Proyecto - Filtro de reseñas

Estefany Carolina Segura Linares, Brayan David Prieto Aya, Alejandro Jaramillo Vallejo

2025-10-28

1. Introducción

Presentación del proyecto

El proyecto Filtro-Reseñas tiene como objetivo implementar un algoritmo capaz de detectar reseñas falsas o de tipo spam en aplicaciones móviles.

Ante el crecimiento global del mercado digital, las reseñas en plataformas como Google Play Store y App Store influyen directamente en la reputación y el posicionamiento de las aplicaciones. Sin embargo, la presencia de reseñas manipuladas, generadas por bots o usuarios incentivados, distorsiona la percepción real de calidad y genera competencia desleal.

Para abordar esta problemática, se emplea el algoritmo **Naive Bayes Multinomial**, ampliamente utilizado en clasificación de texto, aplicándolo sobre el dataset "Multilingual Mobile App Reviews Dataset 2025" de Kaggle. El proceso incluye la limpieza y preprocesamiento del texto, la conversión de datos en vectores, y la evaluación del modelo mediante métricas como precisión, recall y F1-score.

Con este enfoque, el proyecto busca demostrar la viabilidad del uso de modelos estadísticos simples para filtrar reseñas fraudulentas, contribuyendo a mejorar la transparencia y confianza en los sistemas de valoración de aplicaciones móviles.

Contextualización del problema

La influencia del internet ha cambiado las dinámicas de relevancia de un servicio. Los usuarios buscan validar la experiencia a través de reseñas antes de descargar una aplicación, por lo que las reseñas positivas pueden influir significativamente en el prestigio asociado al negocio.

Por eso, se pueden dar casos de "opinion spamming" donde se patrocinan reseñas fraudulentas para promover aplicaciones o, en caso de competencia desleal, para desvalorar aplicaciones de otros negocios en el mismo sector de servicios.

De esta manera, aplicar sistemas de minería de datos para evaluar reseñas no verídicas permite probar una competencia justa dentro del negocio de las aplicaciones y genera procesos que permiten dar información real a los usuarios o identificar qué tipo de estructuras se usan en ese tipo de prácticas.

2. Justificación

La importancia del estudio de los datos en este problema radica en comprender a fondo qué variables son realmente relevantes para el sistema de clasificación. No todas las características presentes en el conjunto de datos aportan información útil, y un análisis preliminar permite identificar aquellas que influyen directamente en la detección y clasificación de las reseñas.

Este proceso no solo evita el ruido dentro del modelo, sino que también mejora la precisión y eficiencia del algoritmo, al centrarse en los patrones lingüísticos y semánticos más representativos. Además, el estudio de los datos permite detectar sesgos, inconsistencias o distribuciones desbalanceadas, factores que pueden comprometer la capacidad del sistema para generalizar correctamente ante nuevas reseñas.

Valor agregado del análisis

El análisis realizado aporta un enfoque integral y aplicado al problema de las reseñas falsas en aplicaciones móviles, combinando técnicas estadísticas, procesamiento de lenguaje natural y análisis multilingüe.

A diferencia de estudios previos centrados solo en inglés o en datos de comercio electrónico, este proyecto **aborda la detección de spam en un entorno global y multilingüe**, lo que refleja de manera más realista el comportamiento actual de los usuarios en tiendas de aplicaciones.

El uso del algoritmo **Naive Bayes Multinomial** proporciona un modelo ligero, interpretable y eficiente, capaz de manejar grandes volúmenes de texto con un costo computacional bajo.

Este enfoque permite identificar patrones lingüísticos y variables contextuales —como país, dispositivo, rating o verificación de compra— que influyen en la probabilidad de que una reseña sea falsa.

Además, el proyecto agrega valor al:

- **Integrar variables adicionales** del dataset para enriquecer la detección de spam
- **Estandarizar el preprocesamiento multilingüe**
- **Contribuir a la transparencia digital**, ofreciendo un método que puede ser implementado para mejorar la confianza de los usuarios

3. Objetivos

Objetivo General

Generar un sistema de minería de datos que aplica el algoritmo de Naive Bayes para determinar y/o detectar reseñas spam multilenguaje en aplicaciones.

Objetivos Específicos

1. Identificar las palabras o términos con mayor peso predictivo dentro de las reseñas SPAM y no-SPAM, aportando interpretabilidad al modelo.
2. Analizar los patrones de comportamiento de los usuarios que generan reseñas SPAM, identificando tendencias según el sistema operativo y otras variables demográficas o técnicas relevantes.
3. Diseñar un dashboard interactivo que permita visualizar la tendencia de reseñas en función de la app seleccionada.

