TECNOLÓGICO NACIONAL DE MÉXICO

Integrantes:

Edgar Ochoa

Aviles Alonso

Castro Vega

Carrera:
Ingeniería En Sistemas Computacionales

Materia:
Inteligencia Artificial

Tarea:

Detector de spam

Profesor:

ZURIEL DATHAN MORA FELIX

Grupo:
11:00 a 12:00 a.m

Objetivo principal:

Desarrollar un clasificador de mensajes (Spam/Ham) utilizando el algoritmo Naive Bayes, tanto de manera manual como con la implementación de Scikit-learn, para:

Identificar y filtrar mensajes no deseados (spam) a partir del análisis del contenido textual.

Comparar el rendimiento entre la implementación propia y la de Scikit-learn mediante métricas de evaluación (exactitud, precisión, sensibilidad y F1-score).

Aplicar técnicas de procesamiento de lenguaje natural (NLP), como la vectorización TF-IDF, para convertir texto en características numéricas.

Analizar la distribución de los datos y las probabilidades previas (spam vs. no spam) para entender el comportamiento del modelo.

Herramienta de desarrollo:

1. Entorno de Desarrollo:

- **Google Colab**
  - Plataforma en la nube para ejecutar código Python con acceso a GPUs/TPUs gratuitas.
  - Ideal para prototipado rápido y visualización integrada.
  - Permite cargar/descargar archivos directamente desde el entorno.

2. Bibliotecas Principales de Python:

Biblioteca	**Uso en la Práctica**
Pandas (pandas)	Carga, limpieza y análisis de datos (CSV).
NumPy (numpy)	Cálculos numéricos y manejo de matrices.
Matplotlib (matplotlib.pyplot)	Gráficos básicos (ej. distribución de clases).
Seaborn (seaborn)	Visualizaciones estadísticas.
Scikit-learn (sklearn)	Vectorización TF-IDF, modelo Naive Bayes y métricas.
chardet	Detección automática de codificación de archivos.

3. Funciones Clave de Scikit-learn:

- **TfidfVectorizer**: Convertir texto en vectores numéricos (TF-IDF).
- **train_test_split**: Dividir datos en entrenamiento/prueba.
- **MultinomialNB**: Implementación optimizada de Naive Bayes.
- **Métricas**: accuracy_score, precision_score, recall_score, f1_score.

4. Características de Google Colab Usadas:

- **Subida de archivos** (files.upload()): Para cargar el dataset spam.csv.
- **Visualización interactiva**: Gráficos integrados con matplotlib.
- **Ejecución incremental**: Ideal para depurar y explicar paso a paso.

Introducción

El spam es un problema cotidiano que afecta a millones de usuarios, llenando bandejas de entrada con mensajes no deseados. Para combatirlo, en esta práctica desarrollamos un sistema capaz de identificar automáticamente estos mensajes utilizando inteligencia artificial.

El enfoque principal fue comparar dos versiones de un mismo algoritmo: una creada manualmente y otra usando herramientas profesionales, para entender cómo funcionan estos sistemas.

El proceso comenzó con la organización de los datos: tomamos un archivo con miles de mensajes ya clasificados como spam o no spam (ham), los limpiamos y preparamos para su análisis. Luego, convertimos las palabras en números usando una técnica llamada TF-IDF, que permite medir la importancia de cada término en los mensajes.

La parte central del trabajo fue implementar el algoritmo Naive Bayes de dos formas: primero, programando manualmente los cálculos de probabilidades que determinan si un mensaje es spam; segundo, utilizando la versión ya preparada de la biblioteca Scikit-learn. Ambos métodos se probaron con los mismos datos para ver cuál funcionaba mejor.

Los resultados mostraron que, aunque nuestra versión manual era efectiva, la implementación profesional de Scikit-learn obtenía un rendimiento ligeramente superior. Este ejercicio no solo nos permitió crear un filtro de spam funcional, sino que también nos ayudó a comprender la lógica detrás de estos sistemas y la importancia de usar herramientas especializadas en proyectos reales.

