Nota de Aceptación

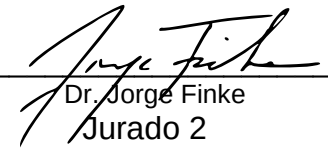
Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar el título de Ingeniero de Sistemas y computación.

Dr. Camilo Rocha
Decano de la Facultad de Ingeniería

Dr. Gerardo Sarria
Director Carrera Ingeniería de Sistemas y Computación.

Dra. María Constanza Pabón
Director(a) Trabajo

Dr. Diego Linares
Jurado 1



Dr. Jorge Finke
Jurado 2

Pontificia Universidad JAVERIANA Cali

Facultad de Ingeniería y Ciencias

Acta de Correcciones al Proyecto de Grado
Ingeniería de Sistemas y Computación

Fecha: 01/09/ 2021

Autores: Brayan David Vera Leyton

Nombre del Proyecto de Grado: Detección de publicaciones generadas por bots en twitter

Directora: María Constanza Pabón

Como indica el artículo 2.27 de las Directrices de Trabajo de Grado, he verificado que los estudiantes indicados arriba han implementado todas las correcciones que los Jurados del Proyecto de Grado definieron que se efectuaran, como consta en el Acta de Calificación correspondiente.

Firma de Director(a) del Proyecto de Grado

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería.
Ingeniería de Sistemas y Computación.
Anteproyecto de Grado.

Detección de publicaciones generadas por bots en twitter

Brayan David Vera Leyton

Director: Dr. Maria Constanza Pabón

Julio del 2021

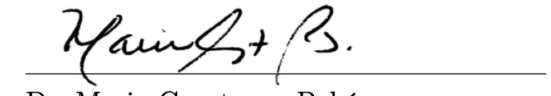
Santiago de Cali, Julio del 2021.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria Montemiranda
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo.

Por medio de la presente me permito informarle que he revisado el proyecto de grado titulado "Detección de publicaciones generadas por bots en twitter" del estudiante de Ingeniería de Sistemas y Computación Brayan David Vera Leyton (cod: 8918691), del cual soy directora y lo considero apto para ser presentado y sometido a consideración del jurado.

Atentamente,



Dr. Maria Constanza Pabón
Directora Trabajo de Grado
Pontificia Universidad Javeriana Cali

Santiago de Cali, Julio del 2021.

Señores
Pontificia Universidad Javeriana Cali.
Director Carrera de Ingeniería de Sistemas y Computación.
Dr. Gerardo Mauricio Sarria Montemiranda
Cali.

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado "Detección de publicaciones generadas por bots en twitter" con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el proyecto de grado y posteriormente optar al título de Ingeniero de Sistemas y Computación.

Atentamente,

Brayan David Vera Leyton
Código: 8918691

Abstract

The presence of bots on social media platforms such as Twitter, Facebook and Instagram, have generated substantial problems in the user community, since most of these bots are malicious and seek to disseminate information, which in many cases can be erroneous and detrimental to the good name of the community that uses these types of platforms. Therefore, the detection of social bots is a defense mechanism made for users against defamation and damage to the image of each of those affected by automated accounts. In addition, it helps companies to filter large volumes of misinformation that damages the image and the reputation of these companies.

In this paper, an investigation of studies conducted in recent years in detecting bots on social networks was carried out. This study has been made with the objective of selecting two machine learning techniques used for this specific task. Decision tree and convolutional neural network were the techniques selected, implemented and evaluated by means of the most used metrics in the articles examined, in addition to comparing the results obtained between both techniques.

Keywords::

- Bots
- Machine Learning
- Metrics
- Preprocessing
- Twitter

Resumen

La presencia de los bots en plataformas de redes sociales tales como Twitter, Facebook e Instagram, han generado una gran problemática en la comunidad de usuarios, ya que la mayoría de estos bots son maliciosos y buscan difundir información, que en muchos casos puede ser errónea y perjudicial para el buen nombre de la comunidad que usa este tipo de plataformas. Por lo cual, la detección de bots sociales es un mecanismo de defensa para los usuarios en contra de la difamación y el daño al buen nombre de cada uno de los afectados por cuentas automatizadas. Además, ayuda a las empresas a filtrar grandes volúmenes de información errónea que perjudica la imagen y el buen nombre de dichas empresas.

En este proyecto de grado se realizó una investigación de los estudios realizados en los últimos años en la detección de bots en las redes sociales. Este estudio se realizó con el objetivo de seleccionar dos técnicas de aprendizaje automático utilizadas para esta tarea en especifico. Árbol de decisión y red neuronal convolucional, fueron las técnicas seleccionadas, implementadas y evaluadas por medio de las métricas más utilizadas en los artículos examinados, además de comparar los resultados obtenidos entre ambas técnicas.

Palabras Clave:

- Bots
- Aprendizaje Automático
- Métricas
- Preprocesamiento
- Twitter

Índice general

Introducción

El lenguaje escrito es un mecanismo, que desde su invención en el siglo IV, ha permitido que las diferentes culturas plasmen y comuniquen sus conocimientos e ideas a otras culturas. Lo cual es importante porque ayudó a masificar la información, ya que genera un puente entre culturas para reproducir y compartir la información con todos los integrantes que lo deseen. No obstante, presenta grandes retos debido a la variedad de culturas y comunidades que lo utilizan,ya que la interpretación de la información compartida por medio de este mecanismo puede ser variable dependiendo del contexto de cada comunidad.

El lenguaje escrito fue evolucionando con el pasar de los años y debido a la aparición de las nuevas tecnologías se ha implementado como una herramienta para lograr la comunicación a distancia. El internet y las plataformas, conocidas actualmente como redes sociales, utilizan el lenguaje escrito como forma de comunicación e interconexión cultural a distancia. Por otro lado, la amplitud y facilidad de uso de las redes sociales las ha convertido en un entorno ideal para la aparición de agentes conocidos como bots sociales, los cuales son robots que generan contenido automáticamente. Estos interactúan con otras personas, intentan imitar y cambiar el comportamiento de los demás [1].

Las cuentas de redes sociales automatizadas y semi automatizadas pueden generar una problemática social, ya que en su mayoría, los bots apuntan a tener más visitantes a sus sitios web maliciosos, influir en la comunidad sobre un tema específico, manipular a las personas para tomar decisiones, propagar algunas noticias falsas y chantajear a las personas para no difundir su información privada. Es por esto que es importante la detección de bots sociales, que tiene como objetivo ayudar a mantener la estabilidad social, evitar estafas en la red y garantizar la seguridad de la privacidad de los usuarios.

Por otro lado, en este trabajo se aplicaron las técnicas de árbol de decisión y red neuronal convolucional en la clasificación de los datos. Los resultados obtenidos de la clasificación muestran un alto porcentaje en *precision* y el *recall* para las dos técnicas, el árbol de decisión obtuvo 99,992 % en *precision* y 99,997 % en *recall*, la red convolucional obtuvo 95,70 % en *precision* y 95,54 % en *recall*.

Capítulo 1

Descripción del Problema

1.1. Planteamiento del Problema

El desarrollo de Internet y el crecimiento exponencial de la popularidad de las redes sociales tales como Twitter, Facebook e Instagram, ha dado origen a una gran problemática en la sociedad que es la cantidad de cuentas que están controladas por agentes automatizados, programados para imitar mediante técnicas sofisticadas y engañosas el comportamiento de las cuentas de usuarios humanos. Según un estudio de Varol et al., entre el 9 % y el 15 % de las cuentas de Twitter son cuentas de bot [2]. Por lo general, estos bots o cuentas de spam están diseñados para transmitir o difundir cierto tipo de información como, por ejemplo, los vendedores que intentan ofrecer productos o atraer usuarios a nuevos sitios. Otros tipos de bots son mucho más maliciosos y difunden información errónea que perjudica o engaña a los usuarios. Dichos bots (cuentas falsas) pueden tener graves consecuencias, ya que pueden influir en una comunidad sobre un tema específico, difundir información errónea, reclutar personas para organizaciones ilegales o manipular personas para tomar acciones que afecten a alguien más.

Debido a la presencia de los bots en la redes sociales y que las redes sociales se han convertido en uno de los factores determinantes e influenciadores en la toma de decisiones en la vida cotidiana de los usuarios, permitiendo que estas "cuentas falsas" tengan todo el potencial de influir drásticamente en los eventos de la sociedad.

Por tal motivo, la comunidad tecnológica ha comenzado a investigar múltiples soluciones que han tenido como objetivo abordar los problemas asociados con el uso de cuentas automatizadas en las redes sociales proponiendo diferentes métodos por los cuales los bots sociales pueden ser detectados y bloqueados. Por ejemplo, un estudio con un enfoque basado en wavelet para la clasificación de cuentas sybil(cuentas administradas por un bot) un nuevo esquema de ponderación, llamado LBCA (*Lexicon Based Coefficient Attenuation*) mediante técnicas como bosques aleatorios y perceptrones multicapa; obtuvo como resultado que el esquema LBCA superó las técnicas presentadas en la literatura y lograron una precisión media del 94 % [3]. En otro estudio, se utilizó la minería de patrones temporales sobre las actividades de los bot en Twitter mediante la aplicación de algoritmos de minería de patrones de series de tiempo de última generación en series de tiempo de actividad tales como descubrimiento de motivos, descubrimiento de la discordia, subsiguiente unión y dinámica racimo. Los autores concluyeron que los aspectos temporales de los bots han sido ignorados en gran medida y descubrieron que estos aspectos tienen un gran impacto en la detección de bots. [4].

Este trabajo presenta la implementación de dos técnicas de aprendizaje para entrenar algunos agentes con el fin de detectar si una cuenta es administrada por un bot o por un ser humano. Esto ayudará a las empresas para poder filtrar grandes volúmenes de cuentas que escribieron post en las redes sociales generando información sobre estas que puede ser información perjudicial para la imagen o buen nombre de dicha empresa, ya que aproximadamente el 32 % del contenido que se publica en Twitter es generado por bots [5]. En el mercado, el nivel de competencia es alto entre las empresas y, en ocasiones, utilizan técnicas para desprestigiar a otras a través de las redes sociales. Debido a esto, la detección de contenido generado por bots es un mecanismo de defensa contra la difamación del buen nombre de las empresas.

1.1.1. Formulación

¿Cómo detectar publicaciones generadas por bots en tweets mediante técnicas de aprendizaje automático supervisado ?

1.1.2. Sistematización

- ¿Cuáles métodos de aprendizaje automático han tenido mejores resultados en la detección de publicaciones generadas por bots?
- ¿Cuáles técnicas de preprocesamiento se han aplicado para la detección de publicaciones generadas por bots?
- ¿Cómo evaluar los resultados obtenidos de la aplicación de las técnicas de aprendizaje automático en la detección de publicaciones generadas por bots?
- ¿Cómo difieren los resultados de aplicar técnicas de aprendizaje automático usando el texto de los tweets y los metadatos (como la hora de generación y la fecha de publicación), a comparación de aplicar las técnicas de aprendizaje automático con sólo el texto de los tweets para la detección de publicaciones generadas por bots?

1.2. Objetivos

1.2.1. Objetivo General

Detectar publicaciones generadas por bots en twitter mediante técnicas de aprendizaje automático

1.2.2. Objetivos Específicos

- Identificar los métodos de aprendizaje automático que se han aplicado en la detección de publicaciones generadas por bots en twitter.

- Identificar las técnicas de preprocesamiento que se han aplicado para la detección de publicaciones generadas por bots según la literatura.
- Generar modelos de predicción de tweets publicados por bots, aplicando dos técnicas de aprendizaje automático usando dos tipos de dataset: el primero con el texto de los tweets y los metadatos (como la hora de generación y la fecha de publicación) y el segundo solamente con el texto del tweet para la detección de publicaciones generadas por bots.
- Comparar los resultados obtenidos de aplicar las dos técnicas previas de aprendizaje automático usando el texto de los tweets y los metadatos (como la hora de generación y la fecha de publicación), y sólo usando el texto de los tweets para la detección de publicaciones generadas por bots.

1.3. Justificación

Los *social bots*, son agentes que habitan en una plataforma social para producir contenido automáticamente o tratan de imitar y alterar el comportamiento de los humanos que interactúan con los bots. Los usuarios ilegales, a menudo, usan bots sociales para manipular opiniones públicas, difundir rumores y producir valoraciones o reseñas falsas. Por lo tanto, estos robots sociales maliciosos traen efectos negativos a la seguridad pública e individual, ya que las redes sociales han ido evolucionando gradualmente como la plataforma principal para los usuarios de Internet. Los problemas causados por los bots sociales son cada vez más visibles. Por ejemplo, en Siria, se utilizó un bot social para inundar Twitter con información relacionada con la guerra civil con temas irrelevantes que distrajeron la atención de los usuarios de las controvertidas acciones del gobierno [6].

Otro ámbito que se puede ver afectado es el sector empresarial, debido a las grandes cantidades de información que pueden generar los bots sociales e inclusive, información errónea que puede afectar el buen nombre de una empresa, como lo fue el caso de Cynk, causándole grandes daños financieros mediante una caída de 220 veces en el precio de mercado como resultado de las actividades de los robots sociales automatizados del comercio de acciones[7].

Partiendo de lo anterior, se puede observar que es importante obtener información detallada sobre las capacidades y limitaciones de las técnicas de detección de bots sociales que se utilizan actualmente en las plataformas de redes sociales como Twitter y mediante la comparación, evaluación y un enfoque basado en el aprendizaje automático , podemos desarrollar una comprensión de las soluciones disponibles que podemos aplicar para identificar cuáles tecnologías logran la mejor precisión y resultados de detección de bots. Por lo cual, el grupo de investigación Destino de la Pontificia Universidad Javeriana Cali ha desarrollado un interés en este tema. A través de este proyecto se busca establecer un conocimiento base en el área de la detección de bots sociales,de tal manera que se abra la posibilidad para que el grupo de investigación pueda continuar desarrollando este campo en futuras investigaciones.

Capítulo 2

Marco De Referencia

2.1. Áreas Temáticas

- Computing methodologies - Machine learning – Learning paradigms – Supervised learning
- Computing methodologies - Machine learning – Machine learning approaches.
- Computing methodologies – Artificial intelligence – Natural language processing.

2.2. Marco Teórico

En esta sección se describen temas como el preprocesamiento de datos, tipos de aprendizaje automático, tareas del aprendizaje automático, técnicas de validación de los modelos y métricas para la evaluación de los modelos. Todos estos temas son importantes para la realización de un proyecto como este, en el cual se busca aplicar diferentes técnicas de aprendizaje automático. A continuación, se presentarán los temas y los elementos principales de los mismos.

2.2.1. Preprocesamiento

El preprocesamiento de datos es una área amplia, la cual consta de diferentes técnicas y estrategias, mediante las cuales busca adaptar los datos de la forma más adecuada para realizar la minería de datos utilizando las diferentes formas de aprendizaje automático.Por otro lado, el preprocesamiento ayuda a identificar y tomar decisiones sobre las posibles soluciones frente a situaciones en las que la información está incompleta o es inconsistente. A continuación, se presentan las técnicas más comunes que se utilizan durante el preprocesamiento de información y la descripción de dichas técnicas basadas en el libro de Pang, Steinbach, Kumar[8].

