UNIVERSIDAD DEL VALLE DE GUATEMALA
CC3085 - Inteligencia artificial
Sección 30
Ing. Javier Fong

Proyecto 2

Jose Santisteban, 21153

GUATEMALA, 6 de enero de 2024

MENSAJES SPAM / HAM

Análisis exploratorio

A continuación se mostrarán algunos resultados derivados del análisis exploratorio realizado con los datos de 'spam.csv', como primera instancia se hizo una exploración de existencia de datos dentro del set de datos, el cual tuvo como resultado lo siguiente:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
count	5572	5572	50	12	6
unique	2	5169	43	10	5
top	ham	Sorry, I'll call later	bt not his girlfrnd... G o o d n i g h t . . .@"	MK17 92H. 450Ppw 16"	GNT:-)"
freq	4825	30	3	2	2

Figura 1: análisis de datos superficiales

En donde se puede observar que hay un poco más de 5000 datos y que las columnas importantes son las v1 y v2 (tipo de mensaje y contenido de mensaje respectivamente). Además se realizó un análisis previo a la limpieza de datos para determinar cuáles eran las palabras más comunes en los dos conjuntos de datos, los resultados son los siguientes:

```
Palabras más comunes en mensajes ham:
[('to', 1530), ('you', 1458), ('I', 1436), ('the', 1019), ('a', 969), ('and', 738), ('i', 736), ('in', 734), ('u', 645), ('is', 638)]

Palabras más comunes en mensajes spam:
[('to', 604), ('a', 358), ('your', 187), ('call', 185), ('or', 185), ('the', 178), ('2', 169), ('for', 169), ('you', 164), ('is', 143)]
```

Figura 2: palabras más comunes en los conjuntos de datos

Se puede observar que las palabras más comunes en los dos tipos de mensajes son palabras de conexión, pronombres, etc, lo que no nos da información relevante sobre los dos conjuntos.

También se realizó un análisis de la distribución de cantidad de mensajes en el set de datos, para poder visualizar la cantidad de mensajes contra la longitud de los mismos.

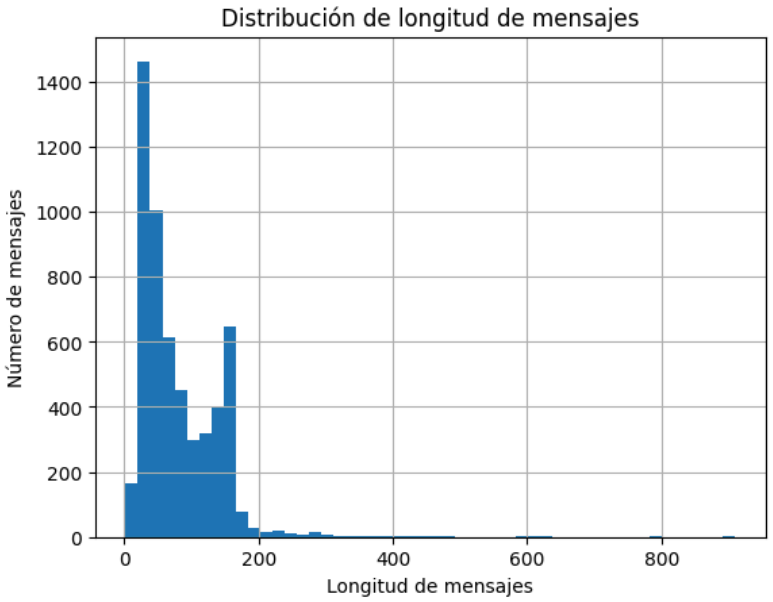

Figura 3: Distribución de longitud de mensajes

Como se puede observar en la figura 3 , la mayoría de los mensajes que se han enviado tienen una longitud menor a la de 200 caracteres. Este resultado se puede ver más detalladamente en las siguientes gráficas.

