Mundo Fesc E-ISSN: 2216-0388 P- ISSN: 2216-0353 Artículo Original

# Modelo de aprendizaje supervisado para la identificación automática de spam en entornos de mensajería móvil

## Supervised Learning Model for the Automatic Identification of Spam in Mobile Messaging Environments

**Recibido:** 6 de septiembre del 2024
**Aprobado:** 23 de diciembre del 2024
**publicación:** 1 de Mayo del 2025

**Forma de citar:** C. Esquea Osorio, M. . Riascos Carvajal, and D. Guevara Ibarra, "Modelo de Aprendizaje Supervisado para la Identificación Automática de Spam en Entornos de Mensajería Móvil", Mundo Fesc, vol. 15, no. 32,pp. 385-399 May 2025, doi: 10.61799/2216-0388.1797.

**Camilo Esquea Osorio.**
Ingeniero electrónico,
camiloandresesos@ufps.edu.co,
https://orcid.org/0000-0002-6734-5076,
Universidad Francisco de Paula Santander,
Cúcuta, Colombia

**Melissa Riascos Carvajal.**
Ingeniero electrónico,
melissarc@ufps.edu.co,
https://orcid.org/0009-0009-2047-678X,
Universidad Francisco de Paula Santander,
Cúcuta, Colombia.

**Dinael Guevara.**
Doctor en ingeniería,
dinaelgi@ufps.edu.co,
https://orcid.org/0000-0003-3007-8354,
Universidad Francisco de Paula Santander,
Cúcuta, Colombia.

***Autor para correspondencia:**
Email: E-mail: camiloandresesos@ufps.edu.co

 

# Modelo de Aprendizaje Supervisado para la Identificación Automática de Spam en Entornos de Mensajería Móvil


## Resumen

Este artículo presenta un modelo de aprendizaje supervisado para la detección automática de mensajes no deseados (spam) en entornos de mensajería móvil, utilizando datos del repositorio Hugging Face. La metodología incluyó las etapas de limpieza y normalización del texto, traducción al español, análisis exploratorio y representación de los mensajes mediante la técnica de bolsa de palabras (CountVectorizer). Posteriormente, el conjunto de datos fue dividido en particiones de entrenamiento y prueba, y se entrenó un clasificador Random Forest con hiperparámetros ajustados manualmente. Los resultados muestran una exactitud global del 96 %, una precisión de 1.00 para la clase spam y una mejora significativa del F1-score (0.82). Además, la validación cruzada de cinco pliegues obtuvo una media de exactitud del 94.4 %, evidenciando estabilidad en el modelo. En conjunto, estos hallazgos confirman que el modelo propuesto es una alternativa eficaz para la detección de spam en aplicaciones de mensajería móvil.


**Palabras clave:** aprendizaje automático, clasificación,mensajería móvil, modelo supervisado, procesamiento de texto, Random Forest, spam, vectorización de texto.

# Supervised Learning Model for the Automatic Identification of Spam in Mobile Messaging Environments


## Abstract

This article presents a supervised learning model for the automatic detection of unwanted messages (spam) in mobile messaging environments, using data obtained from the Hugging Face repository. The methodology included text cleaning and normalization, translation into Spanish, exploratory analysis, and message representation through the bag-of-words technique (CountVectorizer). The dataset was then split into training and testing sets, and a Random Forest classifier was trained using manually tuned hyperparameters. The results show an overall accuracy of 96%, a precision of 1.00 for the spam class, and a significant improvement in its F1-score (0.82). Additionally, five-fold cross-validation reported a mean accuracy of 94.4%, demonstrating the model's stability. Overall, the proposed approach proves to be an effective alternative for spam detection in mobile messaging applications.


**Keywords: classification, machine learning, mobile messaging, Random Forest, spam, supervised model, text processing, text vectorization.**

## Introducción

Hoy en día, la forma en que nos comunicamos ha cambiado drásticamente gracias al crecimiento de aplicaciones que permiten una mensajería instantánea como WhatsApp, Telegram y Signal la cuales, se han convertido en herramientas comunes tanto para conversaciones personales como incluirse en asuntos laborales. No obstante, con su popularidad también ha aumentado de manera preocupante la presencia de mensajes no deseados, conocidos como spam [1]. Este tipo de contenido no solo afecta la privacidad de los usuarios [2], [3], sino que también satura los espacios de comunicación y puede llegar a generar riesgos importantes en términos de seguridad de los usuarios. Además, a diferencia de lo que ocurre con el correo electrónico [4] o los mensajes de texto cortos, el spam en servicios de mensajería instantánea suele pasar por alto los filtros tradicionales obligando a buscar métodos más avanzados para poder detectarlo de forma automática y eficaz [1].

En la detección de spam en mensajería móvil se ha visto que los modelos de aprendizaje supervisado funcionan muy bien, ya que permiten entrenar clasificadores con ejemplos previamente etiquetados ayudando a reconocer patrones tanto en el contenido como en el contexto donde aparecen dichos mensajes [5], [6], [7]. Además, se han estudiado varias técnicas como Naive Bayes, Support Vector Machines (SVM) y Random Forest para abordar esta problemática [3], [7], [8]. Entre ellas, Random Forest ha sido especialmente valorada obteniendo resultados satisfactorios por su capacidad para trabajar con datos con ruido [9], [10], permitiendo evitar el sobreajuste y obtener eficacia en la clasificación de textos [11], [12]. Por otro lado, diferentes investigaciones han destacado su buen rendimiento al aplicarse en mensajes poco estructurados, como los que suelen circular en aplicaciones de mensajería móvil o plataformas similares [8], [13].

