
# FACULTAD DE CIENCIAS
# INTELIGENCIA ARTIFICIAL

---

# Clasificación de Texto

---

Equipo: `Skynet Scribes`

Número de practica: `08`

**Sarah Sophía Olivares García**
318360638

**Marco Silva Huerta**
316205326

**Carlos Daniel Cortés Jiménez**
420004846

**Laura Itzel Tinoco Miguel**
316020189

**Luis Enrique García Gómez**
315063880

**Fernando Mendoza Eslava**
319097690

Profesora: Cecilia Reyes Peña
Ayudante teoría: Karem Ramos Calpulalpan
Ayudante laboratorio: Tania Michelle Rubí Rojas

**Semestre 2024-2**

Fecha de entrega:
**01 de Mayo del 2024**

# Índice



# 1. Investigación

## 1.1. Algoritmos de aprendizaje supervisados

Los algoritmos de aprendizaje supervisado experimentan un conjunto de datos que contiene características, pero cada ejemplo también está asociado con una etiqueta u objetivo. Por ejemplo, el conjunto de datos de Iris está anotado con la especie de cada planta de iris. Un algoritmo de aprendizaje supervisado puede estudiar el conjunto de datos de Iris y aprender a clasificar las plantas de iris en tres especies diferentes según sus mediciones. [GBC16]

Es decir aprenden a asociar alguna entrada con alguna salida, dado un conjunto de entrenamiento de ejemplos de entradas $x$ y salidas $y$. En muchos casos, los resultados y pueden ser difíciles de recopilar automáticamente y deben ser proporcionados por un *supervisor* humano, pero el término aún se aplica incluso cuando los objetivos establecidos de capacitación se recopilaron automáticamente.[GBC16]

Brevemente el contraste con los algoritmos de aprendizaje no supervisados experimentan un conjunto de datos que contiene muchas características y luego aprenden propiedades útiles de la estructura de este conjunto de datos. En el contexto del aprendizaje profundo, generalmente queremos aprender la distribución de probabilidad completa que generó un conjunto de datos, ya sea explícitamente como en la estimación de densidad o implícitamente para tareas como síntesis o eliminación de ruido. Algunos otros algoritmos de aprendizaje no supervisados desempeñan otras funciones, como el clustering, que consiste en dividir el conjunto de datos en

grupos de ejemplos similares.[GBC16]

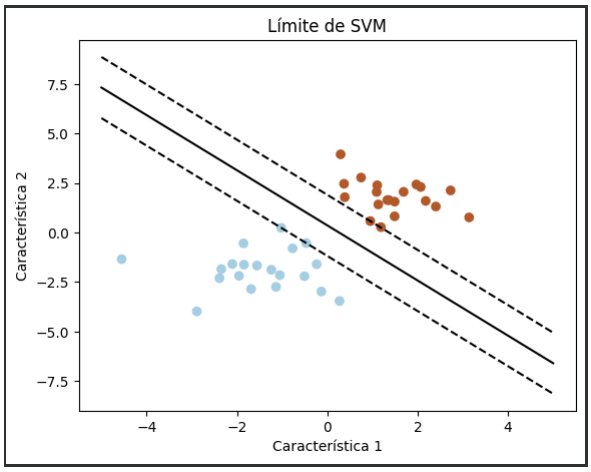
## 1.2. Support Vector Machine

Uno de los enfoques más influyentes para el aprendizaje supervisado es la máquina de vectores de soporte. Este modelo es similar a la regresión logística en el sentido de que está impulsado por una función lineal $w^\intercal x+b$. A diferencia de la regresión logística, la máquina de vectores de soporte no proporciona probabilidades, sino que solo emite una identidad de clase. La SVM predice que la clase positiva está presente cuando $w^\intercal x + b$ es positivo. De manera similar, predice que la clase negativa está presente cuando $w^\intercal x + b$ es negativo.[GBC16]

Una máquina de vectores de soporte (SVM) es un modelo de aprendizaje automático potente y versátil, capaz de realizar clasificación, regresión e incluso detección de novedades lineales o no lineales. Las SVM brillan con conjuntos de datos no lineales de tamaño pequeño y mediano (es decir, de cientos a miles de instancias), especialmente para tareas de clasificación. Sin embargo, como verá, no se adaptan muy bien a conjuntos de datos muy grandes. [Gér22]

La idea fundamental detrás de las máquinas de vectores de soporte (SVM) es encontrar el hiperplano que mejor separa dos clases en un espacio de características, maximizando el margen entre las instancias de entrenamiento más cercanas a él. En términos simples, busca la mejor *avenida* que divide dos grupos de puntos en un espacio multidimensional.