- Agregación: es la combinación de varios objetos en un solo objeto. Esto reduce los cientos o miles de objetos que pueden presentarse en un conjunto de datos y por lo tanto, la agregación puede permitir el uso de algoritmos de minería de datos más costosos ya que al reducir la cantidad de objetos el costo de procesamiento aceptable.
- Discretización y binarización: esta técnica es aplicada principalmente cuando se van a utilizar algoritmos de clasificación, ya que estos requieren que los datos estén en forma de atributos categorizados, por lo cual esta técnica busca convertir atributos continuos en categóricos (discretización) o convertir atributos continuos o discretos en atributos binarios (binarización).

- Reducción de la dimensionalidad: los conjuntos de datos llegan a tener muchos atributos, la reducción de dimensionalidad es una técnica que se aplica para crear nuevos atributos que son una combinación de otros atributos, esto puede ser una ventaja ya que puede eliminar atributos irrelevantes o el ruido en la información para poder generar un nuevo conjunto de datos comprensible. Por último, al aplicar esta técnica puede llegar a reducir el tiempo y la memoria requerida por un algoritmo de minería de datos.

- Selección de un subconjunto de atributos: esta técnica selecciona un subconjunto de datos en el cual todos los atributos son relevantes para el sistema, buscando así eliminar atributos irrelevantes para el mismo.

2.2.1.1. Preprocesamiento de texto

1. Extracción de características: La extracción de características tiene cómo objetivo transformar la información en características o atributos que tenga una significancia alta para el modelo [9].

   - N-gramas: Los N-gramas son un conjunto de n palabras consecutivas presentes en un documento de texto. Los n-gramas son utilizados para ver cómo es posible que una palabra aparezca seguida de otras palabras siendo útiles en aplicaciones de análisis de sentimiento, clasificación de texto y generación de texto.
   - Stemming: Es el proceso de reducir una palabra que tiene prefijos o sufijos a la raíz de la palabra. Con este proceso se busca reducir la cantidad de palabras únicas a una raíz en común y aumentar la frecuencia de aparición de dicha raíz en el documento. Por ejemplo: “Likes”, “liked”, “likely” se reducen a la raíz “like”.
   - Tokenizar: Consiste en tomar un texto y cortar este en partes o dividirlo en pequeños elementos con significado propio, en el caso de los texto, las palabras.
   - *Part of Speech*: Cataloga cada una de las palabras de un texto en Sustantivos, adjetivos, verbos, entre otros. Este proceso puede ser útil en el análisis de sentimiento debido a que una misma palabra puede tener diferentes significados dependiendo del contexto.

2. Selección de características: La selección de características se realiza para filtrar los atributos con mayor grado de relevancia para el modelo y se descartan aquellos atributos que no [10].

   - *Term Frequency (TF)*: Indica la frecuencia con la que aparece un término en un documento, en este caso, los términos corresponden a palabras o frases. la frecuencia del término se divide por el número total de términos en el documento (Ver Ecuación 2.1), ya que es posible que un término aparezca con más frecuencia en documentos más largos que en documentos más cortos.

     t = Término
     X = Número de veces que aparece el término.

T = Cantidad total de términos en el documento.

$$TF(t) = \frac{X}{T} \tag{2.1}$$

- *Chi-square*: Se utiliza como una medida estadística para probar la independencia entre dos variables. Si las variables están relacionadas *Chi-square* devuelve un valor bajo, mientras que si las variables no lo están el valor que retorna es alto.
- *Information Gain*: Se utiliza como una medida que indica cuánta información nos puede suministrar un atributo acerca de una clase. Es decir, ese atributo, que tanta ganancia genera para la clasificación.

2.2.2. Tipos de aprendizaje automático:

Una de las maneras que se pueden clasificar los tipos de aprendizaje automático son mediante la forma como "aprenden" o mejoran sus resultados. A continuación, se presentan los tres tipos de aprendizaje más comunes:

- Aprendizaje supervisado [11]: se basa en la información de una muestra de datos etiquetados con el resultado que el modelo debe responder. Después de que se ejecuta el modelo se obtiene la salida y esta va a ser comparada con las salidas esperadas. Es decir, los resultados conocidos de los datos para realizar ajustes y revaluar los datos convirtiendo el error como su nuevo aprendizaje para obtener un mejor entrenamiento.

- Aprendizaje no supervisado[11]: este aprendizaje no requiere que la muestra de datos se encuentre etiquetada con el resultado que el modelo debe responder. Solo necesita recibir los datos, ya que cuenta con la capacidad de aprender y organizar los datos sin necesidad de revaluar los errores para evaluar la posible respuesta.

- Aprendizaje reforzado[12]: está enfocado en realizar funciones de recompensa o castigo al algoritmo después de que este realizó un conjunto de acciones. Se puede evidenciar que el aprendizaje reforzado no es supervisado, ya que no requiere que el conjunto de datos esté etiquetado con la respuesta que se desea. Por otro lado, tampoco es no supervisado puesto que el modelo sí sabe la respuesta frente a un conjunto de acciones que vendría a ser la recompensa.

2.2.3. Métodos de clasificación

A continuación se presentan y se describen los métodos de aprendizaje automático y aprendizaje estadístico más utilizados en la detección de bots en redes sociales, basados en la literatura recopilada.

- *Decision tree*: Los árboles de decisión representan muy bien las relaciones no lineales y pueden ser utilizados en problemas de regresión o problemas de clasificación. Un árbol de decisión genera la clasificación del individuo que se esta analizando a partir de los posibles valores que pueden tomar uno o varios atributos [8]. En los árboles de decisión cada nodo que no sea terminal representa un atributo y cada arista es una respuesta al atributo del nodo que sale, los tipos de nodos son:

  Raíz - Representa a toda la población o muestra y esto se divide en dos o más conjuntos homogéneos, no tiene aristas de entrada pero cuenta con cero o más aristas de salida.

  Nodo Interno (padre e hijo) - Cuenta con una arista de entrada y al menos dos de salida.

  Nodo Terminal (hoja) - Representa una clase, tiene una sola arista de entrada y no cuenta con aristas de salida.

  Debido a las diferentes divisiones que pueden generar los nodos, resultan diferentes árboles de decisión de un mismo conjunto de atributos. Con el objetivo de obtener el árbol que genere la mejor exactitud se han desarrollado algunos criterios (o índices) para dividir los nodos y obtener obtener una separación de datos más pura. Los criterios más utilizados son:

$$Gini(t) = \sum_{i=0}^{c} [p(i|t)]^2 \tag{2.2}$$

$$Entropy(t) = \sum_{i=0}^{c} p(i|t) \log_2 p(i|t) \tag{2.3}$$

$$Classification\ error(t) = 1 - max[p(i|t)] \tag{2.4}$$

  Donde p(i|t) es la cantidad de registros que pertenecen a una clase i en un nodo t, c es la cantidad de clases, y t es un nodo del árbol.

- *Bayesian Classifiers*: Basados en los diferentes artículos recopilados, se presentan diferentes tipos de clasificadores bayesianos que utilizaron para la detección de bots en redes sociales [13], mediante el uso del teorema de bayes (Ver Ecuación 2.5). Los tipos más comunes son:

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)} \tag{2.5}$$

  1. *Naive Bayes*: El clasificador bayesiano ingenuo *(Naive Bayes)* utiliza el teorema de Bayes,este clasificador asume que los atributos son condicionalmente independientes.

     El *Naive Bayes* es más práctico porque no requiere un conjunto de entrenamiento muy grande para obtener una buena estimación de la probabilidad [8], ya que mediante el

supuesto de independencia condicional solo se debe estimar la probabilidad condicional de cada X dado Y (Ver Ecuación 2.6).

$$P(Y|X) = \frac{P(Y)\prod_{i=1}^{b} P(Y|X)}{P(X)} \tag{2.6}$$

2. *Bayesian Belief Networks* (BBN): Este método nos permite especificar qué par de atributos son condicionalmente independientes , a diferencia del método *Naive Bayes* que requiere que todos los atributos sean condicionalmente independientes. Una (BBN) proporciona una representación gráfica de las relaciones probabilísticas entre un conjunto de variables aleatorias (Ver Figura 2.1)

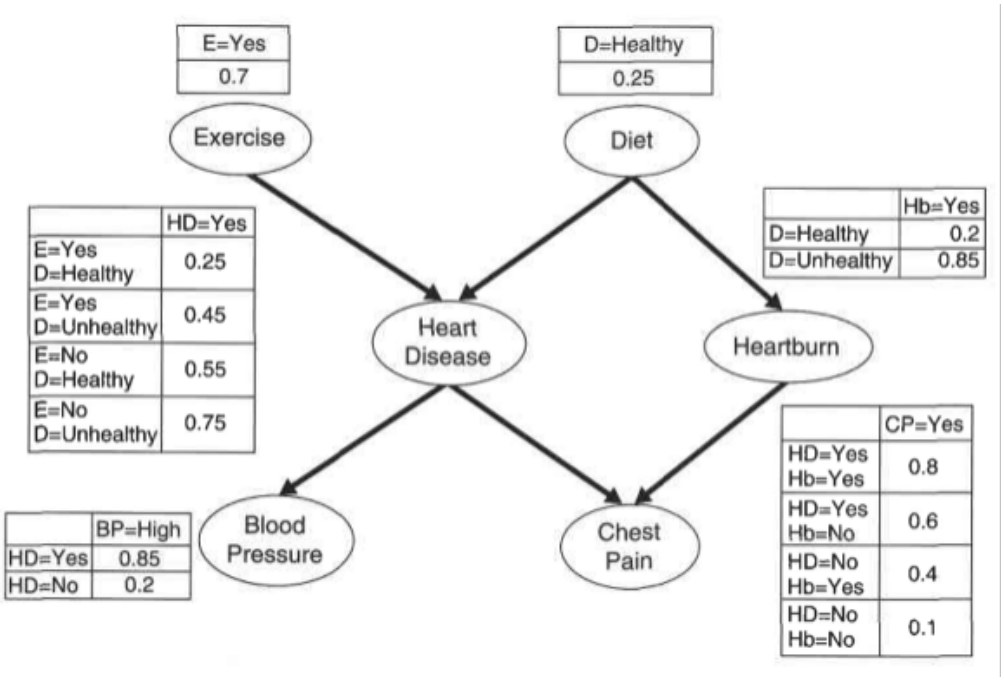


Figura 2.1: BBN para detectar acidez estomacal y problemas cardiacos. Tomado de [8]

En la figura se observan dos componentes importantes para la construcción de una (BBN), el primero una tabla que asocia cada nodo a sus padres inmediatos mediante una probabilidad, el segundo un grafo que represente las relaciones de cada nodo.

- *Support Vector Machines*: Las maquinas de vectores de soporte (SVM) se utilizan para dar solución a los problemas de regresión o de clasificación, también tienen una alta ventaja al tratar con datos de alta dimensionalidad.

La maquina de vectores de soporte busca el hiperplano que tenga la máxima distancia (margen) que separe de forma óptima los puntos de una clase de la de otra. De esta forma, los

puntos del vector que son etiquetados con una categoría estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado [8] (Ver Figura 2.2).

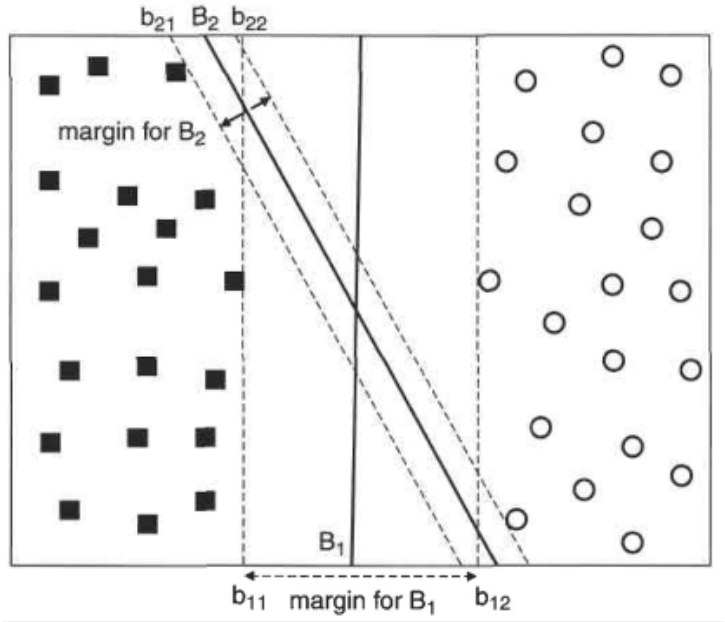


Figura 2.2: SVM máximo margen. Tomado de [8]

En la figura 2.2 se puede observar que tanto el hiperplano B1 como B2 separan las clases, pero B1 presenta un mayor margen de distancia, lo cual es lo ideal para lograr una buena diferenciación entre clases y evitar errores. En el caso de B2 sucede lo opuesto, ya que el margen de B2 menor que el de B1 dando posibilidad de que se presente una mayor cantidad de errores diferenciando entre las clases.

- *Random forests*: El *random forest* es un método que combina las diferentes predicciones realizadas por múltiples árboles de decisión, generados a partir de vectores aleatorios de un conjunto de datos [8].

  Random forest hace uso del concepto de *bagging* para generar los conjuntos de entrenamiento de cada árbol, el *bagging* consiste en seleccionar varias muestras aleatorias de datos para generar múltiples subconjuntos, la selección se realiza con reemplazo, es decir que un dato tiene la posibilidad de aparecer dentro de un subconjunto varias veces [14].

  La construcción del *random forest* se da mediante múltiples vectores compuestos de una selección aleatoria de k atributos de los m atributos totales del conjunto de entrenamiento, con los cuales se crean los n árboles diferentes. Los datos que no se seleccionaron para pertenecer al

conjunto de entrenamiento conforman un conjunto denominado *Out of bag data (OOB data)*, con el cual se calculará el error de clasificación del modelo *random forest*. Una vez que se han construido los árboles, las clasificaciones se combinan utilizando un esquema de votación por mayoría, es decir la clase que obtenga la mayor cantidad de respuestas por parte de los árboles es donde se clasificará al individuo [15]. Un ejemplo más claro de la construcción de un *random forest* se puede observar en la Figura 2.3.

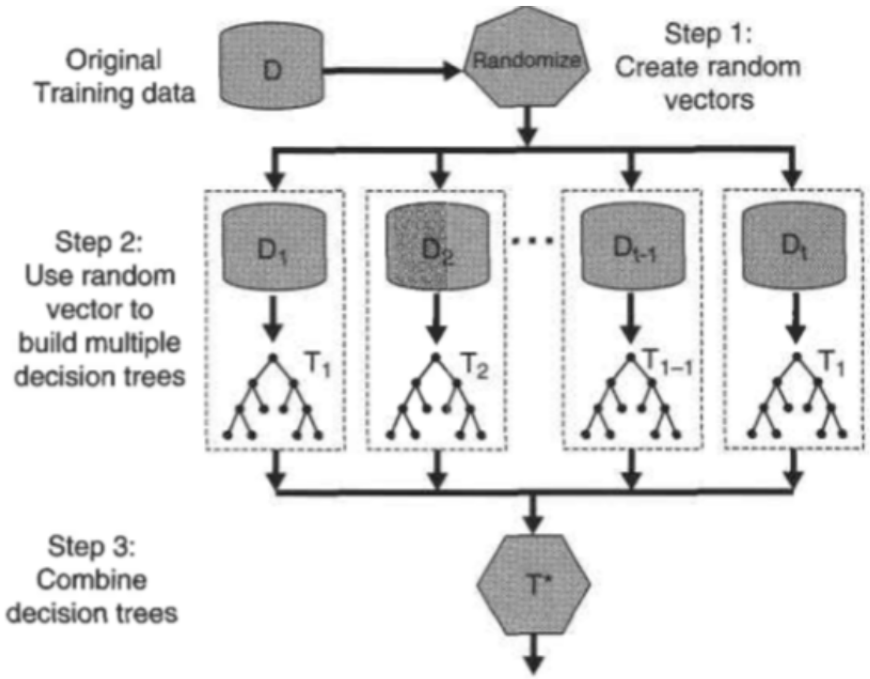


Figura 2.3: *Random forest*. Tomado de [8]

- *Convolutional neural network* (CNN): Una CNN es una técnica de aprendizaje profundo que pertenece a la familia de las redes neuronales artificiales. Las CNN se utilizan generalmente para procesar datos que están representados en múltiples matrices y multidimensionales, como por ejemplo: las imágenes que están compuestas de matrices 2D, los vídeos representados por matrices 3D y el lenguaje como matrices 1D (Ver Figura 2.4) [16].