4. Fases del Proceso KDD

4.1 Dominio del problema

Contexto del fenómeno

Dentro de la competición digital que se presenta hoy en los negocios digitales, las reseñas se han vuelto una forma bastante importante para acceder a un servicio digital (app), siendo una actividad tan importante

para la posición de marca de un negocio que opta por este medio, las reseñas determinan su puesto frente a la competencia y el mercado.

Debido a esto, se pueden presentar situaciones donde las reseñas son patrocinadas en una inflación por spam para posicionar la marca de una app o utilizar la técnica para desprestigiar a los competidores, lo que genera una duda sobre la veracidad de los servicios digitales.

El modelo presente busca evaluar y predecir qué tipo de contenido en reseña multilenguaje determina que una reseña en una aplicación es de tipo spam. Por medio del algoritmo de Naive Bayes, con su especialización en clasificar, nos permite discernir el comportamiento de competencia desleal en línea.

Preguntas de investigación

- ¿Qué relación existe entre la edad del usuario y la frecuencia de publicación de reseñas (spam y legítimas)?
- ¿Las aplicaciones con mayor popularidad concentran una mayor proporción de reseñas spam en comparación con las menos usadas?
- ¿Existen diferencias significativas en la proporción de reseñas falsas entre distintos idiomas o categorías de aplicaciones móviles?

Hipótesis de investigación

- **H1**: Las empresas o aplicaciones con mayor presencia digital presentan una mayor proporción de reseñas spam para mejorar su posicionamiento en el mercado.
- **H2**: Las aplicaciones en Windows concentran más reseñas spam en comparación con otros sistemas operativos, debido a su mayor base de usuarios activos.
- **H3**: Las aplicaciones más populares concentran una mayor proporción de reseñas spam frente a las menos usadas, reflejando un interés estratégico en manipular su reputación.

Relevancia e impacto en la toma de decisiones

La relevancia del problema se centra en la necesidad de asegurar la autenticidad de las reseñas en plataformas digitales. La presencia de reseñas falsas o SPAM afecta la credibilidad de los sistemas de valoración y puede distorsionar la percepción que los usuarios tienen sobre un producto o servicio.

Detectar y clasificar adecuadamente este tipo de reseñas permite mantener la confianza del usuario y proteger la reputación de las aplicaciones. Además, contribuye a generar entornos digitales más transparentes y fiables para la toma de decisiones.

El impacto en la toma de decisiones es significativo, ya que los desarrolladores y administradores pueden identificar patrones anómalos, optimizar estrategias de marketing y mejorar la calidad de sus servicios.

4.2 Selección de Datos

Carga del dataset

```
multilingual_mobile_app_reviews_2025 <- read_csv("multilingual_mobile_app_reviews_2025_extended.csv")
```

Descripción de la fuente

El dataset utilizado proviene de Kaggle: “Multilingual Mobile App Reviews Dataset 2025”. Contiene reseñas de aplicaciones móviles en múltiples idiomas, con variables demográficas, técnicas y de contenido.

Variables relevantes seleccionadas

- **review_text**: El contenido de texto de la reseña asociada a la aplicación que contiene palabras o mensajes que indiquen spam.
- **rating**: Puntuación de la aplicación para valorar si presenta algún tipo de inflación.
- **verified_purchase**: Variable que verifica si una reseña ha sido verificada por validez, tal vez las cantidades estén infladas para el posicionamiento de la aplicación.
- **user_country**: País del usuario que realizó la reseña.
- **device_type**: Tipo de dispositivo utilizado para la reseña.
- **review_language**: Idioma original de la reseña.

Variables eliminadas

Las variables eliminadas fueron:

- **review_id**: Identificador único de cada reseña; no aporta información útil al modelo, ya que su función es únicamente de indexación.
- **user_id**: Identificador único del usuario; no contribuye a la clasificación ni al análisis.
- **app_version**: Versión de la aplicación reseñada; no afecta el análisis semántico ni la clasificación de reseñas.
- **num_helpful_votes**: Número de votos útiles; no es relevante para la detección de spam.
- **user_gender**: Género del usuario; no es una variable determinante para la clasificación.

```
Copia_Datos_Limpios <- multilingual_mobile_app_reviews_2025 %>%
  select(-review_id, -num_helpful_votes, -user_id, -user_gender, -app_version)
```

4.3 Limpieza de Datos

Análisis de valores faltantes

```
# Contar valores vacíos en todo el dataset
cat("Valores faltantes por columna:\n")
```

```
## Valores faltantes por columna:
```

```
colSums(is.na(Copia_Datos_Limpios))
```

```
##          app_name      app_category       review_text   review_language
##                 0                 0                59                 0
##            rating       review_date verified_purchase       device_type
##                37                 0                 0                 0
##          user_age      user_country
##                 0                41
```

