Análisis de sentimientos sobre la percepción ciudadana de la vacunación del COVID-19 en Colombia

Héctor Leonardo Arias García

Luis Carlos Doria Pérez

Proyecto de Grado

Directores

Elio H. Cables Pérez, Ph.D.

Edison Leonardo Neira Espitia, Esp.

Especialización en Gobierno de Datos

Facultad de ingeniería, Universidad Antonio Nariño

2021

Contenido

Índice de figuras

Índice de tablas

Resumen

En el presente trabajo de grado se realiza un análisis de sentimiento de la percepción de la vacunación del COVID-19 en Colombia, tomando como fuente de datos las publicaciones en la red social Twitter. Con las actividades realizadas se logró obtener información de los Tweets desde el día 15 de marzo al día 25 de abril del año 2021 por medio de la API streaming proporcionada por la red social, se almaceno la información en bases de datos MongoDB en la nube. Se utilizó Python como lenguaje de programación para la implementación del código fuente mediante la creación de notebooks. La metodología utilizada está comprendida en las siguientes etapas: Identificación de datos, extracción de datos, preprocesamiento, procesamiento y resultados. Se aplico la técnica de aprendizaje supervisado para la clasificación de Tweets positivos, negativos y neutrales, se implementaron los algoritmos de machine learning para el entrenamiento de modelos con el fin de evaluar el desempeño en las predicciones de los datos clasificados. De acuerdo con el análisis se identificó que las diferencias en los porcentajes de precisión de los modelos no representaron una cantidad significativa. Este estudio pretende servir como base para posibles trabajos futuros donde se implementen las técnicas de machine learning para identificar patrones de comportamiento a partir de las opiniones de los ciudadanos en redes sociales.

Introducción

Un factor determinante en la globalización ha sido internet, el cual ha permitido la intercomunicación, desde sus orígenes cuando se creó la primera red informática que permitió la comunicación entre diversas universidades en Estados Unidos. Desde hace aproximadamente cincuenta años no ha parado de evolucionar y hoy día se puede afirmar que el vertiginoso avance de internet ha cambiado la forma como se interactúa, dejando de ser un mecanismo estático para evolucionar al contenido social. Las redes sociales hacen referencia a aquellas plataformas digitales que nos permiten estar comunicados a través de una herramienta informática en la cual es posible compartir todo tipo de información, como mensajes de texto, imágenes digitales, vídeo, audio, entre otros.

Del mismo modo ha cambiado la forma como la sociedad piensa sobre determinado tema en particular, ya que esta forma de comunicación permite expresar opiniones sobre temas específicos influenciando nuestra forma de pensar y actuar; al tener acceso a gran variedad de contenido, bien sea mensajes de texto, imágenes digitales, vídeo, audio entre otros (Mathkour et al., 2015). Para citar un ejemplo la red social Twitter, en la cual se pueden compartir mensajes mediante los denominados Tweets, que es un mensaje digital que se envía a través de la red social Twitter. Estos mensajes permiten expresar opiniones sobre diversos temas que se comparten en la red social, exponiendo un punto de vista, es así como esta red social influye de cierta manera en nuestra forma de pensar y actuar, permitiendo convertir temas de interés en tendencias a través de los comentarios masivos que se llegan a popularizar respecto a un tema específico.

Para el análisis de los textos de los tweets es importante considerar la gran variedad de contenidos en forma de artículos, por ejemplo los publicados en la revista ELSEVIER, los cuales

hacen referencia a la necesidad de ser capaz de almacenar alto volumen de información, hace muchos años atrás se pudo evidenciar cómo nacieron los sistemas de almacenamiento de información estructurados, estos sistemas si bien en la actualidad son muy usados por la industria ya que permite tener una información organizada en formato de texto en filas y columnas, la necesidad de almacenamiento y captura de diversos tipos de formatos llevó a la evolución en la cual estamos hoy día y que conocemos como datos no estructurados y semi estructurado, es así como el 80% de las información relevante de una empresa se encuentra en datos no estructurados (Narvaiza Cortes & Medina Valverde, 2020).

De tal modo, a través de los comentarios de una red social se puede deducir que piensan las personas sobre determinado tema de interés, en este trabajo en particular se hará referencia al análisis de los comentarios publicados frente a la aceptación de la vacunación del COVID-19 en Colombia. Con el fin de realizar un análisis respecto a esta temática se propone realizar tareas de **text mining** extrayendo información de los textos escritos para analizarlos, procesarlos y lograr identificar patrones de comportamiento que pueden contribuir a mejorar la aceptación o rechazo de un producto una idea o un tema en particular (Mathkour et al., 2015).

Formulación y Descripción Del Problema

En un mundo como el actual, donde la era digital replica y transmite la información en tiempo real a cualquier rincón del planeta, donde existe una crisis global y una emergencia de salud pública como la pandemia del Covid 19 se exige realizar un análisis de los pensamientos de las personas al respecto de la vacunación al Covid 19 y tener en cuenta tanto la participación como la opinión ciudadana.

La vacunación del Covid es un tema de alta discusión y de posiciones encontradas, la tendencia de las personas frente a su aceptación o no en Colombia está influenciada en gran medida por la información que consumen de los medios de comunicación y las redes sociales. Por otra parte, no todas las fuentes de información son fidedignas, existen fake news (noticias falsas) divulgadas especialmente en las redes sociales que genera confusión en las personas y hacen que tomen una posición distante a la realidad frente a la vacunación resultado de esta información poco confiable. Las tendencias positivas o negativas de la percepción de la vacunación en Colombia hacen que los gobiernos tanto a nivel nacional como municipal tomen decisiones frente a sus planes de divulgación, concientización y aceptación de la vacuna del Covid 19 con el fin de mejorarlos y mitigar los efectos negativos o adversos que esto podría representar en la salud pública del país.

Por lo expuesto anteriormente se identifica la siguiente problemática: no considerar las posiciones a favor o en contra de los ciudadanos, la divulgación de falsa información y el hecho de que la decisión de la vacunación es voluntaria podría dejar sin efecto los esfuerzos publicitarios que se ejecutan actualmente para la divulgación de información real sobre efectos y beneficios de la inmunización ante el virus. Es importante analizar el fenómeno social de la vacunación en Colombia e identificar los posibles impactos en los medios de divulgación que deben velar por establecer canales cercanos de interacción con la ciudadanía.

Pregunta de investigación

¿Cómo clasificar el grado de aceptación de la vacunación del Covid-19 en Colombia a partir del análisis de sentimientos de los tweets publicados en la red social Twitter?

Objetivos

Objetivo General

Clasificar los tweets sobre la vacunación del Covid-19 en Colombia según su sentimiento: Positivo, negativa o neutral, para los meses de marzo y abril de 2021, basado en herramientas de machine learning.

Objetivos Específicos

Caracterizar los métodos, procedimientos y modelos utilizados para la predicción a partir del estudio de redes sociales.

Obtener información publicada en los tweets de la red social Twitter entre los meses de marzo y abril del 2021 mediante las APIs proporcionadas por la misma plataforma, referentes a la vacunación del covid-19.

Aplicar los procedimientos existentes para la limpieza, procesamiento y análisis de información referida a los tweets emitidos en Colombia.

Evaluar el desempeño de diversos modelos de predicción de machine learning con aprendizaje supervisado para la clasificación de los tweets.

Marco referencial

Marco teórico

De acuerdo con las diferentes técnicas que se aplicarán en este proyecto de aplicación, como también los procedimientos y metodologías, es importante contextualizar al lector del presente documento respecto a los términos técnicos de los cuales se ha hecho referencia hasta el momento.

Inteligencia artificial

Es importante recordar que la inteligencia artificial no es un término reciente, su origen data de la década de los cincuenta donde los departamentos de defensa de las potencias mundiales apostaron por desarrollar sistemas capaces de identificar patrones con el fin de que las computadoras tomarán decisiones respecto a la aparición de dichos patrones (SAS, 2021). Como se puede observar el término es de vieja data y su desarrollo ha sido constante con el paso de las décadas, debido a la poca capacidad para almacenar y procesar grandes volúmenes de datos su evolución inicial estuvo enfocada más en la teoría, teoría que es la base fundamental de la práctica actual en donde se cuenta con la tecnología necesaria para procesar esos grandes volúmenes.

Se podría decir que son tres las etapas en la evolución de la inteligencia artificial. La primera comprendida entre la década de los cincuenta y setentas, en donde el desarrollo de redes neuronales para crear máquinas pensantes puso sobre la mesa uno de sus principios fundamentales. La segunda etapa entre la década de los ochenta y la primera década del siglo XXI desarrollando el aprendizaje autónomo (machine learning) (Valcárcel Asencios, 2004). Finalmente, la presente era en donde el deep learning entró a escena como actor principal. Actualmente el término ha tomado una relevancia diferente y distante de aquella que compartían las películas y series de ciencia ficción donde existían los robots autónomos con un sistema de aprendizaje que se podrían salir de control

y convertirse en una amenaza para la humanidad. Por el contrario, hoy en día la industria ha permitido que el sistema de aprendizaje sea aplicado en la red computacional con el fin de mejorar los negocios en todo el espectro del mercado (SAS, 2021).

Data mining

Cuando se habla de Data mining se hace referencia a una técnica que permite explorar los datos tomando como base a la estadística, con el fin de encontrar patrones dentro de grandes volúmenes de datos (Valcárcel Asencios, 2004). En lo que concierne al presente proyecto esta técnica apoyará el análisis de datos con el fin de identificar patrones relevantes en la posición de los ciudadanos colombianos respecto a la aceptación o no de la vacunación del Covid-19 en Colombia.