De esta manera, se ha comprobado que escoger bien las técnicas de preprocesamiento, como normalizar el texto, eliminar ruido lingüístico presente en los datos y representar los mensajes con métodos como CountVectorizer, marca una gran diferencia en cómo se comportan estos modelos de clasificación [14], teniendo en cuenta una gran relevancia en el entorno móvil, donde los mensajes suelen ser cortos, informales y muchas veces incluyen elementos que no son netamente textuales [6]. De igual forma, en áreas exploradas como la detección de noticias falsas, el análisis de emociones en redes sociales [15] o el estudio de contenido engañoso en plataformas digitales y plataformas web [16], también se ha demostrado el potencial que tienen los modelos supervisados y otros más complejos como las redes neuronales profundas. Aun así, la clasificación de mensajes spam sigue siendo una prioridad, porque afecta directamente la experiencia de los usuarios y puede comprometer la seguridad de los sistemas de mensajería [17].

## Metodología

Esta investigación tuvo como objetivo analizar el funcionamiento del modelo de clasificación Random Forest para detectar automáticamente mensajes como spam en aplicaciones de mensajería instantánea móvil. Para lograrlo, se partió de un conjunto de datos ya etiquetado y se aplicaron pasos previos de limpieza y preparación del texto, desarrollando un sistema que detecte eficazmente mensajes no deseados en estas aplicaciones. Todo el proceso se llevó a

cabo tomando en cuenta las recomendaciones y avances que se han publicado en investigaciones anteriores sobre el tema.

La metodología para el desarrollo de esta investigación se organizó en cinco etapas que se desarrollaron de manera progresiva, desde la recopilación de los datos de texto a utilizarse hasta el entrenamiento del modelo para detectar mensajes de spam, el proceso se puede visualizar en la Figura 1, en esta se resumen cada una de las fases junto con su función dentro del flujo de trabajo de la investigación. En primer lugar, se recolectó un conjunto de mensajes y texto en idioma español y ya clasificados como spam o no spam, a partir de fuentes confiables de base de datos para clasificadores y redes neuronales [3], [18], [19].

Seguidamente, se pasó a la etapa de preprocesamiento del texto, etapa crucial en la que se limpiaron los textos y se les aplicaron ajustes básicos como eliminar signos de puntuación y espacios vacíos, convertir todo el texto a minúsculas, quitar palabras poco relevantes y dividir el contenido en fragmentos más cortos y útiles mediante técnicas sencillas de tokenización.

Seguidamente, en la tercera fase, se transformaron esos mensajes cortos en datos numéricos que los algoritmos pudieran entender y clasificar, utilizando la técnica conocida como bolsa de palabras [20], [21] a través de CountVectorizer. Una vez hecho esto, se dividió el conjunto de datos en dos partes, la primera para entrenar el modelo y la segunda destinada a pruebas, buscando que los resultados fueran lo más justos y precisos posible. Finalmente, se construyó y entrenó el modelo utilizando el algoritmo Random Forest, que, con toda la información procesada, se permitió el entrenamiento para identificar si un mensaje era spam o no. Cada una de estas etapas está pensada para aportar de forma ordenada a la solución de esta problemática planteada, ayudando a crear un sistema que realmente funcione y que pueda ser evaluado con claridad.

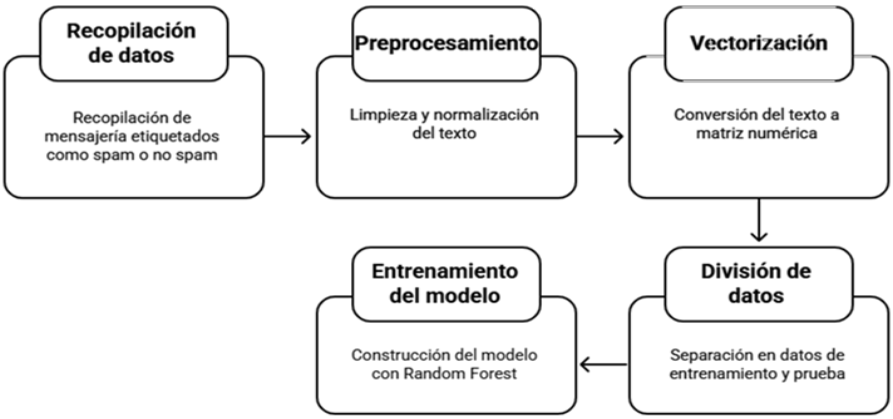


Figura 1. Flujo de trabajo de la metodología.