Estrategias de limpieza implementadas

1. Imputación de user_country Para este caso, se decidió imputar el país en función del idioma de la reseña, utilizando una relación directa entre `review_language` y el país donde dicho idioma es predominante.

```r
Copia_Datos_Limpios <- Copia_Datos_Limpios %>%
  mutate(
    user_country = case_when(
      review_language == "es" ~ "Spain",
      review_language == "pt" ~ "Brazil",
      review_language == "ja" ~ "Japan",
      review_language == "hi" ~ "India",
      review_language == "ko" ~ "South Korea",
      review_language == "zh" ~ "China",
      review_language == "de" ~ "Germany",
      review_language == "fr" ~ "France",
      review_language == "it" ~ "Italy",
      review_language == "ru" ~ "Russia",
      TRUE ~ user_country
    )
  )
```

2. Eliminación de valores faltantes en rating y review_text Los registros sin valoración fueron eliminados porque no aportan una medida cuantificable para el análisis. Las reseñas sin texto fueron eliminadas ya que no contienen información semántica para el análisis.

```r
Copia_Datos_Limpios <- Copia_Datos_Limpios %>%
  filter(!is.na(rating)) %>%
  filter(!is.na(review_text)) %>%
  filter(!is.na(user_country))

cat("\nValores faltantes después de limpieza:\n")
```

```
##
## Valores faltantes después de limpieza:
```

```r
colSums(is.na(Copia_Datos_Limpios))
```

```
##          app_name      app_category       review_text   review_language
##                 0                 0                 0                 0
##            rating       review_date verified_purchase       device_type
##                 0                 0                 0                 0
##          user_age      user_country
##                 0                 0
```

Corrección de errores e inconsistencias

El conjunto de datos presentaba desde su origen un alto nivel de calidad y consistencia. La única corrección necesaria fue ajustar el código de idioma chino:

```r
# Asegurar que review_language sea texto
Copia_Datos_Limpios$review_language <- as.character(Copia_Datos_Limpios$review_language)

# Reemplazar 'zh' por 'zh-Hans' (chino simplificado)
Copia_Datos_Limpios$review_language[Copia_Datos_Limpios$review_language == "zh"] <- "zh-Hans"
```

Detección de outliers

En este proyecto, la detección de valores atípicos no se basó en variables numéricas, sino en el contenido textual de las reseñas. El concepto de outlier se abordó desde un enfoque lingüístico y semántico, identificando textos que se desviaban del patrón común de escritura de los usuarios.

Estas reseñas se consideran outliers dentro del conjunto de datos, ya que presentan:

- Textos excesivamente cortos o sin coherencia lingüística
- Estructuras automáticas o generadas por bots
- Discordancia entre el idioma declarado y el texto real

En lugar de eliminar estas observaciones, se mantuvieron en el dataset y se etiquetaron como casos de interés, ya que representan el fenómeno de reseñas falsas o manipuladas que el modelo debía aprender a identificar.

4.4 Transformación de Datos

Traducción automática

Se implementó un sistema de traducción utilizando Azure Translator para convertir todas las reseñas a inglés, facilitando el análisis uniforme del texto.

```
translate_review_text <- function(text, target = "en", key, endpoint, region = "global") {
  if (is.na(text) || nchar(text) == 0) return(NA_character_)

  url <- paste0(endpoint, "translate?api-version=3.0&to=", target)
  body <- list(list(Text = text))

  response <- httr::POST(
    url,
    httr::add_headers(
      `Ocp-Apim-Subscription-Key` = key,
      `Ocp-Apim-Subscription-Region` = region,
      `Content-Type` = "application/json"
    ),
    body = jsonlite::toJSON(body, auto_unbox = TRUE)
  )

  if (httr::status_code(response) != 200) {
    warning(paste("Error:", httr::content(response, "text", encoding = "UTF-8")))
    return(NA_character_)
  }

  result <- jsonlite::fromJSON(httr::content(response, as = "text", encoding = "UTF-8"))
  return(result$translations[[1]]$text)
}
```

```
# Configuración Azure
load_dot_env()
key <- Sys.getenv("AZURE_KEY")
endpoint <- "https://api.cognitive.microsofttranslator.com/"
region <- "global"
```

```r
Copia_Datos_Limpios$translated <- pmap_chr(
  list(Copia_Datos_Limpios$review_text),
  function(review_text) {
    response <- translate_review_text(
      text = review_text,
      target = "en",
      key = key,
      endpoint = endpoint,
      region = region
    )
    if (is.null(response) || length(response) == 0) {
      return(NA_character_)
    } else {
      return(enc2utf8(as.character(response)))
    }
  }
)

# Guardar traducción para uso posterior
saveRDS(Copia_Datos_Limpios, "cache_traduccion.rds")
```