Text mining

El text mining es una técnica de extracción de datos que permite obtener información significativa sobre orígenes de datos de tipo texto (Bian et al., 2014). Su valor real se encuentra en la programación realizada en la máquina para que ésta pueda realizar lecturas sobre datos no estructurados y reportar resultados con valor real para cualquier investigación. Teniendo en cuenta que la opinión ciudadana es diversa y más la socialización de ideas por medio de las redes sociales, se considera que el Text mining como técnica para la identificación de relaciones con la información textual analizada, permitirá identificar tendencias e influencias en la posición tomada ya sea a favor o en contra de la vacunación contra el virus. En la ejecución de esta técnica se suelen encontrar palabras vacías que son aquellas sin significado tales como: artículos, pronombres, preposiciones, etc. que son filtradas antes o después del procesamiento de datos en lenguaje natural (texto). Hans Peter Luhn es uno de los pioneros en la recuperación de información, a quien se le

atribuye la acuñación de la locución inglesa stop words y el uso del concepto para la identificación de palabras conectoras aplicando análisis estadísticos en los textos.

Procesamiento de lenguaje natural (PLN)

El procesamiento de lenguaje natural se basa en la investigación de mecanismos capaces de interpretar la interacción entre el lenguaje humano y las máquinas. Para estos casos es necesario un intérprete construido bajo código fuente que logra entregar resultados derivados de esta interacción, Python es un lenguaje de programación que soporta este tipo de lógica interpretativa con un rendimiento de procesamiento bastante eficiente para la entrega de reportes finales. El PLN es de utilidad para la identificación de patrones significativos respecto a las opiniones expresadas por los ciudadanos en un Tweet o imagen respecto a un tema específico

Machine learning

Basado en los modelos analíticos como el socializado en el presente documento, el machine learning busca que las máquinas tengan la capacidad de tomar sus propias decisiones, se espera que dichas decisiones sean cada vez más acertadas y con menos intervención humana (SAS, 2021). En el mercado su aplicación es muy importante porque gracias a su capacidad de almacenamiento y procesamiento de grandes volúmenes de datos permite identificar oportunidades de mejora para el negocio y evitar posibles riesgos que afecten la operación de este, tarea que con esfuerzo humano llevaría mucho más tiempo de lograr.

Deep learning

Este término hace parte del machine learning, pero con un enfoque más profundo en donde se busca que las máquinas sean capaces de identificar los patrones no solo de texto sino de fuentes

humanas más complejas como videos, imágenes, expresiones humanas con el fin de predefinir parámetros para que la máquina sea capaz de mejorar su aprendizaje aplicando modelos cada vez más novedosos (SAS, 2021).

Web Scraping

Es una técnica para recolectar información de cualquier sitio web de forma automática, consultando y extrayendo los contenidos de interés desde diferentes fuentes. Es necesario conocer la estructura HTML del sitio web consultado (Gonzalez Peña et al., 2014).

Twitter Streaming API2

Es una herramienta que entre sus variadas prestaciones se encuentra la recolección de datos mediante la captura de tweets. Esta API proporciona facilidad de captura bajo código programado en lenguaje Python (Kabir & Madria, 2020). Otras características importantes que posee para realizar la preparación de la data para su posterior análisis son (Twitter Inc, n.d.):

- Aplicar filtros mediante operadores para segmentar la muestra capturada.
- Identificación de las interacciones sobre los tweets publicados.
- Realizar seguimiento de conversaciones basadas en las respuestas de un tweet específico. Este comportamiento en la red social es denominado “Hilo”.

Firehose de Twitter

Consiste en tener acceso a la Base de Datos de Twitter donde se almacenan los tweets, esta es una de las mejores formas de acceder a la información de la red social, sin embargo, para utilizar esta herramienta es necesario realizar un pago económico.

Estado del Arte

Desarrollar una metodología que permita construir un modelo para la extracción, análisis y procesamiento de información de una red social o realizar un análisis de sentimientos, implica tener conocimiento de las diferentes técnicas de minería de texto, por tanto, a continuación, se describe los aportes más relevantes de autores sobre este tema, es importante tener una base que nos permita abordar de la mejor forma este tema:

Según artículo "Computers in Human Behavior - Selection criteria for text mining approaches" de H. Hashimi, A. Hafez y H Mathkour publicado en la revista ELSEVIER en el año 2015, los autores proporcionan una aproximación sobre las técnicas más utilizadas en la minería de texto, de cómo pueden ser utilizadas para encontrar información relevante de nuestro interés, las herramientas de extracción de texto que permiten identificar el tema principal de un documento (Mathkour et al., 2015). De igual manera presentan una aproximación a las diferentes áreas de aplicación de la minería de texto y finalmente proponen una serie de criterios que son utilizados para determinar la efectividad de una u otra de las técnicas que son más empleadas para el análisis de texto aplicadas a la minería de texto. Esta investigación permite tener una guía de selección de la técnica adecuada para su aplicación de acuerdo con el problema de análisis de información. El tema central será "Contenidos emocionales de textos en redes sociales online" (Mathkour et al., 2015), que aborda el análisis y estudio de los patrones de comportamiento de los usuarios de la red social Twitter.

Por otra parte, teniendo en cuenta la temática anteriormente expuesta se encontró que mediante la aplicación de text mining se realizó un estudio sobre la "terapia de acupuntura en la hipertensión en la medicina en China", ya que esta modalidad terapéutica es muy importante en la

medicina tradicional en el país asiático (Bian et al., 2014). El estudio consistió en “extraer un conjunto de datos de la literatura de SinoMed”, se consultó la aparición de la palabra “Hipertensión” en el idioma chino en la literatura indicada, el método usado para este análisis fue dividido en varias fases: recolección, pretratamiento y visualización de los datos.

En el estudio se obtuvo un conjunto de datos, los cuales fueron guardados en un archivo .txt codificado en ANSI, luego estos datos fueron almacenados en una base de datos para analizarlos y contar la palabra clave repetida de cada documento solo una vez mediante un software de visualización lograron mostrar en un diagrama de redes los puntos de acupuntura más usados utilizados en el tratamiento de la hipertensión (Bian et al., 2014).

En la misma línea de la Salud, se encontró un estudio que se realizó sobre la aplicación del tratamiento de acupuntura en enfermedades de las arterias coronarias (Bian et al., 2014). Mediante un estudio de minería de datos que fue realizado en China el cual consistió en el análisis de texto sobre la literatura de los diferentes estudios aplicados a la acupuntura y una revisión de los documentos, se extrajo información sobre los meridianos, puntos de acupuntura, así como los métodos más efectivos que constantemente se utilizan para tratar enfermedades del corazón, siendo la acupuntura una de las técnicas terapéuticas más usadas en China.

El método usado para este estudio consistió en la recopilación de los datos del portal médico SinoMed en el cual se consultó el término “enfermedad coronaria” obteniendo la información concerniente a este estudio, posteriormente se prepararon los datos para luego procesarlos, analizarlos y finalmente presentar los resultados. Este estudio demostró ser útil en la investigación clínica como también para la práctica médica del tratamiento de esta enfermedad (Bian et al., 2014).

Otro estudio donde se utilizó minería de texto fue en Italia, consistió en analizar la información publicada en la prensa popular italiana sobre la percepción de la sociedad sobre el SIDA, se pretendía analizar los principales temas publicados por la prensa y analizar las tendencias que se publicaban sobre el SIDA durante las últimas décadas. El estudio consistió en analizar los artículos publicados por dos de los principales periódicos de Italia "entre 1985 y 1990 y entre 2005 y 2010", se utilizó una muestra equivalente a 446 artículos de los periódicos sobre el tema de estudio, se realizó un análisis de contenido asistido por computador que permitió identificar cinco temas que más se publicaron en los artículos periodísticos, atención médica, apoyo familiar, debate sobre ciencia y religión, exclusión social y políticas sanitarias. Este análisis logró determinar los temas más relevantes en la población que debían ser fortalecidos para plantear una solución a la problemática de salud relacionada con el SIDA (Caputo et al., 2016).

Recientemente se han realizado análisis de sentimientos (Nielsen, 2011), basados en los contenidos de las redes sociales específicamente Twitter, ya que es una de las plataformas donde se comparte una mayor parte de contenido significativo. A continuación, se presentan los estudios más recientes sobre este tema.

Uno de los estudios más recientes acerca del COVID-19, pretendía realizar un análisis de sentimientos mediante la clasificación de tweets en puntuación positivas, negativas y neutrales, para demostrar cómo la popularidad de los comentarios en la red social está afectando la precisión y veracidad de la información publicada por las autoridades mundiales de la salud (Chakraborty et al., 2020). El estudio consistió en extraer los tweets que contienen la palabra COVID-19 y OMS, para analizar dicha información y determinar si esta información es relevante y veraz para ayudar a guiar a las personas de edad avanzada que se encontraban encerradas. En una primera fase se extrajeron los tweets desde el 1 de enero hasta el 23 de marzo de 2020 donde se determinó que la

información publicada en los comentarios refleja un sentimiento neutral o negativo. El análisis de los datos recopilados entre diciembre de 2019 y mayo de 2020 logró determinar cuál fue el número de tweets positivos y neutrales que se publicaron.

Sin embargo, el estudio demostró que, si bien la mayor parte de la información es positiva y veras, se evidenció que se compartió información irreal que no tiene fundamento científico y carece de toda verdad ya que una gran cantidad de usuarios se dedicó a retuitear los comentarios negativos, lo que masifica a la desinformación.

Por ende, esto puede afectar la persecución que se tiene del nuevo Covid 19, si bien es cierto que este nuevo virus obligó a guardar cuarentena y estar en distanciamiento por largo tiempo causando en muchas personas sentimientos de desesperación y angustia. La desinformación producida por comentarios negativos en las redes sociales contribuyó a aumentar el estrés y generar todo tipo de pánico en la población (Kabir & Madria, 2020).