Por ejemplo, para la tarea de los correos electrónicos, clasificarlos como spam o no spam, tenemos un conjunto de datos con correos etiquetados y cada correo tiene características como la frecuencia de ciertas palabras clave, la longitud del mensaje, etc. Utilizando SVM, el algoritmo busca un hiperplano en este espacio de características que separe claramente los correos de spam de los correos legítimos. El margen entre este hiperplano y las instancias más cercanas de cada clase es maximizado, lo que ayuda a mejorar la capacidad de generalización del modelo.



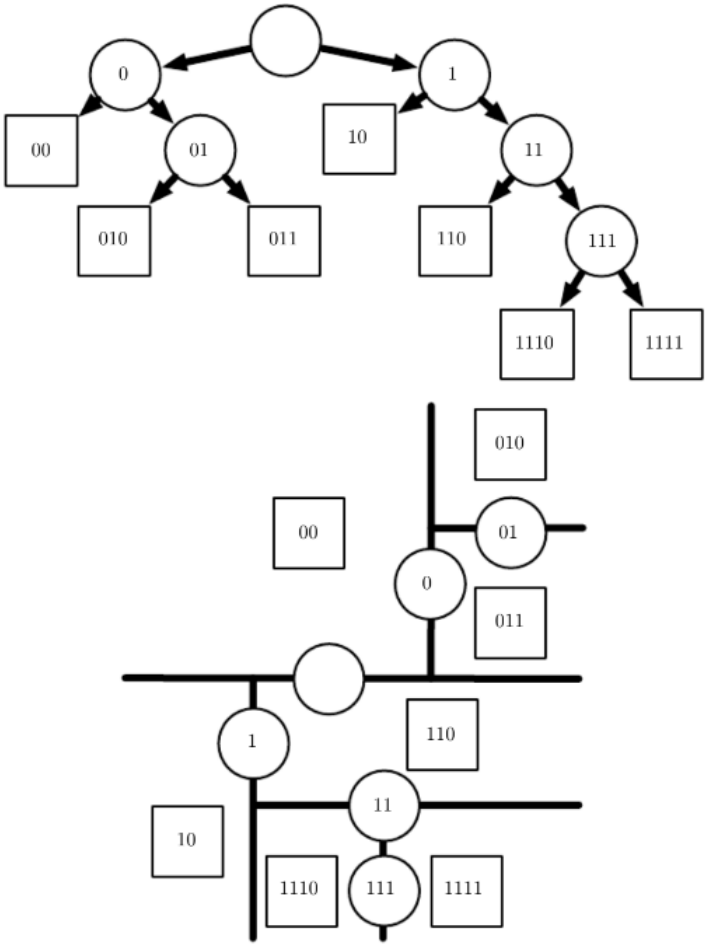

## 1.3. Decision Tree

Los árboles de decisión son algoritmos versátiles de aprendizaje automático que pueden realizar tareas de clasificación y regresión, e incluso tareas de múltiples salidas. Son algoritmos potentes, capaces de ajustar conjuntos de datos complejos. Los árboles de decisión también son los componentes fundamentales de los bosques

aleatorios, que se encuentran entre los algoritmos de aprendizaje automático más poderosos disponibles en la actualidad. [Gér22]

El algoritmo puede considerarse no paramétrico si se le permite aprender un árbol de tamaño arbitrario, aunque los árboles de decisión generalmente se regularizan con restricciones de tamaño que los convierten en modelos paramétricos en la práctica. Los árboles de decisión, tal como se utilizan habitualmente, con divisiones alineadas con los ejes y resultados constantes dentro de cada nodo, tienen dificultades para resolver algunos problemas que son fáciles incluso para la regresión logística. Por ejemplo, si tenemos un problema de dos clases y la clase positiva ocurre dondequiera que $x_2 > x_1$, el límite de decisión no está alineado con el eje. Por lo tanto, el árbol de decisión necesitará aproximarse al límite de decisión con muchos nodos, implementando una función de paso que recorra constantemente hacia adelante y hacia atrás a través de la verdadera función de decisión con pasos alineados con el eje. [GBC16]

**Cómo funciona un árbol de decisión.**




Cada nodo del árbol elige enviar el ejemplo de entrada al nodo secundario de la izquierda (0) o al nodo secundario de la derecha (1). Los nodos internos se dibujan como círculos y los nodos de las hojas como cuadrados. Cada nodo se muestra con un identificador de cadena binaria correspondiente a su posición en el árbol, que se obtiene agregando un bit a su identificador principal (0=elija izquierda o arriba, 1=elija derecha o abajo). (Abajo) El árbol divide el espacio en regiones.