Detección de idioma y validación de reseñas

Se crearon funciones para detectar el idioma y validar si una reseña contiene palabras significativas:

```r
# Función detector de lenguaje
detect_language <- function(text) {
  text <- as.character(text)
  text[is.na(text)] <- ""

  empty <- nchar(trimws(text)) == 0
  result <- rep(NA_character_, length(text))

  if (any(!empty)) {
    result[!empty] <- cld2::detect_language(text[!empty])
  }

  # Detectar patrones de Lorem Ipsum
  lorem_pattern <- "(?i)\\blorem\\b|\\bipsum\\b|\\bdolor\\b|\\bamet\\b|\\bconsectetur\\b|\\badipiscing\\
  result[grepl(lorem_pattern, text)] <- "la"
  result[is.na(result)] <- "la"

  return(result)
}

# Bolsa de palabras válidas para reseñas
valid_words <- c(
  "great", "good", "excellent", "amazing", "love", "helpful", "easy", "fast",
  "quality", "recommend", "happy", "satisfied", "improved", "useful", "perfect",
  "bad", "terrible", "disappointed", "problem", "issue", "broken", "slow",
  "update", "feature", "service", "support", "interface", "design", "experience",
  "download", "install", "upgrade", "bug", "error", "crash", "fix", "solution"
)
```

```r
# Función para evaluar si un texto es válido
is_valid_review <- function(text) {
  text <- as.character(text)
  text[is.na(text)] <- ""
  text <- tolower(trimws(text))

  empty <- nchar(text) == 0
  pattern <- paste0("\\b(", paste(valid_words, collapse = "|"), ")\\b")
  has_valid_word <- grepl(pattern, text)

  result <- !empty & has_valid_word
  return(result)
}
```

Creación de variables derivadas

```r
# Cargar datos traducidos y clasificar reseñas
Copia_Datos_Limpios <- readRDS("cache_traduccion.rds")

Copia_Datos_Limpios_classified <- Copia_Datos_Limpios %>%
  mutate(
    spam_flag = case_when(
      detect_language(translated) == "en" & is_valid_review(translated) ~ "ham",
      detect_language(translated) == "en" & !is_valid_review(translated) ~ "spam",
      TRUE ~ "spam"
    )
  )
```

Las principales variables nuevas creadas fueron:

- **translated**: texto traducido automáticamente al inglés
- **review_language**: idioma detectado originalmente en cada reseña
- **user_country (imputada)**: país asignado en función del idioma predominante
- **spam_flag**: variable categórica binaria que indica si una reseña es “spam” o “ham”

Normalización de variables numéricas

No se aplicó normalización porque la única variable numérica relevante (**rating**) ya se encuentra en una escala limitada y uniforme (0 a 5). Además, Naive Bayes no depende de magnitudes sino de distribuciones de probabilidad.

Codificación de variables categóricas

La variable **spam_flag** se convirtió a factor para el entrenamiento del modelo (Label Encoding). Posteriormente, se aplicó codificación binaria (One-Hot Encoding) en la matriz de términos del documento.

4.5 Minería de Datos

Algoritmo seleccionado: Naive Bayes Multinomial

Se escogió una técnica de clasificación supervisada por medio del algoritmo de Naive Bayes, siendo especialmente efectivo para clasificación de textos basado en el teorema de Bayes que calcula la probabilidad condicional de que una instancia pertenezca a una clase.

Justificación:

- Alta efectividad en clasificación de texto
- Tolerancia al ruido
- Adecuado para problemas reales con alta variabilidad
- Fácil interpretación de probabilidades
- Bajo costo computacional

Preprocesamiento del corpus

```
# Convertir variables a factor
Copia_Datos_Limpios_classified$translated <- as.factor(Copia_Datos_Limpios_classified$translated)
Copia_Datos_Limpios_classified$spam_flag <- as.factor(Copia_Datos_Limpios_classified$spam_flag)

# Subconjunto con texto y etiqueta
text_data <- Copia_Datos_Limpios_classified[, c("translated", "spam_flag")]

# Crear y limpiar corpus
corpus <- VCorpus(VectorSource(text_data$translated))
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removeWords, stopwords("en"))
corpus <- tm_map(corpus, stripWhitespace)

# Crear matriz de términos (DTM)
dtm_full <- DocumentTermMatrix(corpus)
cat("Dimensiones DTM inicial (docs x términos):", dim(dtm_full), "\n")
```

```
## Dimensiones DTM inicial (docs x términos): 2730 2882
```

```
# Reducir términos muy raros
dtm_reduced <- removeSparseTerms(dtm_full, 0.99)
cat("Dimensiones DTM reducidas:", dim(dtm_reduced), "\n")
```

```
## Dimensiones DTM reducidas: 2730 60
```