En este mismo sentido los autores Y. Kabir y S. Madria, crearon una aplicación web interactiva, que permite extraer datos de los tweets sobre el COVID-19, en tiempo real de las publicaciones de Twitter, analizando más de 200 millones de tweets en Estados Unidos, permitiendo analizar durante un periodo de tiempo los temas relacionados con el COVID-19, para desarrollar esta aplicación utilizaron Twitter Streaming API2[1] y la librería Tweepy3[2] de Python. Realizaron un análisis de sentimientos que permitió comprender mejor las emociones humanas, los autores compartieron públicamente los datos procesados para que los investigadores utilizarán estos datos ya procesados y los analizaran, para así contribuir su lucha contra el COVID-19 (Kabir & Madria, 2020).

En el estudio denominado “Análisis de sentimiento y sus aplicaciones en la lucha contra el COVID-19 y las enfermedades infecciosas: una revisión sistemática” (Alamoodi et al., 2021). Los autores realizaron un análisis de sentimientos que fue enfocado en descubrir la literatura sobre las enfermedades infecciosas de los últimos diez (10) años entre enero de 2010 y junio de 2020. Esta investigación fue motivada por la reciente propagación masiva del COVID-19.

En la investigación se describen las fases y métodos utilizados para el desarrollo de esta. Este trabajo sirve de referencia para entender mejor en qué consiste el análisis de sentimientos ya que detalla cómo se debe realizar, las fases que se deben tener en cuenta para identificar y dar uso de la mejor herramienta en la extracción de información de una red social. Como resultado final se presentaron los patrones de comportamiento más significativos en las personas frente a estas enfermedades.

Según la investigación, los autores hacen referencias del modo en que los estudios realizados demuestran que el uso de aplicaciones de redes sociales logra influenciar la forma en que las personas se comportan, ya que dedican una gran cantidad de tiempo al uso de estas aplicaciones para compartir todo tipo de contenido, así mismo afirman que “debido a la gran cantidad de información que se encuentra en una red social, estas plataformas de redes se consideran el centro global del big data”.

Otro estudio significativo denominado “Twitter e Investigación: Un Resumen Sistemático de Literatura A Través de la Minería de Textos.” (KARAMI et al., 2020), se esforzó en identificar los principales temas de investigación basados en Twitter, de los años comprendidos desde el 2016 al 2019, permitiendo caracterizar la literatura de los temas más relevantes. Se recopilaron artículos de mayor importancia de tres bases de datos (Web of Science (WOS)[3], EBSCO[4], y IEEE[5]).

Analizando los artículos que contenían la palabra twitter, se logró realizar la clasificación de los principales artículos permitiendo establecer el nivel de relevancia que han tenido los diferentes temas de investigación basados en twitter.

Impacto

El presente proyecto busca presentar un análisis de sentimiento de los comentarios en los tweets sobre el tema de la vacunación del COVID-19 en Colombia que permita establecer patrones de comportamiento que pueden influenciar de forma negativa, positiva o neutral frente a la vacunación del COVID-19. Dicho análisis permitirá a cualquier ente gubernamental y/o privado tener una guía sobre la aceptación ciudadana, y le permita al gobierno establecer políticas necesarias para el mejoramiento del plan de vacunación que permita la inmunidad de rebaño de la población colombiana.

Componente de innovación

Si bien la existencia de metodologías de minería de datos es ampliamente conocidas y comúnmente aplicadas. Resulta importante presentar un buen uso de estas técnicas enfocadas a temáticas de interés público, analizando problemáticas actuales para presentar resultados relacionadas con el COVID-19, puntualmente con la percepción de la ciudadanía colombiana con respecto a la vacunación teniendo en cuenta su impacto científico, educativo, político y en la salud pública.

Metodología

De acuerdo con el planteamiento del problema y los objetivos del proyecto. La metodología aplicada para el desarrollo de este proyecto está basada en el artículo “Análisis de sentimiento y sus aplicaciones en la lucha contra el COVID-19 y las infecciones enfermedades: una revisión

sistemática" (Alamoodi et al., 2021) previamente descrito en el estado del arte del presente documento y el cual es guía para la ejecución de la propuesta presentada.

Esta metodología permite abordar el análisis de sentimientos desde la perspectiva de la interpretación del comportamiento social de acuerdo con las relaciones existentes entre las opiniones publicadas en las redes sociales y la interpretación de texto. Como finalidad se busca establecer una clasificación de tweets sobre la vacunación en Colombia ya sea negativo, positivo o neutral. De acuerdo a esto para desarrollar el análisis de sentimientos es necesario establecer una variable que de acuerdo a su valor exprese de forma cuantitativa cuál es el sentimiento de un tweet, siendo este: positivo, negativo o neutral teniendo valores que se pueden expresar de la siguiente forma, por ejemplo: 0, 1 y -1.

Es decir, a partir de la obtención de tweets con textos que contienen opiniones cualitativas, plantear los criterios necesarios para ejecutar la transformación de los datos de un ámbito cualitativo a cuantitativo.

En la figura 1 se ofrece una representación visual de cada una de las etapas que conforman la metodología de desarrollo del proyecto con su respectiva descripción.

Figura 1. Metodología de desarrollo del proyecto. Diseño: Elaboración propia. Fuente: Sentiment analysis and its applications in fighting COVID-19 and infectious diseases: A systematic review. Iconografía: Canva.

Etapas de la metodología

Como se pudo observar en la figura 1, la metodología implementada comprende una serie de etapas donde es necesario ejecutar cada una en el orden presentado en dicho diagrama. Para comprender mejor el trabajo en cada una de las etapas. En el siguiente apartado se describe de manera general el trabajo realizado en cada una de ellas, en la continuación del documento y la ejecución del proyecto se detallarán cada una de las tareas realizadas con el fin de presentar los resultados obtenidos de la investigación.

Identificación de datos

En esta primera parte de la metodología se identificarán los tweets que tengan palabras claves de la vacunación en Colombia. En ese sentido se tendrán en cuenta también los hashtags relacionados como #Covid19, #vacunacion, #Colombia, #Yomevacuno, #Yonomevacuno entre otros. Con el fin de segmentar la temática de los tweets a extraer se tendrán en cuenta los tweets relacionados con el Covid 19 y la vacunación ya que la información generada en la red social respecto a la pandemia es muy grande.

Extracción de datos

Continuando con la metodología, la extracción de datos se realiza mediante el uso de la API pública de desarrollo de Twitter en compañía del uso de código en Python haciendo llamado a las respectivas librerías y las llaves de conexión proporcionadas por la misma API de la red social. El periodo de tiempo para la extracción de datos fue desde el 15 de marzo y el 25 de abril del año 2021.

Preprocesamiento

En esta etapa se obtuvo la información almacenada en las bases de datos MongoDB, se aplicó una primera limpieza general de los datos dado que la información que se analizó estaba en formato de texto, por lo tanto, se realizó una limpieza inicial de los mismos. Se crearon dos archivos .csv, uno con la información que se extrajo de las bases de datos y otro con un filtro específico de la información que fue objeto de análisis. Se estructuro la información ya que fue necesario realizar la clasificación manual para cada uno de los tweets, seleccionando las columnas de interés que se consideraron necesarias para el análisis.

Procesamiento

Se realizaron trabajos con el fin de establecer las frecuencias y los patrones que indicaron una posición positiva, neutral o negativa de cada uno de los Tweets obtenidos. En esta etapa se

realizó una limpieza profunda de los datos. Se aplicó un modelo estadístico con el cual se clasificaron los tweets de acuerdo con el sentimiento expresado entre: positivo, negativo o neutral. Así mismo se entrenaron los modelos utilizando el método de aprendizaje supervisado e implementando los algoritmos de machine learning Logistic Regression, Gaussian Naive Bayes (GaussianNB), SVM (Support Vector Machines), Random Forest Classifier y Decision Tree Classifier. Para evaluar los modelos mencionados se estableció una base de entrenamiento del 80% de los datos clasificados y un 20% de pruebas para establecer la efectividad de la predicción en los modelos.

Resultados

Finalmente, la presentación de los resultados se entrega un informe acorde al análisis realizado con el fin de presentar las tendencias, producto del procesamiento de los datos de la opinión ciudadana frente a la vacunación del Covid 19.

De acuerdo con la metodología presentada en el anterior apartado, en la figura 2 se puede apreciar el diagrama de arquitectura propuesto para el desarrollo de la aplicación, esta arquitectura es la necesaria para realizar el análisis de sentimientos de la temática abordada.

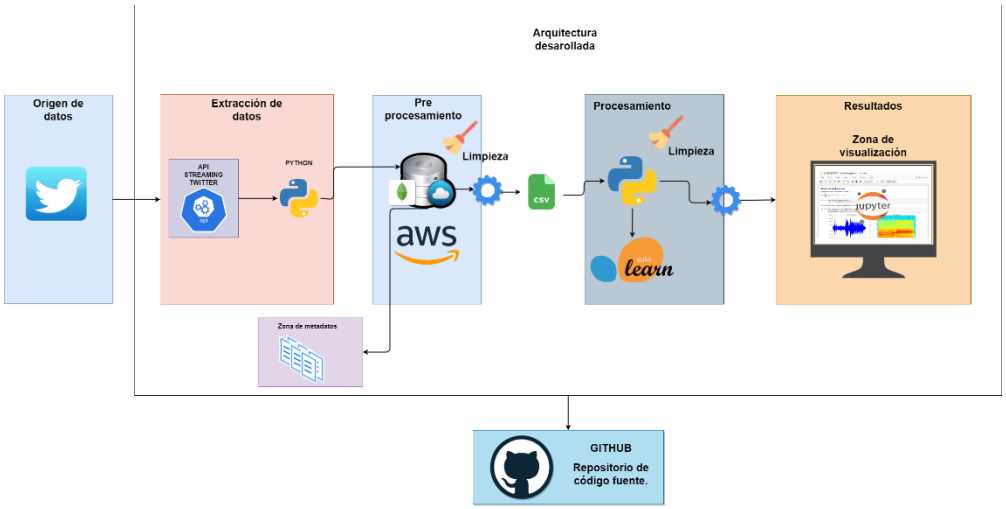


Figura 2. Diagrama de arquitectura de la solución propuesta en el proyecto de aplicación.

