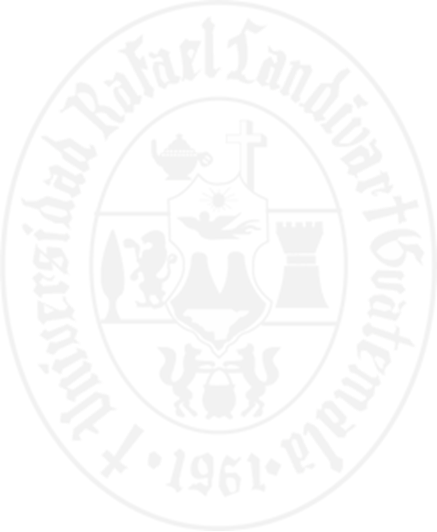
**UNIVERSIDAD RAFAEL LANDÍVAR**

FACULTAD DE INGENIERÍA

INTELIGENCIA ARTIFICIAL



**PROYECTO: Clasificación usando Naïve Bayes**

RAFAEL ANDRÉS ALVAREZ MAZARIEGOS 1018419

JOSÉ DANIEL DE LEÓN CHANG 1170419

CARLOS ENRIQUE LAPARRA ROBLEDO 1031120

GUATEMALA DE LA ASUNCIÓN, ABRIL DE 2025

CAMPUS CENTRAL “SAN FRANCISCO DE BORJA, S. J” DE LA CIUDAD DE GUATEMALA

# Introducción

Se desarrolló el proyecto titulado **"PROYECT RAFAELITOS"**, enfocado en aplicar de manera práctica los conceptos de clasificación de texto mediante el algoritmo de Naïve Bayes.

El objetivo principal de este proyecto es demostrar cómo la inteligencia artificial puede utilizarse para categorizar noticias en diferentes temáticas, como política, deportes, negocios, tecnología y entretenimiento, de forma automática y precisa. Para ello, trabajamos con el dataset de noticias proporcionado por la BBC, el cual nos permitió entrenar y evaluar nuestro modelo.

Además de la implementación algorítmica, integramos el modelo en una aplicación web sencilla e intuitiva, permitiendo que cualquier usuario pueda ingresar una noticia y obtener su clasificación correspondiente en tiempo real. Esta integración nos permitió abordar no solo el desarrollo del modelo de IA, sino también su despliegue práctico mediante una interfaz web accesible.

El proyecto también nos proporcionó la oportunidad de reforzar conceptos esenciales de la inteligencia artificial, tales como el preprocesamiento de datos textuales, la evaluación de modelos mediante métricas estándar (precisión, recall, F1-score) y la importancia de una correcta comunicación entre un motor de inferencia y una aplicación frontend.

# Definición del Problema y Objetivos (generales y específicos)

Objetivo general:

Diseñar e implementar un sistema de clasificación automática de noticias utilizando el algoritmo de Naïve Bayes, integrando un motor de inferencia en Python con una interfaz web funcional para la interacción con el usuario.

Objetivos específicos:

* Aplicar técnicas de preprocesamiento de texto sobre el dataset de noticias.
* Entrenar un clasificador Naïve Bayes desde cero, sin utilizar bibliotecas externas que resuelvan el modelo.
* Evaluar el rendimiento del modelo mediante métricas estándar como precisión, recall, F1-score y matriz de confusión.
* Implementar un motor de inferencia que reciba textos nuevos y devuelva su clasificación en tiempo real.
* Desarrollar una interfaz web amigable que permita a los usuarios ingresar noticias y visualizar la categoría correspondiente.
* Integrar correctamente el modelo con el frontend para asegurar una experiencia fluida.
* Documentar adecuadamente el proceso de desarrollo y justificar las decisiones tomadas.

# Descripción del dataset utilizado.

Para nuestro proyecto usamos un dataset llamado BBC News Summary. (Proporcionado en el enunciado del proyecto)

Este dataset tiene noticias reales de la BBC y están divididas en cinco categorías:

* Business (Negocios)
* Entertainment (Entretenimiento)
* Politics (Política)
* Sport (Deportes)
* Tech (Tecnología)

Cada noticia está guardada en un archivo de texto (.txt) y las noticias de la misma categoría están dentro de la misma carpeta.

-¿Qué hicimos con el dataset?

Leímos cada archivo y vimos de qué categoría era según la carpeta donde estaba.