El plano 2D muestra cómo un árbol de decisión podría dividir $\mathbb{R}^2$. Los nodos del árbol se trazan en este plano, con cada nodo interno dibujado a lo largo de la línea divisoria que utiliza para categorizar ejemplos,

y los nodos de hoja dibujados en el centro de la región de ejemplos que reciben. El resultado es una función constante por partes, con una parte por hoja. Cada hoja requiere al menos un ejemplo de entrenamiento para definirse, por lo que no es posible que el árbol de decisión aprenda una función que tenga más máximos locales que el número de ejemplos de entrenamiento. [GBC16]

## 1.4. Random Forest

El concepto de sabiduría de la multitud se refiere a la idea de que la respuesta colectiva de un grupo de personas al azar a menudo supera la respuesta individual de un experto. De manera similar, en el aprendizaje conjunto, al combinar las predicciones de múltiples predictores, como clasificadores o regresores, se obtienen predicciones más precisas que las de un único predictor. Por lo tanto, el aprendizaje conjunto implica el uso de algoritmos que combinan múltiples predictores para mejorar la precisión y la robustez de las predicciones.

Que significa todo lo de arriba? Los Random Forest (bosques aleatorios) es un algoritmo de aprendizaje supervisado que crea un bosque y lo hace de alguna manera aleatorio. Para decirlo en palabras simples: el bosque aleatorio crea múltiples árboles de decisión y los combina para obtener una predicción más precisa y estable. [Gon19]

Como ejemplo de método de conjunto, puede entrenar un grupo de clasificadores de árboles de decisión, cada uno en un subconjunto aleatorio diferente del conjunto de entrenamiento. Luego puede obtener las predicciones de todos los árboles individuales, y la clase que obtiene más votos es la predicción del conjunto. Este conjunto de árboles de decisión se denomina **bosque aleatorio** y, a pesar de su simplicidad, es uno de los algoritmos de aprendizaje automático más potentes. [Gér22]

Entonces un bosque aleatorio es un conjunto de árboles de decisión, generalmente entrenados mediante el método de embolsado, normalmente con $max\_samples$ establecido en el tamaño del conjunto de entrenamiento. En lugar de crear un BaggingClassifier y pasarle un DecisionTreeClassifier, puede usar la clase **RandomForestClassifier**, que es más conveniente y está optimizada para árboles de decisión (de manera similar, existe una clase RandomForestRegressor para tareas de regresión). El siguiente código es un ejemplo de como entrena un clasificador de bosque aleatorio con 500 árboles, cada uno limitado a un máximo de 16 nodos hoja, utilizando todos los núcleos de CPU disponibles:[Gér22]

```
1    from sklearn.ensemble import RandomForestClassifier
2
3    rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1,
     random_state=42)
4
5    rnd_clf.fit(X_train, y_train)
6
7    y_pred_rf = rnd_clf.predict(X_test)
```

Listing 1: bosque aleatorio con 500 árboles

# 2. Desarrollo

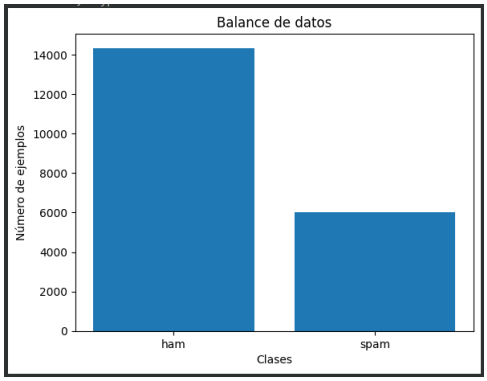
## 2.1. Revisión de datos

Una vez cargado el data set, imprimimos la grafica de barras usando:

```
    # Graficar un grafico de barras para visualizar el balance de clases
    plt.bar(class_counts.index, class_counts.values)
    plt.xlabel('Clases')
    plt.ylabel('Numero de ejemplos')
    plt.title('Balance de datos')
    plt.show()
```

Listing 2: codigo grafca de barras

Dentro de la columna $text\_type$ esta si el texto es spam o ham, al ver la grafica podemos ver como ser carga por casi más del doble los datos ham, dándonos un desbalanceo.