Desarrollo de la propuesta

Configuración del ambiente de almacenamiento

Como primera actividad antes de realizar la captura de Tweets se crea y configura el ambiente contenedor y administrador de los datos a almacenar.

Se crea un servidor en la nube de MongoDB Atlas con el fin de crear la base de datos para el almacenamiento de los Tweets. El ambiente se configura con las siguientes características:

- Cluster Free. Este servidor a pesar de ser gratuito tiene las características necesarias para el desarrollo del proyecto y su aplicación.
- Servicio de proveedor AWS ubicado en Virginia, Estados Unidos.
- 512 MB de almacenamiento para los documentos dentro de la base de datos.
- Nombre personalizado para el actual proyecto.

Para revisar los detalles de la configuración del ambiente de almacenamiento favor consultar el "*Anexo 1 Creación base de datos MongoDB en la nube*".

En la figura 3 se presenta la configuración general del cluster con la base de datos en MongoDB alojada en la nube de AWS (Amazon Web Services). Entre otras características se puede observar el nombre asignado al servidor, la ubicación física del servidor y el tráfico generado durante la creación de esta.

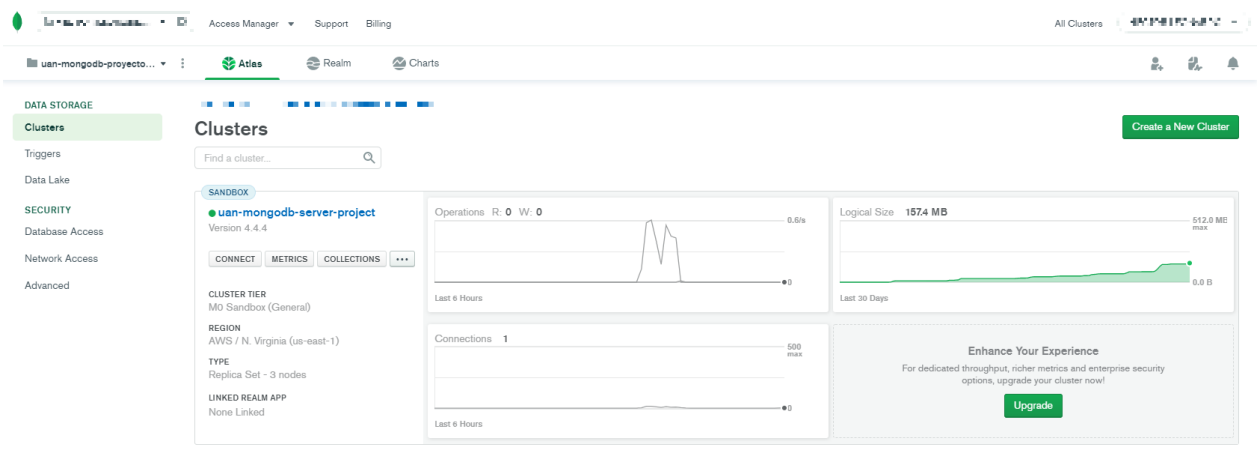


Figura 3. Servidor de almacenamiento en la nube de la base de datos MongoDB destinada al desarrollo del proyecto.

Entre los parámetros de acceso configurados al servidor se encuentran:

- Configuración del usuario administrador y de acceso a la base de datos.
- Configuración de acceso desde la web, solo las direcciones IP configuradas y con la cadena de conexión respectiva pueden acceder al servidor.

Repositorio y administración de versiones

El versionamiento de código fuente y el manejo de branches se realizó mediante la plataforma de GitHub. Se creó un proyecto denominado **uan-gb-final-project-repository**, en el cual se administró el código y la documentación como se puede apreciar en la tabla 1:

Tabla 1. Branch creados en el desarrollo del proyecto.

Branch	Especificación
master	Este branch contiene la versión definitiva del código fuente que se desarrolló para el proyecto.
desarrollo	Este branch contiene la versión de desarrollo en la cual también se realizaron pruebas para verificar el funcionamiento correcto del código.

documentación	Branch dedicado a la documentación de todos los productos documentales creados en el desarrollo del proyecto. Comprende el presente documento y los anexos asociados al mismo.

Tabla con la descripción de cada una de los branch creados para el proyecto de aplicación.

En la figura 4 se muestra el proyecto creado en GitHub, la una descripción de este y los branches descritos en la tabla 1. Los cambios de código, así como de estructura del proyecto se fue persistiendo en el Branch de desarrollo, en el Branch master se publicó la versión definitiva y en el de documentación al igual que en desarrollo se fueron publicando las versiones más recientes de los archivos generados.

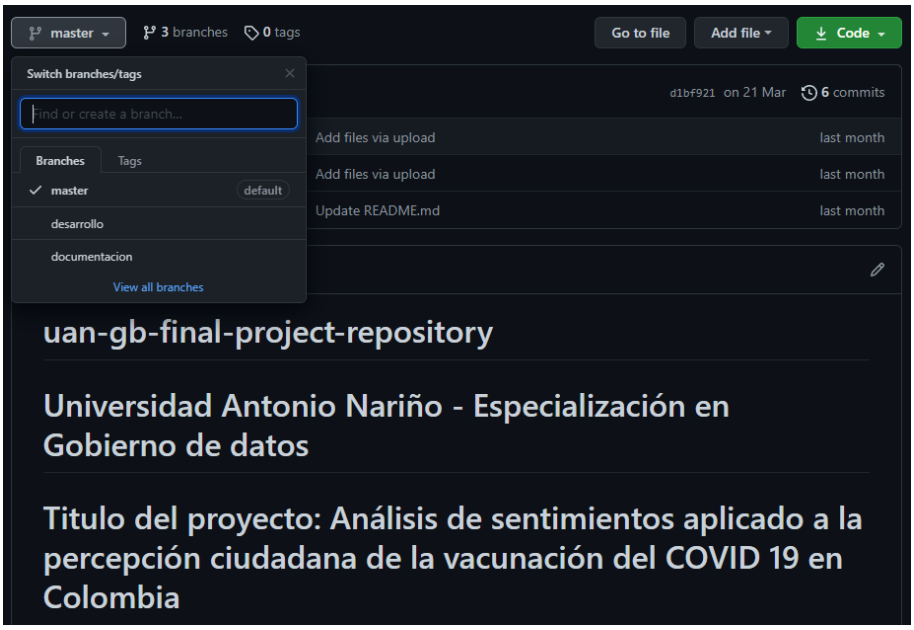


Figura 4. Versionamiento y administración del proyecto en GitHub.

Para obtener más detalle de la administración de versionamiento de código y la documentación del proyecto consultar el *"Anexo 4 - Versionamiento, administración y repositorio de código fuente"*.

Estructura del Proyecto en notebooks

En el desarrollo del proyecto se utilizaron las herramientas de Jupyter. En esta etapa se creó un proyecto en Jupyter–Anaconda3 con el fin de crear notebooks para su desarrollo e implementar el código en Python para la extracción, preprocesamiento, procesamiento y presentación de resultados producto del análisis de los datos. En la figura 5 se aprecia la estructura del proyecto descrita anteriormente.

Figura 5. Estructura del proyecto en Jupyter.

Identificación de datos - fuente de información

Para desarrollar la investigación que logre determinar la percepción que tienen las personas sobre el proceso de vacunación contra el COVID-19 en Colombia, a través de análisis de sentimientos, se tomó como fuente de información los tweets publicados en la red social Twitter, estableciendo como periodo de captura de datos desde marzo de 2021 hasta abril de 2021.

Extracción de datos

Utilizar Twitter como fuente de información se fundamenta principalmente en que en esta red social la mayoría de los contenidos que comparten o publican los usuarios son públicos. Existen

varias técnicas para la recolección de información de redes sociales o sitios web, dando uso estas fueron las utilizadas en el presente proyecto:

API Twitter: Esta herramienta proporcionada por la red social de forma gratuita, fue utilizada para la extracción de datos. Su invocación y uso se llevó a cabo mediante la implementación de código en Python. De esta manera se definieron las palabras claves para identificar los tweets válidos y realizar el posterior almacenamiento en la base de datos (Martinez, 2016).

Para el desarrollo de este trabajo se utilizó la API gratuita la cual proporciona Twitter, del mismo modo se desarrolló el código en Python para utilizar este API haciendo referencia a la librería Tweepy. Otra librería importante que se utilizó en el código fue pymongo, esta permite interpretar el código desarrollado para la conexión a la base de datos de MongoDB.

A través del portal para desarrolladores de la red social Twitter proporciona dos tipos de APIs, el API REST y el API STREAMING, para poder usar cualquier de estas dos funcionalidades se debe solicitar la creación de una cuenta de desarrollador, una vez aprobada la solicitud y se puede hacer uso de la herramienta para la creación de una aplicación con el fin de usar la API para obtener información de la red social.

El API STREAMING, esta funcionalidad se ejecutó en el código fuente del notebook "**1. Extracción y almacenamiento. ipynb"** del proyecto de aplicación. Se realizó captura de datos en tiempo real desde Twitter (Twitter Inc, 2021).

Por lo tanto, con el fin de garantizar el almacenamiento de la mayor cantidad de datos, con un volumen representativo, la captura en tiempo real y con histórico de 7 días fue crucial.

Para hacer uso de las dos APIs mencionadas anteriormente, es necesario hacer la solicitud a la red social para obtener las Keys y los Tokens. Estas credenciales permitirán establecer la conexión entre el código de programación bajo lenguaje Python y el flujo en Streaming de la red social.