Limpiamos el texto: le quitamos signos raros, convertimos todo a minúsculas, y quitamos palabras que no aportan mucho (como "the", "is", "at", etc.).

Unimos todo en un archivo más ordenado (en formato CSV) para poder entrenar nuestro modelo.

Este dataset fue ideal porque ya tenía noticias de varios temas diferentes, lo cual nos ayudó a entrenar mejor el modelo para clasificar nuevas noticias que los usuarios escriban.

# Descripción del preprocesamiento aplicado

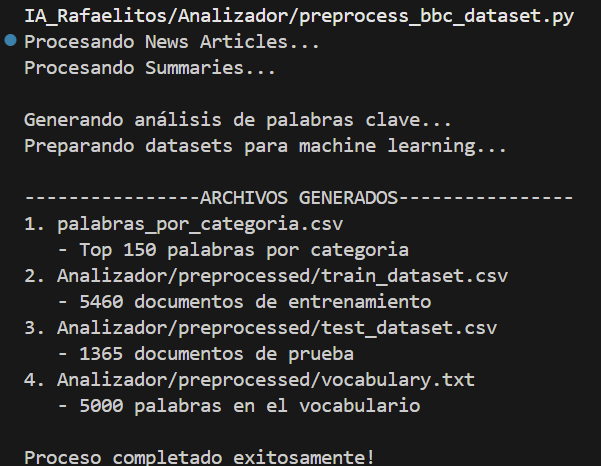
Para el preprocesamiento del dataset se implementó una clase llamada “preprocess\_bbc\_dataset.py”, en la cual se incluyó la lectura de archivos de las cinco categorías (business, entretainment, politics, sport, tech) desde las dos fuentes del dataset (New Artricles, Summaries)

Se le aplicaron los siguientes pasos a cada uno de los archivos .txt:

* Conversión a minúsculas
* Eliminación de caracteres no alfabéticos mediante expresiones regulares
* Tokenización del texto en palabras individuales
* Eliminación de stopwords y palabras de menos de tres letras

Luego agrupamos por categoría y se generaron documentos de 100 palabras para crear un conjunto de entrenamiento y prueba, manteniendo una proporción de 80/20. Se construyo un vocabulario con las 5000 palabras más frecuentes y después se vectorizó cada documento como un diccionario de frecuencia de palabras.

Cuando se ejecuta el código del “preprocess\_bbc\_dataset.py”:



*Imagen No.1 Salida de Consola de datos preprocesados(Fuente propia)*

Se guardaron tres archivos:

* train\_dataset.csv
* test\_dataset.csv
* vocabulary.txt

Agregando al archivo de preprocesamiento con las 150 palabras con mas frecuencia en cada una de las categorías llamado “palabras\_por\_categoria.csv”.

* Los datos de “palabras\_por\_categoria.csv” se muestran de la siguiente forma:

Tabla

El contenido generado por IA puede ser incorrecto.

*Imagen No.2 Salida de datos preprocesados (Fuente propia)*

* Los datos de “test\_dataset.csv” se muestran de la siguiente forma:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

El contenido generado por IA puede ser incorrecto.*Imagen No.3 Inicio de dataset para el test (Fuente propia)*

*Pantalla de computadora

El contenido generado por IA puede ser incorrecto.*

*Imagen No.4 Final de dataset para el test (Fuente propia)*

Para el csv del dataset de entrenamiento es el mismo solo que tiene más datos que el de prueba.

Explicación del algoritmo Naïve Bayes y justificación

El modelo de Naïve Bayes que se implementó en el proyecto con el objetivo de una clasificación probabilística basado en el Teorema de Bayes, se siguieron una serie de pasos para nuestra implementación:

1. Entrenamiento en la clase “naive\_bayes.py”
   * Se calcula la probabilidad a priori de cada clase.
   * Se calcula la probabilidad condicional de cada palabra dada una clase, con el suavizado de Laplace
2. Predicción
   * Para un nuevo texto vectorizado, se calcula el logaritmo de probabilidad de cada una de las clases, en donde se escoge la clase con la probabilidad más alta.
3. Evitar errores
   * Para minimizar los errores comunes que nos pueden causar problema en el futuro utilizamos
     + Math.log para evitar un underflow numérico
     + Utilizamos la probabilidad mínima (1e-10) si una palabra no aparece en el modelo de entrenamiento
     + Implementamos la formula del Suavizamiento de Laplace

Ahora que ya tenemos el Naive Bayes se utiliza el train\_model.py para poder entrenar el modelo utilizando naive bayes.