## 2.2. Oversampling de la Clase Minoritaria

- El oversampling implica aumentar la cantidad de ejemplos en la clase minoritaria para igualarla a la cantidad de ejemplos en la clase mayoritaria.
- Esto se puede hacer mediante duplicación de instancias existentes o generación de ejemplos sintéticos.
- Una técnica común es el método SMOTE (Synthetic Minority Over-sampling Technique), que crea ejemplos sintéticos interpolando entre instancias cercanas en el espacio de características.

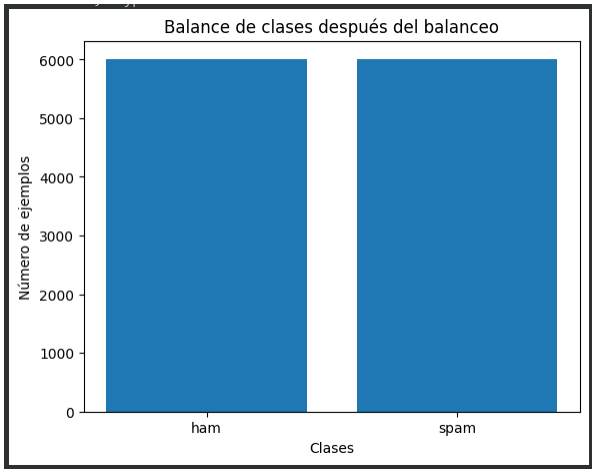
## 2.3. Undersampling de la Clase Mayoritaria

- El undersampling implica reducir la cantidad de ejemplos en la clase mayoritaria para igualarla a la cantidad de ejemplos en la clase minoritaria.
- Esto se puede hacer seleccionando aleatoriamente un subconjunto de instancias de la clase mayoritaria.
- Sin embargo, debes tener cuidado de no perder información importante al reducir el tamaño del conjunto de datos.

## 2.4. Elección de balanceo

La ventaja de Oversampling es poder conservar todos los datos disponibles sin embargo esto nos podría introducir un riesgo de sobreajuste si se aumenta demasiado la clase minoritaria, vemos que aumenta casi el doble por lo que al tener datos suficientes no usaremos Oversampling.

Mientras que para Undersampling al reducir el desbalance, puede mejorar la capacidad del modelo para aprender patrones contenidos en nuestros datos, sabemos que esto nos lleva a una perdida de información en filas que se eliminan pero en este caso consideramos más importante trabajar con datos nuestros.




## 2.5. Stopwords

Las stopwords, también conocidas como palabras vacías, son palabras comunes que aparecen con frecuencia en un idioma pero que tienen poco o ningún significado semántico. Estas palabras no aportan mucha información al significado general de un texto y, de hecho, pueden dificultar el procesamiento del lenguaje natural (PLN). Por lo tanto, eliminar las stopwords de un conjunto de datos de texto suele ser un paso previo importante en muchas tareas de PLN, como el análisis de sentimientos, la clasificación de texto y la recuperación de información.

## biblioteca NLTK

La biblioteca NLTK (Natural Language Toolkit) es una herramienta popular para el procesamiento del lenguaje natural en Python. NLTK incluye un conjunto de stopwords predefinido para varios idiomas, un ejemplo es:

```
import nltk

# Cargar el conjunto de datos de texto
texto = "Este es un ejemplo de texto para eliminar stopwords."

# Tokenizar el texto en palabras
palabras = nltk.word_tokenize(texto)

# Cargar las stopwords en espanol
stopwords = set(nltk.corpus.stopwords.words('spanish'))

# Eliminar stopwords de la lista de palabras
palabras_sin_stopwords = [palabra for palabra in palabras if palabra not in
stopwords]

# Unir las palabras restantes en una cadena de texto
texto_sin_stopwords = ' '.join(palabras_sin_stopwords)

print(texto_sin_stopwords)
```

Listing 3: bosque aleatorio con 500 árboles

## 2.6. TfidfVectorizer

TfidfVectorizer es una clase en la biblioteca scikit-learn de Python que se utiliza para convertir un conjunto de documentos de texto en vectores numéricos. Estos vectores, conocidos como vectores TF-IDF (Term Frequency-Inverse Document Frequency), representan la importancia de cada palabra en cada documento.