Continuando con el desarrollo del modelo de clasificación de mensajes, se utilizó el conjunto de datos titulado *SMS Spam Multilingual Collection*, el cual se encuentra disponible públicamente a través del repositorio de Hugging Face. Esta colección de datos de texto contiene mensajes en diversos idiomas, en el que está disponible las columnas de etiqueta de mensaje y dichos datos ya están debidamente etiquetados como spam o ham (no spam), lo que la convierte en

una fuente adecuada para tareas de clasificación mediante modelo de aprendizaje supervisado, considerando el enfoque del proyecto en el idioma español, se seleccionaron únicamente las columnas correspondientes a los mensajes en español (text_es) y sus respectivas etiquetas (label).

Antes de comenzar con el procesamiento de los datos se partió de un conjunto de 5.572 mensajes escritos en español, de los cuales 4.825 estaban etiquetados como mensajes regulares o normales (no spam) y 747 como spam, mostrando un desequilibrio entre ambas categorías, algo que resulta relevante al momento de entrenar y evaluar el modelo, ya que puede influir en su rendimiento de clasificación. Por otro lado, la selección de este conjunto de datos se hizo no solo porque está disponible de manera abierta y libre, sino también porque se ajusta bien tanto en lenguaje como en contenido al objetivo del proyecto, que es detectar automáticamente mensajes no deseados en situaciones reales y conversacionales dentro de aplicaciones de mensajería móvil.

Luego de reunir el conjunto de mensajes etiquetados que se obtuvo del repositorio, se dio inicio a la fase de preprocesamiento de este conjunto de datos, cuyo propósito principal fue preparar adecuadamente los textos para que luego pudieran ser representados en forma numérica, para esto se incluyó varios pasos clave, como la limpieza y normalización del contenido textual, además de un análisis general en los datos permitiendo entender mejor la estructura general del conjunto de datos.

Lo primero que se hizo fue revisar cómo estaban distribuidas las categorías de spam y mensajes normales con un claro desbalance entre las categorías, se revisó si había datos duplicados, y al encontrarlos, se eliminaron para evitar distorsiones en el aprendizaje automático y reducir la posibilidad de sesgos en la clasificación.

Seguidamente, se pasó a normalizar el texto convirtiendo todos los caracteres a minúsculas, quitar signos de puntuación y eliminar aquellos símbolos que no aportaban significado alguna al clasificador. Con estos ajustes, fue posible avanzar a la tokenización de los datos, es decir, dividir el texto en unidades de datos más manejables, lo que ayudó a preparar mejor los datos para la siguiente fase de vectorización, para este proceso no se consideró la eliminación de palabras poco útiles, debido a que se quería dejar el contexto del mensaje tratando de mantener un sentido conversacional de estos datos, lo cual es más común en los mensajes de spam, con estas acciones realizadas se logra reducir el ruido en los datos y dejando una base firme para transformar los textos en representaciones numéricas útiles para el clasificador.

Continuando con la vectorización de este conjunto de datos es necesario que para que los algoritmos de aprendizaje supervisado pudieran procesar los mensajes de texto es necesario transformar estos datos no estructurados en una representación numérica adecuada, estos modelos de machine learning [22], como regresión logística o *Random Forest* implementado en esta investigación requieren que las variables predictoras (x) sean una matriz o dataframe de números reales, mientras que la variable objetivo (y) sea un vector numérico. Como los mensajes en lenguaje natural no cumplen con este formato, se aplicaron técnicas de extracción de características textuales para convertirlos en vectores numéricos.

En esta investigación se utilizó una técnica conocida como *CountVectorizer* la cual permitió convertir mensajes como los que se intercambian en aplicaciones de mensajería, en vectores

con una longitud fija. Esta herramienta se basa en el modelo de “bolsa de palabras”, donde cada mensaje se transforma en un vector en el que cada posición representa una palabra específica del vocabulario que se aprendió previamente, y el número que aparece en esa posición indica cuántas veces se usó esa palabra en el mensaje introducido.

El proceso de *CountVectorizer* se lleva a cabo en dos etapas principales, analizando en primer lugar todo el conjunto de mensajes para detectar las palabras que lo componen y asignarles un número identificador único; esta parte se conoce como la fase de ajuste del mensaje. Seguidamente, en la fase de transformación, se construye una matriz donde cada fila representa un mensaje y cada columna una palabra del vocabulario que se aprendió, con el cual se obtiene una especie de mapa de frecuencias que muestra cuántas veces aparece cada palabra en cada mensaje.

Este procedimiento se puede observar en la Figura 2, en la que CountVectorizer introducido en el algoritmo convierte los mensajes en español en una matriz numérica, en el que a partir de dicha representación que, aunque es sencilla resulta bastante útil para organizar los textos de manera que los algoritmos de clasificación pudieran procesarlos sin problemas [23], [24], [25]. De esta manera, se sentaron las bases para construir el modelo que permitiría identificar los mensajes spam.

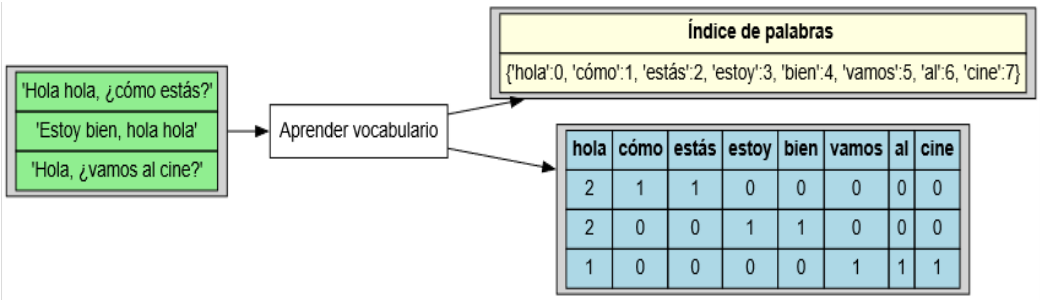


Figura 2. Representación de mensajes mediante la técnica CountVectorizer.