Dentro del proyecto cuando ya se ejecute el train\_model.py nos saldrá la siguiente salida:

Texto

El contenido generado por IA puede ser incorrecto.

*Imagen No.5 Salida del train model (Fuente propia)*

¿Por qué se utilizó de esa manera?

Principalmente por la clasificación del texto, ya que eran muchas palabras y archivos que se tenían que leer, así mismo funciona muy bien para vocabularios grandes y así poder interpretar fácilmente las probabilidades y contribuciones de cada palabra. Cuando termina de ejecutar guarda el modelo en bbc\_classifier.pkl para poder utilizarlo luego con las métricas

Dentro del archivo. pkl nos muestra lo siguiente

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

*Imagen No.6 Salida del pkl(Fuente propia)*

# Explicación de la evaluación del modelo

Para la evaluación del modelo se utilizaron las siguientes métricas de rendimiento:

* Precisión (Accuracy): Numero de predicciones correctas sobre las positivas predichas
* Recall: Verdaderos positivos correctamente clasificados
* F1-Score: Se utiliza la precisión y el recall para la media de ambos

# Texto El contenido generado por IA puede ser incorrecto.

*Imagen No.7 Salida del evaluate model (Fuente propia)*

Se coloco las métricas generales que son todas las noticias evaluadas y como es que lo toma el modelo, luego se hizo para cada una de las categorías su precisión, recall y el F1-Score

Se puede observar en la Imagen No.7 que hay un 100% de recall para sport y esto se debe a que es una de las categorías que tenia más archivos y esto nos ayudo a entrenar el modelo en base a la cantidad de archivos proporcionados en el dataset.

Pantalla de computadora

El contenido generado por IA puede ser incorrecto.

*Imagen No.8 Código del evaluate model (Fuente propia)*

# Diagramas:

**Arquitectura de la solución (motor de inferencia + página web)**

Diagrama

El contenido generado por IA puede ser incorrecto.

*Imagen No.9 Arquitectura (Fuente propia)*

Usuario:

* Es quien ingresa la noticia que quiere clasificar desde el navegador
* Interactúa con la interfaz hecha con React

Frontend (React)

* Encargado de presentar la caja de texto al usuario para ingresar la noticia
* Cuando se da click en “Verify” envia la solicit HTTP tipo POST al backend con la noticia

Backend (Python, Flask)

* Recibe el texto
* Lo preprocesa
* Llama al motor de Naïve Bayes para hacer la predicción
* El backend y el frontend se comunican por medio de flask-cors

Motor Naïve Bayes

* Utiliza las probabilidades previamente entrenadas para calcular la categoría más probable de la noticia
* Si aplica también calcula la segunda categoría mas probable

**Diagramas de Casos de Uso**

Diagrama

El contenido generado por IA puede ser incorrecto.

*Imagen No.10 Casos de Uso (Fuente propia)*

Actores

* Usuario: Persona que accede desde el navegador web
* Backend: Contiene la caja de texto para la solicitar al Naive Bayes

Casos de Uso

* Ingresar noticia: El usuario escribe la noticia que desea clasificar
* Solicitar Clasificación: Se envía al backend via solicitud POST
* Recibir resultados
  + La categoría más probable
  + El porcentaje de categoría
  + Si aplica la segunda categoría más probable

**Diagrama de Flujo General**

Diagrama

El contenido generado por IA puede ser incorrecto.

*Imagen No.11 Flujo completo (Fuente propia)*

**Diagrama de Componentes**

Diagrama

El contenido generado por IA puede ser incorrecto.

