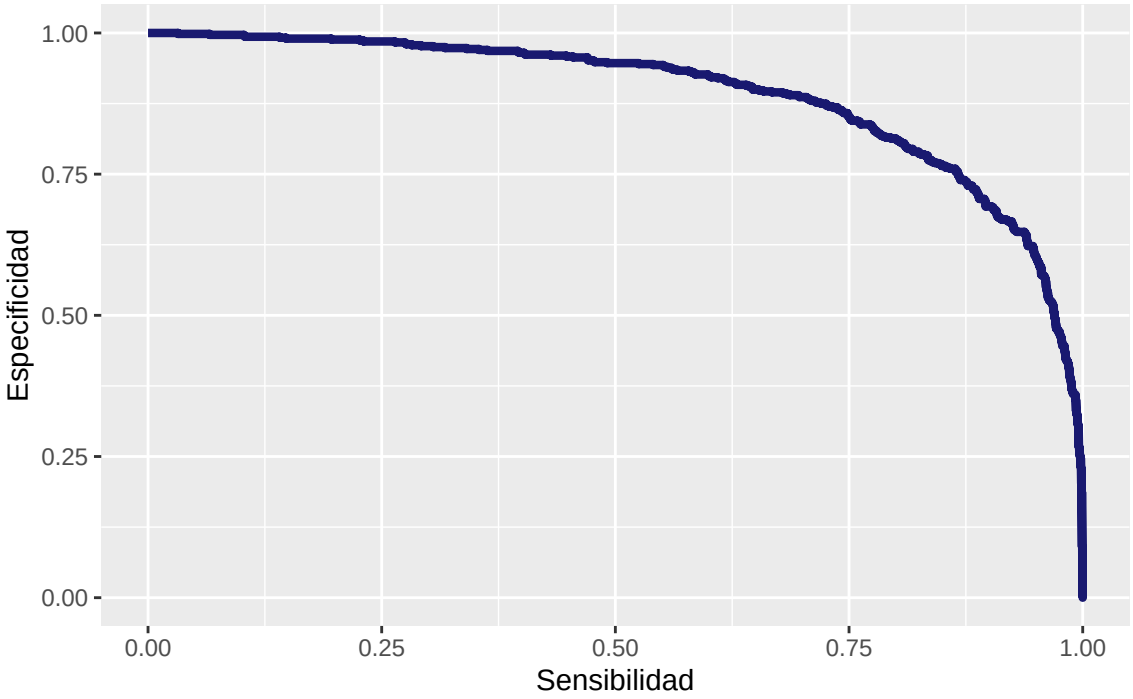
Como se observa en la figura anterior, las categorías y los grupos que son mejores para la predicción son:

- **single serve capsules and pods**
- **grupo grocery**
- **categoría iced tea**

```
validation_data_ranger <-
  validation_data %>%
  select(-c(ProductId, UserId,Summary,
            Nrev,Length, ID_review))
validation_data_ranger$Score <-
  factor(validation_data_ranger$Score, levels = c(1, 0))
validation_data_ranger<-clean_names(validation_data_ranger)
names(validation_data_ranger) <- make.names(names(validation_data_ranger))
validation_data_ranger[is.na(validation_data_ranger)] = 0
eval$pred_rf <- drop(predict(rf_best, validation_data_ranger)$predictions[,1])
```

```
roc.rf <- roc_curve(data = select(eval, c("Score", "pred_rf")),
                              truth=Score, "pred_rf",
                              event_level= "first")
roc.rf %>%
  ggplot(aes(x = sensitivity, y = specificity)) +
  geom_line(color = "midnightblue",size = 1.5) +
  xlab("Sensibilidad") +
  ylab("Especificidad") +
  labs(title = "Figura 18. Curva ROC para el Random Forest")
```

Figura 18. Curva ROC para el Random Forest



Como se observa en la curva *ROC* asociada al *Random Forest* que estimamos se tiene un buen nivel tanto de sensibilidad como de especificidad, además, se gano especificidad en relación al modelo *Lasso.*

XGBoosting

Dado la dificultad para *tunear* los parámetros de este modelo estimamos tres de ellos hasta encontrar el mejor dentro del poder de cómputo disponible.