Para poder evaluar de manera objetiva cómo se comportaba el modelo de clasificación, se decidió dividir el conjunto de datos en dos partes: una para la fase de entrenamiento y otra para ponerlo a prueba, con esto se permitió verificar si el modelo era capaz de aplicar lo que había aprendido a datos nuevos ingresados de forma independiente en la estructura del algoritmo, datos que no había visto antes. Para hacer esta separación se utilizó la función *train_test_split*, asignando el 80 % de los datos para entrenar el modelo y dejando el 20 % restante para evaluar su desempeño. Además, se definió un valor específico de semilla aleatoria (*random_state*), lo que ayuda a que los resultados sean reproducibles en futuras ejecuciones del algoritmo, entrenando en múltiples ocasiones.

Esta división también fue útil para evitar que el modelo se ajustara demasiado a los datos de entrenamiento, lo que podría perjudicar su capacidad de adaptarse a nuevas situaciones. Por otro lado, al mantener una distribución proporcional entre las categorías, se garantizó que tanto el conjunto de entrenamiento como el de prueba conservaran una representación adecuada de los mensajes spam y no spam. De esta manera, fue posible realizar una evaluación más justa y cercana a lo que ocurriría en un escenario real donde de igual manera la mayoría de mensajes

son considerados regulares y son pocos mensajes de spam.

Se continua con el entrenamiento del modelo supervisado con los datos ya vectorizados, buscando que con este modelo se pueda identificar de forma automática los mensajes clasificados como spam. Para este propósito se eligió el algoritmo *Random Forest*, esta técnica basada en el ensamblado de múltiples árboles de decisión contribuyo con su capacidad para ofrecer alta precisión, manejar conjuntos de datos desequilibrados y reducir el sobreajuste, características especialmente útiles en tareas de clasificación binaria como la detección de spam.

Dicho modelo se implementó mediante la clase *RandomForestClassifier* de la biblioteca *scikit-learn* en Python, con esta librería se proporciona una implementación robusta del algoritmo, con opciones de configuración que permiten ajustar parámetros clave del modelo. En este caso, se emplearon los valores por defecto para varios hiperparámetros, excepto por el número de árboles para garantizar un equilibrio entre rendimiento y tiempo de entrenamiento del modelo, ya establecida una semilla aleatoria para asegurar la reproducibilidad de los resultados. El entrenamiento se realizó utilizando exclusivamente los datos de entrenamiento generados en la fase anterior, este conjunto de entrenamiento incluía tanto mensajes spam como mensajes legítimos (ham), ya transformados en vectores numéricos mediante la técnica de *CountVectorizer*. El modelo fue ajustado (*fit*) sobre estos vectores junto con sus respectivas etiquetas de clase.

Una vez entrenado, el modelo fue evaluado utilizando el conjunto de prueba separado previamente. Esta evaluación preliminar permitió observar el comportamiento general del clasificador que será profundizado en la siguiente sección.

## Resultados

### Análisis Exploratorio y Preparación del Conjunto de Datos.

El conjunto de datos inicial, obtenido del repositorio de Hugging Face, contenía mensajes etiquetados como “ham” (no spam) y “spam”. Tras aplicar un proceso de limpieza, traducción al español y eliminación de duplicados, se conservaron un total de 5.138 mensajes, distribuidos en 4.498 mensajes ham y 640 mensajes spam. Esta distribución pone de manifiesto un importante desbalance de clases, con una proporción de aproximadamente 7 mensajes ham por cada spam, lo cual representa un reto para los algoritmos de clasificación, al aumentar el riesgo de sesgo hacia la clase mayoritaria.

Como parte del análisis exploratorio textual, se examinó la longitud de los mensajes (en caracteres) en cada clase. La Tabla I resume la distribución de mensajes por clase y las estadísticas descriptivas asociadas a la longitud de los mensajes.

Tabla I. Estadísticas descriptivas de la longitud de los mensajes por clase.

| Clase | Cantidad | Longitud media | Desviación estándar | Mínimo | Percentil 25 | Mediana (P50) | Percentil 75 | Máximo |
|---|---|---|---|---|---|---|---|---|
| Ham | 4498 | 74.63 | 57.77 | 2 | 36 | 56.5 | 96 | 751 |
| Spam | 640 | 149.00 | 38.23 | 10 | 130 | 160 | 172 | 298 |

Se observó que los mensajes spam son, en promedio, significativamente más largos que los mensajes ham. En particular, los mensajes spam tienen una longitud media de 149 caracteres con una desviación estándar de 38.23, mientras que los mensajes ham presentan una media de 74.63 caracteres con una desviación estándar mayor (57.77). Además, el mensaje spam más largo alcanzó los 298 caracteres, mientras que el mensaje ham más extenso llegó a 751 caracteres, aunque este último es un caso atípico. La mayoría de los mensajes spam se ubican entre 130 y 172 caracteres, lo que sugiere una estructura más uniforme y extensa, posiblemente debido a la inclusión de ofertas, enlaces o instrucciones detalladas.