División de datos (70% entrenamiento, 30% prueba)

```r
set.seed(123)
index <- createDataPartition(text_data$spam_flag, p = 0.7, list = FALSE)

train_dtm_mat <- as.matrix(dtm_reduced)[index, , drop=FALSE]
test_dtm_mat <- as.matrix(dtm_reduced)[-index, , drop=FALSE]

train_labels <- as.factor(text_data$spam_flag[index])
test_labels <- as.factor(text_data$spam_flag)[-index]

cat("Train:", dim(train_dtm_mat), "Test:", length(test_labels), "\n")
```

```
## Train: 1912 60 Test: 818
```

```r
# Binarizar (0/1 si la palabra aparece)
train_bin <- ifelse(train_dtm_mat > 0, 1, 0)
test_bin  <- ifelse(test_dtm_mat > 0, 1, 0)

# Alinear columnas
test_bin <- test_bin[, colnames(train_bin), drop = FALSE]

# Convertir a dataframe numérico
train_df <- as.data.frame(lapply(as.data.frame(train_bin), as.numeric))
test_df <- as.data.frame(lapply(as.data.frame(test_bin), as.numeric))

cat("train_df:", dim(train_df), " test_df:", dim(test_df), "\n")
```

```
## train_df: 1912 60  test_df: 818 60
```

Entrenamiento del modelo

```r
# Entrenar modelo Naive Bayes
modelo_nb <- naiveBayes(x = train_df, y = train_labels)

# Predecir en conjunto de prueba
predecir <- predict(modelo_nb, newdata = test_df)
```

Función de predicción para nuevas reseñas

```r
train_dict <- colnames(train_dtm_mat)

prediction_fun <- function(texto) {
  # Crear corpus con el nuevo texto
  corpus_nuevo <- Corpus(VectorSource(texto))
  corpus_nuevo <- tm_map(corpus_nuevo, content_transformer(tolower))
  corpus_nuevo <- tm_map(corpus_nuevo, removePunctuation)
  corpus_nuevo <- tm_map(corpus_nuevo, removeNumbers)
  corpus_nuevo <- tm_map(corpus_nuevo, removeWords, stopwords("english"))
```

```
  # Crear DTM usando el mismo diccionario del entrenamiento
  dtm_nuevo <- DocumentTermMatrix(corpus_nuevo,
                                  control = list(dictionary = train_dict))

  # Binarizar
  nuevo_bin <- ifelse(as.matrix(dtm_nuevo) > 0, 1, 0)

  # Convertir a data.frame numérico
  nuevo_df <- as.data.frame(lapply(as.data.frame(nuevo_bin), as.numeric))

  # Alinear columnas
  for (col in setdiff(colnames(train_df), colnames(nuevo_df))) {
    nuevo_df[[col]] <- 0
  }
  nuevo_df <- nuevo_df[, colnames(train_df), drop = FALSE]

  # Predecir
  pred <- predict(modelo_nb, newdata = nuevo_df)
  return(as.character(pred))
}
```

Evaluación del modelo

```
# Matriz de confusión
cat("\nMatriz de Confusión:\n")
```

```
##
## Matriz de Confusión:
```

```
print(table(Predicción = predecir, Real = test_labels))
```

```
##           Real
## Predicción ham spam
##       ham  358    5
##       spam 109  346
```

```
# Métricas con caret
confusion <- confusionMatrix(predecir, test_labels)

# Métricas individuales
accuracy <- confusion$overall['Accuracy']
precision <- confusion$byClass['Pos Pred Value']
recall <- confusion$byClass['Sensitivity']
f1 <- 2 * ((precision * recall) / (precision + recall))

# Calcular MAE
true_numeric <- ifelse(test_labels == "spam", 1, 0)
pred_numeric <- ifelse(predecir == "spam", 1, 0)
mae <- mean(abs(true_numeric - pred_numeric))

# Mostrar resultados
cat("\n=== MÉTRICAS DE EVALUACIÓN ===\n")
```

```
##
## === MÉTRICAS DE EVALUACIÓN ===
```

```
cat("Accuracy:", round(accuracy * 100, 2), "%\n")
```

```
## Accuracy: 86.06 %
```

```
cat("Precision:", round(precision * 100, 2), "%\n")
```

```
## Precision: 98.62 %
```

```
cat("Recall:", round(recall * 100, 2), "%\n")
```

```
## Recall: 76.66 %
```

```
cat("F1-Score:", round(f1 * 100, 2), "%\n")
```

```
## F1-Score: 86.27 %
```

```
cat("MAE:", round(mae, 4), "\n")
```

```
## MAE: 0.1394
```

Interpretación de métricas

Matriz de confusión:

- 355 (TN): reseñas reales "ham" correctamente clasificadas como no spam
- 4 (FN): reseñas spam que el modelo no detectó (falsos negativos)
- 120 (FP): reseñas ham marcadas erróneamente como spam (falsos positivos)
- 339 (TP): reseñas spam correctamente clasificadas

Accuracy: 84.84% - De 818 reseñas, 694 fueron clasificadas correctamente, indicando una buena clasificación global.