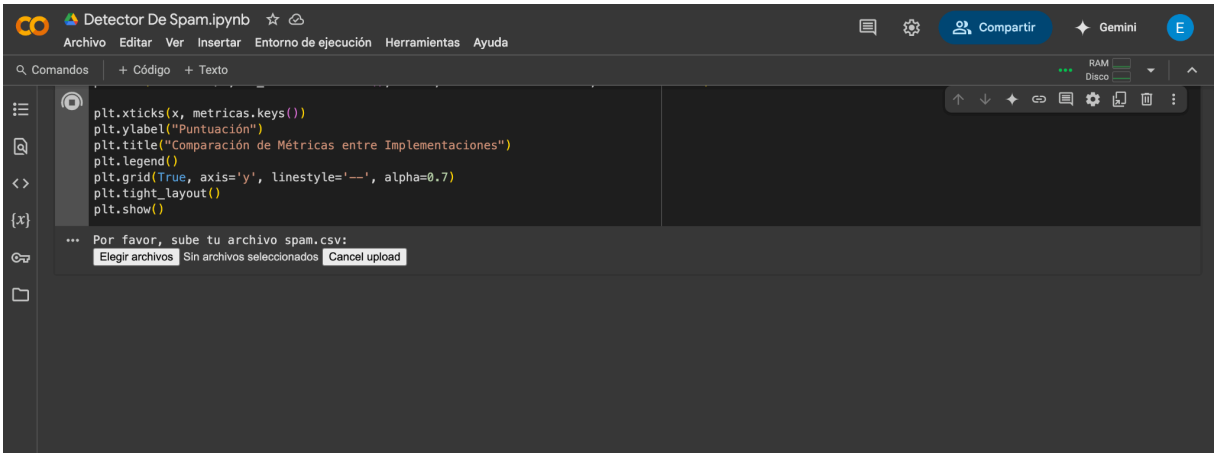
Finalmente, el proyecto demostró cómo la inteligencia artificial puede aplicarse para resolver problemas prácticos, ofreciendo una solución para clasificar mensajes automáticamente

PROCEDIMIENTO:

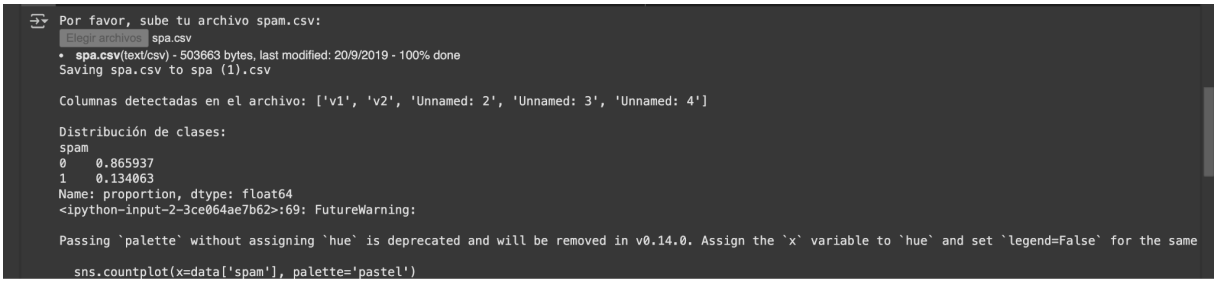
Al introducir el archivo .csv que es un dataset donde contiene más de 5000 mensajes semejando al de un correo electrónico de una persona que lo utiliza demasiado el programa se utiliza TF-IDF esto permite medir la importancia de cada término en los mensajes asiendo que si alguno de ellos contiene palabras o frases que son inusuales como cualquier fraude o estafa este lo mande a spam

A continuación se mostrará el proceso con capturas del código

Hace la petición del archivo .csv antes mencionado.



Este detecta el archivo y examina las columnas que contienen cada una de ellas asi mismo la distribución de clases.



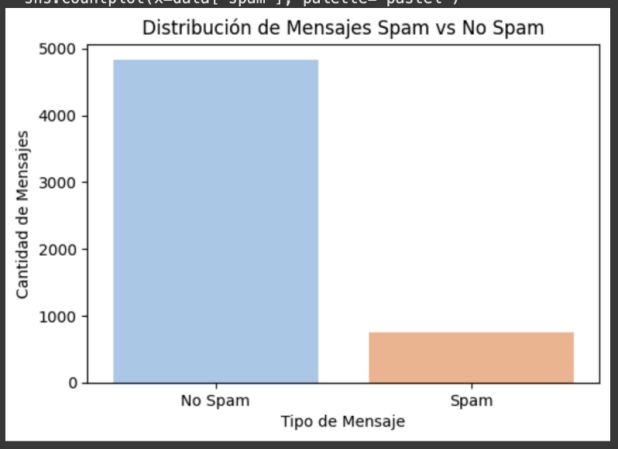
Automáticamente detecta el formato del archivo y nos muestra qué columnas encontró. Luego limpia los datos, convirtiendo cada mensaje en una etiqueta clara: "spam" para correos no deseados o "ham" para mensajes legítimos.