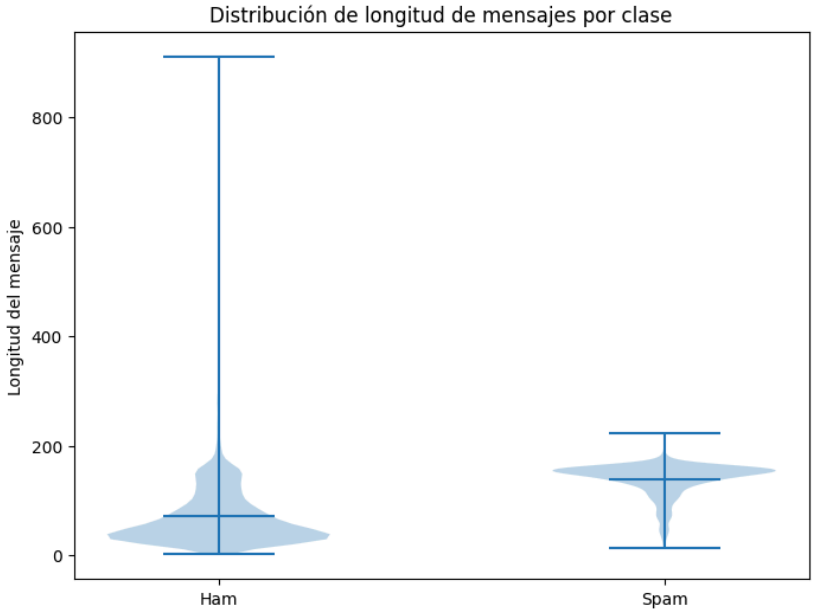

Figura 4: Distribución de longitud de mensajes por clase

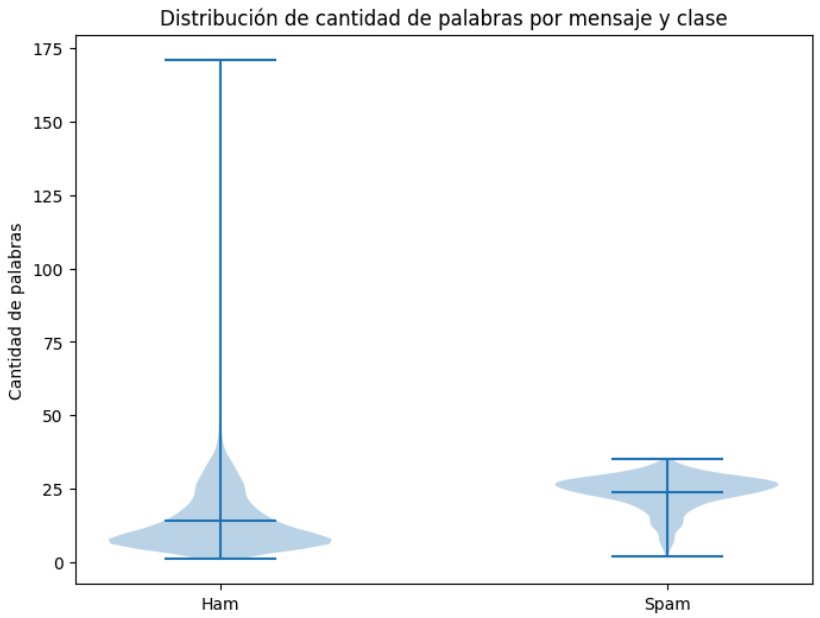

Figura 5: Distribución de cantidad de palabras por mensaje y clase

Como se puede observar en las gráficas superiores en donde los mensajes ham son lo s que tienen una longitud menor de mensajes a comparación de los mensajes spam, sin embargo por la naturaleza de los mensajes ham, es mucho más común que se tengan mensajes con longitudes atípicos ya que a pesar de que usualmente tienen una longitud más corta, esto no significa que no pueden haber mensajes largos.