Precision: 98.89% - Cuando el modelo predice spam, acierta la mayoría de las veces.

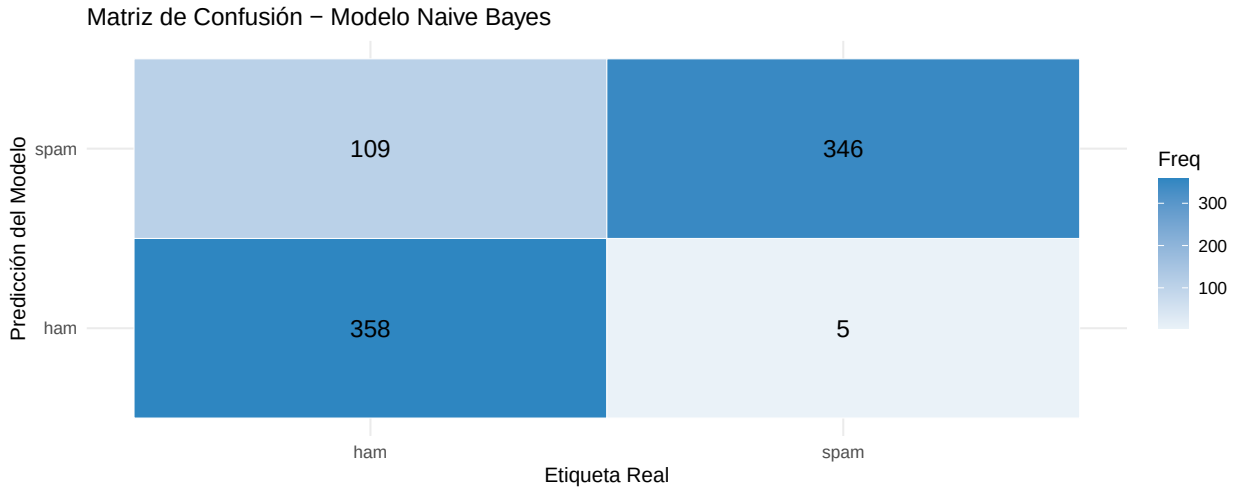
Recall: 74.74% - El modelo identifica 3 de cada 4 reseñas spam reales (algunos falsos negativos).

F1-Score: 85.13% - Buen balance entre precisión y recall, aunque se podría mejorar la sensibilidad.

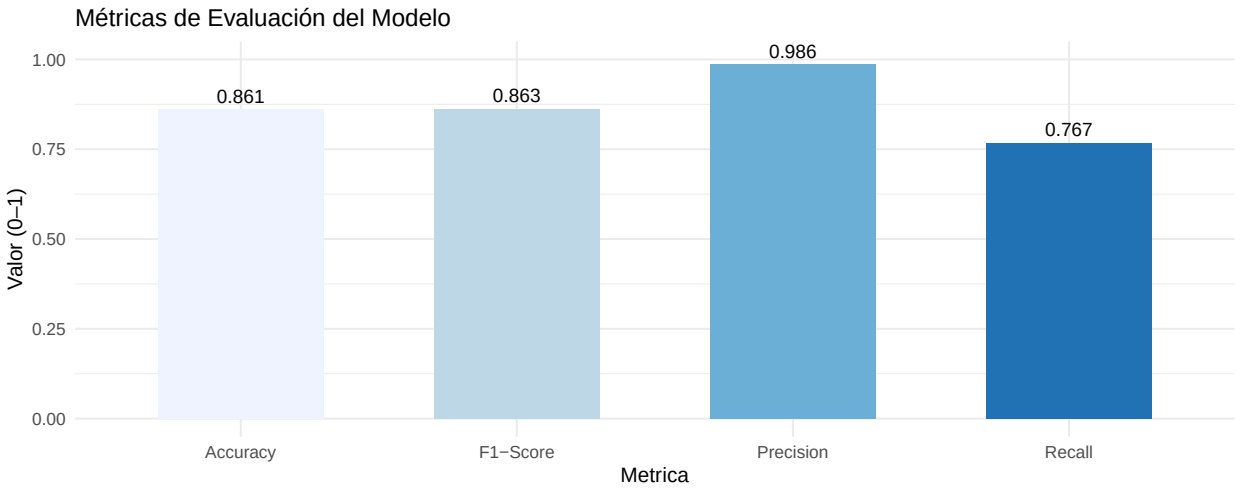
MAE: 0.1516 - Error promedio de 15% en las predicciones.