Básicamente la API proporciona dos funcionalidades:

Obtención de las keys y tokens

Para la obtención de estas claves y el uso de la API es necesario seguir lo siguientes pasos:

1. Crear una cuenta en Twitter.
2. Obtener la cuenta de desarrollador en Twitter.
3. Llenar los formularios respectivos y justificar su uso. En este caso, la finalidad de la investigación académica.
4. Crear un proyecto y una aplicación en Twitter.
5. Consultar las keys y los tokens para su posterior uso en el código desarrollado en Python.

Para consultar el detalle del proceso de obtención de keys y tokens del API de Twitter consultar el "*Anexo 2 - Obtención Keys y Tokens de Twitter*".

Captura de datos de Twitter en Python

El lenguaje seleccionado para realizar la captura de los Tweets fue Python. Para dicha captura se creó el notebook "**1. Extracción y almacenamiento. ipynb"** con las siguientes funciones:

1. getAayth(): Función donde se definieron los atributos de conexión al API de Twitter con las keys y tokens correspondientes. Su ejecución retorna la cadena de conexión a la red social.
2. connMongoDb(): Esta función retorna la cadena de conexión correspondiente al servidor y la base de datos MongoDB en donde se almacenaron los tweets. En este caso fueron utilizados tres servidores en la nube.

Para más detalle respecto a la construcción del código fuente para la captura de datos en Python consultar el "*Anexo 3 Extracción de datos*".

En el proceso de captura de datos de los tweets se utilizó la API proporcionada por la red social la cual devuelve la información en formato JSON, se almacenó en una base de datos MongoDB ya que este tipo de bases de datos permite el almacenamiento de documentos.

En MongoDB, se creó la base de datos llamada **twitterdb** adicionalmente en esta base de datos se creó la colección llamada **tweetsCovid**, en esta colección se almacenaron los documentos tipo JSON que corresponden a los tweets recopilados mediante la API de Twitter. Se utilizaron 3 servidores con capacidad de almacenamiento de 512 MB cada uno como lo muestra la figura 6.

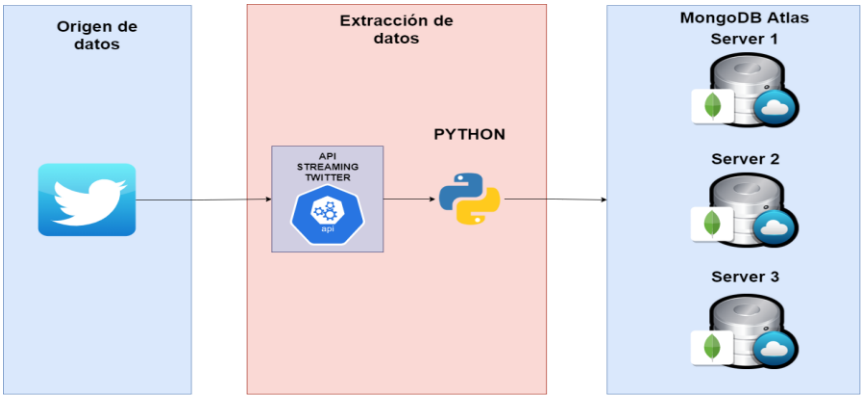


Figura 6. Ingesta de datos (Arquitectura).

Se creó un arreglo de palabras que fue usado por la API para filtrar información relacionada con la vacunación y el covid-19, las siguientes palabras fueron usadas para realizar el filtro de información:

['vacuna','covid','coronavirus']

Los datos fueron recopilados diariamente durante un periodo de tiempo de 2 meses, se exceptuaron algunos días donde no se recolectaron datos, esto debido a que la API tiene algunas limitaciones para el filtrado de información se capturó tweets de todo el mundo que incluyeran las palabras antes indicadas.

El proceso de captura de datos se describe a detalle en el anexo Nro. Anexo 3 - Extracción de datos.

Descripción de los Datos

Como se ha indicado anteriormente el API de Twitter retorna información de los tweets en formato JSON, el cual es muy sencillo de entender y manipular algunos de los atributos que se tuvieron en cuenta para realizar el análisis de la información:

Diccionario de datos de tweets

En la tabla 2 descrita a continuación, se muestra el diccionario de datos para los atributos de primer nivel (objetos primarios) que se tuvieron en cuenta para el análisis de información (Twitter Inc, 2021).

Tabla 2. Diccionario de datos de los Tweets capturados para el análisis de sentimientos.

Atributo	Tipo	Descripción
created_at	String	Hora UTC cuando se creó este Tweet. Ejemplo: *"created_at": "Wed Oct 10 20:19:24 +0000 2018"*
id_str	String	Representación de cadena del identificador único de este Tweet. Las implementaciones deberían usar esto en lugar del entero grande en id. Ejemplo:

		"id_str":"1050118621198921728"
text	String	El texto UTF-8 real de la actualización de estado. Consulte twitter-text para obtener detalles sobre qué caracteres se consideran válidos actualmente. Ejemplo: *"text":"To make room for more expression, we will now count all emojis as equal—including those with gender and skin t… https://t.co/MkGjXf9aXm"*
truncated	Boolean	Indica si el valor del parámetro de texto se truncó, por ejemplo, como resultado de un retweet que excede el límite de longitud del texto del Tweet original de 140 caracteres. El texto truncado terminará en puntos suspensivos, como este ... Dado que Twitter ahora rechaza los Tweets largos en lugar de truncarlos, la gran mayoría de los Tweets lo tendrán configurado como falso. Tenga en cuenta que, si bien los retweets nativos pueden tener su propiedad de texto de nivel superior acortada, el texto original estará disponible en el objeto retweeted_status y el parámetro truncado se establecerá en el valor del estado original (en la mayoría de los casos, falso). Ejemplo: *"truncated":true*

user	User object	El usuario que publicó este Tweet. Consulte el Diccionario de datos de usuario para obtener una lista completa de atributos. Ejemplo resaltando atributos seleccionados: *{ "user": {* *"id": 6253282,* *"id_str": "6253282",* *"name": "Twitter API",* *"screen_name": "TwitterAPI",* *"location": "San Francisco, CA",* *"url": "https://developer.twitter.com",* *"description": "The Real Twitter API. Tweets about API changes, service issues and our Developer Platform. Don't get an answer? It's on my website.", ...*
retweeted_status	Tweet	Los usuarios pueden amplificar la transmisión de Tweets creados por otros usuarios retwitteando. Los retweets se pueden distinguir de los tweets típicos por la existencia de un atributo **retweeted_status**. Este atributo contiene una representación del Tweet original que se retuiteó. Tenga en cuenta que los retweets de retweets no muestran representaciones del retweet intermediario, sino solo el Tweet original. (Los usuarios también pueden cancelar un retweet que crearon eliminando su retweet).

Fuente: (Twitter Inc, 2021).

Preprocesamiento de datos

Una vez finalizado el proceso de extracción y almacenamiento de la información de los tweets, se realizó la etapa de preprocesamiento. Se creó un nuevo notebook denominado **2. Pre-Procesamiento.ipynb**.

De las tareas más importantes que se ejecutó en la etapa de preprocesamiento fue la limpieza de los datos. Teniendo en cuenta que la data obtenida a través de los Tweets no cuenta con ningún filtro y el texto es tal cual fue escrito por cada usuario, existen muchas palabras como conectores gramaticales y otras que no cuentan con un valor representativo para el estudio y la evaluación de positividad, negatividad y neutralidad para el análisis de sentimientos.

Limpieza general de datos

El proceso de limpieza de datos consistió técnicamente en tomar el campo **text** o **full_text**, de la estructura JSON en donde se guarda el texto que un usuario publica en un tweet, en esta limpieza general se quitaron caracteres especiales, emoticones y demás símbolos ya que el análisis fue exclusivo sobre las palabras contenidas en los tweets. Se convirtió el texto en minúscula y se quitaron las referencias a otras cuentas de tweets que por lo general se agregan al realizar una publicación.

Para realizar este proceso se creó un notebook en Python llamado **2. Pre-Proceamiento.ipynb**. Se implementó una función llamada connMongoDb(), que se encarga de retornar la cadena de conexión de la base de datos MongoDB, ya que para el almacenamiento de información se utilizaron tres servidores diferentes en la nube como se documentó anteriormente.

Para obtener la información de la base de datos de los tweets almacenados se realizaron dos filtros iniciales. El primer filtro que se utilizó consistió en validar la existencia de la etiqueta **text** en los documentos json, mientras que el segundo filtro que se aplicó fue sobre la llave **user.location** en la cual se encontrará la palabra **Colom**, con el fin de analizar los tweets que se originaron o retuitearon en Colombia.

Se extrajeron las siguientes columnas de los documentos almacenados en la Base de Datos, con el fin de almacenar los datos en un dataframe por servidor. A continuación, se creó un

dataframe con los datos consolidados de los tres anteriormente descritos. La estructura del dataframe es la descrita en la tabla 3.

Tabla 3. Estructura del dataframe con los datos unificados de los orígenes de datos.

Nombre JSON - en Tweets	Nombre en DataFrame
["created_at"]	FechaCreacion
["id_str"]	Id_tweest
["truncated"]	Truncado
["user"]["id_str"]	Id_Usuario
["user"]["name"]	Nombre_Usuario
["user"]["location"]	Ubicacion
["text"]	Texto
["quoted_status"]["created_at"]	Q_FechaCreacion
["quoted_status"]["id_str"]	Q_Id_tweest
["quoted_status"]["truncated"]	Q_Truncado
["quoted_status"]["user"]['id_str']	Q_Id_Usuario
["quoted_status"]["user"]['name']	Q_Nombre_Usuario
["quoted_status"]["user"]['location']	Q_Ubicacion
["quoted_status"]["extended_tweet"]["full_text"]	Q_Texto
["retweeted_status"]["created_at"]	R_FechaCreacion
["retweeted_status"]["id_str"]	R_Id_tweest
["retweeted_status"]["truncated"]	R_Truncado
["retweeted_status"]["user"]['id_str']	R_Id_Usuario
["retweeted_status"]["user"]['name']	R_Nombre_Usuario
["retweeted_status"]["user"]['location']	R_Ubicacion
["retweeted_status"]["extended_tweet"]["full_text"]	R_Texto

Fuente: Propia.