Cómo último punto de la exploración de los datos fue ver la cantidad de mensajes en cada categoría como se puede ver en la siguiente gráfica:

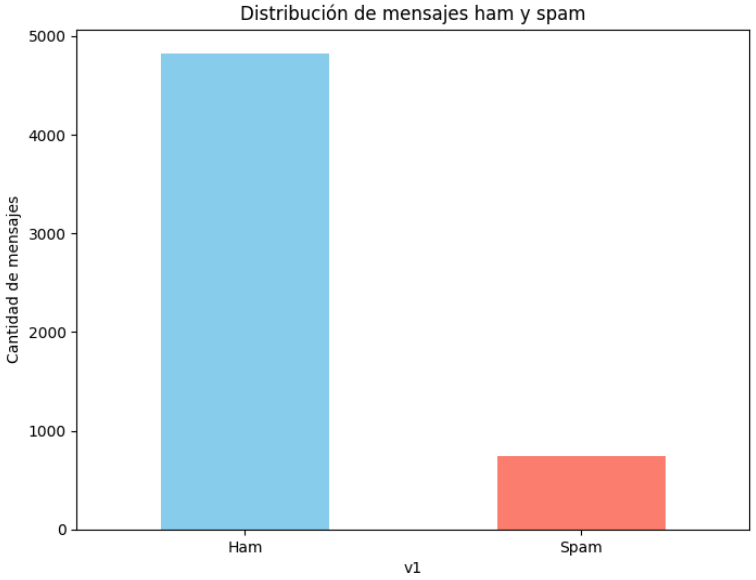

Figura 6: distribución de mensajes de ham y spam

Gracias a esta gráfica podemos observar que la cantidad de mensajes de ham sobrepasa de gran manera a los mensajes de spam.

Limpieza de datos

Objetivo:
El objetivo de este proceso es preparar datos de texto para su posterior análisis o modelado. Esto implica eliminar caracteres no deseados, convertir el texto a un formato uniforme, y reducir las palabras a su forma base para simplificar el análisis.

Herramientas Utilizadas:
- Python como lenguaje de programación.
- Biblioteca NLTK (Natural Language Toolkit) para el procesamiento de lenguaje natural.
- Expresiones regulares (re module) para manipulación de texto.

Pasos Realizados:
1. **Descarga de Recursos de NLTK:**
   Se descargó el tokenizador de NLTK utilizando la función nltk.download('punkt'). Este tokenizador es necesario para dividir el texto en palabras individuales.

2. **Importación de Bibliotecas y Recursos:**
   Se importaron las siguientes bibliotecas y recursos de NLTK:
   - stopwords: lista de palabras comunes que se pueden ignorar durante el análisis.
   - PorterStemmer: algoritmo para realizar stemming en palabras.
   - WordNetLemmatizer: algoritmo para lematizar palabras.

   Además, se importó el módulo re para trabajar con expresiones regulares.

3. **Definición de Stop Words, Stemming y Lematización:**
   Se definió un conjunto de stop words en inglés utilizando stopwords.words('english'). También se inicializaron objetos para realizar stemming y lematización en el texto.

4. **Función para Limpiar el Texto:**
   Se creó la función clean_text(text) que toma un texto como entrada y realiza las siguientes operaciones:
   - Remover caracteres especiales y números utilizando expresiones regulares.
   - Convertir el texto a minúsculas para uniformidad.

- Tokenizar el texto en palabras individuales.
- Remover las stop words del texto.
- Lematizar las palabras para reducirlas a su forma base.
- Aplicar stemming para reducir las palabras a su raíz léxica.

5. **Aplicación de la Función de Limpieza al DataFrame:**
   Se aplicó la función clean_text() a la columna de texto del DataFrame original, creando una nueva columna con el texto limpio y preprocesado.

6. **Creación de un Nuevo DataFrame con el Texto Limpio:**
   Se creó un nuevo DataFrame que contiene solo las columnas relevantes (etiqueta del mensaje y texto limpio) para facilitar el análisis posterior.

7. **Cambio de Nombres de Columnas:**
   Se cambiaron los nombres de las columnas para hacerlos más descriptivos y comprensibles.