Posteriormente, el conjunto fue dividido en dos particiones: un 80 % para entrenamiento y un 20 % para prueba, manteniendo la proporción original de clases mediante muestreo estratificado. La representación textual se realizó mediante la técnica de CountVectorizer, que permitió convertir los mensajes en una matriz dispersa de ocurrencias de palabras (bolsa de palabras), base para el entrenamiento del modelo de clasificación.

**Entrenamiento del modelo.**

Para entrenar el modelo se empleó el algoritmo Random Forest Classifier con los hiperparámetros ajustados de acuerdo con pruebas preliminares que buscaban mejorar el rendimiento del clasificador ante datos desbalanceados. No se realizó un ajuste fino mediante búsqueda en rejilla (grid search) en esta etapa, pero se establecieron valores razonables para mejorar la capacidad de generalización del modelo. En la Tabla II se presentan los principales hiperparámetros configurados en el modelo de Random Forest.

Tabla II. Parámetros configurados para el modelo Random Forest.

| Hiperparámetro | Valor asignado |
|---|---|
| Número de árboles | 300 |
| Profundidad máxima | 20 |
| Muestras mínimas por hoja | 5 |
| Semilla aleatoria | 42 |

El parámetro Número de árboles define la cantidad total de árboles que conforman el bosque aleatorio. Se eligieron 300 árboles para lograr un buen balance entre capacidad de generalización y tiempo de entrenamiento. La Profundidad máxima limita la profundidad que puede alcanzar cada árbol, con el objetivo de evitar que crezcan demasiado y sobreajusten el modelo a los datos de entrenamiento; en este caso se estableció en 20. El parámetro Muestras mínimas por hoja indica el número mínimo de datos necesarios para que un nodo sea considerado una hoja terminal, lo cual ayuda a suavizar el modelo y evitar que se ajuste demasiado a casos específicos, estableciendo un valor de 5. Finalmente, la Semilla aleatoria asegura que los resultados sean reproducibles, usando el valor 42 como una convención común para mantener consistencia en los experimentos.

**Evaluación del modelo.**

Para evaluar el rendimiento del modelo de clasificación de mensajes de texto en español, se emplearon diversas métricas que permiten obtener una visión más completa, especialmente en

contextos de clases desbalanceadas. Entre las métricas consideradas se encuentran accuracy (exactitud), recall (sensibilidad o exhaustividad), F1-score y el área bajo la curva ROC (AUC-ROC).

Tras su entrenamiento y evaluación sobre un conjunto de prueba compuesto por 1.115 mensajes, se obtuvo una exactitud global (accuracy) del 96%. A nivel de clases, se observó un rendimiento diferenciado: para la clase mayoritaria (ham), el modelo alcanzó una precisión del 96%, un recall del 100% y un F1-score de 0.98. En contraste, para la clase minoritaria (spam), el modelo logró una precisión del 100%, un recall del 70% y un F1-score de 0.82, lo cual representa una mejora importante respecto a iteraciones anteriores donde el F1-score de spam fue de 0.66. Estos resultados sugieren que el modelo no solo mantiene una excelente capacidad para identificar mensajes no spam (ham), sino que también ha mejorado considerablemente su habilidad para detectar mensajes de spam, una tarea crítica dada su menor representación en los datos. La alta precisión para spam (1.00) indica que los mensajes clasificados como spam realmente lo son, mientras que el recall del 70% revela que aún hay un 30% de mensajes spam no detectados.

Al analizar los resultados obtenidos en la evaluación del modelo, uno de los elementos clave fue la matriz de confusión, que se muestra en la figura 3, con esta herramienta se permitió visualizar de forma clara cómo se desempeñó el clasificador al identificar correctamente los distintos tipos de mensajes, teniendo en cuenta los datos usados el modelo logró clasificar correctamente 966 mensajes considerados normales (ham), es decir, aquellos que no eran spam, además de identificar de forma acertada 104 mensajes que sí correspondían a la categoría de spam.

Sin embargo, también se observó que 45 mensajes que en realidad eran spam fueron clasificados por el modelo como si fueran mensajes no spam, y era un margen de error esperado y significativo porque afecta directamente el rendimiento del modelo en lo que respecta a la clase minoritaria, es decir, los mensajes de spam. Específicamente este tipo de error tiene impacto sobre la métrica conocida como recall calculada en el entrenamiento de datos para medir la capacidad del modelo de poder encontrar todos los casos positivos dentro de una categoría especifica.

Por lo cual, considerando que existe una proporción considerable de mensajes spam que pasaron desapercibidos, el valor del recall para esa clase se vio reducido colocando en evidencia una de las principales dificultades al trabajar con conjuntos de datos desbalanceados, donde hay muchas más muestras de una clase que de otra, en específico muchos más mensajes normales que mensajes spam. Por otro lado, estos resultados también invitan a reflexionar sobre posibles mejoras en el sistema, como aplicar técnicas adicionales de balanceo o ajustar ciertos parámetros del modelo para afinar aún más su precisión en la detección de mensajes no deseados y asi se podrían minimizar los errores en clasificación.