El proceso de conversión se realiza en dos pasos:

1. Cálculo de la frecuencia de término (TF):

   Para cada palabra en cada documento, se calcula la frecuencia de término (TF), que es la cantidad de veces que aparece la palabra en el documento.

2. Cálculo de la frecuencia inversa de documento (IDF):

   Para cada palabra en todo el conjunto de documentos, se calcula la frecuencia inversa de documento (IDF). La IDF mide qué tan rara es una palabra en el conjunto de documentos. Se calcula como el logaritmo del número total de documentos dividido por el número de documentos que contienen la palabra.

Finalmente la combinación de TF y IDF: para cada palabra en cada documento, se calcula el producto de TF e IDF. Este valor, conocido como puntuación TF-IDF, representa la importancia relativa de la palabra en el documento.

# 3. Análisis e interpretación de resultados

## 3.1. Código

Este es el código usado para imprimir la matriz, se uso en cada uno del los modelos para ver de mejor manera esta información.

```
    plt.figure(figsize=(8, 6))
    sns.heatmap(nombre_modelo, annot=True, fmt='d', cmap='Blues', cbar=False)
    plt.xlabel('Prediccion ')
    plt.ylabel('Valor Real')
    plt.title('Matriz de Confusion para <nombre modelo>')
    plt.show()
```

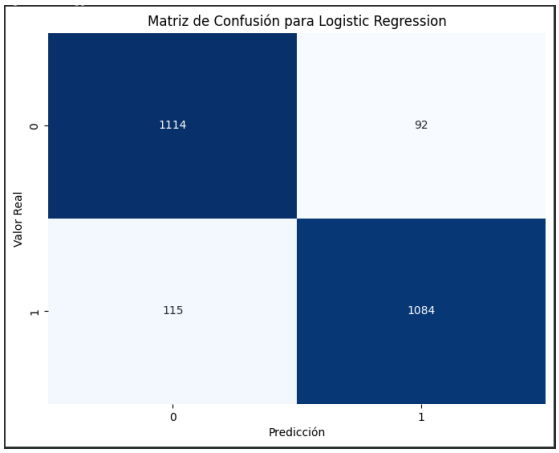
Listing 4: código de grafico de matriz

## 3.2. Logistic Regression

**Métricas de evaluación**

- Precisión: para la clase ham es del $91\,\%$, este es el porcentaje de los mensajes clasificados como ham fueron realmente ham.
- Recall: para la clase ham es del $92\,\%$, es decir el modelo identificó correctamente el $92\,\%$ de todos los mensajes ham.
- F1-score: para ambas clases (ham y spam) se obtuvo un equilibrio del $91\,\%$.
- Accuracy: La exactitud general del modelo es del $91\,\%$.

### Matriz de confusión



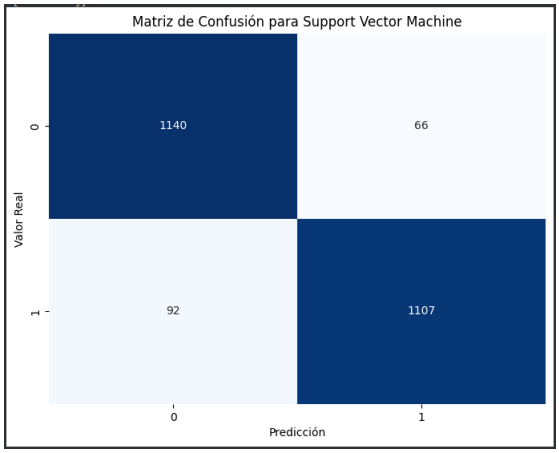

- Verdaderos positivos (TP): 1114 (mensajes 'ham' clasificados correctamente como 'ham').
- Falsos positivos (FP): 92 (mensajes 'spam' clasificados incorrectamente como 'ham').
- Falsos negativos (FN): 115 (mensajes 'ham' clasificados incorrectamente como 'spam').
- Verdaderos negativos (TN): 1084 (mensajes 'spam' clasificados correctamente como 'spam').