8. **Verificación del Proceso de Limpieza:**
   Finalmente, se mostraron las primeras filas del DataFrame limpio para verificar que el proceso de limpieza se realizó correctamente. El DataFrame resultante se puede observar en la siguiente imagen.

	label	message
0	ham	[go, jurong, point, crazi, avail, bugi, n, gre...
1	ham	[ok, lar, joke, wif, u, oni]
2	spam	[free, entri, wkli, comp, win, fa, cup, final,...
3	ham	[u, dun, say, earli, hor, u, c, alreadi, say]
4	ham	[nah, dont, think, go, usf, life, around, though]

Figura 7: DataFrame resultante de la limpieza

Modelo

Para realizar el modelo se realizaron una serie de pasos requeridos para obtener las probabilidades solicitadas.

Probabilidad de que una palabra sea spam:

1. Recuento de Mensajes Spam y No Spam: Se contó el número total de mensajes etiquetados como spam y no spam en el conjunto de datos limpio.

Esto proporcionó los totales necesarios para calcular las probabilidades a priori.

2. Cálculo de Probabilidades a Priori: Se calcularon las probabilidades a priori de que un mensaje sea spam (P(S)) y de que sea no spam (P(H)) utilizando la frecuencia relativa de mensajes spam y no spam en el conjunto de datos.

3. Función para Calcular Probabilidades Condicionales: Se definió una función que calcula las probabilidades condicionales de que una palabra dada aparezca en un mensaje spam (P(W|S)) y en un mensaje no spam (P(W|H)). Esto se logró contando la frecuencia de ocurrencia de la palabra en cada tipo de mensaje y aplicando el suavizado de Laplace para evitar problemas de probabilidad cero.

4. Función para Calcular la Probabilidad de Spam dado una Palabra: Se creó una función que utiliza el teorema de Bayes para calcular la probabilidad de que un mensaje sea spam dado que contiene una palabra específica (P(S|W)). Esto se calculó utilizando las probabilidades a priori y las probabilidades condicionales de la palabra.

5. Cálculo de la Probabilidad de Palabras como Spam: Se seleccionó una palabra específica (en este caso, la palabra 'free') y se calculó la probabilidad de que un mensaje que contiene esa palabra sea spam utilizando la función previamente definida.

Forma de uso:

```
word = 'free'

P_spam_given_word = calc_spam_probability(word)

print(f'La probabilidad de que el mensaje sea spam dado que contiene la palabra "{word}" es: {P_spam_given_word:.4f}')
```

En donde el resultado fue el siguiente:

La probabilidad de que el mensaje sea spam dado que contiene la palabra "free" es: 0.7419

En donde se puede observar que tiene una probabilidad que tiene sentido ya que en el contexto de una conversación normal, no es usual usar la palabra ‘free’.

Probabilidad de que un texto sea spam

1. Función para Calcular Probabilidad de Spam dado un Texto: Se definió una función calc_spam_probability_text(text) que toma un texto como entrada y calcula la probabilidad de que sea spam. Esto se logró dividiendo el texto en palabras, calculando la probabilidad de spam para cada palabra utilizando la función calc_spam_probability() previamente definida, y luego combinando estas probabilidades para obtener la probabilidad total utilizando el teorema de Bayes.
2. División del Texto en Palabras y Cálculo de Probabilidades de Spam para Cada Palabra: Se dividió el texto en palabras utilizando la función clean_text() previamente definida. Luego, se calculó la probabilidad de spam para cada palabra utilizando la función calc_spam_probability().
3. Combinación de Probabilidades de Palabras para Obtener la Probabilidad Total del Texto: Se combinaron las probabilidades de spam para cada palabra utilizando el teorema de Bayes para obtener la probabilidad total de que el texto sea spam. Esto se realizó utilizando funciones de reducción para multiplicar y sumar las probabilidades de las palabras.
4. Presentación de Resultados: Se imprimió la probabilidad calculada para el texto dado como un mensaje informativo.

Forma de uso

```
text = 'claim money'
P_spam_given_text = calc_spam_probability_text(text)

print(f'La probabilidad de que el mensaje sea spam dado el texto "{text}" es: {P_spam_given_text:.4f}')
```

En este ejemplo se utilizó el texto de claim money como ejemplo de una gran probabilidad de spam y el resultado fue el siguiente:

La probabilidad de que el mensaje sea spam dado el texto "claim money" es: 0.8939.