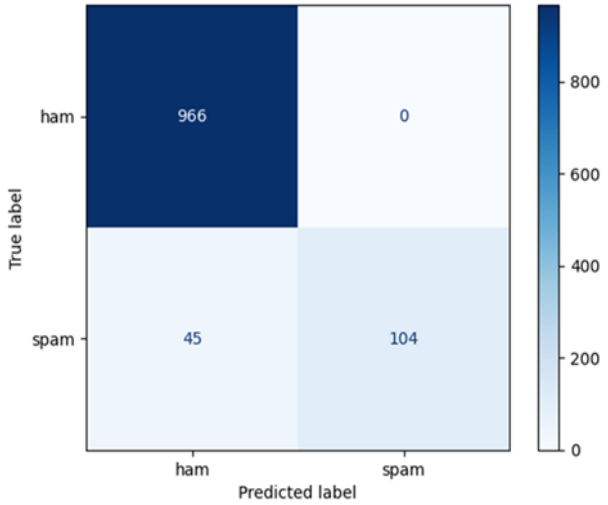


Figura 3. Matriz de confusión del modelo Random Forest sobre el conjunto de prueba.

Dentro del análisis de desempeño del modelo se decidió usar dos métricas como lo son el F1-score y la curva ROC, en primer lugar, el F1-score es especialmente útil porque combina dos aspectos fundamentales en la tarea de clasificación de este algoritmo tanto la precisión, que nos permitió conocer cuántos de los mensajes identificados como spam realmente si lo son; y la sensibilidad del modelo, que indica cuántos de los mensajes spam fueron detectados de forma correcta. En el caso de esta investigación, el modelo obtuvo un F1-score de 0.82 para la clase spam, lo cual puede considerarse un resultado bastante aceptable en aplicaciones prácticas considerando que en aplicaciones de mensajería instantánea no se encontraron resultados similares en los antecedentes. Sin embargo, todavía hay espacio para mejorar este rendimiento mediante estrategias de modificación del peso de las clases dentro del modelo (reponderación) y ajustar los parámetros internos del algoritmo.

Por otro lado, la curva ROC que se muestra en la figura 4 ofrece una mirada más amplia sobre cómo el modelo se comporta bajo diferentes niveles de exigencia, en esta imagen la curva permite visualizar el balance entre verdaderos positivos y falsos positivos a medida que se modifica el umbral de decisión, es decir, el punto a partir del cual el modelo decide si un mensaje ingresado es spam o no. Lo más destacable en este análisis es que se logró un área bajo la curva (AUC) de 0.99, lo cual refleja que el modelo tiene una excelente capacidad para diferenciar entre ambas clases, incluso cuando las condiciones se van cambiando, este valor indica que, en general, el sistema no solo funciona bien, sino que también es confiable al enfrentar nuevas situaciones o mensajes que no estaban en los datos de entrenamiento.

Por lo tanto, tomando en cuenta tanto el F1-score como el AUC, se lleva a que el modelo ofrece resultados sólidos y promete un buen rendimiento en escenarios reales.

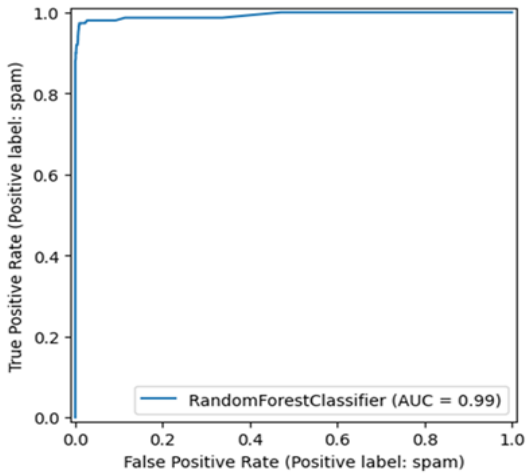


Figura 4. Curva ROC del modelo Random Forest para la detección de spam en mensajería móvil.

Para cerrar con el proceso de evaluación del modelo, se aplicó una validación cruzada de cinco pliegues sobre el conjunto de entrenamiento, por lo cual se dividió los datos en cinco partes y se alternaron en cuál de ellas se utiliza para probar mientras las demás se usan para entrenar, lo cual permitió comprobar si el modelo mantiene un desempeño constante sin importar qué datos se le estén asignando. Gracias a este procedimiento, se logró obtener una media de precisión del 94.4 %, lo que sugiere que el modelo es bastante estable y no depende de una única partición para ofrecer buenos resultados en la clasificación de mensajes.

Además de la validación cruzada, se hizo un análisis más detallado de las métricas de desempeño mencionadas anteriormente, en la Tabla III se resumen los valores de precisión, recall y F1-score para cada una de las clases evaluadas, de igual forma se incluyó el valor de soporte, que representa la cantidad real de muestras que pertenecen a cada clase en el conjunto de prueba.