  Normalmente para la construcción de una CNN se utilizan diferentes tipos de capas (Ver Figura 2.4). En la Figura 2.4 se pueden observar los diferentes tipos de capas que se utilizan para la construcción de una CNN, las cuales son:

  1. Capas convolucionales: Las funciones principales de este tipo de capas es aplicar la opera-

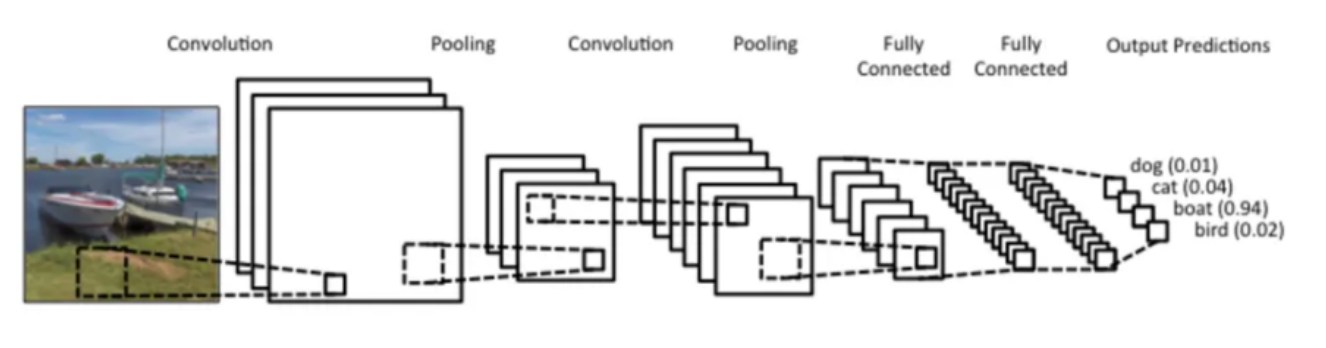


Figura 2.4: Arquitectura de una CNN, Tomado de [17]

ción llamada "Convolución", la cual es la encargada de extraer las características óptimas que representen los datos y detectar conjunciones locales de las características provenientes de la capa anterior [18].

2. Capas de reducción o *pooling*: se encargan de reducir las dimensiones de las características que provienen de la capa convolucional o de otra capa de reducción, para realizar la reducción generalmente se usan operaciones como el *averagepooling*, esta operación aplica un promedio sobre una región de la matriz de datos para generar una reducción a lo alto y a lo ancho de la matriz original; También el *MaxPooling*, consiste en tomar una región de la matriz original y seleccionar el valor más alto [18].
3. Capa clasificadora totalmente conectada: Generalmente esta capa se encuentra al final de la red en la etapa de clasificación y es la encargada de tomar todas las neuronas de la capa anterior para luego conectarlas con cada una de las neuronas de la capa actual para generar la clasificación la cual es entregada por una ultima capa de salida [18].

Por otro lado, para procesar un conjunto de datos que posee una estructura unidimensional o que contiene datos vectorizados, como lo es el lenguaje escrito, se utiliza una red neuronal convolucional unidimensional (1D CNN), la cual difiere de las CNN 2D y 3D ya que los datos cuentan con una dimensionalidad menor y solo fluyen en una dirección [19].

2.2.4. Tareas del aprendizaje automático

Para poder categorizar los métodos de aprendizaje automático, se debe definir el problema que se espera resolver para así identificar la tarea de aprendizaje automático adecuada que ayude a solucionarlo. Las principales tareas son [20]:

- Clasificación (aprendizaje supervisado): se utiliza para poder determinar a qué clase pertenece un conjunto de instancias etiquetadas; la clasificación tiene dos subclasificaciones las cuales son:

  - Clasificación binaria: a partir de un umbral de puntuación y el conjunto de datos etiquetado con los valores de 0 para los datos que no sobrepasan el umbral o 1 para los datos que sobrepasen el umbral, lo cual permite generar un clasificador, que puede ser usado para predecir la clase de las nuevas instancias que aún no están etiquetadas.
  - Clasificación multiclase: esta clasificación no tiene que elegir un umbral de puntuación y mediante un conjunto de datos etiquetados busca obtener un clasificador para predecir la clase (categoría) de las nuevas instancias de los datos.

- Regresión (aprendizaje supervisado): busca predecir valores continuos o discretos basándose en un conjunto de atributos con características que se encuentran relacionadas a este. Por ejemplo, la estimación del precio de venta de un articulo basados en las características que tiene, en el caso de un automóvil el modelo, la capacidad de pasajeros, etc.

- Agrupación (aprendizaje no supervisado): esta tarea se utiliza para agrupar elementos de instancias en grupos que contengan características similares.

2.2.5. Técnicas de validación de modelos

Son técnicas utilizadas para validar el entrenamiento de un modelo que ha sido entrenado. A continuación se presentan dos de las técnicas mas comunes [21]:

- *Holdout*: Generalmente se toma el conjunto de dato y se divide en tres subconjuntos, el conjunto de entrenamiento, validación y pruebas. El conjunto de validación se usa para validar la correctitud del modelo construido en base al conjunto de entrenamiento.

- *Cross-Validation*: En esta técnica se elige un k el cual es la cantidad de subconjuntos en los que se divide el conjunto de datos. Uno de los K conjuntos es utilizado para la validación y el resto k-1 conjuntos, como los datos de entrenamiento.Este proceso se repite k-veces con un conjunto diferente cada vez.

2.2.6. Métricas

Las métricas son la manera de evaluar y comparar los modelos entrenados mediante algoritmos de aprendizaje automático. Por consiguiente, es necesario analizar qué métricas se utilizarán en cada proceso, ya que el rendimiento de un algoritmo puede ser deficiente en una métrica pero sobresaliente en otra. Debido a esto, se debe realizar un análisis para poder identificar qué métricas se deben utilizar para medir el desempeño de un modelo específico. Ahora bien, las métricas más comunes son [22]:

- Matriz de confusión: una matriz de confusión es una tabla en la cual se genera una comparacion entre los resultados esperados (actuales) y la prediccion del algoritmo. Un ejemplo de una matriz de confusión binaria consta de los siguientes elementos: (Ver Figura 2.5)

  - TP = *True positive*: hace referencia a los datos que debían ser clasificados de manera positiva y el algoritmo los predijo de esta manera.

- FN = *False negative*: hace referencia a los datos que debían ser clasificados de manera positiva pero el algoritmo los clasificó de manera negativa.
- FP = *False positive*: hace referencia a los datos que debían ser clasificados de manera negativa pero el algoritmo los clasificó de manera positiva.
- TN = *True negative*: hace referencia a los datos que debían ser clasificados de manera negativa y el algoritmo los clasificó de esta manera.

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

Figura 2.5: Matriz de confusión. Tomado de [22]

- *Accuracy*: se puede definir como la cantidad de datos correctamente clasificados dividido por total de datos. Esta medida nos muestra con que exactitud el clasificador esta realizando las predicciones de las clases. No obstante, para datos desequilibrados la exactitud puede ser muy alta y no haber clasificado bien uno de los conjuntos de elementos.

- *F1 Score*: es utilizado para determinar el desempeño del modelo, indicando que tan preciso es el modelo mediante un valor que se miden entre 0 y 1, siendo 0 el peor caso y 1 el mejor. Busca realizar un promedio de *Precision* y *Recall*, ya que un *Recall* alto y *Precision* baja significa un gran número de resultados con un porcentaje bajo de predicciones correctas. Por el contrario, un *Recall* bajo y una alta *Precision* da a entender que se obtuvo pocos resultados pero un alto porcentaje de precisión. Se define como $2 * \frac{PrecisionRecall}{Precision+Recall}$.

- *Mean Absolute Error*: el error promedio de la diferencia entre los valores originales y los valores predichos.

- *Mean Squared Error*: toma el error promedio de la diferencia al cuadrado entre los valores originales y los valores predichos.

- *Sensitivity (Recall)*: es utilizado para clasificaciones binarias. Se define como $\frac{TP}{TP+FN}$. Con esta métrica podemos observar de todos los datos que realmente eran positivos y cuántos se clasificaron correctamente como positivos.

- *Precision* : este es utilizado para clasificaciones binarias. Se define como $\frac{TP}{TP+FP}$. Con esta métrica podemos observar todos los datos de la predicción que fueron clasificados como positivos y nos dice cuántos realmente eran positivos. Este, a comparación de *Sensitivity*, se basa en las predicciones y no en las clasificaciones reales.

2.2.7. Trabajos Relacionados

En las redes sociales existen "cuentas falsas", las cuales son cuentas administradas por bots que buscan difundir cierto tipo de información que en muchos casos puede ser información errónea que afecta la imagen de los usuarios, también manipulan a los usuarios para acceder a sitios maliciosos. Debido a lo anterior y a que las redes sociales poseen un alto impacto en la toma de decisiones de los usuarios en su vida cotidiana, los bots logran obtener el potencial de influir drásticamente en los eventos de la sociedad, por lo cual la presencia de bots en las redes sociales se convierte en una problemática social. Basado en lo anterior es importante dar solución a la problemática de las "cuentas falsas" en las redes sociales, para evitar que eventos de la vida cotidiana de un usuario sean manipulados por bots. La comunidad tecnológica ha propuesto diferentes enfoques que permiten abordar una solución para poder identificar "cuentas falsas" en la redes sociales tales como:

Trabajos que usan análisis estadístico:

- *"Detecting and analyzing automated activity on twitter"* [23].

  Tiene como objetivo presentar una técnica para determinar si una cuenta de Twitter emplea bots para generar publicaciones o no, mediante el análisis de la periodicidad de tiempo con la que la cuenta genera publicaciones. Bajo el supuesto de que las cuentas automatizadas muestran ciertos patrones de tiempo en las publicaciones, los cuales no se manifiestan en publicaciones realizadas por cuentas de usuarios humanos. Por lo tanto, una secuencia de los tiempos de cada publicación debe parecer extraída aleatoriamente de una distribución uniforme. Los tweets generados por humanos, aunque no son completamente uniformes, carecen de agrupaciones o patrones notables. Las cuentas automatizadas, por otro lado, pueden presentar distribuciones de tiempo que llevan a una no uniformidad detectable debido a que los scripts publican tweets programados en cada periodo de tiempo. Entonces, se encontrarán los tiempos de tweets agrupados en esos minutos programados.

  Utilizaron la prueba de Pearson $(x)^2$ para evaluar si un conjunto de tiempos de publicaciones es consistente con las distribuciones de segundos y minutos esperadas de los usuarios humanos, la cual retorna la probabilidad de que se produzca una distribución de los tiempos de cada publicación. Una probabilidad baja, indica que la cuenta tiene un comportamiento en el que la elección de los segundos o minutos para publicar un mensaje no presenta patrones, lo que indica que eran publicaciones generadas por humanos; así mismo, si la probabilidad es demasiado alta, sugiere que la cuenta está utilizando un mecanismo de automatización.

En conclusión, encontraron que el 16 % de las 19.436 cuentas que probaron muestran un comportamiento altamente automatizado. También, encontraron que de las cuentas verificadas(cuentas de personas famosas como: cantantes, actores, celebridades, etc.), las más seguidas y los seguidores de estas mismas, tienen tasas de automatización más bajas que las cuentas públicas, es decir cuentas de personas no famosas, (6.9 %, 12 % y 4.2 %, respectivamente)

- *"Temporal Patterns in bot activities"* [4].

  En este trabajo se realizó minería de patrones temporales en la serie de actividades de los bots en twitter, tales como descubrimiento de motivos (comportamiento repetitivo), discordias (comportamiento anómalo), unión de series temporales (correlación de dos series de tiempo similares) y clusters dinámicos (grupos de bots que cambian su patrón de actividad). Para mostrar la diferencia entre cuentas administradas por bots y cuentas reales los autores plantearon cuatro indicadores de alto riesgo:

  1. "El número de actividades por usuario en dos horas, esta es una característica genérica que se centra en las actividades generales. Los bots suelen ser muy activos.
  2. El número de eliminaciones de tweets por usuario en dos horas indica si el usuario mantiene o no un perfil bajo en el número acumulado de tweets para evitar parecer un bot. Similar a las actividades generales, los robots eliminan los tweets con más frecuencia que las cuentas benignas.
  3. El porcentaje de tweets que contienen URLs indica qué fracción del contenido de los tweets está potencialmente fuera de Twitter.
  4. El porcentaje de los tweets duplicados indica la fracción de los tweets que es generada por el usuario automáticamente. Consideramos todos los tweets con texto idéntico como duplicados. Este conjunto incluye los retweets por definición. Las fuentes originales de estos contenidos duplicados suelen ser celebridades, políticos, deportistas y noticias."

  Los autores concluyeron que los aspectos temporales de los bots han sido ignorados en gran medida y descubrieron que estos aspectos tienen un gran impacto en la detección de bots, ya que mediante estos aspectos obtuvieron una precisión alta de acuerdo a los indicadores que plantearon durante la ejecución de la investigación.

Trabajos que usan aprendizaje automático:

- "Classification of Twitter Accounts into Automated Agents and Human Users"[24].

  Z. Gilani, E. Kochmar, and J. Crowcroft capacitaron a un agente para clasificar las cuentas de Twitter en usuarios "bots" o "humanos". Para lograr mejorar la precisión de la clasificación, describieron una metodología sistemática que consisten en:

1. Recolectar entre 2.5 y 3 millones de tweets de diferentes cuentas, la recolección se dio mediante la plataforma Stweeler.
2. Dividir los datos recolectados en cuatro subconjuntos de cuentas seleccionadas aleatoriamente basados en la popularidad, es decir la cantidad de seguidores. Los subconjuntos quedaron conformados de la siguiente forma: el primero de 50 cuentas con más de 9M de seguidores y 150,336 de tweets, el segundo incluye 746 cuentas entre 900K y 1.1M de seguidores y 303,517 de tweets, el tercero consta de 1,447 cuentas entre 90K y 110K de seguidores y 230,577 de tweets, y el último de 1.293 cuentas entre 0.9K y 1.1K de seguidores y 37,679 de tweets.
3. Realizar el proceso de etiquetado, el cual fue de manera manual por medio de científicos informáticos capacitados. En este proceso se descartaron las cuentas no disponibles o cuentas bloqueadas, también usaron el coeficiente kappa de Cohen para mantener la fiabilidad del proceso de etiquetado.
4. Extraer 15 características relevantes de los tweets, las cuales son la edad de la cuenta, "me gusta" de todos los tweets, listas por usuario, la proporción de seguidores y amigos, los "me gusta" del usuario, "me gusta"/favoritos por tweet, retweets por tweet, respuestas de los usuarios, tweets del usuario, retweets del usuarios, frecuencia de tweets, las URLs, tipo de fuente de actividad (navegador, aplicaciones móviles, aplicaciones de medios sociales, automatización, marketing, noticias, otros.), retweets de fuentes ,tamaño del contenido del CDN.
5. Aplicar un clasificador random forest para realizar tres experimentos en los que aplicaron cross validation con k = 5. Los resultados de estos experimentos tienen un accuracy del 86,44 % y un F1 del 83 %.