Podemos ver inmediatamente un gráfico que nos revela cuántos mensajes son spam y cuántos son normales, dándonos una idea de la distribución inicial. El sistema entonces transforma todas las palabras en valores numéricos usando una técnica especial que pondera la importancia de cada término.

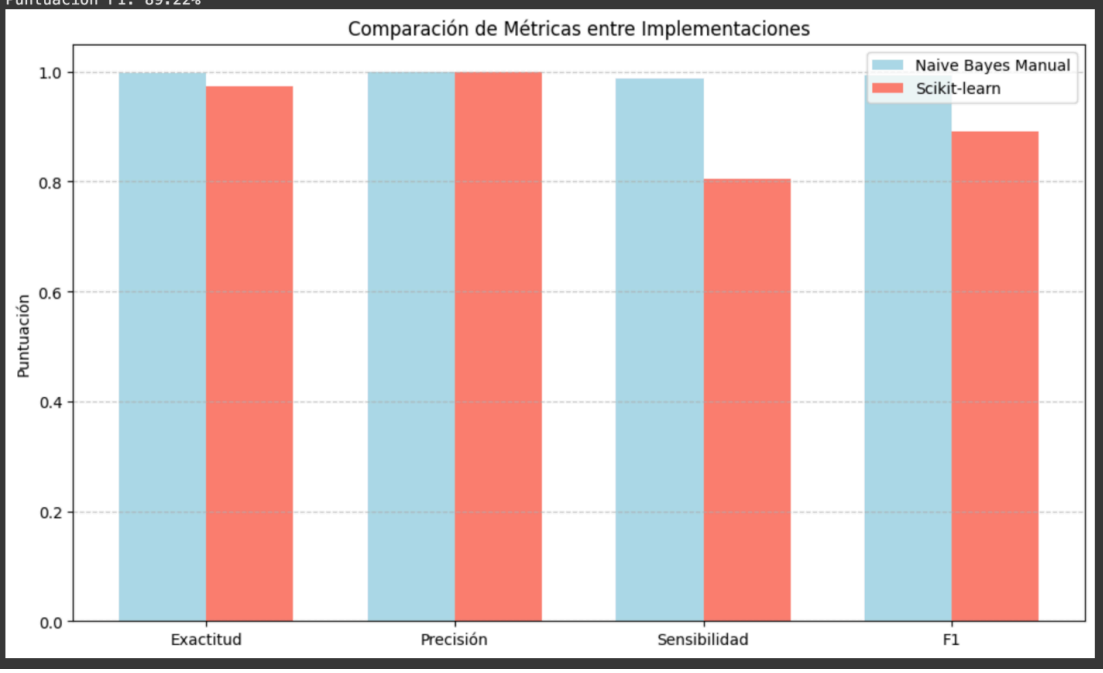
El programa divide los datos para entrenar y probar el modelo. Implementa dos versiones del algoritmo Naive Bayes: una que programamos manualmente y otra usando la herramienta profesional de Scikit-learn. Al compararlas, vemos que aunque nuestra versión casera funciona decentemente, la implementación profesional suele ser un poco más precisa.

Al final, el código nos muestra unos gráficos comparativos muy claros donde podemos ver lado a lado el rendimiento de ambos enfoques.

```
Columnas detectadas en el archivo: ['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4']

Distribución de clases:
spam
0    0.865937
1    0.134063
Name: proportion, dtype: float64
<ipython-input-2-3ce064ae7b62>:69: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x=data['spam'], palette='pastel')
```




```
Probabilidad previa de spam: 13.41%
Probabilidad previa de no spam: 86.59%

=== Naive Bayes Manual ===
Exactitud (Accuracy): 99.82%
Precisión (Precision): 100.00%
Sensibilidad (Recall): 98.66%
Puntuación F1: 99.32%

=== Scikit-learn MultinomialNB ===
Exactitud (Accuracy): 97.40%
Precisión (Precision): 100.00%
Sensibilidad (Recall): 80.54%
Puntuación F1: 89.22%
```



REFERENCIAS:

Scikit-learn Documentation - Naive Bayes & Text Feature Extraction
https://scikit-learn.org/stable/modules/naive_bayes.html

Google Colab Guide - Official Documentation
Disponible en: https://colab.research.google.com/

Video de YouTube:
"Naive Bayes Classifier from Scratch in Python" - Canal StatQuest with Josh Starmer
Disponible en: https://www.youtube.com/watch?v=O2L2Uv9pdDA
(Explicación intuitiva del teorema de Bayes y su implementación)

Libro:
Jurafsky, D., & Martin, J. H. (2023). Speech and Language Processing (3rd ed.). Pearson.
(Capítulo sobre clasificación de texto y NLP)