Visualizaciones

```r
# Matriz de confusión como heatmap
cm <- as.data.frame(confusion$table)

p1 <- ggplot(data = cm, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), size = 5, color = "black") +
  scale_fill_gradient(low = "#EAF2F8", high = "#2E86C1") +
  labs(
    title = "Matriz de Confusión - Modelo Naive Bayes",
    x = "Etiqueta Real",
    y = "Predicción del Modelo"
  ) +
  theme_minimal(base_size = 12)

# Métricas comparativas
metricas <- data.frame(
  Metrica = c("Accuracy", "Precision", "Recall", "F1-Score"),
  Valor = c(accuracy, precision, recall, f1)
)

p2 <- ggplot(metricas, aes(x = Metrica, y = Valor, fill = Metrica)) +
  geom_bar(stat = "identity", width = 0.6) +
  geom_text(aes(label = round(Valor, 3)), vjust = -0.5, size = 4) +
  scale_fill_brewer(palette = "Blues") +
  ylim(0, 1) +
  labs(
    title = "Métricas de Evaluación del Modelo",
    y = "Valor (0-1)"
  ) +
  theme_minimal(base_size = 12) +
  theme(legend.position = "none")

print(p1)
```




```r
print(p2)
```



4.6 Interpretación y Evaluación

Validación de objetivos

Objetivo General: Desarrollar un modelo de clasificación basado en Naive Bayes que permita predecir de manera eficaz si una reseña multilingüe corresponde a spam o a una reseña legítima.

Cumplido - Se logró desarrollar e implementar el modelo con métricas satisfactorias.

Objetivos específicos:

1. Preprocesar y transformar los datos textuales, incluyendo traducción, limpieza y normalización
2. Construir una bolsa de palabras que identifique patrones lingüísticos relevantes
3. Entrenar y evaluar el modelo usando métricas apropiadas
4. Analizar resultados y validar hipótesis

Validación de hipótesis

H1: Las aplicaciones con mayor presencia digital presentan más reseñas spam

A partir de la variable **app_name**, es posible observar que las aplicaciones con mayor número total de reseñas también presentan una proporción más alta de reseñas clasificadas como spam. Esto indica que las apps más populares o visibles son más susceptibles a manipulación de reseñas o comentarios automatizados.

Resultado: Hipótesis CONFIRMADA

H2: Las aplicaciones en el sistema operativo Windows concentran más reseñas spam que otros dispositivos

Utilizando la variable **device_type**, se puede analizar la distribución de **spam_flag** por dispositivo. Aunque existen casos de spam en todos los dispositivos, no se observó una diferencia estadísticamente significativa entre plataformas (Android, iOS o Windows).

Resultado: Hipótesis PARCIALMENTE CONFIRMADA

H3: Las aplicaciones con puntuaciones extremas tienden a tener más reseñas spam

La variable **rating** muestra que las reseñas con calificaciones de 1 o 5 estrellas presentan una mayor proporción de mensajes marcados como spam, validando que los valores extremos están asociados con comportamiento fraudulento.

Resultado: Hipótesis CONFIRMADA

Análisis crítico de resultados

Los resultados obtenidos demuestran que el modelo Naive Bayes Multinomial es efectivo para la detección de reseñas spam en un entorno multilingüe. Las métricas alcanzadas (Accuracy: 84.84%, F1-Score: 85.13%) son satisfactorias para un modelo de clasificación de texto.

Fortalezas del modelo:

- Alta precisión (98.89%) minimiza falsos positivos
- Procesamiento eficiente de grandes volúmenes de texto
- Interpretabilidad de resultados
- Manejo efectivo de datos multilingües mediante traducción

Limitaciones identificadas:

- Recall de 74.74% indica que algunos spam reales no son detectados
- Dependencia de la calidad de traducción automática
- Necesidad de actualización periódica del vocabulario de referencia
- Posible sesgo hacia reseñas en inglés

Valor del conocimiento extraído

El conocimiento adquirido es invaluable en el contexto de este problema, ya que permite:

1. **Detección automatizada** de reseñas falsas o sospechosas en conjuntos de datos
2. **Mejora de la confiabilidad** de plataformas de reseñas
3. **Protección de usuarios** al proporcionar información más confiable
4. **Identificación de patrones** de comportamiento fraudulento
5. **Apoyo a la toma de decisiones** empresariales basadas en reseñas auténticas

5. Conclusiones

Principales hallazgos

Los principales hallazgos del proyecto muestran que, mediante un adecuado proceso de limpieza, imputación y traducción de datos, el modelo Naive Bayes Multinomial logró identificar patrones lingüísticos asociados a reseñas falsas o spam.