- "Detecting Automation of Twitter Accounts: Are You a Human, Bot, or Cyborg?"[25].

  Los autores realizaron un estudio de las características relacionadas con el comportamiento de los tweets y su contenido, también estudiaron las propiedades de las cuentas para detectar si eran humanas, de bots o de cyborgs (es decir, cuentas administradas por un humano mediante el uso de programas automatizados). Obtuvieron un conjunto de datos de aproximadamente 500.000 cuentas mediante el API de twitter, de las cuales tomaron muestras de forma aleatoria y las etiquetaron manualmente para formar un conjunto de entrenamiento con 6.000 cuentas y 8,350,095 tweets clasificadas en, 2.000 cuentas de humanos, 2.000 de bots y 2.000 cuentas de cyborgs. Propusieron un sistema de clasificación que incorpora 3 componentes:

  1. "El componente de entropía usa el intervalo de tweeteo como una medida de la complejidad del comportamiento y detecta el tiempo periódico y regular, el cual es un indicador de automatización.
  2. El componente de detección de spam utiliza contenido de tweets para verificar si los patrones de texto contienen spam o no.

3. El componente de propiedades de la cuenta emplea propiedades útiles de la misma, como la configuración del dispositivo de tweeteo, la relación de URL, para detectar desviaciones de lo normal."

El sistema de clasificación también plantea un tomador de decisiones, el cual aplica un algoritmo random forest para determinar a qué clase pertenece una cuenta, ya sea humano, bot o cyborg. Para tomar una decisión, el clasificador se basa en las características generadas por los tres componentes nombrados anteriormente. Aplicaron cross validation con 10 subconjuntos para entrenar y probar el clasificador sobre el conjunto etiquetado previamente. Por último, emplearon una matriz de confusión para medir el rendimiento del sistema, el cual logró una precisión promedio del 96 %.

- *The Paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race*[26].

En este trabajo los autores realizaron un estudio exhaustivo frente a la nueva generación de spambots sociales en Twitter. Compararon varias técnicas de clasificación de bots presentadas en la literatura académica para evaluar el desempeño de los modelos frente a la clasificación de los nuevos spambots sociales. La descripción detallada de los conjuntos de datos utilizados en este estudio se puede observar en la siguiente figura.

		statistics			
dataset	**description**	accounts	tweets	year	**used in section**
`genuine accounts`	verified accounts that are human-operated	3,474	8,377,522	2011	3.1, 3.2
`social spambots #1`	retweeters of an Italian political candidate	991	1,610,176	2012	3.1, 3.2
`social spambots #2`	spammers of paid apps for mobile devices	3,457	428,542	2014	3.1, 3.2
`social spambots #3`	spammers of products on sale at *Amazon.com*	464	1,418,626	2011	3.1, 3.2
`traditional spambots #1`	training set of spammers used by Yang *et al.* in [44]	1,000	145,094	2009	3.1
`traditional spambots #2`	spammers of scam URLs	100	74,957	2014	3.1
`traditional spambots #3`	automated accounts spamming job offers	433	5,794,931	2013	3.2
`traditional spambots #4`	another group of automated accounts spamming job offers	1,128	133,311	2009	3.2
`fake followers`	simple accounts that inflate the number of followers of another account	3,351	196,027	2012	3.1
`test set #1`	mixed set of 50% `genuine accounts` + 50% `social spambots #1`	1,982	4,061,598	–	4, 5
`test set #2`	mixed set of 50% `genuine accounts` + 50% `social spambots #3`	928	2,628,181	–	4, 5

Figura 2.6: Estadísticas sobre los conjuntos de datos utilizados para este estudio. Tomado de [26]

El conjunto *test set 2* lo utilizaron para replicar modelos presentados en la literatura académica. Algunos de los modelos implementados en este estudio son:

- *BotOrNot: A System to Evaluate Social Bots*[27]. El sistema de clasificación de BotOrNot es un modelo que genera más de 1.000 atributos mediante el uso de los metadatos disponibles e información extraída de patrones de interacción y contenido. Los 1.000 atributos se dividen en clases como: red, usuario, amigos, temporal, contenido, sentimiento.

- *Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers*[28].

  En este trabajo presentan una serie de características nuevas la cuales las agruparon en las siguientes clases: vecinos, automatización, temporal.

Los resultados que obtuvieron mediante la implementación de los modelos anteriores son: 0.635 *Precision*, 0.950 *Recall* y 0.981 *Specificity* para *BotOrNot*; En el caso de *Evolving Twitter Spammers*. Obtuvieron 0.727 *Precision*, 0.409 *Recall*, 0.848 *Specificity*.

Los autores concluyeron que la variación en los resultados obtenidos es debido a que la clasificación manual de las cuentas no es suficiente para garantizar una buena clasificación de los nuevos bots catalogados como spambots.

Capítulo 3

Metodología

En este capitulo se describe todo el proceso de construcción de los modelos y el tratamiento de los datos, este proceso se realizo previo a la clasificación.

3.1. Datos

En esta sección se describe la elección, el manejo y el tratamiento de los conjuntos de datos.

3.1.1. Conjunto Original

Para este trabajo se tomó parte del conjunto de datos de una investigación previa de los autores S.Cresci, A.Spognardi et al. en [26]. La investigación cuenta con un conjunto de datos separados en varios archivos; uno de usuarios de cuentas reales y los demás, de cuentas gestionadas por bots:

1. Cuentas genuinas, este archivo contiene todas las cuentas y los tweets verificados que son de usuarios humanos. El proceso de verificación se llevó a cabo seleccionando de manera aleatoria usuarios, a quienes se contactó para hacerles una pregunta sencilla. Todas las respuestas fueron verificadas de forma manual para certificarlos como usuarios humanos.

2. Spambots sociales #1, incluye cuentas que hicieron retweet de los tweets de un candidato a la alcaldía de Roma en 2014, estas cuentas fueron catalogadas manualmente por los investigadores quienes, certificaron que pertenecen a cuentas automatizadas.

3. Spambot sociales #2, las cuentas de este archivo promocionan el #TALNTS. Talnts es una aplicación móvil para contactar artistas que trabajan en los campos de la escritura, la fotografía digital, la música, etc. La clasificación de estas cuentas se hizo de forma manual por parte de los investigadores, quienes certificaron que son cuentas automatizadas.

4. Spambot sociales #3, este archivo contiene cuentas que generan spam mediante la publicación de tweets con URLS anunciando productos a la venta en Amazon.com. De igual forma que los archivos anteriores, el proceso de clasificación fue de manera manual.

5. Conjunto de prueba #1, Este archivo fue generado a partir de la selección aleatoria del 50 % de cuentas genuinas y el 50 % de spambots sociales # 1.

6. Conjunto de prueba #2, Este archivo fue generado a partir de la selección aleatoria del 50 % de cuentas genuinas y el 50 % de spambots sociales # 3.

Una descripción más detallada del número de cuentas y de tweets por cada archivo se puede ver en el Cuadro 3.1.

Conjunto de datos	Descripción	Cuentas	Tweets	Año
Cuentas genuinas	Cuentas verificadas que son operadas por humanos	3.474	8,377.522	2011
Spambots sociales #1	Retweeters de un candidato político italiano	991	1,610.176	2012
Spambots sociales #2	Spams de aplicaciones pagas para dispositivos móviles	3.457	428.542	2014
Spambots sociales #3	Spams de productos a la venta en Amazon.com	464	1,418.626	2011
Conjunto prueba #1	50 % de cuentas genuinas + 50 % de spambots sociales #1	1.982	4,061.598	-
Conjunto prueba #2	50 % de cuentas genuinas + 50 % de spambots sociales #3	928	2,628.181	-

Cuadro 3.1: Conjunto de datos. Tomado y adaptado de [26]

3.1.2. Conjuntos Generados

Los conjuntos de datos usados en este trabajo se generaron a partir de los conjuntos mencionados en la sección anterior. Estos conjuntos de datos están compuestos de la siguiente manera:

- Primer Conjunto: Se tomó la decisión de mezclar los archivos cuentas genuinas, spambot sociales #2 y spambot sociales #3, presentados por S.Cresci, A.Spognardi et al. Esta decisión se tomó ya que estos archivos tienen los tweets en inglés y los demás no. Para la construcción del conjunto 1 se realizó una limpieza de datos y un balanceo de clases,descritos en la sección 3.2.1.

  Después del proceso de limpieza y balanceo el conjunto 1 quedó conformado por un total 2,571.244 de tweets distribuidos en 1,285.622 tweets humanos, 1,285.622 tweets que son bots.

  Por otro lado para el primer conjunto se decidió tener en cuenta el texto del tweet y los atributos obtenidos de las cuentas, por ejemplo: el número de seguidores, el número de amigos, el número de me gusta por tweet, el número de urls por tweet, el número de hashtag por tweet, el número de menciones por tweet.

- Segundo Conjunto: Para el conjunto 2 se mezclaron los mismos archivos que en el conjunto 1, además se realizó el mismo proceso de limpieza y balanceo de datos descritos en la sección 3.2.1, con la diferencia de que no se tomó en cuenta los atributos obtenidos de las cuentas, solo el texto de los tweets. Al igual que el primer conjunto, el conjunto 2 quedó conformado por un total 2,571.244 de tweets distribuidos en 1,285.622 tweets humanos, 1,285.622 tweets que son bots.

La diferencia entre los dos conjuntos descritos previamente es el uso de los metadatos para el conjunto 1 y en el conjunto 2 no, esto es debido a que para este trabajo se planteo implementar una técnica que sacara el mayor provecho del texto solamente, y compararla con una técnica que usara texto y metadatos.

Por otro lado, al ejecutar este proyecto se presento una limitación en el procesamiento de la máquina, debido al tamaño de los conjuntos de datos. Se inicio con una maquina de 8 GB de RAM y se paso a otra con 16 GB de RAM, pero la limitación seguía presente, por ende, se tomó la decisión de reducir los conjuntos de datos (Ver Figura 3.1 y 3.2), ya que la máquina no cuenta con la suficiente memoria RAM para ejecutar las técnicas de aprendizaje automático seleccionadas para este proyecto. La selección de los nuevos conjuntos se realizó de manera aleatoria y respetando el balance entre clases. Los conjuntos de datos quedaron conformados de la siguiente manera:

- Primer conjunto: Un total 1,000.000 de tweets distribuidos en 500.000 tweets humanos, 500.000 tweets que son bots. Se tomó en cuenta el texto de los tweets y los metadatos.
- Segundo conjunto: Un total 1,000.000 de tweets distribuidos en 500.000 tweets humanos, 500.000 tweets que son bots. Se tomó en cuenta sólo el texto de los tweets.

3.1.2.1. Conjuntos de entrenamiento y pruebas

Los conjuntos de entrenamientos y pruebas para el conjunto 1 se separaron en un 80 % y 20 % respectivamente (Ver Figura 3.1). La selección del 20 % para el conjunto de pruebas se realizo de manera aleatoria manteniendo la proporción de balanceo. Obteniendo así un conjunto de entrenamiento de 800.000 tweets, 400.000 humanos y 400.000 bots. Y un conjunto de pruebas de 200.000 compuesto de 100.000 humanos y 100.000 bots.

El segundo conjunto se separó en conjunto de entrenamiento 70 %, pruebas 20 % y validación 10 % (Ver Figura 3.2). La separación del conjunto se realizó de manera aleatoria manteniendo la proporción de balanceo. Los conjuntos obtenidos quedaron conformados de la siguiente manera:

- Conjunto de entrenamiento, el cual quedó con 700.000 tweets en total, distribuidos en 350.000 humanos y 350.000 bots.
- Conjunto de pruebas, quedó conformado por un total de 200.000 tweets, divididos en 100.000 humanos y 100.000 bots.
- Conjunto de validación, este conjunto quedó compuesto de 100.000 tweets en total, lo cuales están distribuidos en 50.000 humano y 50.000 bots

3.2. Preprocesamiento

Las figuras de las matrices de confusión en esta sección pueden ser analizadas mediante la siguiente información:

- La tonalidad del color azul representa la cantidad de datos clasificados en ese cuadrante, entre más claro el color indica que la cantidad es menor y entre más oscuro significa mayor cantidad de datos.
- La etiqueta 1 es la clase que representa los tweets clasificados como bots y 0 son los tweets clasificados como humanos.

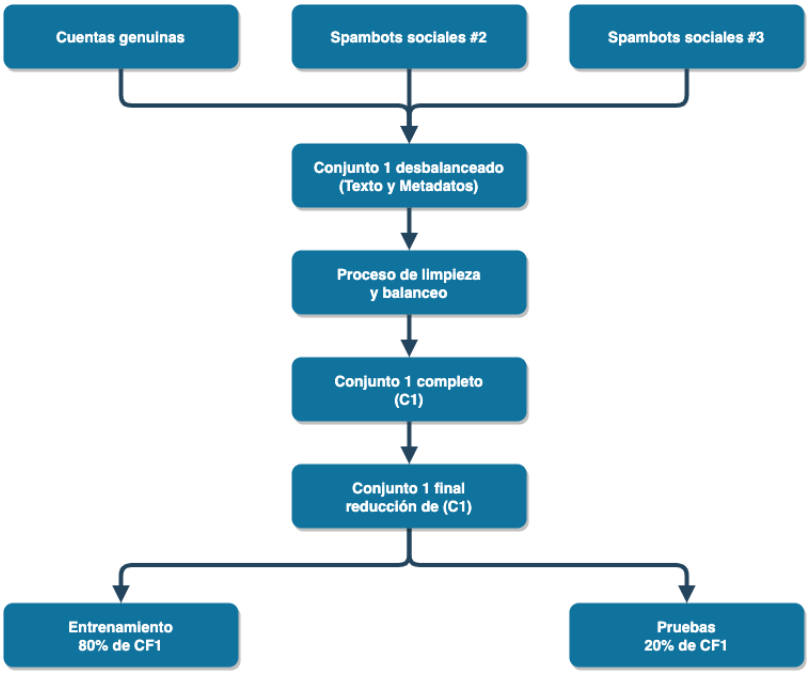


Figura 3.1: Generación del primer conjunto de datos

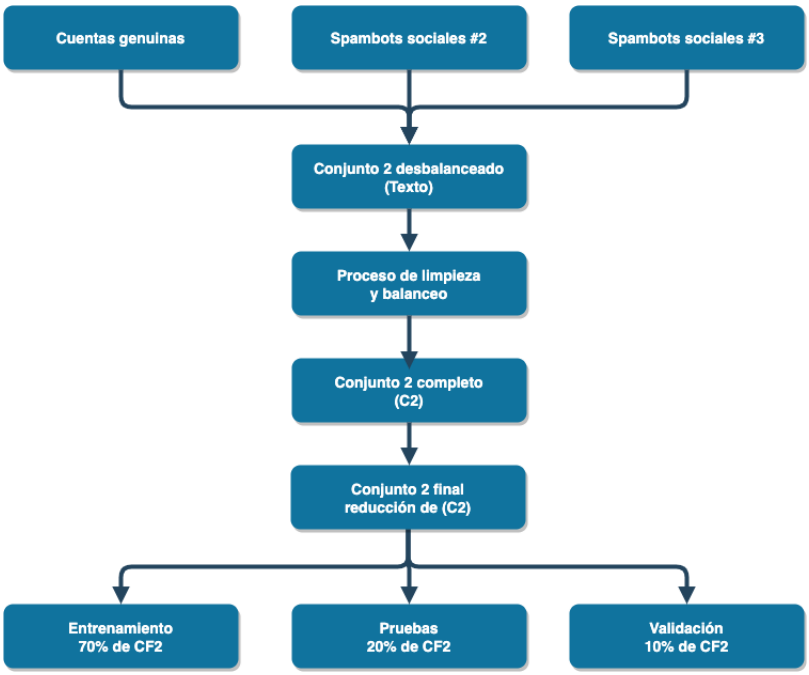


Figura 3.2: Generación del segundo conjunto de datos

3.2.1. Limpieza y balanceo de datos

En el proceso de limpieza de los datos se tuvieron en cuenta las sugerencias encontradas en Towards Data Science [29], donde presentan múltiples conocimientos para la comprensión de la ciencia de datos.