## 3.3. Support Vector Machine

### Métricas de evaluación

- Precisión: para la clase 'ham' es del $93{,}43\,\%$, es decir ese porcentaje de mensajes clasificados como 'ham' fueron realmente 'ham'
- Recall: identificó correctamente el $95\,\%$ de todos los mensajes 'ham'
- F1-score: para ambas clases (ham y spam) se obtuvo un equilibrio del $93\,\%$.
- Accuracy: La exactitud general del modelo es del $93\,\%$.

### Matriz de confusión




- Verdaderos positivos (TP): 1140 (mensajes 'ham' clasificados correctamente como 'ham').

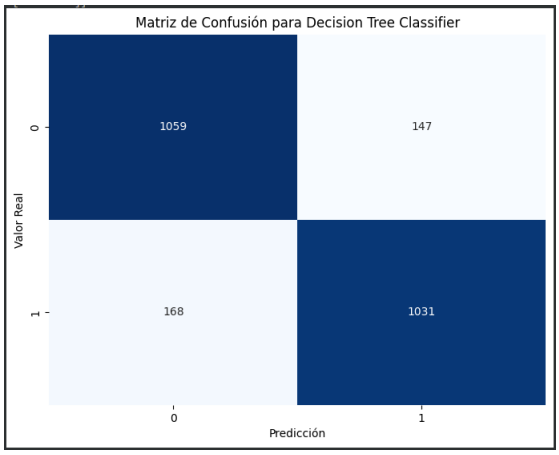
- Falsos positivos (FP): 66 (mensajes 'spam' clasificados incorrectamente como 'ham').
- Falsos negativos (FN): 92 (mensajes 'ham' clasificados incorrectamente como 'spam')
- Verdaderos negativos (TN): 1107 (mensajes 'spam' clasificados correctamente como 'spam')

## 3.4. Decision Tree

### Métricas de evaluación

- Precisión: $86{,}90\,\%$ de los mensajes clasificados como 'ham' fueron realmente 'ham'.
- Recall: el modelo identificó correctamente el $88\,\%$ de todos los mensajes 'ham'
- F1-score: para ambas clases (ham y spam) se obtuvo un equilibrio del $87\,\%$.
- Accuracy: La exactitud general del modelo es del $87\,\%$

### Matriz de confusión



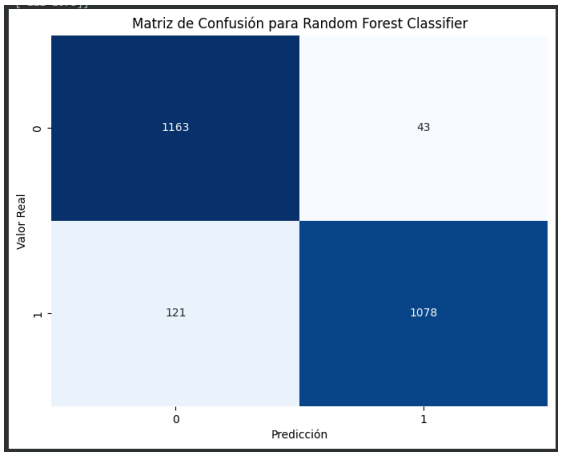

- Verdaderos positivos (TP): 1059 (mensajes 'ham' clasificados correctamente como 'ham').
- Falsos positivos (FP): 147 (mensajes 'spam' clasificados incorrectamente como 'ham').
- Falsos negativos (FN): 168 (mensajes 'ham' clasificados incorrectamente como 'spam').
- Verdaderos negativos (TN): 1031 (mensajes 'spam' clasificados correctamente como 'spam').

## 3.5. Random Forest

### Métricas de evaluación

- Precisión: el $93{,}18\,\%$ de los mensajes clasificados como 'ham' fueron realmente 'ham'.
- Recall: el modelo obtuvo un $96\,\%$ correcto de todos los mensajes 'ham'.
- F1-score: para ambas clases (ham y spam) se obtuvo un equilibrio del $93\,\%$.
- Accuracy: La exactitud general del modelo es del $93\,\%$

**Matriz de confusión**




- Verdaderos positivos (TP): 1163 (mensajes 'ham' clasificados correctamente como 'ham').
- Falsos positivos (FP): 43 (mensajes 'spam' clasificados incorrectamente como 'ham').
- Falsos negativos (FN): 121 (mensajes 'ham' clasificados incorrectamente como 'spam').
- Verdaderos negativos (TN): 1078 (mensajes 'spam' clasificados correctamente como 'spam').