*Imagen No.12 Componentes (Fuente propia)*

**Diagrama de secuencias (Modelar la interacción entre el usuario, el frontend y el backend)**

**Diagrama

El contenido generado por IA puede ser incorrecto.**

*Imagen No.13 Secuencias (Fuente propia)*

Usuario: Interactúa con la interfaz e ingresa una noticia

Backend: Aplica el procesamiento al texto y llama al motor de naive bayes. Recibe la predicción y la convierte en una respuesta

Motor Naive Bayes: se encarga de la predicción y el procesamiento

# Evidencias de funcionamiento (capturas de pantalla de la web app y el modelo en acción).

**Frontend**

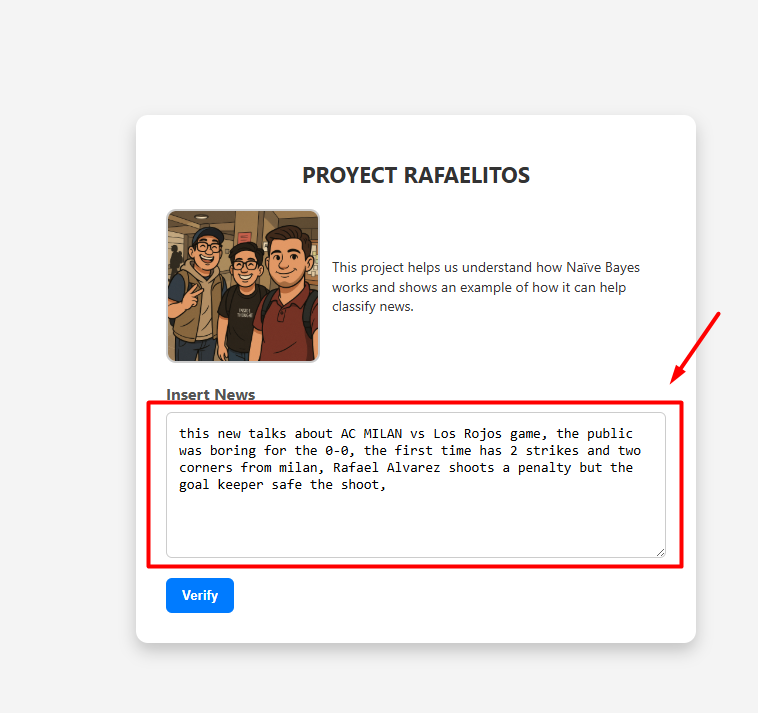
* Al inciar se vera de esta forma la pagina inicial

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

*Imagen No.14 (Fuente propia)*

En esta ventana deberemos escribir la noticia en el text box de enmedio.



Se presiona el botón Verify

*Imagen No.15 (Fuente propia)*

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

*Imagen No.16 (Fuente propia)*

* Nos regresara un model con la categoría o categorías con su probabilidad

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

*Imagen No.17 (Fuente propia)*

* Si en dado caso no se obtiene un 100% de accuracy, se muestran los dos porcentajes más altos.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

**Backend**

* Para correr el motor de naive bayes se puede desde la consola con el comando “python api.py”

\*Se necesita estar dentro de la ubicación del proyecto …../ProyectoIA\_Rafaelitos\Analizador

Texto

El contenido generado por IA puede ser incorrecto.

* Si necesita ver las métricas es con el siguiente comando desde consola con el comando “python Analizador/evaluate\_model.py”

\*Se necesita estar dentro de la ubicación del proyecto …../ProyectoIA\_Rafaelitos

Texto

El contenido generado por IA puede ser incorrecto.

# Conclusiones y aprendizajes

El desarrollo de este proyecto permitió comprender con mayor profundidad cómo funciona el algoritmo de Naïve Bayes y su utilidad en la clasificación de textos. Al implementarlo desde cero, se lograron fortalecer competencias clave como el tratamiento de datos textuales, el manejo de estructuras en Python y la construcción de un motor de inferencia funcional.

Uno de los aprendizajes más significativos fue reconocer la importancia del preprocesamiento de los datos. Una adecuada limpieza del texto influye directamente en la precisión del modelo, lo que evidenció cómo una buena preparación de los datos puede marcar la diferencia en los resultados obtenidos. Asimismo, las métricas de evaluación utilizadas nos brindaron una base sólida para interpretar el desempeño del modelo y tomar decisiones con mayor criterio técnico.

La creación de la interfaz web utilizando React ofreció la oportunidad de aplicar conocimientos de integración entre el frontend y el backend, enfrentando desafíos reales en la comunicación entre ambas capas. Además, el trabajo colaborativo y la necesidad de organizar entregables fomentaron una mejor planificación, así como la capacidad de comunicar con claridad las soluciones implementadas.

En conclusión, este proyecto demostró que, incluso con algoritmos relativamente simples como Naïve Bayes, es posible obtener resultados efectivos en problemas del mundo real, siempre que se realice un diseño cuidadoso y una implementación rigurosa.