Del mismo modo se desarrolló la función clean(), para quitar los saltos de línea y los emoticones de los tweets. Esta acción se realizó para algunas columnas del dataframe como: nombre usuario, ubicación y texto.

La función remove_url(), se usó para hacer una primera limpieza general eliminando las referencias a urls de los tweets y a los nombres de usuario cuando se hace referencia a un retweet que corresponde una publicación de otro usuario ejemplo: @MrDoctorOficial.

Seguidamente se exportó la información a un archivo CSV llamado **“Base_Original_1.csv”**. Los campos seleccionados fueron los informados anteriormente, este archivo se creó con el fin de realizar ejercicios de prueba con el procesamiento de los datos.

Finalmente, se creó un nuevo Dataframe en el cual se seleccionaron las columnas que se tuvieron en cuenta para el proceso de análisis de sentimientos: **'FechaCreacion', 'Id_tweest', 'Ubicacion', 'Texto', 'R_Texto'**. Luego en este mismo dataframe a las columnas **'Texto'** y **'R_Texto'** se les aplicó la función remove_url(), seguidamente en la columna **'Texto'** se eliminaron las filas en donde su longitud fuera menor o igual a uno. Luego se eliminaron los registros duplicados.

Se agregó una nueva columna al dataframe llamada **‘Tipo’** para clasificar los tweets y los retweets. Esta acción nació a partir de la revisión del inicio de los caracteres en los valores de la columna **'Texto'**, donde los textos que iniciaron con la palabra **’RT’** se clasificaron como ‘**Retweet’** mientras que los que no como ‘Tweet’. Esta columna se tomó para actualizar el contenido de la columna **'R_Texto'** de la siguiente manera: cuando la clasificación fue ‘Retweet’ se conservó el valor de la columna **'R_Texto'** en caso contrario se actualizó con el valor de la columna **‘Texto’**. La tarea anterior se realizó para consolidar los textos tanto de tweets como retweets en una sola columna ya que los registros de los tweets tenían valor vacío en la columna **'R_Texto',** columna esencial para los análisis finales. Seguidamente se descartaron los tweets que en el texto no contenían la palabra ‘vacun’ y se adicionó la columna index para reconstruir los índices del dataframe con datos consolidados.

Este set de datos se utilizó para el resto del proceso de análisis de sentimientos. La figura 7 visualiza su estructura:

	Index	FechaCreacion	Id_tweest	Ubicacion	Texto	R_Texto	Tipo
0	0	2021-03-15 00:15:08+00:00	1371253596411588620	Colombia	RT ¿CÓMO ME RECUPERÉ DE #COVID19?¡3 MESES DESP...	¿CÓMO ME RECUPERÉ DE #COVID19? ¡3 MESES DESPUÉS...	Retweet
1	1	2021-03-15 00:15:09+00:00	1371253598638768128	Bogotá, Colombia	RT Respetado si hay ajustes en las cifras de a...	Respetado si hay ajustes en las cifras de ayer...	Retweet
2	2	2021-03-15 00:15:28+00:00	1371253678330494979	Bogotá - Colombia	WEBINAR EXCLUSIVO PARA VENDEDORES DE #CAFÉ TOS...	WEBINAR EXCLUSIVO PARA VENDEDORES DE #CAFÉ TOS...	Tweet
3	3	2021-03-15 00:15:44+00:00	1371253746903175169	Colombia	RT Los test serológicos miden la presencia d l...	Los test serológicos miden la presencia d los ...	Retweet
4	4	2021-03-15 01:53:20+00:00	1371278308487348228	Bogotá, Colombia	RT Los científicos del momento Son pareja en l...	Los científicos del momento Son pareja en la v...	Retweet
...	...	...	...	...	...	...	...
2796	2796	2021-04-25 02:12:46+00:00	1386141103238299648	Barranquilla, Colombia	Una vacunación sin agendamiento es lo ideal, d...	Una vacunación sin agendamiento es lo ideal, d...	Tweet
2797	2797	2021-04-25	1386389601980346368	Colombia	Bien por el saludo... Favor no descuidar el	Bien por el saludo... Favor no descuidar el	Tweet

Figura 7. Dataframe con los datos específicos para el análisis de sentimientos.

Para obtener mayores detalles del proceso consultar el anexo *'Anexo 5 - Preprocesamiento'*.

Este dataframe sirvió de insumo para crear el archivo **'Base_original.csv'** que se utilizó en las siguientes fases.

Clasificación

Para realizar el análisis de sentimientos se aplicó el tipo de aprendizaje supervisado, por lo tanto, fue necesario utilizar el archivo anterior al cual se le agregó la columna **'Sentiment'** para clasificar manualmente cada uno de los tweets. Dicha clasificación consistió en agregar el valor con el cual se consideró que el tweet expresó un sentimiento negativo, positivo o neutral asignando la etiqueta -1, 1 y 0 respectivamente. En la figura 8 se puede observar la estructura y la data del dataframe creado.

	Index	FechaCreacion	Id_tweest	Ubicacion	Texto	R_Texto	Tipo	Sentiment
0	0	2021-03-15 01:57:03+00:00	1.37128e+18	Bogota Colombia	RT [Video] Engañan a otra abuela y no le aplic...	[Video] Engañan a otra abuela y no le aplican ...	Retweet	-1
1	1	2021-03-15 01:57:18+00:00	1.37128e+18	Barranquilla, Colombia	RT y la vacuna pa cuando? @	y la vacuna pa cuando?	Retweet	-1
2	2	2021-03-15 01:58:38+00:00	1.37128e+18	colombia, Córdoba-Monteria	RT Como Gobernador solícito q revisen bien las...	Como Gobernador solícito q revisen bien las ci...	Retweet	0
3	3	2021-03-15 01:58:38+00:00	1.37128e+18	Valledupar, Cesar, Colombia	RT Los que andaban criticando a por la vacunac...	Los que andaban criticando a por la vacunación...	Retweet	-1
4	4	2021-03-15 01:58:55+00:00	1.37128e+18	Bogotá, D.C., Colombia	RT ATENCIÓN : Irlanda recomienda suspender por...	ATENCIÓN : Irlanda recomienda suspender por "p...	Retweet	-1
...	...	...	...	...	...	...	...	...
1500	1500	2021-04-25 02:12:37+00:00	1.38614e+18	Colombia	Busca en la página del ministerio los puntos d...	Busca en la página del ministerio los puntos d...	Tweet	1
1501	1501	2021-04-25 02:12:46+00:00	1.38614e+18	Barranquilla, Colombia	Una vacunación sin agendamiento es lo ideal, d...	Una vacunación sin agendamiento es lo ideal, d...	Tweet	1

Figura 8. Dataframe con la clasificación de sentimientos de los tweets.

Procesamiento de datos

Se creó el notebook "**3. Procesamiento.ipynb**" donde se hizo una lectura del archivo csv para la creación de un dataframe donde se aplicó una limpieza profunda sobre los valores de la columna **'R_Texto'** como se describe a continuación.

Se aplicó la función llamada **clean_tweets**, la cual recibió como parámetro el valor de la columna **'R_Texto'** y retorno el texto limpio. Esta limpieza consistió en la eliminación de números, caracteres especiales y dobles espacios. Para desarrollar esta función se usó la librería ***re*** de Python la cual permite manipular expresiones regulares (Python Software Foundation, 2009).

Después de aplicar la limpieza de los datos en los textos se procedió a remover las letras repetidas y se reemplazaron de las letras tildadas con la función **removeWord**.

Las palabras más comunes de la lengua española, las llamadas **StopWords** que son palabras tales como preposiciones, artículos, pronombres, etc., fueron descartadas. La función **removeStopwords** recibe una cadena de texto y devuelve el texto excluyendo las palabras más comunes o **StopWords,** para desarrollar esta función se utilizó la librería de Python NLTK. De acuerdo con el análisis realizado no representaban mayor valor al estudio y era necesario descartarlas.

El siguiente paso que se realizó para dejar la información lo más estandarizada posible, consistió en ejecutar el steaming, que consiste en reducir las palabras a su raíz recortándolas para dejarlas estandarizadas. Para este proceso se creó la función llamada **steaming**, la cual recibe una cadena de texto y devuelve el texto reemplazando las palabras por su correspondiente raíz, para implementar esta función utilizamos la librería de Python NLTK.

Del mismo modo se creó la función **tokenizar_tweets** para remover aquellas palabras que solo contenía dos caracteres o menos con el fin de evitar ruido de palabras no representativas para el análisis. El resultado se puede observar en la figura 9.