Tabla III. Métricas de desempeño del modelo Random Forest por clase.

| Clase | Accuracy | Recall | F1-Score | Soporte |
|---|---|---|---|---|
| Ham | 0.96 | 1.00 | 0.98 | 966 |
| Spam | 1.00 | 0.70 | 0.82 | 149 |
| Accuracy global | – | – | 0.96 | 1115 |

Los resultados muestran que la clase “ham” o no spam obtuvo una precisión del 96 %, una sensibilidad o recall de 1.00 y un F1-score de 0.98, lo que indica que el modelo identifica correctamente la mayoría de los mensajes normales. Ahora bien, en el caso de la clase spam, aunque la precisión fue muy alta (1.00), la sensibilidad bajó a 0.70, lo que se reflejó en un F1-score de 0.82, por lo cual quiere decir que, si bien los mensajes detectados como spam efectivamente lo eran, hubo algunos que pasaron desapercibidos. En total, se evaluaron 1.115 mensajes, de los cuales 966 eran ham y 149 eran spam, y la precisión general del modelo se ubicó en un sólido 96 %.

Tomando en cuenta estos resultados se afirma que el modelo ofrece un rendimiento bastante bueno en la clasificación automática de mensajes estructurados de texto, presentando un margen de mejora sobre todo en la detección de mensajes spam, los valores obtenidos confirman que el enfoque metodológico es sólido y que podría ser perfectamente aplicable en escenarios reales donde se requiere filtrar este tipo de contenido, específicamente en aplicaciones de mensajería instantánea.

## Conclusiones

En este trabajo se desarrolló un modelo de clasificación para detección de mensajes spam en español a partir de un conjunto de datos desbalanceado obtenido del repositorio Hugging Face. La limpieza, traducción y preprocesamiento permitieron obtener un corpus balanceado para el entrenamiento y evaluación del modelo. Se evidenció que los mensajes spam tienden a ser más largos y con menor variabilidad en su longitud en comparación con los mensajes ham.

El modelo entrenado con Random Forest, ajustado con hiperparámetros específicos para controlar el sobreajuste, mostró un desempeño global satisfactorio, alcanzando una exactitud del 96% en el conjunto de prueba. En particular, el clasificador mantuvo una alta precisión en la detección de mensajes no spam, con un recall perfecto del 100% para esta clase, mientras que mejoró considerablemente la capacidad para identificar mensajes spam, logrando un F1-score de 0.82, superior a versiones previas del modelo. El valor del área bajo la curva ROC (0.99) indica una alta capacidad discriminativa del modelo en diferentes umbrales de clasificación.

Finalmente, se concluye que el modelo es estable y robusto, como lo confirma la validación cruzada realizada. No obstante, futuras mejoras podrían orientarse a la aplicación de técnicas avanzadas de balanceo de clases, ajuste fino de hiperparámetros y métodos de aumento de datos, con el fin de optimizar la detección de spam y reducir los errores de clasificación.

## Referencias

[1] B. Sonare, G. J. Dharmale, A. Renapure, H. Khandelwal, and S. Narharshettiwar, “E-mail Spam Detection Using Machine Learning,” *2023 4th International Conference for Emerging Technology, INCET 2023*, 2023, doi: 10.1109/INCET57972.2023.10170187.

[2] A. Alzahrani and D. B. Rawat, “Comparative Study of Machine Learning Algorithms for SMS Spam Detection,” *Conference Proceedings - IEEE SOUTHEASTCON*, vol. 2019-April, Apr. 2019, doi: 10.1109/SOUTHEASTCON42311.2019.9020530.

[3] K. Aparna and S. Halder, “Detection of Multilingual Spam SMS Using NaïveBayes Classifier,” *5th IEEE International Conference on Cybernetics, Cognition and Machine Learning Applications, ICCCMLA 2023*, pp. 89–94, 2023, doi: 10.1109/ICCCMLA58983.2023.10346960.

[4] K. Debnath and N. Kar, “Email Spam Detection using Deep Learning Approach,” *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing, COM-IT-CON 2022*, pp. 37–41, 2022, doi: 10.1109/COM-IT-CON54601.2022.9850588.

[5] J. Mythili, B. Deebeshkumar, T. Eshwaramoorthy, and J. N. Ajay, "Enhancing Email Spam Detection with Temporal Naive Bayes Classifier," *2024 International Conference on Communication, Computing and Internet of Things, IC3IoT 2024 - Proceedings*, 2024, doi: 10.1109/IC3IOT60841.2024.10550229.

[6] N. Ramya, M. K. Devi, K. Nithya, V. Hema, and R. Thomas Abragam Walker, "Detection of Malicious Messages from Mobile Computing Devices Using NLP and Slack Integration," *Proceedings of the 2023 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems, ICSES 2023*, 2023, doi: 10.1109/ICSES60034.2023.10465341.

[7] P. Santhiya, S. Kavitha, T. Aravindh, S. Archana, and A. V. Praveen, "Fake News Detection Using Machine Learning," *2023 International Conference on Computer Communication and Informatics, ICCCI 2023*, 2023, doi: 10.1109/ICCCI56745.2023.10128339.

[8] X. Fei, J. Li, Y. Gao, and Y. Zhou, "SMS Text Classification Model Based on Machine Learning," *2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP 2021*, pp. 289–292, 2021, doi: 10.1109/ICCWAMTIP53232.2021.9674063.