Los pasos del proceso de limpieza descritos a continuación se realizaron para el primer conjunto.

1. Iniciando se tenían 44 atributos (Ver Cuadro 3.2), de los cuales se eliminaron atributos en los que sus valores eran nulos y/o la cantidad de nulos fuera superior al 30 %. Al final se obtuvieron 16 atributos (Ver Cuadro 3.3).

2. Se verificaron los tipos de datos de cada atributo y en caso de estar erróneo se hizo la corrección mediante el Cast respectivo, los atributos con un tipo de dato erroneo fueron: followers_count, created_at_x y created_at_y.

3. Se eliminaron registros que tuvieran tres o más atributos nulos.

Los pasos a continuación se aplicaron para ambos conjuntos.

1. Reemplazar URLS, Hashtags y menciones a usuarios por las palabras "hashtag", "url" y "name" respectivamente, debido a que al ser partes del texto que se repiten con poca frecuencia y además son extensas, pueden generar una mayor cantidad de atributos que no aportarían valor en la clasificación.

2. El balanceo de la clase objetivo se realizó mediante funciones de la librería de Pandas, conservando un 50 % bots y un 50 % humanos.

3. Se realizó la eliminación de *StopWords*, las *StopWords* son aquellas palabras que no tienen significado propio, tales como; artículos,pronombres y preposiciones.

4. Se eliminaron los emojis, ya que uno de los objetivos planteados en este trabajo de grado es analizar el texto del tweet y no los emojis.

5. Se eliminaron los signos de puntuación, ya que en este trabajo se buscaba analizar la ganancia a partir del texto o partes que componen el texto, es decir, las palabras para la clasificación de bots.

3.2.2. Extracción de características

La extracción de características tiene como objetivo reducir la cantidad de características en un conjunto de datos creando nuevas características a partir de las existentes [30]. Las siguientes técnicas se aplicaron al atributo "text" :

id	text	source
name	screen_name	in_reply_to_screen_name
truncated	in_reply_to_status_id	in_reply_to_user_id
geo	place	contributors
retweeted_status_id	retweet_count	reply_count
statuses_count	followers_count	friends_count
favourites_count	listed_count	notifications
geo_enabled	profile_image_url	profile_banner_url
profile_text_color	created_at_x	profile_link_color
description	age	following
num_hashtags	num_urls	num_mentions
created_at_y	protected	location
favourite_count	follow_request_sent	possibly_sensitive
default_profile	default_profile_image	utc_offset
geo_enabled	test	—

Cuadro 3.2: Atributos Originales

Atributo	**Descripción**
statuses_count	El número de tweets generados por la cuenta
followers_count	El número de seguidores que tiene esta cuenta actualmente
friends_count	El número de usuarios que la cuenta sigue
favourites_count	La cantidad de tweets que le han gustado a este usuario durante la vida de la cuenta
favourite_count	La cantidad de likes que tiene el tweet
listed_count	El número de listas públicas de las que es miembro este usuario
description	La descripción del perfil del usuario
num_hashtags	Número de hashtag presentes en el tweet
num_urls	Número de urls presentes en el tweet
num_mentions	Número de menciones a otros usuarios en el tweet
retweet_count	Número de veces que se ha hecho retweet a este tweet
text	El texto escrito en el tweet
created_at_x	La fecha y hora cuando se creó este tweet
created_at_y	La fecha y hora en que se creó la cuenta en Twitter
age	Número de dias desde que se abrio la cuenta hasta que publicaron este tweet
test	Etiqueta si es humano o bot

Cuadro 3.3: Descripción de atributos finales

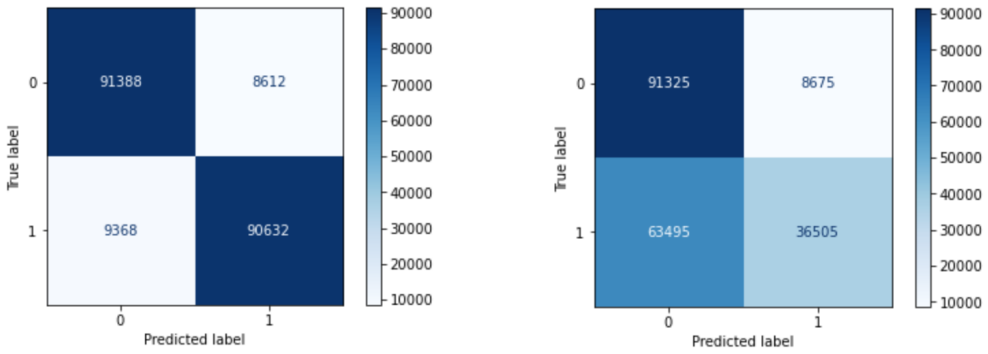
- Stemming: Es el proceso de reducir una palabra que tiene prefijos o sufijos a la raíz de la palabra. Con este proceso se busca reducir la cantidad de palabras únicas a una raíz en común y aumentar la frecuencia de aparición de dicha raíz en el documento. Por ejemplo: “Likes”, “liked”, “likely” se reducen a la raíz “like”.

  Para realizar este proceso se utilizó el algoritmo de Stemmer de Porter [31] mediante la librería de NLTK de Python.

- N-gramas: Los N-gramas son un conjunto de n palabras consecutivas presentes en un documento de texto. Los n-gramas son utilizados para ver una secuencias de n palabras que ocurren en el texto siendo útiles en aplicaciones de análisis de sentimiento, clasificación de texto y generación de texto [32]. Se realizó una extracción de unigramas y bigramas para los conjuntos de entrenamiento y pruebas.
  En el presente trabajo se utilizaron unigramas, después de comparar los resultados de entrenar un árbol de decisión estándar con 1000 unigramas y 1000 bigramas (Ver Figura 3.3). Basados en los resultados de esta prueba (Ver Cuadro 3.4) se opta por trabajar con los unigramas, ya que los unigramas obtuvieron mejores resultados en 2 de las 3 métricas utilizadas para evaluar el modelo.

Datos	Precision	Recall	F1_score
Unigramas	0,9132	0,9053	0,9017
Bigramas	0,9079	0,3640	0,5328

Cuadro 3.4: Resultados árbol de decisión estándar Unigramas y Bigramas



(a) Unigramas (b) Bigramas

Figura 3.3: Matrices de confusión árbol de decisión estándar Unigramas y Bigramas

3.2.3. Selección de características

La selección de características es el proceso mediante el cual se filtran los atributos con mayor grado de relevancia para el modelo y se descartan aquellos atributos que no representan una ganan-

cia de información relevante para el modelo [30].

Los unigramas se seleccionaron mediante la medida de Chi-square con el objetivo de medir el nivel de independencia de cada unigrama frente a la clase objetivo "test". Se seleccionó un total de 1.000 unigramas, esta cantidad se seleccionó debido a la limitación del procesamiento de la máquina mencionado en la sección 3.1.2. Se utilizó la librería de Sklearn para aplicar la medida Chi-square y seleccionar los unigramas. Esta selección se realizó para el conjunto de entrenamiento y de pruebas

En la selección de los atributos diferentes de los unigramas, es decir los metadatos, se optó por realizar una matriz de correlación (Ver Figura 3.4) entre ellos para analizar el coeficiente de correlación de cada atributo frente al atributo objetivo para decidir si se conservar o se elimina. Mediante la siguiente información se puede analizar la matriz de correlación:

- Un coeficiente cercano a 0 indica que hay poca correlación entre los dos atributos, y por lo tanto esos atributos son independientes entre sí.
- Un coeficiente cercano a 1 indica una alta correlación directamente proporcional, es decir, si el valor de uno de los atributos aumenta, el valor del otro atributo aumenta también.
- Un coeficiente cercano a -1 indica una fuerte correlación inversamente proporcional, es decir, que el valor de un atributo aumenta al disminuir el del otro y viceversa.
- Cada celda presenta un color, los tonos que tienden a ser más claros o más oscuros, significa que el coeficiente de correlación es más cercano a 1 o -1 lo que indica que hay un alto grado de correlación directamente proporcional o inversamente proporcional.

De acuerdo a la matriz de correlación se optó por no tener en cuenta el atributo "age", ya que este se obtiene mediante un calculo entre los atributos "created_at_x" y "created_at_y".

3.2.4. Características para la CNN

La red neuronal convolucional (CNN) recibe como entrada una secuencia numérica, la cual es la representación de un tweet, por lo tanto el proceso de extracción y selección de características fue diferente del primer conjunto. De acuerdo a lo anterior los pasos que se siguieron para la extracción y selección fueron:

1. Aplicar el stemmer de Porter en cada palabra de los tweets.
2. Tokenizar el tweet.
3. Unir los tokens nuevamente en una frase.
4. Convertir la frase en una secuencia de números enteros.

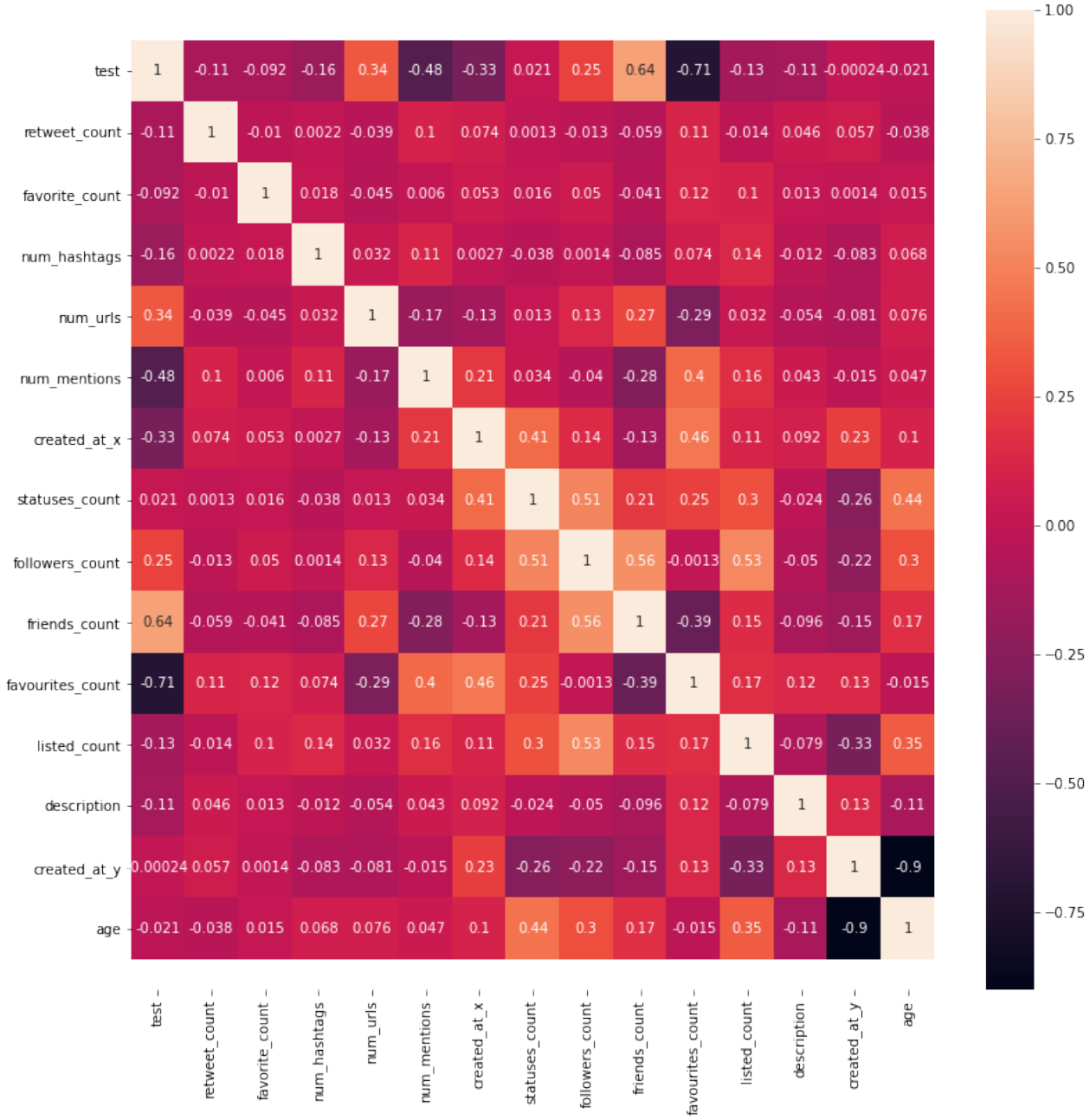


Figura 3.4: Matriz de correlación

5. Por último, se realizó un relleno de ceros al inicio de cada tweet con el fin de unificar la longitud de todas las secuencias numéricas con base en el tweet más largo, esta decisión se tomó ya que los tweets cuentan con diferentes cantidades de palabras.

3.2.5. Estimación de hiperparámetros

3.2.5.1. Árbol de Decisión

A partir de las sugerencias propuestas por Alawad et al. [33] y Mantovani et al. [34] donde indican los hiperparámetros más relevantes para generar un árbol de decisión según la literatura recopilada, se optó por explorar la profundidad del árbol, el mínimo de elementos para separar un nodo y el mínimo de elementos para convertir un nodo en hoja.

Para la construcción del árbol de decisión se utilizó la librería Sklearn la cual utiliza el algoritmo de CART (Classification and Regression Tree).

Al iniciar las pruebas, el primer hiperparámetro evaluado fue la profundidad del árbol, para el cual se realizó una búsqueda de amplio espectro donde se varió la profundidad del árbol entre 1 y 120 con el fin de analizar el comportamiento del *accuracy*. Basados en los resultados de la prueba anterior (Ver Figura 3.5), se decide realizar una búsqueda en el área específica donde se obtuvo el mejor resultado (Ver Figura 3.6). De acuerdo con las pruebas anteriores se estableció que los valores óptimos para la profundidad del árbol son 20, 30, 70 y 90.