	Index	FechaCreacion	Id_tweest	Ubicacion	Texto	R_Texto	Tipo	Sentiment
0	0	2021-03-15 01:57:03+00:00	1.37128e+18	Bogota Colombia	RT [Video] Engañan a otra abuela y no le aplic...	vide engañ abuel aplic vacun medelin jering vaci	Retweet	-1
1	1	2021-03-15 01:57:18+00:00	1.37128e+18	Barranquilla, Colombia	RT y la vacuna pa cuando? @	vacun	Retweet	-1
2	2	2021-03-15 01:58:38+00:00	1.37128e+18	colombia, Córdoba-Monteria	RT Como Gobernador solícito q revisen bien las...	gobern solicit revis bien cifr estan public re...	Retweet	0
3	3	2021-03-15 01:58:38+00:00	1.37128e+18	Valledupar, Cesar, Colombia	RT Los que andaban criticando a por la vacunac...	andab critic vacunacion enter mas pais vacun	Retweet	-1
4	4	2021-03-15 01:58:55+00:00	1.37128e+18	Bogotá, D.C., Colombia	RT ATENCIÓN : Irlanda recomienda suspender por...	atencion irland recomiend suspend precaucion v...	Retweet	-1
...	...	...	...	...	...	...	...	...
1500	1500	2021-04-25 02:12:37+00:00	1.38614e+18	Colombia	Busca en la página del ministerio los puntos d...	busc pagin ministeri punt vacunacion autoriz	Tweet	1
1501	1501	2021-04-25 02:12:46+00:00	1.38614e+18	Barranquilla, Colombia	Una vacunación sin agendamiento es lo ideal, d...	vacunacion agend ideal deberi prolong inici asi	Tweet	1
1502	1502	2021-04-25 18:40:13+00:00	1.38639e+18	Colombia	Bien por el saludo... Favor no descuidar el pr...	bien salud favor descuid proces vacunacion	Tweet	1
1503	1503	2021-04-25 18:40:15+00:00	1.38639e+18	Bogotá - Colombia	Interesante opinión sobre el plan vacunación ¿...	interes opinion plan vacunacion chil estan dif...	Tweet	0
1504	1504	2021-04-25 18:40:17+00:00	1.38639e+18	Ibagué, Colombia	RT Oites Tola, esperar la segunda dosis de la ...	oit tol esper segund dosis vacun duch tod enja...	Retweet	0

1505 rows × 8 columns

Figura 9. Dataframe resultante después de ejecutar las tareas de limpieza.

Se realizaron cálculos como por ejemplo la cantidad de tweets agrupados por la clasificación del sentimiento tal como se puede observar en la figura 10:

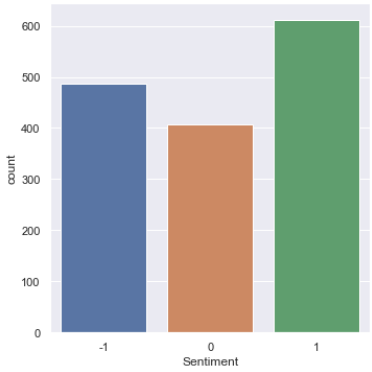


Figura 10. Cantidad de los tweets agrupados por la clasificación del sentimiento.

Los valores obtenidos del procesamiento anterior y de acuerdo con la gráfica anterior fueron los siguientes: los tweets positivos fueron 612, los negativos 486 y los neutrales 407. Así las cosas, los tweets positivos representan el 40.66%, los negativos 32.29% y los neutrales 27.04%.

Del mismo modo se clasificaron tanto los tweets como los retweets de acuerdo con la columna **Tipo** en el dataframe **tw** como lo muestra la figura 11. Donde la cantidad de tweets originales de un usuario fueron 609 y la cantidad de retweets fueron 896 lo que representa el 40.46% y 59.53% respectivamente.

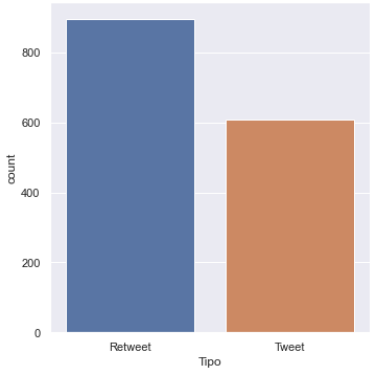


Figura 11. Clasificación de tweets y retweets.

Una vez se contó con la información depurada que se obtuvo en el paso anterior se creó el dataframe **df_tw** con las columnas **R_Texto** y **Sentiment** ya que este se utilizó para aplicar los diferentes modelos de predicción de machine learning. La figura 12 presenta la estructura anterior.

	R_Texto	Sentiment
0	vide engañ abuel aplic vacun medelin jering vaci	-1
1	vacun	-1
2	gobern solicit revis bien cifr estan public re...	0
3	andab critic vacunacion enter mas pais vacun	-1
4	atencion irland recomiend suspend precaucion v...	-1
...	...	...
1500	busc pagin ministeri punt vacunacion autoriz	1
1501	vacunacion agend ideal deberi prolong inici asi	1
1502	bien salud favor descuid proces vacunacion	1
1503	interes opinion plan vacunacion chil estan dif...	0
1504	oit tol esper segund dosis vacun duch tod enja...	0

1505 rows × 2 columns

Figura 12. Función para obtener las palabras más frecuentes.

Se creó la función **dic_frecuecia_palabras** que recibió el listado de palabras y retorno un diccionario con la frecuencia de cada una. A partir del resultado obtenido se visualizaron las 15 palabras más frecuentes en los textos procesados como se muestra en la figura 13.

	Palabra	Frecuencia
4	vacun	1360
34	vacunacion	549
84	cov	411
26	dosis	239
36	mas	171
164	colombi	164
169	mayor	129
3	aplic	116
67	person	107
286	recib	102
125	segund	102

Figura 13. Lista de palabras más frecuentes en los textos de los tweets.

A continuación, se creó el corpus a partir de la columna **R_Texto** ya que esta información se utilizó para crear la matriz de frecuencia de palabras teniendo en cuenta aquellas que tuvieron

una frecuencia mayor o igual a 10, ya que se consideró que las palabras con menos frecuencia no representaban mayor relevancia en el análisis. La figura 14 muestra el corpus creado con un total de 1505 filas.

```
0        vide engañ abuel aplic vacun medelin jering vaci
1                                                   vacun
2       gobern solicit revis bien cifr estan public re...
3            andab critic vacunacion enter mas pais vacun
4       atencion irland recomiend suspend precaucion v...
                              ...
1500         busc pagin ministeri punt vacunacion autoriz
1501      vacunacion agend ideal deberi prolong inici asi
1502           bien salud favor descuid proces vacunacion
1503    interes opinion plan vacunacion chil estan dif...
1504    oit tol esper segund dosis vacun duch tod enja...
Name: R_Texto, Length: 1505, dtype: object
```

Figura 14. Corpus de datos destinado a la investigación del proyecto.

Para la creación de la matriz de frecuencia se utilizó el siguiente procedimiento. Cada registro correspondió a cada uno de los tweets y cada columna a cada palabra, en la intersección donde se cruzó cada fila con cada columna se visualizó el valor numérico que hizo referencia al número de veces que se repitió esa palabra dentro del tweet. Al final se agrega la columna "**Sentiment"** que corresponde al sentimiento asignado al tweet. Por ejemplo, se requiere expresar el siguiente texto de un tweet en la matriz de frecuencia: "**La llegada de la vacuna enciende una luz de esperanza**". Este proceso se conoce como **countVectorizer** que realiza un conteo de palabras en un texto determinado.

colombia	continua	llegada	de	la	tiene	todos	vacuna	enciende	proceso	una	luz	esperanza	Sentiment
0	0	1	2	2	0	0	1	1	0	1	1	1	1

-1 Negativo; 1 Positivo; 0 Neutral

Figura 15. Matriz de intersección entre palabras y tweets.

En la figura 15 se puede apreciar la columna *Sentiment*, que corresponde a la clasificación dada al tweet, luego se tienen todas las variables que corresponden a cada una de las columnas que pueden ser todas las palabras que aparezcan en los tweets.

En cada una de las columnas donde aparece la palabra que se encuentra en el tweet se marcó con 1, donde no apareció está marcada con 0, en los casos donde una palabra específica apareció más de una vez en un mismo tweet se marcó el campo con el número de veces repetidas.

Finalmente, la matriz quedó representada por todas las filas que corresponden a cada uno de los tweets y las columnas en este caso las variables serán todas las palabras que aparezcan en los tweets. La tabla 4 muestra la forma en que se extrajeron las características de cada tweet.

Tabla 4. Estructura de la matriz de clasificación y definición de sentimiento de los tweets.

tweets	Sentiment	Palabra 1	Palabra 2	Palabra 2	. . .	Palabra n
1	1	0	0	1	. . .	. . .
2	1	1	1	0	. . .	. . .
3	-1	0	. . .	. . .	. . .	. . .
. . .	0	. . .	. . .	. . .	. . .	. . .
n	. . .	. . .	. . .	. . .	. . .	. . .

La matriz de frecuencia de palabras obtenidas fue la que se visualiza en la tabla 5. Esta matriz no contiene la columna **Sentiment**, la cual se agregó después de generada.

Tabla 5. Matriz de Frecuencias de palabras.

	abril	abuel	abuelit	aca	acab	aceler	acompañ	activ	actual	acuerd	...	vam	van	ver	vez	via	vid	vide	virus	viv	yomevacun
0	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1500	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1501	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1502	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1503	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1504	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

1505 rows × 354 columns

Una vez creada la matriz de frecuencias, se agregó la columna **Sentiment**, con el valor que le corresponde a cada tweet según la clasificación como se ve en la tabla 6.

Tabla 6. Matriz de Frecuencias con la columna Sentiment.

	Sentiment	abril	abuel	abuelit	aca	acab	aceler	acompañ	activ	actual	...	vam	van	ver	vez	via	vid	vide	virus	viv	yomevacun
0	-1	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
1	-1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	-1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	-1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1500	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1501	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1502	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1503	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1504	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

1505 rows × 355 columns

Se evaluaron los modelos de predicción de machine learning: Logistic Regression, Gaussian Naive Bayes (GaussianNB), SVM (Support Vector Machines), Random Forest Classifier y Decision Tree Classifier. Para esto se utilizó la librería **scikit-learn** en donde se utilizó el mismo dataset distribuido de la siguiente manera: 20% de los datos para probar los modelos y el 80% para el entrenamiento de estos (Martinez, 2016). La figura 16 describe la distribución realizada.

Figura 16. Distribución de datos de prueba y entrenamiento.

La distribución respecto a cantidades quedó de la siguiente manera: positivos 489, negativos 389 y neutrales 326 para el entrenamiento. Para pruebas: positivos 123, negativos 97 y neutrales 81.