Modelo 1

```
# Base de entrenamiento
y_train.boost <- as.numeric(smoted_data$Score)
X_train.boost <-select(smoted_data, -c(Score))
X_train.boost <- sparse.model.matrix(~ . + 0, data = X_train.boost)
xgboosting1 <-
  xgboost(data = X_train,label = y_train.boost ,objective = "binary:logistic",
          nrounds = 10, max.depth = 6, eta = 0.05, gamma = 0.1,
          lambda = 0.05,early_stopping_rounds = 20, eval_metric = "error" )
```

```
# Base de validación
validation_data.xgb <-
  data %>%
  inner_join(test_set, by = c("ID_review"))
X_validation.xgb <-
  validation_data.xgb %>%
  select(-c(Score, ProductId, UserId, Summary,
            Nrev, Length, ID_review))
```

```
X_validation.xgb$Time <- as.numeric(X_validation.xgb$Time)
X_validation.xgb <- clean_names(X_validation.xgb)
y_validation.xgb <- validation_data.xgb$Score
X_validation.xgb <- lapply(X_validation.xgb, as.integer)
X_validation.xgb <- data.frame(X_validation.xgb)
X_validation.xgb <- sparse.model.matrix(~.+0, data = X_validation.xgb,
                                        drop.unused.levels = TRUE)
dim(X_validation.xgb)
```

```
# Predicción
pred.xgb <- predict(xgboosting1, X_validation.xgb)
head(pred.xgb)
```

```
cbind(pred.xgb>0.5,y_validation.xgb) %>%
  data.frame() %>%
  table() %>%
  confusionMatrix()
```

Para el primer intento obtuvimos los siguientes resultados:

- **Accuracy:** 0.83
- **Sensitivity:** 0.22
- **Specificity:** 0.98
- **auc:**0.75

Modelo 2

```
xgboosting2 <-
  xgboost(data = X_train,label = y_train.boost ,objective = "binary:logistic",
          nrounds = 100, max.depth = 8, eta = 0.05,
          gamma = 5, lambda = 0.05,early_stopping_rounds = 10, eval_metric = "error" )
```

```
pred.xgb2 <- predict(xgboosting2, X_validation.xgb)
head(pred.xgb)
```

```
cbind(pred.xgb2>0.5,y_validation.xgb) %>%
  data.frame() %>%
  table() %>%
  confusionMatrix()
```

Para el segundo intento obtuvimos los siguientes resultados:

- **Accuracy:** 0.85
- **Sensitivity:** 0.39
- **Specificity:** 0.96
- **auc:**0.83

Modelo 3

```
xgboosting3 <-
  xgboost(data = X_train,label = y_train.boost ,objective = "binary:logistic",
          nrounds = 100, max.depth = 8, eta = 0.1,
          gamma = 5, lambda = 0.05,early_stopping_rounds = 10, eval_metric = "error" )
```

```
pred.xgb3 <- predict(xgboosting3, X_validation.xgb)
head(pred.xgb)
```

```
cbind(pred.xgb3>0.5,y_validation.xgb) %>%
  data.frame() %>%
  table() %>%
  confusionMatrix()
```

Para el tercer intento obtuvimos los siguientes resultados:

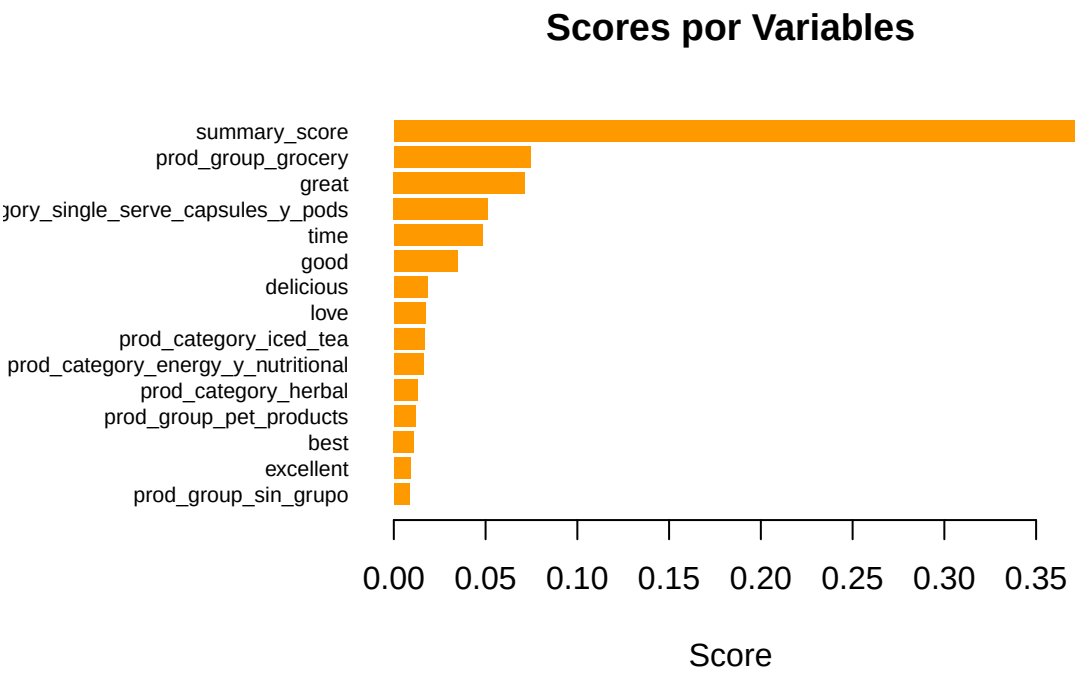
- **Accuracy:** 0.86
- **Sensitivity:** 0.43
- **Specificity:** 0.96
- **auc:**0.85

```
importance_matrix <- xgb.importance(model = xgboosting3)[1:15,]
print(importance_matrix)
```