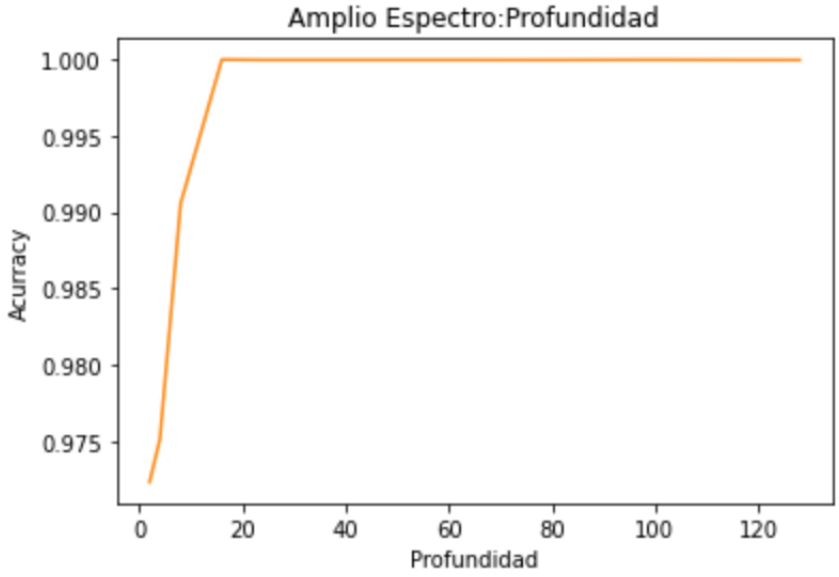


Figura 3.5: Amplio espectro Profundidad - Accuracy

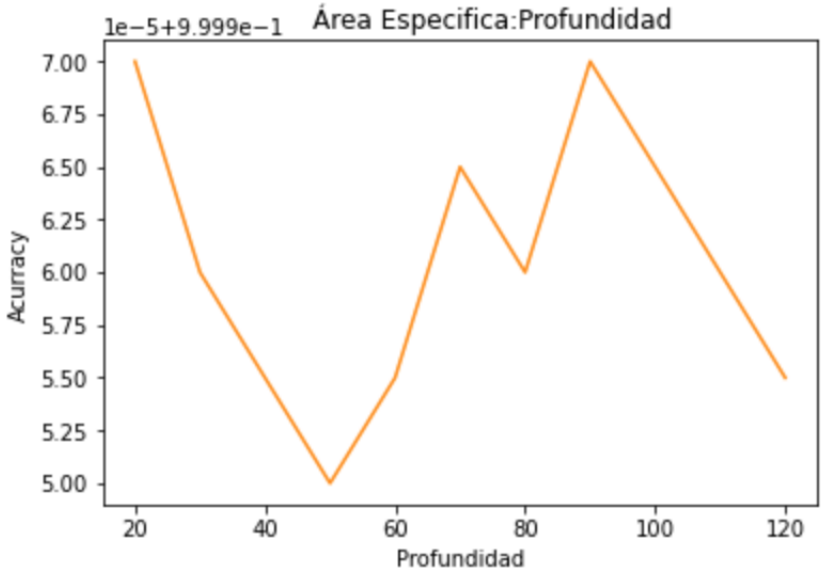


Figura 3.6: Área específica Profundidad - Accuracy

El siguiente hiperparámetro que se analizó fue la cantidad mínima de datos para convertir un nodo en una hoja del árbol. Se ejecutaron las mismas pruebas que en el caso anterior que consta de una búsqueda de amplio espectro y área específica, los resultados de las dos búsquedas se pueden observar en las gráficas 3.7 y 3.8 respectivamente. De la prueba de área específica se puede analizar que el *accuracy* no tiene una variación significativa por lo cual se opta por establecer que el rango óptimo para este hiperparámetro es entre 1 y 8.

Después se analizó el hiperparámetro de la cantidad mínima de elementos para separar un nodo interno, es decir cuántos elementos debe tener un nodo como mínimo para que un nodo pueda ser separado. Se continuó con las pruebas de búsqueda de amplio espectro y de área específica, a partir de los resultados obtenidos de estas pruebas (Ver Figura 3.9) y (Ver Figura 3.10) se observó que la cantidad mínima de elementos donde se obtuvieron los mejores resultados fueron 2, 4, 6 y 14.

Finalmente se realizó una búsqueda exhaustiva, mediante una función de Sklearn GridSearchCV, la cual entrena el modelo con todas las posibles combinaciones de los parámetros dados, para obtener la combinación de hiperparámetros con mejor resultado, además de realizar un entrenamiento con *Cross Validation* definido a 4. En esta prueba se utilizaron las siguientes opciones:

- Máxima profundidad del árbol: 20, 30, 70 y 90
- Cantidad mínima de elementos para separar un nodo interno: 2, 4, 6 y 14
- Cantidad mínima de datos generar una hoja: 1 a 8

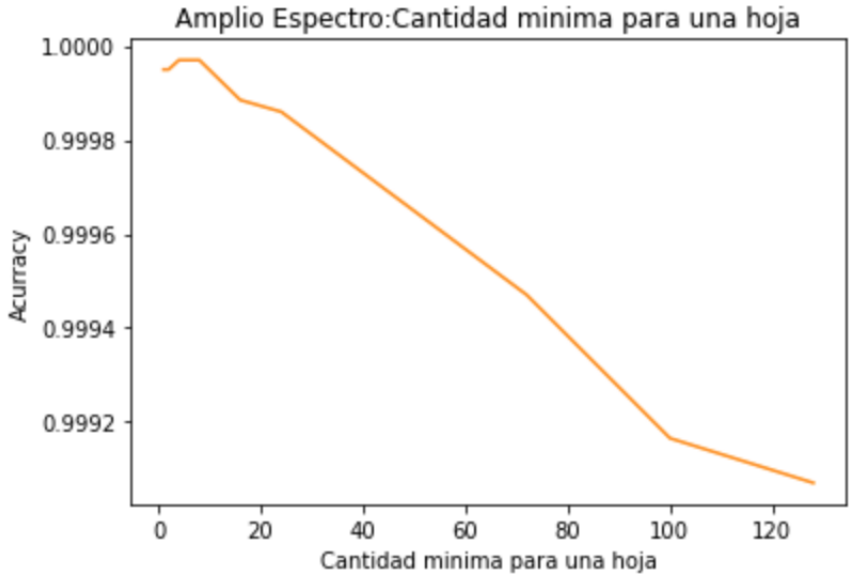


Figura 3.7: Amplio espectro cantidad mínima para generar una hoja

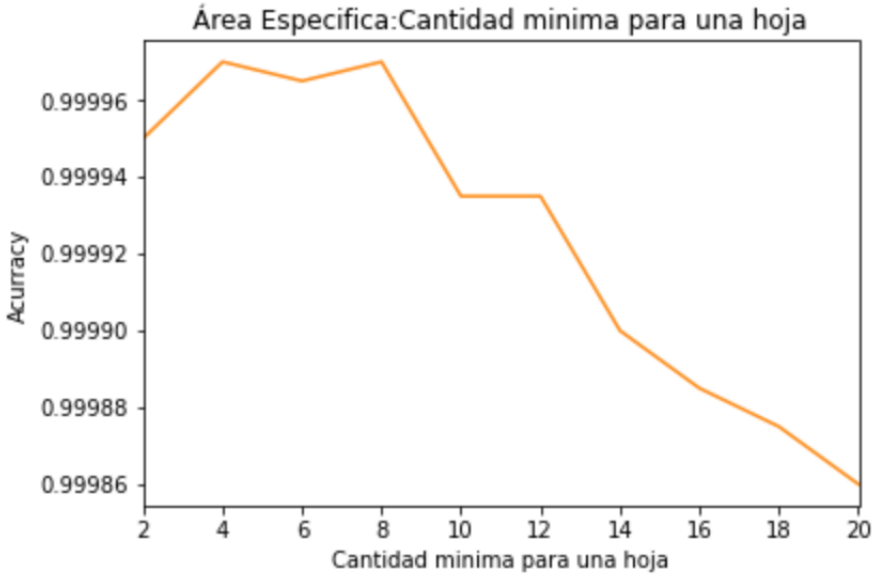


Figura 3.8: Área específica cantidad mínima para generar una hoja

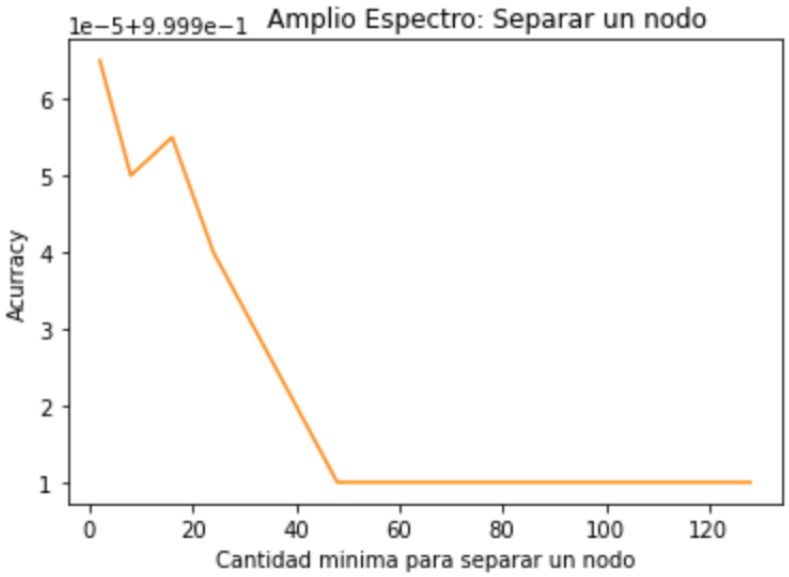


Figura 3.9: Amplio espectro cantidad mínima para para separar un nodo interno

La prueba se realizó 10 veces. Los resultados se pueden observar en el cuadro 4.1. La selección de los hiperparámetros para el modelo del árbol de decisión se basó en el cuadro de resultados y fueron seleccionados de la siguiente manera:

- Máxima profundidad del árbol: 20
- Cantidad mínima de elementos para separar un nodo interno: 4
- Cantidad mínima de datos generar una hoja: 1

Esta decisión se basó en la cantidad de veces que apareció esta combinación de hiperparámetros en los resultados del GridSearch.

3.2.5.2. Análisis de los datos

De acuerdo con los resultados obtenidos en la estimación de los hiperparámetros para el árbol de decisión se observó que los valores obtenidos fueron cercanos al 100 % , por lo cual dado que este es un resultado no esperado en una tarea compleja, se optó por realizar algunos experimentos sencillos adicionales para analizar el comportamiento de los datos y el modelo con el fin de identificar posibles fallos.

El primer experimento se realizó para verificar que el modelo estuviera “aprendiendo”, es decir, que al variar la cantidad de datos de entrada los resultados fueran diferentes. Por otro lado con este experimento se busca analizar que tipo de dato genera un mayor cambio en el rendimiento del

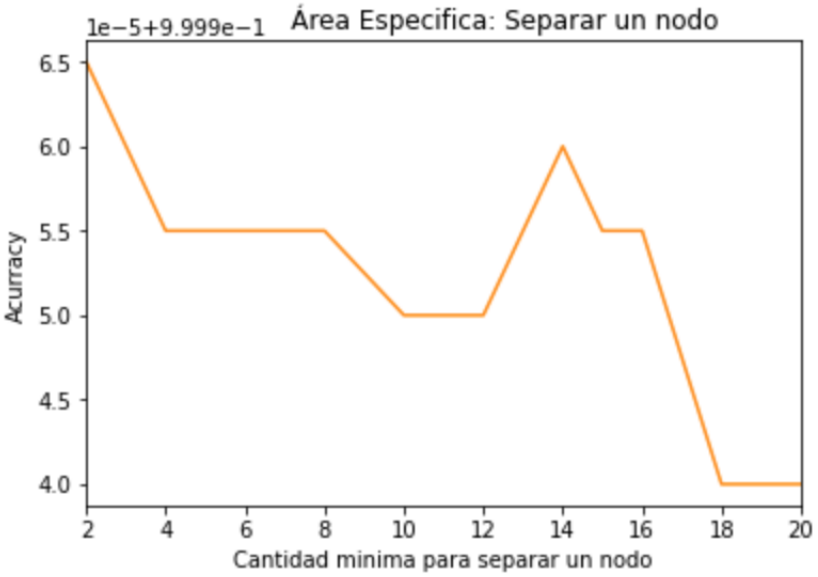


Figura 3.10: Área especifica cantidad mínima para separar un nodo interno

Maxima profundidad	**Cantidad para generar hoja**	**Cantidad para separar nodo**	**Accuracy**
20	1	2	99,9965 %
90	1	2	99,9960 %
20	1	4	99,9975 %
20	1	6	99,9965 %
20	1	4	99,9975 %
20	1	4	99,9975 %
70	1	6	99,9960 %
30	1	2	99,9970 %
20	1	4	99,9975 %
20	1	6	99,9965 %

Cuadro 3.5: Resultados GridSearch árbol de decisión

modelo.

Para esta prueba se decidió generar un modelo para cada tipo de dato, los unigramas, los bigramas y los metadatos. Además cada uno de los modelos generados se entrenaron con tres conjuntos de datos de diferente tamaño, los cuales son: 500.000, 100.000 y 10.000. En el caso de los unigramas y bigramas se conservó la cantidad de 1.000 para cada uno.

El balance entre la clase objetivo se conservó, 50 % bots y 50 % humanos. De igual manera, la proporción de los conjuntos de entrenamiento y pruebas se mantuvo 80 % y 20 % respectivamente para cada conjunto. De los resultados obtenidos (Ver Cuadro 3.6) se puede analizar que de los tres tipos diferentes de datos los que obtuvieron mejores resultados son los unigramas y los metadatos, Por otro lado también se puede observar que los resultados varían frente al cambio en la cantidad de datos, lo cual indica que el modelo está "aprendiendo".

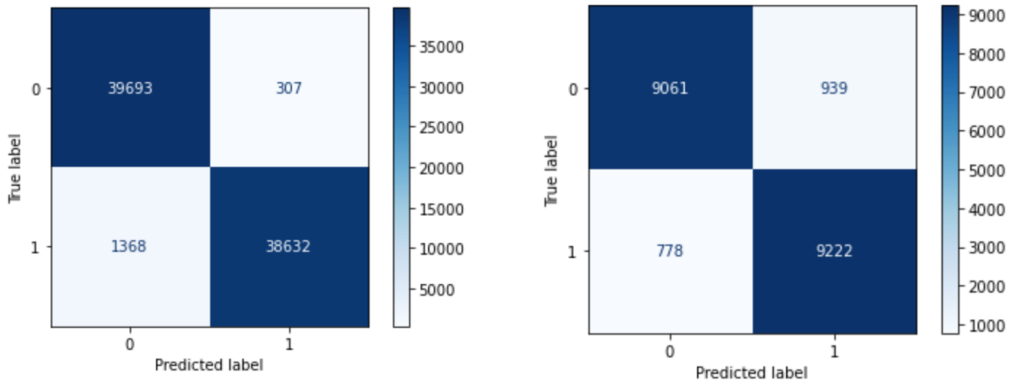
Unigramas			
Cantidad	**Precision**	**Recall**	**F1_score**
500.000	0,9331	0,9337	0,9334
100.000	0,8995	0,9261	0,9126
10.000	0,8258	0,8870	0,8553
Bigramas			
Cantidad	**Precision**	**Recall**	**F1_score**
500.000	0,9698	0,5955	0,7379
100.000	0,9737	0,5858	0,7315
10.000	0,9857	0,5540	0,7093
Metadatos			
Cantidad	**Precision**	**Recall**	**F1_score**
500.000	0,9999	1	0,9999
100.000	0,9994	0,9993	0,9993
10.000	0,9910	0,9950	0,9930

Cuadro 3.6: Resultados del experimento de análisis y aprendizaje de cada tipo de dato

El siguiente experimento se ejecutó una prueba para analizar si el modelo presentaba *overfitting*. En la prueba se utilizó el conjunto de 100.000 datos, se le aplicó a los unigramas, los bigramas y los metadatos. Los resultados de este experimento se pueden ver en el cuadro 3.7 con sus respectivas matrices de confusión en las figuras 3.11 y 3.13. En los resultados se evidencia que los bigramas y metadatos no tiene un *overfitting* alto frente a los unigramas, ya que la variación de los resultados del *train* y *test* es menor que los *train* y *test* de los unigramas.