## 3.6. Análisis equipo

- Marco Silva Huerta

  Para el modelo de Logistic Regression su matriz de confusión muestra un rendimiento bastante bueno con un bajo número de falsos positivos y falsos negativos, eso le da un equilibrio del $91\,\%$ en la clasificación de ham y spam. Pero para el modelo SVM la precisión en números es del $0{,}9343$, esto pasa porque SVM tiene menos falsos positivos que Logistic Regression.

  Ahora cuando volteamos a ver a Decision Tree Classifier, muestra un desempeño aceptable en relación a los dos primeros pues cuenta con un $87\,\%$ de exactitud general, esto se pude explicar ya que Los árboles de decisión son propensos al sobreajuste cuando se entrenan con conjuntos de datos complejos o desbalanceados ya que sus nodos al dividirse pueden tener dificultades para generalizar patrones en datos que no han visto durante el entrenamiento.

  Finalmente el modelo Random Forest Classifier muestra un rendimiento muy muy bueno, fue el que mayor positivos verdaderos tuvo con 1163 y pese a tener también un $93\,\%$ en la exactitud del modelo, queda segundo lugar pues tiene un $0{,}9318$, ligeramente por debajo de SVM pero arriba de su individual árbol de decisión, esto porque Random Forest utiliza múltiples árboles de decisión en lugar de uno solo, la combinación de las predicciones reduce el riesgo de sobreajuste.

- Fernando Mendoza Eslava

  El ajuste de `class_weight='balanced'` ha sido esencial en nuestros modelos, permitiendo que estos presten más atención a la clase minoritaria 'spam'. Este enfoque ha ayudado a mejorar el *recall* de la clase minoritaria sin sacrificar significativamente la precisión, un balance crucial en aplicaciones reales donde el costo de predecir erróneamente un 'spam' como 'ham' puede ser alto.

Elegimos utilizar `TfidfVectorizer` por su capacidad para no solo contar la frecuencia de palabras, sino también ajustar estos conteos basándose en la importancia de las palabras dentro del corpus total del documento. Esta técnica ha probado ser efectiva en resaltar las palabras más relevantes y en minimizar el ruido proveniente de palabras comunes pero irrelevantes.

La evaluación detallada utilizando precisión, *recall*, *F1-score* y matrices de confusión ha revelado la robustez de los modelos SVM y Random Forest, que han mostrado un excelente balance entre precisión y capacidad de *recall*. Estas métricas son fundamentales para entender el verdadero rendimiento en un escenario de clases desbalanceadas.

- Carlos Daniel Cortés Jiménez

  Notemos que los modelos como *Random Forest Classifier*, *Logistic Regression* y *Support Vector Machine (SVM)* estos presentan un mejor rendimiendo a comparación del modelo *Decision Tree Classifier* aunque la diferencia no sea tan grande, además la tasa de verdaderos positivos con la que se puede detectar mejor mensajes de tipo ham es el modelo *Random Forest Classifier* con una precisión del $95\,\%$. Por otro lado el modelo con la mejor detección de mensajes de tipo spam es *Support Vector Machine (SVM)* con un porcentaje del $96\,\%$.

  Podemos decir entonces que los modelos que tienen el mejor equilibrio entre precisión y recall(sensibilidad) son los mdelos anteriores mencionados siendo Random Forest Classifier y *Support Vector Machine (SVM)*, con porcentajes altos de detección de mensajes de tipo spam y ham, siendo lo que deberiamos de usar a la hora de desarrollar programas que tengan el mejor rendimiento y la mejor precisión.

# Referencias


[GBC16] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[Gon19] Ligdi González. *Machine Learning con Python Aprendizaje Supervisado*. 2019.

[Gér22] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2022. ISBN: 9781098122461. URL: https://books.google.com.mx/books?id=X5ySEAAAQBAJ.

[gee24] geeksforgeeks. *Stopwords*. 2024. URL: https://www.geeksforgeeks.org/removing-stop-words-nltk-python/.

[sci24] scikit-learn. *TfidfVectorizer*. 2024. URL: https://www.geeksforgeeks.org/removing-stop-words-nltk-python/.