[9] H. Chen, L. Wu, J. Chen, W. Lu, and J. Ding, "A comparative study of automated legal text classification using random forests and deep learning," *Inf Process Manag*, vol. 59, no. 2, p. 102798, Mar. 2022, doi: 10.1016/J.IPM.2021.102798.

[10] N. Jalal, A. Mehmood, G. S. Choi, and I. Ashraf, "A novel improved random forest for text classification using feature ranking and optimal number of trees," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, pp. 2733–2742, Jun. 2022, doi: 10.1016/J.JKSUCI.2022.03.012.

[11] S. U. Hassan, J. Ahamed, and K. Ahmad, "Analytics of machine learning-based algorithms for text classification," *Sustainable Operations and Computers*, vol. 3, pp. 238–248, Jan. 2022, doi: 10.1016/J.SUSOC.2022.03.001.

[12] B. Zhang, "News Text Classification Algorithm Based on Machine Learning Technology," *Proceedings - 2022 International Conference on Education, Network and Information Technology, ICENIT 2022*, pp. 182–186, 2022, doi: 10.1109/ICENIT57306.2022.00047.

[13] A. Rohalia, Z. Zainuddin, and Z. Tahir, "Classification of Community Responses to Service Offices using a Combined CNN-LSTM Algorithm and Random Forest," *2023 International Conference on Artificial Intelligence Robotics, Signal and Image Processing (AIRoSIP)*, pp. 68–73, Aug. 2023, doi: 10.1109/AIROSIP58759.2023.10873903.

[14] G. Thangarasu and K. R. Alla, "Detection of Cyberbullying Tweets in Twitter Media Using Random Forest Classification," *13th IEEE Symposium on Computer Applications and Industrial Electronics, ISCAIE 2023*, pp. 113–117, 2023, doi: 10.1109/ISCAIE57739.2023.10165118.

[15] V. Mittal, M. Guru, H. Vishwakarma, D. Ganesh, S. Chandrappa, and M. Ram, "Sentimental Analysis of Movie Review Based on Naive Bayes and Random Forest Technique,"

*2023 IEEE 4th Annual Flagship India Council International Subsections Conference: Computational Intelligence and Learning Systems, INDISCON 2023*, 2023, doi: 10.1109/INDISCON58499.2023.10269857.

[16] Y. Sun, Y. Li, Q. Zeng, and Y. Bian, "Application research of text classification based on random forest algorithm," *Proceedings - 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering, AEMCSE 2020*, pp. 370–374, Apr. 2020, doi: 10.1109/AEMCSE50948.2020.00086.

[17] S. Gadde, A. Lakshmanarao, and S. Satyanarayana, "SMS Spam Detection using Machine Learning and Deep Learning Techniques," *2021 7th International Conference on Advanced Computing and Communication Systems, ICACCS 2021*, pp. 358–362, Mar. 2021, doi: 10.1109/ICACCS51430.2021.9441783.

[18] E. Ramanujam, K. Shankar, and A. Sharma, "A Review on Artificial Intelligence Techniques for Multilingual SMS Spam Detection," *Lecture Notes in Electrical Engineering*, vol. 1087, pp. 525–536, 2024, doi: 10.1007/978-981-99-6690-5_40.

[19] M. Johari., "Key insights into recommended SMS spam detection datasets," *Scientific Reports*, vol. 15, no. 1, pp. 1–24, Dec. 2025, doi: 10.1038/S41598-025-92223-1.
[20] C. J. Nusch, "Breve Introducción a la Minería de Textos," Oct. 2024. Accessed: Jul. 08, 2025.

[21] G. Rosenbrock, S. Trossero, and A. Pascal, "Técnicas de análisis de sentimientos aplicadas a la valoración de opiniones en el lenguaje español," *Memorias del Congreso Argentino en Ciencias de la Computación - CACIC 2021*, vol. 1, no. 1, pp. 31–40, 2021, Accessed: Jul. 08, 2025.

[22] L. Vasquez and J. Vivas, "Detección de situaciones de emergencias usando el modelo Naive- Bayes de machine learning.," *Mundo FESC*, vol. 13, no. 25, pp. 20–40, Jan. 2023, doi: 10.61799/2216-0388.1286.

[23] S. Sakthi Vel, "Pre-Processing techniques of Text Mining using Computational Linguistics and Python Libraries," *Proceedings - International Conference on Artificial Intelligence and Smart Systems, ICAIS 2021*, pp. 879–884, Mar. 2021, doi: 10.1109/ICAIS50930.2021.9395924.

[24] H. Qiu, Z. Wu, and X. Zhang, "Exploring Multiple Genres Text Classification: Classifying 61 genres of Mobile App Description based on Naïve Bayes and Count Vectorizer," *Proceedings - 2022 3rd International Conference on Electronic Communication and Artificial Intelligence, IWECAI 2022*, pp. 156–162, 2022, doi: 10.1109/IWECAI55315.2022.00039.

[25] N. O. Lwin, R. Jain, R. Dal, H. Yan, K. K. Thaw, and S. Y. Naung, "Text Classification for Clickbait Detection: A Model-Driven Approach Using CountVectorizer and ML Classifiers," *Journal of Applied Science and Technology Trends*, vol. 6, no. 1, pp. 43–49, Jun. 2025, doi: 10.38094/jastt61237.