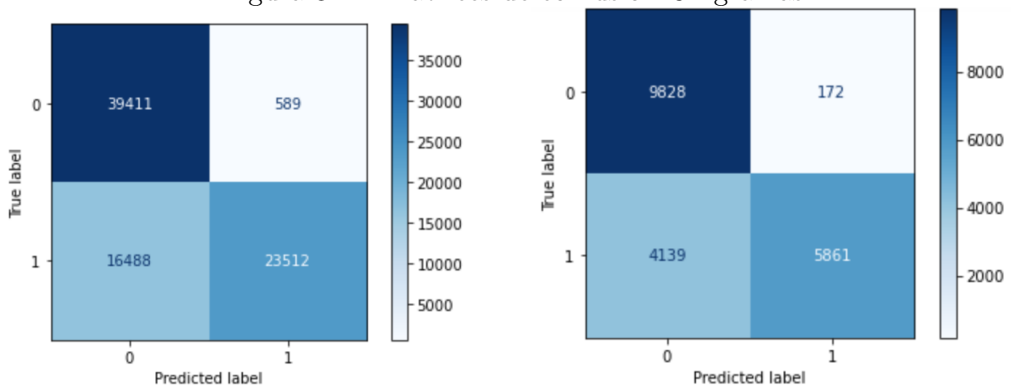
Unigramas			
Conjunto	**Precision**	**Recall**	**F1_score**
train	0,9921	0,9657	0,9787
test	0,9078	0,9222	0,9148
Bigramas			
Conjunto	**Precision**	**Recall**	**F1_score**
train	0,9755	0,5878	0,7311
test	0,9714	0,5861	0,7093
Metadatos			
Conjunto	**Precision**	**Recall**	**F1_score**
train	0,99997	0,99990	0,99993
test	0,9998	0,9997	0,9997

Cuadro 3.7: Resultados de la prueba de *overfitting*



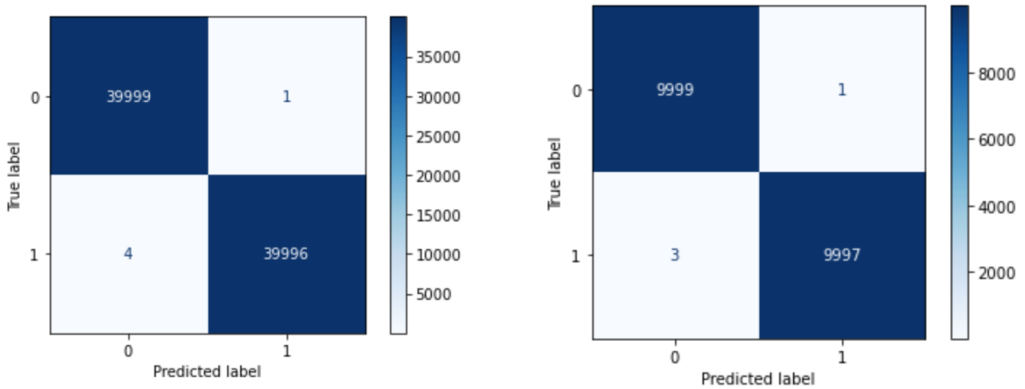
(a) Unigramas-Train (b) Unigramas-Test

Figura 3.11: Matrices de confusión Unigramas



(a) Bigramas-Train (b) Bigramas-Test

Figura 3.12: Matrices de confusión Bigramas



(a) Metadatos-Train

(b) Metadatos-Test

Figura 3.13: Matrices de confusión Metadatos

Se realizó una última prueba con el propósito de analizar el comportamiento de la clasificación después de añadir los metadatos, por lo que se utilizó una combinación de bigramas y metadatos. Esto es debido a que los experimentos realizados con los bigramas dieron menores resultados que los unigramas. Se seleccionó el conjunto de 100.000 datos con 1.000 bigramas y la prueba se llevó a cabo en 3 fases.

- Fase 1: Combinar los 1.000 bigramas con 2 metadatos.
- Fase 2: Al combinación de la fase 1 se le adicionan otros 2 metadatos para un total de 4 metadatos.
- Fase 3: A partir de la fase 2, se adicionan otros 2 metadatos para un total de 6 metadatos.

Los metadatos agregados en las fases previas (Ver Cuadro 3.8) fueron seleccionados de manera aleatoria. Los resultados de la prueba se pueden ver en el cuadro 3.9. Se observa que en la medida que se agregan metadados los resultados mejoran, y esto no es esperado, dado que según la matriz de correlación no hay metadatos con una altísima correlación con la variable objetivo.

num_urls	num_mentions
created_at_x	followers_count
description	statuses_count

Cuadro 3.8: Metadatos seleccionados

3.2.5.3. Red neuronal convolucional (CNN)

En la red neuronal convolucional se tomó como base la arquitectura presentada por Kalchbrenner,Grefenstette, et al. [35] en donde presentan un modelo con un *embedding layer*, dos capas

Datos	Precision	Recall	F1_score
Bigramas + 2 metadatos	0,9359	0,7048	0,8041
Bigramas + 4 metadatos	0,9799	0,9857	0,9828
Bigramas + 6 metadatos	0,9938	0,9973	0,9955

Cuadro 3.9: Resultados Bigramas con 6 metadatos

convolucionales, dos capas de *pooling* dinámico, una capa de *folding* y por último una capa densa de salida. Para este trabajo de grado se tomó la arquitectura descrita anteriormente con algunas diferencias tales como: no incluir la capa de *folding* y no usar un *pooling* dinámico, ya que se opto por realizar una aquitectura mas sencilla debido a la limitación de procesamiento descrita en la sección 3.1.2. Además se tomó la inicialización del *embedding layer* utilizando 3 matrices de vocabulario diferentes, la primera el vocabulario generado por los datos de entrenamiento, la segunda tomando un vocabulario general de GloVe y por último un vocabulario de GloVe enfocado en la clasificación de tweets.

Se realizó una búsqueda exhaustiva, mediante una funcionalidad de TensorFlow que permite utilizar un modelo de keras con funcionalidades de los modelos de Sklearn. La función que se utilizó para realizar la búsqueda exhaustiva fue GridSearchCV, esta función se ejecuto para estimar los mejores valores de los siguientes parametros:

1. Función de activación: Esta función es la que le permite a la neurona reconstruir o predecir con base en los datos que le ingresaron a la neurona. Las funciones evaluadas fueron Relu, Sigmoid, Softmax, Softplus, Softsign, Tanh, Selu, Elu, Exponential [36].

2. El tamaño del kernel: Este valor representa el tamaño de la agrupación de los filtros. Por motivos de consistencia y simplicidad se decidió que el tamaño de los kernels fuera el mismo para todas las capas. De acuerdo a la recopilación de literatura [37] y de varios foros los valores a evaluar fueron 2, 3, 4, 5.

3. El número de filtros o neuronas: Son los encargados de captar las características más significativas provenientes de los datos de entrada. Basado en lo expuesto por Heaton [38], se plantean una serie de rangos para empezar a buscar estos valores.

   - El número de neuronas debe ser menor al doble de la capa de entrada.(menor a 50)
   - El número de neuronas debe estar entre el tamaño de la capa de entrada y la capa de salida.(25 y 2)
   - El número de neuronas debe ser $\frac{2}{3}$ de la capa de entrada más la capa de salida. (18)

   Basados en la recopilación de múltiples foros donde comentan que para obtener una mejora en el tiempo de ejecución de las redes es mejor manejar el número de neuronas en potencias de 2. Para comprobar las sugerencias obtenidas en los foros se realizó una prueba donde se compararon los tiempos de ejecución de los siguientes conjuntos de valores. Conjunto 1 (7, 9,

13, 21, 37), conjunto 2 (2, 4, 8, 16, 32), se observó una disminución pequeña en el tiempo de ejecución del conjunto 2 frente al conjunto 1. Teniendo en cuenta lo anterior y lo planteado por Heaton [38], los valores evaluados al final fueron 2, 4, 8, 16, 32.

Para finalizar la búsqueda exhaustiva se ejecutó 10 veces y se obtuvieron los siguientes resultados (Ver Cuadro 3.10). Dados los resultados presentados se decidió que:

- Para la primera capa convolucional se seleccionó la función de Relu como función de activación, ya que en la mayoría de ejecuciones fue la opción seleccionada, con una ocurrencia de 6 veces.
- En el caso de la segunda capa convolucional hubo una variación, ya que dos funciones aparecieron la misma cantidad de veces, las funciones a explorar fueron Relu y Selu. Para tomar la decisión se tuvo en cuenta la selección realizada en la primera capa, puesto que se observó que en la mayoría de las ocurrencias de la función relu en la primera capa va acompañada de la función Selu en la segunda capa, por ende se decide elegir Selu para la segunda capa.
- Para el tamaño del kernel y el número de filtros se tuvo en cuenta las funciones elegidas para las capas convolucionales, es decir la tupla (Relu, Selu), observando la aparición de esta tupla en las 10 ejecuciones se noto que en la mayoría de sus ocurrencia los valores para el tamaño del kernel fueron 3 y 32 en el número de filtros. Por tal motivo los valores escogidos para el kernel y los filtros fueron 3 y 32 respectivamente.

Ejecución	Conv 1	Conv 2	Kernel	Filtros	Accuracy
1	Relu	Selu	3	32	89,225 %
2	Relu	Selu	3	32	89,219 %
3	Relu	Relu	5	16	89,173 %
4	Selu	Relu	4	16	89,141 %
5	Relu	Relu	4	32	89,218 %
6	Relu	Selu	3	32	89,227 %
7	Tanh	Exponential	2	8	89,118 %
8	Relu	Selu	2	16	89,145 %
9	Selu	Exponential	5	32	89,183 %
10	Softsign	Relu	3	8	89,203 %

Cuadro 3.10: Resultados GridSearch CNN

Finalmente para decidir qué matriz de vocabulario usar en la inicialización de la capa de embedding se realizó un experimento, el cual consistió en evaluar los resultados de la CNN usando las 3 diferentes matrices de vocabulario. La primera matriz generada por el vocabulario de los datos de entrenamiento cuenta con 700.000 tweets de dimensión 25 (Ver Figura 3.14), la segunda matriz tomada de un vocabulario general de GloVe que tiene 2,196.017 de oraciones con dimensión de 300 (Ver Figura 3.15) y por último la tercera matriz construida a partir de un vocabulario de GloVe enfocado en la clasificación de tweets con 1,193.514 de tweets y una dimensión de 200 (Ver Figura 3.16).

La evaluación y comparación de las matrices de vocabulario se llevó a cabo con los datos del *Validation Set*. Los resultados obtenidos en este experimento se pueden ver en el cuadro 3.11

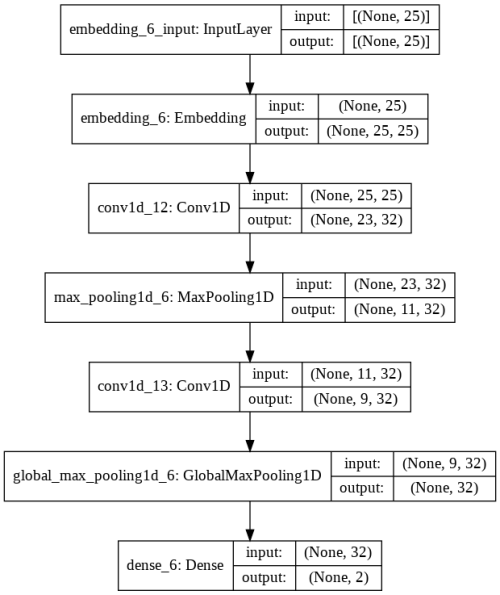


Figura 3.14: Modelo CNN vocabulario de los datos de entrenamiento

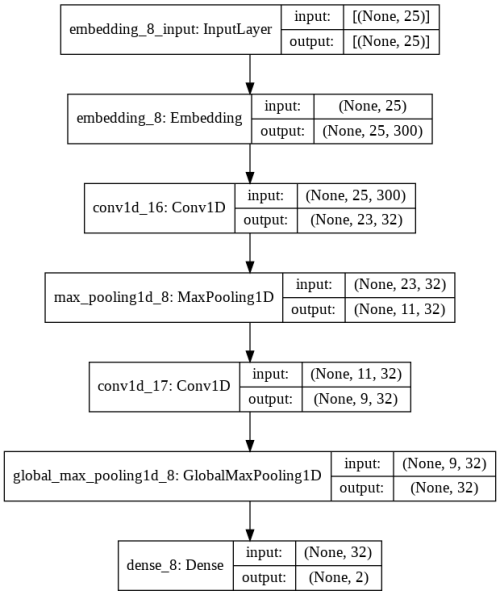


Figura 3.15: Modelo CNN vocabulario GloVe general

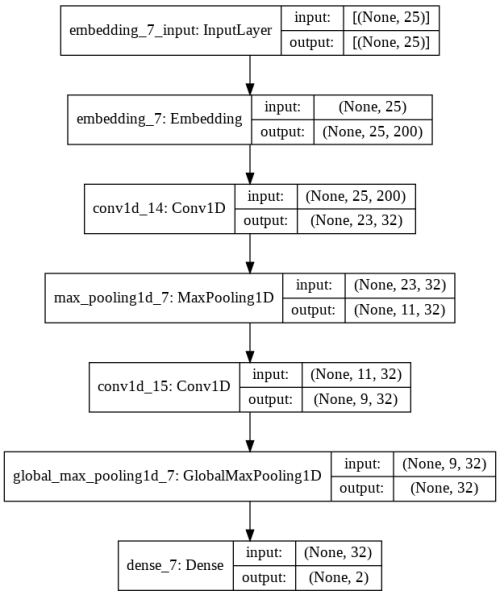


Figura 3.16: Modelo CNN vocabulario GloVe de tweets

Embedding	**Precision**	**Recall**	**F1_score**
Datos propios	0,8926	0,8922	0,8922
GloVe general	0,9568	0,9551	0,9550
GloVe tweets	0,9526	0,9504	0,9503

Cuadro 3.11: Resultados de matrices de embeddings

Capítulo 4

Clasificación

Para esta sección de clasificación los modelos fueron generados nuevamente con los mejores hiperparametros presentados en la sección 3.2.5. Para poder evaluar los modelos, se utilizó los conjuntos de datos Test Set.

Las figuras de las matrices de confusión en esta sección pueden ser analizadas mediante la siguiente información:

- La tonalidad del color azul representa la cantidad de datos clasificados en ese cuadrante, entre más claro el color indica que la cantidad es menor y entre más oscuro significa mayor cantidad de datos.
- La etiqueta 1 es la clase que representa los tweets clasificados como bots y 0 son los tweets clasificados como humanos.