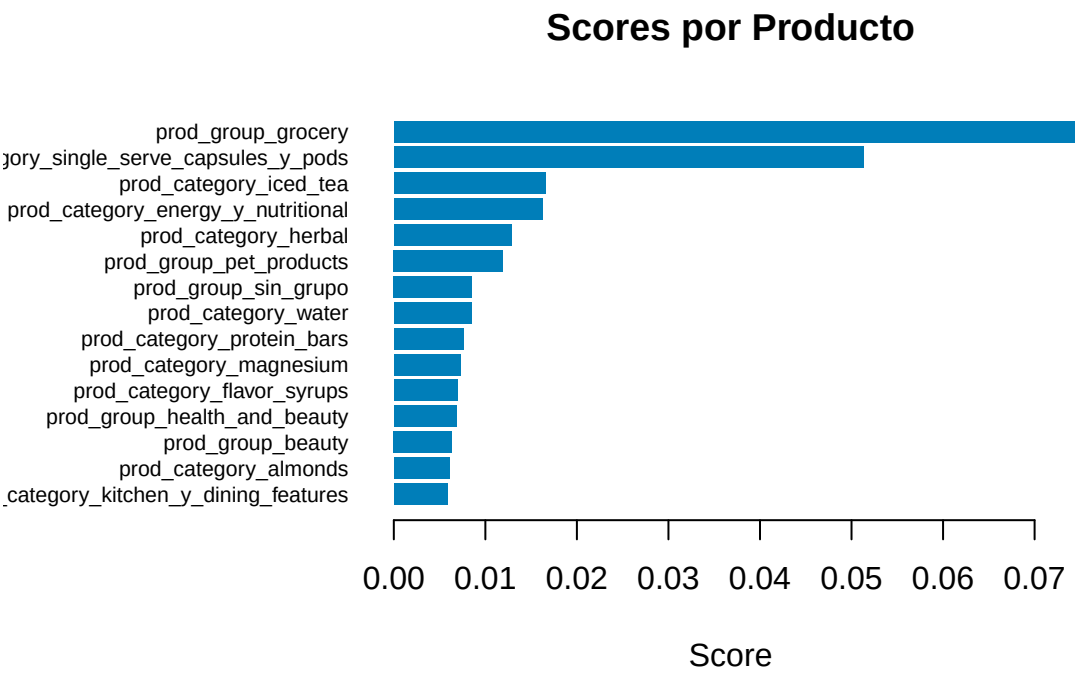
```
##                                       Feature        Gain      Cover
##  1:                             summary_score 0.370865833 0.08828130
##  2:                         prod_group_grocery 0.074348332 0.03639580
##  3:                                      great 0.071552867 0.03984669
##  4: prod_category_single_serve_capsules_y_pods 0.051290594 0.04291963
##  5:                                       time 0.048238882 0.03385641
##  6:                                       good 0.034760099 0.03108687
##  7:                                  delicious 0.018203808 0.02431335
##  8:                                       love 0.017194919 0.02296130
##  9:                       prod_category_iced_tea 0.016571603 0.02288418
## 10:         prod_category_energy_y_nutritional 0.016231546 0.01969670
## 11:                           prod_category_herbal 0.012878968 0.01660139
## 12:                       prod_group_pet_products 0.011921558 0.01593308
## 13:                                       best 0.010955590 0.01640809
## 14:                                  excellent 0.008995206 0.01545976
## 15:                          prod_group_sin_grupo 0.008533889 0.01389012
##      Frequency
##  1: 0.09103169
##  2: 0.02697235
##  3: 0.03034390
##  4: 0.03506406
##  5: 0.15171949
##  6: 0.02832097
##  7: 0.01753203
##  8: 0.01685772
##  9: 0.01955496
## 10: 0.01618341
## 11: 0.01550910
```

```
## 12: 0.01820634
## 13: 0.01213756
## 14: 0.01078894
## 15: 0.01483479
```

```
xgb.plot.importance(importance_matrix = importance_matrix,
                    main ="Scores por Variables", col = "#FF9900",
                    xlab="Score")
```




```
importance_matrix.prod <- xgb.importance(model = xgboosting3)
importance_matrix.prod <- importance_matrix.prod %>%
  filter(str_detect(Feature,"prod"))
importance_matrix.prod <- importance_matrix.prod[1:15,]
xgb.plot.importance(importance_matrix = importance_matrix.prod,
                    main ="Scores por Producto", col = "#007eb9",
                    xlab="Score")
```