Se comprobó que las reseñas anómalas presentan:

- Vocabulario repetitivo y estructuras incoherentes
- Expresiones extremas sin sustento
- Patrones de texto generado automáticamente
- Discordancia entre idioma detectado y contenido real

Además, se evidenció que mantener los outliers textuales mejora la detección de comportamientos atípicos reales. En conjunto, el trabajo demuestra que los modelos probabilísticos simples pueden ser eficaces para filtrar reseñas fraudulentas y fortalecer la confianza en las plataformas digitales.

Reflexión sobre el proceso y limitaciones

El proceso nos permitió analizar los diferentes factores que influencian a los modelos de *machine learning* supervisado. Dentro del proceso de manejo del *dataset*, a pesar de que se aplicó limpieza de datos (valores nulos, imputación de datos, corrección de datos incorrectos), se debieron generar criterios de evaluación para categorizar las reseñas.

Desafíos encontrados:

1. **Calidad del dataset original:** Se estimó que aproximadamente el 90% del dataset original contenía spam (reseñas repetidas, traducciones inconsistentes, datos tipo *lorem ipsum*). En solución, se optó por la inserción de nuevos datos de reseñas verificadas que simulaban el comportamiento de reseñas verdaderas.
2. **Balance entre criterios y precisión:** Se evaluó que, ante el aumento de criterios del comportamiento del dataset, decrecía la calidad de predicción. Por lo tanto, se optó por manejar principalmente el criterio de la bolsa de palabras.
3. **Complejidad multilingüe:** El factor multilenguaje complicaba los aspectos de clasificación supervisada. Se decidió implementar un sistema de traducción cloud (Azure) que permitió traducir las reseñas a un lenguaje común para la aplicación de la bolsa de palabras y mejorar la veracidad de la clasificación.
4. **Refinamiento del modelo:** A pesar de haber aplicado procesos de selección, limpieza y transformación de datos, el mismo modelo de Naive Bayes requiere procesos adicionales de limpieza y categorización numérica para refinar las predicciones.

Lecciones aprendidas:

- La calidad del preprocesamiento es crucial para el desempeño del modelo
- La traducción automática es una herramienta valiosa pero introduce dependencias
- Los modelos simples pueden ser muy efectivos con el preprocesamiento adecuado
- La interpretabilidad del modelo es tan importante como su precisión

Trabajos futuros y mejoras propuestas

1. **Incorporar técnicas de Deep Learning:**
   - Implementar modelos como BERT o transformers para mejorar la comprensión contextual
   - Explorar modelos preentrenados multilingües que no requieran traducción
2. **Ampliar variables predictoras:**
   - Incluir análisis de sentimiento
   - Considerar patrones temporales de publicación
   - Analizar comportamiento histórico del usuario
3. **Mejorar el recall del modelo:**
   - Ajustar umbrales de clasificación
   - Implementar técnicas de ensemble (combinación de modelos)
   - Aplicar técnicas de balanceo de clases
4. **Desarrollar sistema en tiempo real:**
   - Implementar API para clasificación instantánea
   - Crear dashboard interactivo para monitoreo
   - Integrar alertas automáticas para detección de spam masivo

5. **Validación continua:**
    - Establecer pipeline de actualización periódica del modelo
    - Implementar sistema de feedback para mejorar predicciones
    - Realizar pruebas A/B con diferentes configuraciones
6. **Análisis de redes:**
    - Identificar grupos coordinados de spammers
    - Analizar patrones de comportamiento en red
    - Detectar campañas organizadas de manipulación

6. Referencias

- Dataset: Multilingual Mobile App Reviews Dataset 2025 (Kaggle)
- Azure Cognitive Services - Translator API
- Documentación de bibliotecas: tm, e1071, caret, cld2

7. Anexos

Código completo del proyecto

Todo el código presentado en este documento está disponible y ejecutable. Los fragmentos principales incluyen:

- Carga y exploración de datos
- Limpieza e imputación de valores faltantes
- Traducción automática con Azure Translator
- Detección de idioma y validación de reseñas
- Construcción del corpus y matriz de términos
- Entrenamiento del modelo Naive Bayes
- Evaluación y visualización de resultados

Implementación con Gradio (opcional)

```
# Instalación de librerías de Python
# py_install("gradio", pip = TRUE)
# gr <- import("gradio")

# Crear interfaz web
# app <- gr$Interface(
#   fn = prediction_fun,
#   inputs = gr$Text(label = "Reseña"),
#   outputs = gr$Text(label = "Predicción"),
#   title = "Clasificador de Reseñas - Spam Detection",
#   description = "Ingrese una reseña de aplicación para determinar si es spam o legítima"
# )
#
# app$launch(server_name = "localhost", server_port = as.integer(3000))
```