Pruebas de rendimiento

Se realizó un análisis de rendimiento de un modelo de detección de spam utilizando técnicas de procesamiento de lenguaje natural. El modelo se entrenó utilizando un nuevo DataFrame que contiene mensajes etiquetados como spam o no spam, y se evaluó su rendimiento utilizando métricas estándar como precisión, recall y f1-score.

Para hacer el análisis se realizaron los siguientes pasos:

1. **Preparación del DataFrame de Entrenamiento y Pruebas:** Se dividió el DataFrame original en conjuntos de entrenamiento y pruebas utilizando la función train_test_split de scikit-learn. Se aplicaron las funciones de limpieza de texto (clean_text) a los conjuntos de entrenamiento y pruebas para preparar los datos para el modelado.
2. **Funciones de Predicción:** Se definieron funciones para calcular la probabilidad de que un texto dado sea spam (calc_spam_probability_text_predict) y para predecir si un texto es spam o no (predict_spam_probability_text). Estas funciones utilizan las probabilidades calculadas previamente para cada palabra y aplican el teorema de Bayes para realizar la predicción.
3. **Evaluación del Modelo:** Se aplicaron las funciones de predicción al conjunto de pruebas y se generó un DataFrame con las predicciones. Luego, se utilizó la función classification_report de scikit-learn para calcular las métricas de precisión, recall y f1-score del modelo.

Resultados de la evaluación

Clase	Precisión	Recall	F1-Score	Soporte
Ham	0.98	1.00	0.99	965
Spam	0.99	0.88	0.93	150
Accuracy			0.98	1115

- Precisión: La precisión mide la proporción de mensajes clasificados como spam que realmente son spam. En este caso, la precisión para ambas clases (spam y no spam) es alta, lo que indica que el modelo tiene una baja tasa de falsos positivos.
- Recall: El recall mide la proporción de mensajes spam que fueron correctamente identificados como spam. La tasa de recall para la clase spam es ligeramente menor que para la clase no spam, lo que indica que el modelo podría estar perdiendo algunos mensajes spam.
- F1-Score: El f1-score es una medida balanceada entre precisión y recall. En este caso, el f1-score para ambas clases es alto, lo que sugiere que el modelo es capaz de mantener un equilibrio entre precisión y recall.

Conclusiones

- Rendimiento general del modelo: El modelo de clasificación de spam/ham entrenado tiene un excelente rendimiento general, con una precisión (accuracy) del 98%. Esto indica que el modelo es capaz de clasificar correctamente la mayoría de los mensajes como spam o ham.
- Precisión en la clasificación de spam: El modelo tiene una alta precisión (0.99) en la clasificación de mensajes como spam. Esto significa que cuando el modelo predice que un mensaje es spam, tiene una probabilidad del 99% de estar en lo correcto. Esta precisión es importante para evitar el etiquetado incorrecto de mensajes legítimos como spam.
- Recall en la clasificación de spam: Si bien la precisión en la clasificación de spam es alta, el recall (0.88) es ligeramente más bajo. Esto indica que el modelo puede estar perdiendo algunos mensajes de spam reales y clasificándolos erróneamente como ham. Dependiendo de los requisitos del sistema, puede ser necesario ajustar el modelo para mejorar el recall en la clase de spam, incluso si eso implica un pequeño sacrificio en la precisión.
- Rendimiento en la clasificación de mensajes ham: El modelo tiene un excelente rendimiento en la clasificación de mensajes ham, con una precisión de 0.98 y un recall perfecto de 1.00. Esto significa que el modelo es altamente efectivo en identificar correctamente los mensajes legítimos y no clasificarlos erróneamente como spam.
- Desequilibrio de clases: Es importante notar que el conjunto de datos utilizado para el entrenamiento y la evaluación del modelo tenía un

desequilibrio de clases, con 965 instancias de ham y solo 150 instancias de spam. Este desequilibrio puede haber influido en el rendimiento del modelo y debe tenerse en cuenta al interpretar los resultados.

Link de repositorio

https://github.com/Jskenpo/PROYECTO2_IA.git