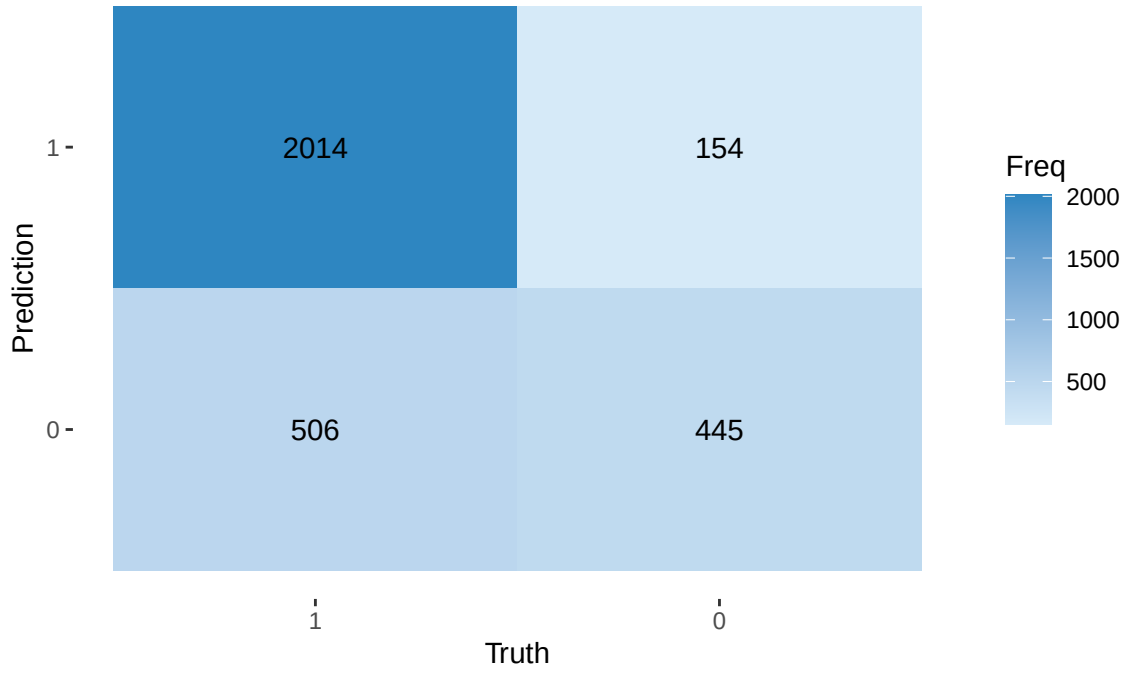


Como se aprecia en las figuras anteriores, la importancia de las variables es muy similar que las del *Random Forest*, esto es de esperarse ya que ambos algoritmos tienen como base estimar árboles, la diferencia es que uno lo hace de forma paralela y el otro de forma secuencial.