4.1. Resultados

En esta sección del documento se presentan los resultados obtenidos de la clasificación de los dos modelos finales implementados para este trabajo.

la selección de los conjuntos de datos para las técnicas se planteo así desde el inicio del proyecto, puesto que se propuso usar una técnica que sacara el mayor provecho del texto solamente, en este caso la red convolucional, y compararla con una técnica donde se usara texto y metadatos, el árbol de decisión.

4.1.1. Árbol de decisión

Para el árbol de decisión se utilizó el primer conjunto de datos compuesto por los tweets y los metadatos de las cuentas de twitter. El conjunto de datos esta compuesto por 1,000.000 de datos con 1.000 unigramas y 12 metadatos seleccionados en la sección 3.2.3, se dividió el conjunto en 2 subconjuntos descritos a continuación:

1. Conjunto de entrenamiento (Train Set),este conjunto está conformado por el 80 % del conjunto original y cuenta con 800.000 tweets.
2. Conjunto de pruebas (Test Set), es el 20 % del conjunto original y tiene 200.000 tweets.

Los hiperparámetros para generar el árbol de decisión fueron:

- Profundidad = 20
- Cantidad Mínima para generar una hoja = 1
- Cantidad mínima para separar un nodo interno = 4

En el cuadro 4.1 se encuentran los resultados del modelo y en la figura 4.1 la matriz de confusión.

Como se esperaba, la combinación de unigramas y metadatos dieron resultados altos, ya que los unigramas por sí solos generan métricas altas. Adicional a eso, se puede concluir que, a partir de los experimentos realizados en 3.2.5.2, el aporte de los metadatos es significativo para la clasificación, lo que generaría un mejor resultado al combinarlos con los unigramas.

Precision	**Recall**	**F1_score**
0,99994	0,99765	0,99884

Cuadro 4.1: Resultados Árbol de decisión

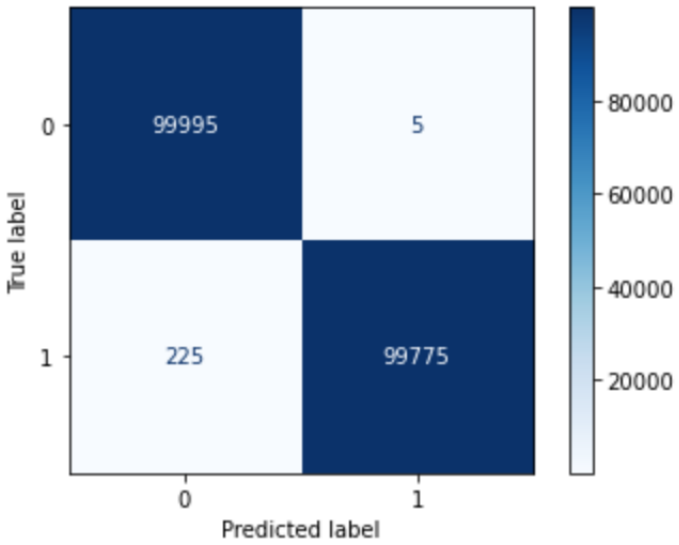


Figura 4.1: Matriz de confusión Árbol de decisión

4.1.2. CNN

En el modelo de la red neuronal convolucional se usó el segundo conjunto de datos compuesto por solo los tweets, el cual esta conformado por 1,000.000 de tweets. Este conjunto de datos se dividió en 2 subconjuntos descritos a continuación:

1. Conjunto de entrenamiento (Train Set), este conjunto está conformado por el 80 % del conjunto original y cuenta con 800.000 de tweets.

2. Conjunto de pruebas (Test Set), es el 20 % del conjunto original y tiene 200.000 tweets.

Los hiperparámetros para generar la red convolucional fueron:

- Embedding layer = embedding GloVe general con dimensión de 300.
- Función de Activación capa 1 = Relu
- Función de Activación capa 2 = Selu
- Tamaño del Kernel = 3
- Numero de filtros o neuronas = 32

En el cuadro 4.2 se encuentran los resultados del modelo y en la figura 4.2 la matriz de confusión.

En la red neuronal convolucional se presentan resultados diferentes al del árbol de decisión. En este caso se obtuvieron resultados menores. Esto, unido a los resultados de los experimentos sencillos realizados en el árbol de decisión (Sección 3.2.5.2) permite plantear la hipótesis de que los metadatos permiten identificar cuando un tweet fue generado por un bot o por un humano. Por otro lado los resultados obtenidos son altos, ya que están por arriba de un 0.95 en las métricas utilizadas en la evaluación del modelo.

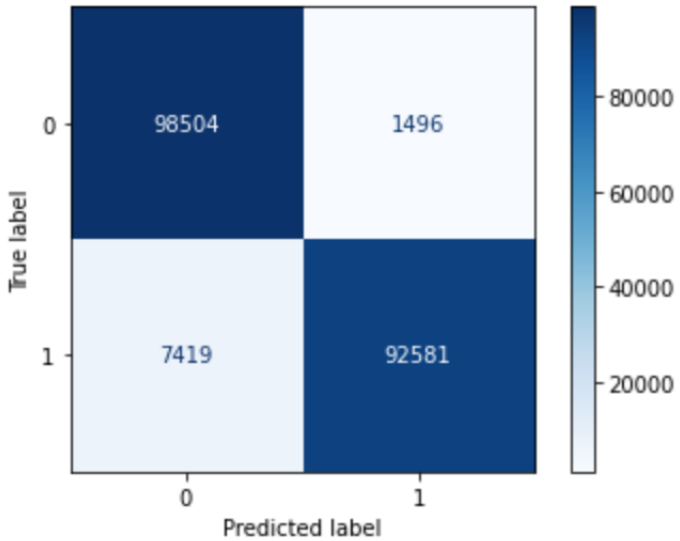


Figura 4.2: Matriz de confusión CNN

Precision	Recall	F1_score
0,9570	0,9554	0,9553

Cuadro 4.2: Resultados de CNN

Capítulo 5

Conclusiones

Durante la experimentación, se observó que el desempeño del árbol de decisión mejora significativamente al aumentar el número de metadatos, por lo que se concluye que de los tipos de datos analizados en 3.2.5.2, los metadatos son los que mayor significancia tienen para la clasificación. Así mismo, se evidencia que los unigramas tienen mayor relevancia que los bigramas de acuerdo a los resultados obtenidos.

Otro aspecto que resalta son los resultados obtenidos en el árbol de decisión, dado que es una tarea tan compleja no se esperaba obtener resultados tan altos. Basados en la conclusión anterior junto al análisis que se realizó en este trabajo, se puede concluir que existe un componente en los metadatos que genera resultados altos en la clasificación, a pesar de que esto no es esperado, ya que según la matriz de correlación no hay metadatos con un alto coeficiente de correlación con la variable objetivo.

En [26] se realizó un modelo de clasificación con aprendizaje no supervisado y los resultados obtenidos del trabajo fueron los siguientes: *precision* del 100 %, *recall* del 85 % y *specificity* del 100 %. Estos resultados son cercanos a los resultados obtenidos en el árbol de decisión. A partir de lo anterior y análisis de los datos realizados en 3.2.5.2 se concluye que los metadatos son el factor diferenciador para la similitud entre los dos modelos.

Por otro lado, los resultados obtenidos en la red neuronal convolucional son similares a los obtenidos por A.David, O.Varol et al. [27], donde presentan un *accuracy* del 95 % en el modelo que desarrollaron. Además son cercanos a los resultados presentados por Z.Chu , S. Gianvecchio, et al. [25], y N. Chavoshi, H. Hamoon, et al. [39] donde obtuvieron un 96 % de *accuracy* y un 94 % de *precision* respectivamente.

Para el modelo de la red neuronal convolucional, se identifica, a partir de los resultados obtenidos en 3.11, que la matriz con mejor desempeño fue la de GloVe general inicializada con una dimensionalidad de 300, por lo que se puede concluir que a mayor dimensionalidad en la capa de embeddings, mejor el desempeño del modelo.

Para finalizar se cumplieron con los objetivos propuestos para este trabajo, ya que se logró implementar dos técnicas de aprendizaje automático para la clasificación de publicaciones realizadas por bots en twitter.

5.1. Trabajo Futuro

Como trabajo futuro, este trabajo puede ser ampliado en el análisis de los datos y su comportamiento en la clasificación para tratar de interpretar cómo los metadatos permiten identificar el origen de los tweets de una manera tan precisa.

Adicionalmente se puede cambiar los conjuntos de datos por conjuntos nuevos, debido a que este trabajo fue realizado con datos obtenidos de otra investigación, seria importante explorar con el cambio de los datos, puesto que pueden traer resultados diferentes, ya que en este tipo de estudio tiene mucha influencia los datos que se usan para entrenar los modelos.

Por otro lado se pueden generar WordEmbeddings y entrenar modelos para la representación de palabras como vectores a partir de conjuntos de datos propios y no modelos preentrenados, con el objetivo de evaluar como es el desempeño de los agentes,en la clasificación bots en twitter.

Bibliografía

[1] M. Jiang, P. Cui, and C. Faloutsos, "Suspicious Behavior Detection: Current Trends and Future Directions," *IEEE Intelligent Systems*, vol. 31, no. 1, pp. 31–39, 2016.

[2] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," *Proceedings of the 11th International Conference on Web and Social Media, ICWSM 2017*, pp. 280–289, 2017.

[3] R. A. Igawa, S. Barbon, K. C. S. Paulo, G. S. Kido, R. C. Guido, M. L. P. Júnior, and I. N. da Silva, "Account classification in online social networks with LBCA and wavelets," *Information Sciences*, vol. 332, pp. 72–83, 2016.

[4] N. Chavoshi, H. Hamooni, and A. Mueen, "Temporal patterns in bot activities," *26th International World Wide Web Conference 2017, WWW 2017 Companion*, pp. 1601–1606, 2019.

[5] Cheng A. & Evans M. Social media monitoring and analytics and Solutions., "Twitter Statistics - An In-depth Report At Most Active Twitter Users." [Online]. Available: https://sysomos.com/inside-twitter/most-active-twitter-user-data/

[6] N. Abokhodair, D. Yoo, and D. W. McDonald, "Dissecting a Social Botnet," pp. 839–851, 2015.

[7] B. Y. E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "P96-Ferrara."

[8] S. M. Pang-ning tan and K. Vipin, *Introduction to Data Minnig*.

[9] L. Hickman, S. Thapa, L. Tay, M. Cao, and P. Srinivasan, "Text Preprocessing for Text Mining in Organizational Research: Review and Recommendations," *Organizational Research Methods*, no. October, 2020.

[10] M. M. Mirończuk and J. Protasiewicz, "A recent overview of the state-of-the-art elements of text classification," *Expert Systems with Applications*, vol. 106, pp. 36–54, 2018.

[11] R. Sathya and A. Abraham, "Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification," *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 2, pp. 34–38, 2013.

[12] A. G. Barto and T. G. Dietterich, "Reinforcement learning and its relationship to supervised learning," *Handbook of Learning and Approximate Dynamic Programming*, pp. 47–63, 2004.

[13] E. Alothali, N. Zaki, E. A. Mohamed, and H. Alashwal, "Detecting Social Bots on Twitter: A Literature Review," *Proceedings of the 2018 13th International Conference on Innovations in Information Technology, IIT 2018*, pp. 175–180, 2019.

[14] J. H. Maindonald, *Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery by Graham Williams*, 2012, vol. 80, no. 1.

[15] M. Reza, S. Miri, and R. Javidan, "A Hybrid Data Mining Approach for Intrusion Detection on Imbalanced NSL-KDD Dataset," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 6, pp. 1–33, 2016.

[16] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[17] J. Martinez, "Redes Neuronales Convolucionales en Profundidad - DataSmarts Español," 2019. [Online]. Available: https://datasmarts.net/es/redes-neuronales-convolucionales-en-profundidad/

[18] T. Guo, J. Dong, H. Li, and Y. Gao, "Simple convolutional neural network on image classification," *2017 IEEE 2nd International Conference on Big Data Analysis, ICBDA 2017*, pp. 721–724, 2017.

[19] Y. Cui, Y. Shi, X. Sun, and W. Yin, "S-net: A lightweight convolutional neural network for n-dimensional signals," *2018 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2018*, pp. 4–7, 2018.

[20] J. Alexander, "Machine learning tasks | Microsoft Docs," 2018. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/machine-learning/resources/tasks

[21] O. Pentakalos, "Introduction to machine learning," in *CMG IMPACT 2019*, 2019. [Online]. Available: https://heartbeat.fritz.ai/introduction-to-machine-learning-model-evaluation-fa859e1b2d7f

[22] A. Mishra, "Metrics to Evaluate your Machine Learning Algorithm," *Towards Data Science*, pp. 1–8, 2018. [Online]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234?fbclid=IwAR2e4qvqwBt1HcdQIUD27vvdpyWXP201mbDvhcDobBFI4KT0gl9JzJR18to

[23] C. M. Zhang and V. Paxson, "Detecting and analyzing automated activity on twitter," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011.

[24] Z. Gilani, E. Kochmar, and J. Crowcroft, "Classification of twitter accounts into automated agents and human users," *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2017*, pp. 489–496, 2017.

[25] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Detecting automation of Twitter accounts: Are you a human, bot, or cyborg?" *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 6, pp. 811–824, 2012.

[26] S. Cresci, A. Spognardi, M. Petrocchi, M. Tesconi, and R. D. Pietro, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," *26th International World Wide Web Conference 2017, WWW 2017 Companion*, pp. 963–972, 2019.

[27] C. A. Davis, O. Varol, E. Ferrara, A. Flammini, and F. Menczer, "BotOrNot: A System to Evaluate Social Bots," pp. 4–5, 2016. [Online]. Available: http://arxiv.org/abs/1602.00975{%}0Ahttp://dx.doi.org/10.1145/2872518.2889302

[28] C. Yang, R. Harkreader, and G. Gu, "Empirical evaluation and new design for fighting evolving twitter spammers," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1280–1293, 2013.

[29] Towards Data Science, "Towards Data Science." [Online]. Available: https://towardsdatascience.com/

[30] EliteDataScience.com, "Dimensionality Reduction Algorithms: Strengths and Weaknesses," 2019. [Online]. Available: https://elitedatascience.com/dimensionality-reduction-algorithms

[31] M. Porter, "Porter- Stemming."

[32] Mathworks, "¿Qué es un n-grama? - MATLAB." [Online]. Available: https://la.mathworks.com/discovery/ngram.html

[33] W. Alawad, M. Zohdy, and D. Debnath, "Tuning Hyperparameters of Decision Tree Classifiers Using Computationally Efficient Schemes," *Proceedings - 2018 1st IEEE International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2018*, pp. 168–169, 2018.

[34] R. G. Mantovani, T. Horváth, R. Cerri, S. B. Junior, J. Vanschoren, and A. C. P. d. L. F. de Carvalho, "An empirical study on hyperparameter tuning of decision trees," 2018. [Online]. Available: http://arxiv.org/abs/1812.02207

[35] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, vol. 1, pp. 655–665, 2014.

[36] K. Team and K. Documentation, "Layer activation functions." [Online]. Available: https://keras.io/api/layers/activations/

[37] Y. Zhang and B. Wallace, "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification," pp. 253–263, 2015. [Online]. Available: http://arxiv.org/abs/1510.03820

[38] J. Heaton, *Introduction to Neural Networks for Java, 2nd Edition.* Heaton Research, Inc., 2013, vol. 99.

[39] N. Chavoshi, H. Hamooni, and A. Mueen, "DeBot: Twitter bot detection via warped correlation," *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 817–822, 2017.