Comparación de Modelos Estimados

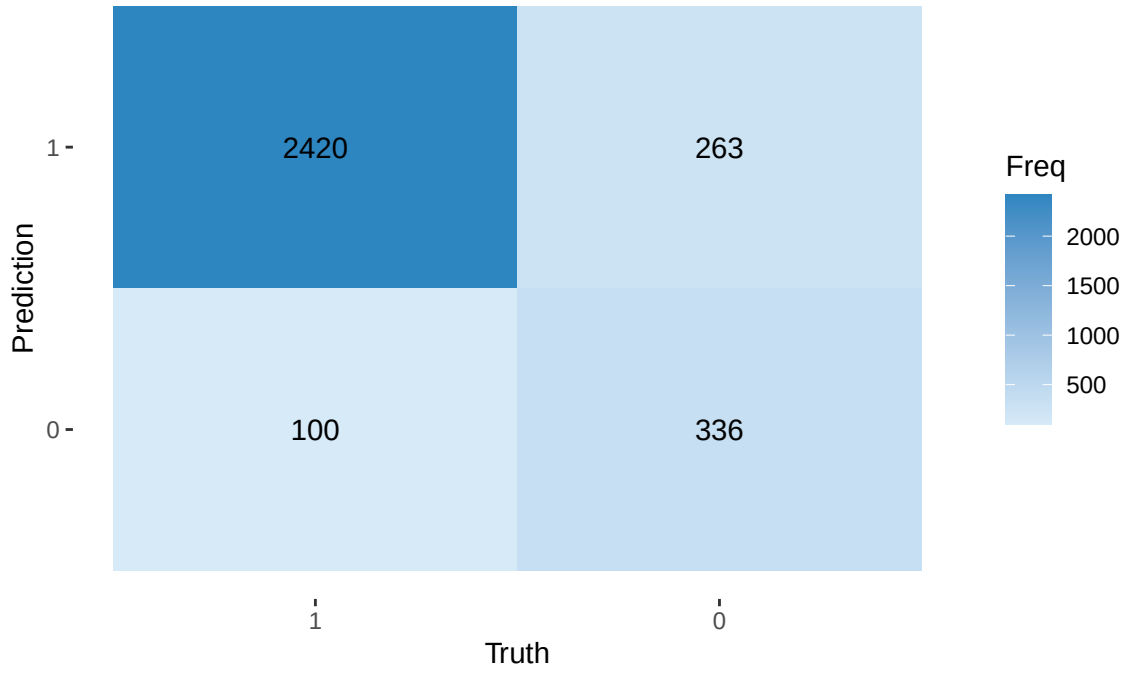
```
# Convertimos todas las probabilidades a unos y ceros
eval$pred_xgb <- drop(predict(xgboosting3, X_validation.xgb))
eval_matrix <- eval
threshold <- .5
eval_matrix[eval_matrix<=threshold] <- 0
eval_matrix[eval_matrix>threshold] <- 1
# Lasso
# Convertimos la variable en factor
eval_matrix$Pred_lasso <- factor(eval_matrix$Pred_lasso, levels = c(1, 0))
mat_lasso <- conf_mat(data = eval_matrix, truth= Score, estimate=Pred_lasso)
autoplot(mat_lasso, type = "heatmap") + ggtitle("Figura 19. Matriz de confusión: Lasso") +
  scale_fill_gradient(low="#D6EAF8",high = "#2E86C1") +
  theme(legend.position = "right")
```

Figura 19. Matriz de confusión: Lasso



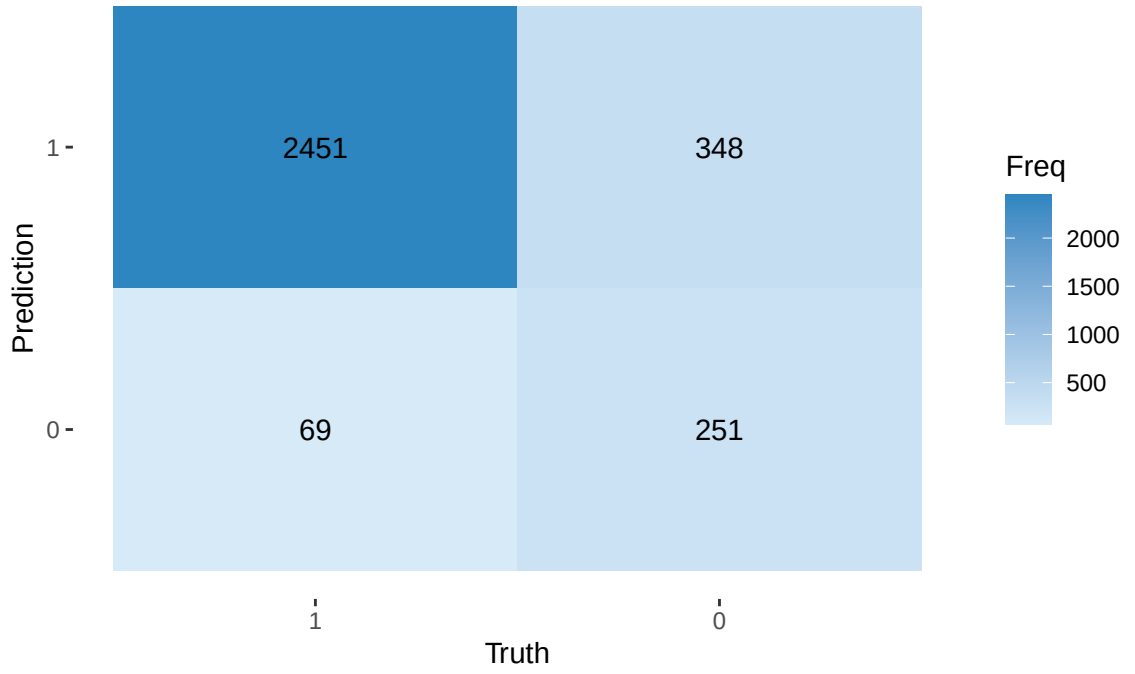

```
# Random Forest
# Convertimos la variable en factor
eval_matrix$pred_rf <- factor(eval_matrix$pred_rf, levels = c(1, 0))
mat_rf <- conf_mat(data = eval_matrix, truth= Score, estimate=pred_rf)
autoplot(mat_rf, type = "heatmap") + ggtitle("Figura 20. Matriz de confusión: Random Forest") +
  scale_fill_gradient(low="#D6EAF8",high = "#2E86C1") +
  theme(legend.position = "right")
```

Figura 20. Matriz de confusión: Random Forest




```
# XGB
# Convertimos la variable en factor
eval_matrix$pred_xgb <- factor(eval_matrix$pred_xgb, levels = c(1, 0))
mat_xgb <- conf_mat(data = eval_matrix, truth= Score, estimate=pred_xgb)
autoplot(mat_xgb, type = "heatmap") + ggtitle("Figura 20. Matriz de confusión: XGB") +
  scale_fill_gradient(low="#D6EAF8",high = "#2E86C1") +
  theme(legend.position = "right")
```

Figura 20. Matriz de confusión: XGB




```
# Corremos la función de roc_auc para cada uno de nuestros modelos
# Lasso:
lasso_auc <- roc_auc(data = select(eval, c(Score, Pred_lasso)),
                             truth=Score, "Pred_lasso",
                             event_level= "first")
lasso_auc$name <- "Lasso"
# Random Forest:
rf_auc <- roc_auc(data = select(eval, c(Score, pred_rf)),
                             truth=Score, "pred_rf",
                             event_level= "first")
rf_auc$name <- "Random_forest"
# XGBoosting:
xgb_auc <- roc_auc(data = select(eval, c(Score, pred_xgb)),
                             truth=Score, "pred_xgb",
                             event_level= "first")
xgb_auc$name <- "XGBoosting"
auc_table <- bind_rows(rf_auc,xgb_auc,lasso_auc)
auc_table <- select(auc_table, c(.estimate, name))
colnames(auc_table) <- c("AUC", "Modelo")

stargazer(auc_table, summary = F, header = F,
          title = "AUC Modelos",type = "latex")
```

Conclusión

A lo largo de la investigación se muestra la relevancia del análisis de sentimientos, como lo vimos en los modelos, la variable *summary_scores* que mide el sentimiento de la reseña es la variable con mayor importancia para la predicción de *Scores.*

Table 6: AUC Modelos

	AUC	Modelo
1	0.891043604420065	Random_forest
2	0.867445411664944	XGBoosting
3	0.844031719532554	Lasso

De los tres modelos estimados, la mejor predicción fue el la del *Random Forest*, seguido por el *XGBoosting* y por último el *Lasso*. Es importante recalcar que para los tres modelos se tienen una muy buena predicción de los verdaderos positivos.

Los resultados son de relevancia para **Amazon** ya que dan a conocer las áreas de oportunidad en sus grupos y categorías por producto. Este análisis da un hincapié para investigaciones futuras que permita focalizar la ubicación del gasto en los productos que presentan mejores reseñas, así como encontrar políticas que mejoren las valuaciones negativas, todo esto sustentado por un análisis robusto de sentimientos. Si bien la propuesta parece estar sustentada por el modelo *Lasso* ya que tiene una mayor interpretabilidad, los otros modelos respaldan el poder predictivo de estas variables.