Es importante tener en cuenta que para la aplicación de los algoritmos de maching learning, se evaluaron los modelos con los valores predeterminados para cada algoritmo, evitando realizar sobreajustes, ya que la finalidad es medir el desempeño de cada método con sus parámetros por defecto.

Evaluación del modelo Logistic Regression

Este modelo mapea una función de las características del grupo de datos con el fin de establecer la probabilidad de que con un nuevo set de datos estos pertenezcan a un sentimiento en particular (Bisong, 2019).

Al ejecutar el modelo se pudo establecer el porcentaje de exactitud y de error tanto en el entrenamiento como en las pruebas. En este caso para el entrenamiento se obtuvo un 73.42% de exactitud y un 26.57% de error. Para las pruebas del modelo arrojó un 51.49% de efectividad y un

48.50% de error en la predicción de la clasificación de datos. Dichas cifras se pueden apreciar en la figura 17. La exactitud de entrenamiento y pruebas para cada uno de los modelos se obtuvo con la función '***score(x, y, sample_weight=None)'*** pensándole como parámetros (datos de entrenamiento y etiquetas de entrenamiento) y (datos de prueba y etiquetas de pruebas) respectivamente mientras que el porcentaje de error se obtuvo restándole a uno (1) el porcentaje de **score** obtenido por la función.

```
SCORE entrenamiento: 0.7342192691029901
Error en entrenamiento: 0.26578073089700993
SCORE prueba: 0.5149501661129569
Error en prueba: 0.48504983388704315
```

Figura 17. Score y error del modelo Logistic Regression tanto en prueba como en entrenamiento.

Se creó la matriz de confusión para cada uno de los modelos con el fin de establecer los valores obtenidos respecto a las siguientes métricas:

- Accuracy (Exactitud): Indica que tan cerca estuvo el modelo de predecir la clasificación de los tweets. Cantidad de predicciones correctas de tweets positivos, negativos y neutrales.
- Precisión (Precisión): El porcentaje de predicción correcta de cada uno de los sentimientos.
- Recall (Cobertura): Toma todos los tipos de sentimiento realmente positivos y establece cuantos se predijeron correctamente.
- F1(Medida F): Compara el modelo a partir de la precisión vs la exactitud. En la figura 18 se puede observar los porcentajes obtenidos de cada medida en la regresión logística.

```
[[63 18 16]
 [30 17 34]
 [17 31 75]]
              precision    recall  f1-score   support

          -1       0.57      0.65      0.61        97
           0       0.26      0.21      0.23        81
           1       0.60      0.61      0.60       123

    accuracy                           0.51       301
   macro avg       0.48      0.49      0.48       301
weighted avg       0.50      0.51      0.51       301
```

Figura 18. Resultado de la matriz de confusión con la evaluación del modelo Logistic Regression.

En la matriz de confusión el eje X representa los datos de predicción de la clasificación de tweets y el eje Y las etiquetas que representan el valor verdadero (datos de entrenamiento) como lo muestra la figura 19.

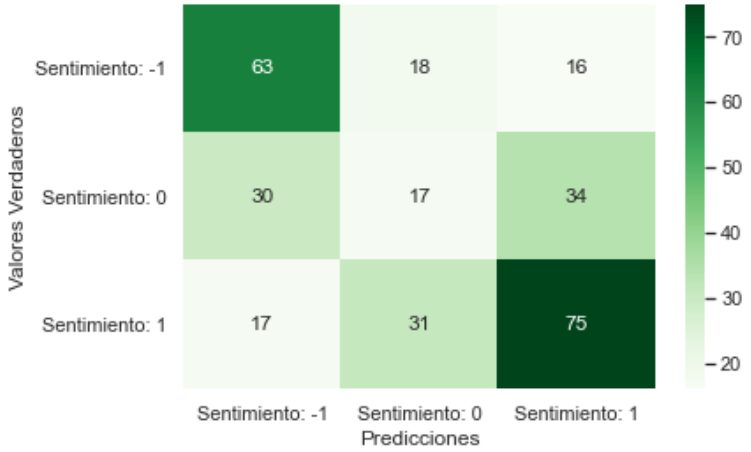


Figura 19. Matriz de confusión del modelo Logistic Regression.

La matriz de confusión anterior se puede interpretar de la siguiente manera:

- La intercepción entre valores verdades y predichos para el sentimiento positivo (1), indica que logro predecir correctamente 75 tweets. Mientras que 31 que deberían ser positivos se predijeron como naturales y 17 que deberían ser positivos se predijeron como negativos.
- La intercepción entre valores verdades y predichos para el sentimiento neutral (0), indica que logro predecir correctamente 17 tweets. Mientras que 34 que deberían ser neutrales se predijeron como positivos y 30 que deberían ser neutrales se predijeron como negativos.
- La intercepción entre valores verdades y predichos para el sentimiento negativo (-1), indica que logro predecir correctamente 63 tweets. Mientras que 18 que deberían ser negativos se predijeron como naturales y 16 que deberían ser negativos se predijeron como positivos.

Evaluación del modelo Gaussian Naive Bayes (GaussianNB)

Este modelo se basa en un grupo de algoritmos donde se aplica el teorema de Bayes para el aprendizaje supervisado, donde se parte de una suposición ingenua que independiza cada característica respecto al valor del sentimiento para establecer su valor predicho (Scikit Learn, 2020). El score y porcentaje de error obtenido tanto de los datos de prueba y entrenamiento se pueden apreciar en la figura 20.

```
SCORE entrenamiento: 0.6112956810631229
Error en entrenamiento: 0.38870431893687707
SCORE prueba: 0.4717607973421927
Error en prueba: 0.5282392026578073
```

Figura 20. Score y error del modelo GaussianNB tanto en prueba como en entrenamiento.

Los porcentajes obtenidos para el modelo Bayes (GaussianNB) se muestran en la figura 21.

```
[[71 19  7]
 [44 21 16]
 [40 33 50]]
              precision    recall  f1-score   support

          -1       0.46      0.73      0.56        97
           0       0.29      0.26      0.27        81
           1       0.68      0.41      0.51       123

    accuracy                           0.47       301
   macro avg       0.48      0.47      0.45       301
weighted avg       0.50      0.47      0.46       301
```

Figura 21. Resultado de la matriz de confusión con la evaluación del modelo GaussianNB.

La figura 22 muestra la matriz de confusión graficada con la escala de calor correspondiente a la concordancia de la predicción versus los valores verdaderos.

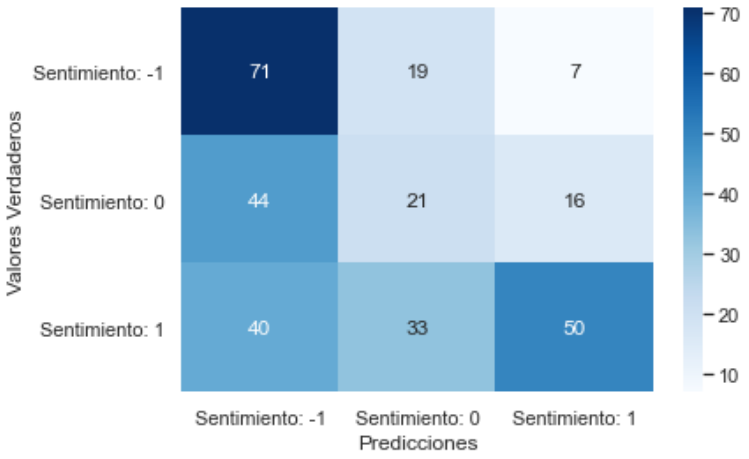


Figura 22. Matriz de confusión del modelo GaussianNB.

La matriz de confusión anterior se puede interpretar de la siguiente manera:

- La intercepción entre valores verdades y predichos para el sentimiento positivo (1), indica que logro predecir correctamente 50 tweets. Mientras que 33 que deberían ser positivos se predijeron como naturales y 40 que deberían ser positivos se predijeron como negativos.
- La intercepción entre valores verdades y predichos para el sentimiento neutral (0), indica que logro predecir correctamente 21 tweets. Mientras que 16 que deberían ser neutrales se predijeron como positivos y 44 que deberían ser neutrales se predijeron como negativos.
- La intercepción entre valores verdades y predichos para el sentimiento negativo (-1), indica que logro predecir correctamente 71 tweets. Mientras que 19 que deberían ser negativos se predijeron como naturales y 7 que deberían ser negativos se predijeron como positivos.

Evaluación del modelo SVM (Support Vector Machines)

Como sus siglas lo indican, son máquinas de soporte vectoriales que bajo el método de aprendizaje supervisado utiliza la clasificación de datos para predecir la agrupación futura de las variables con distintos sets de datos (Scikit Learn, 2020). La figura 23 presenta el porcentaje de score y error tanto en los datos de prueba como de entrenamiento del modelo SVM.

```
SCORE entrenamiento: 0.8222591362126246
Error en entrenamiento: 0.1777408637873754
SCORE prueba: 0.5382059800664452
Error en prueba: 0.46179401993355484
```

Figura 23. Score y error del modelo SVM tanto en prueba como en entrenamiento.

Los porcentajes obtenidos para el modelo SVM se muestran en la figura 24.

```
[[54 11 32]
 [28 18 35]
 [22 11 90]]
              precision    recall  f1-score   support

          -1       0.52      0.56      0.54        97
           0       0.45      0.22      0.30        81
           1       0.57      0.73      0.64       123

    accuracy                           0.54       301
   macro avg       0.51      0.50      0.49       301
weighted avg       0.52      0.54      0.52       301
```

Figura 24. Resultado de la matriz de confusión con la evaluación del modelo SVM.

Al igual que los modelos anteriormente ejecutados y como se muestra en la figura 25, se creó la gráfica de la matriz de confusión con la escala de calor correspondiente a la concordancia de la predicción versus los valores verdaderos.

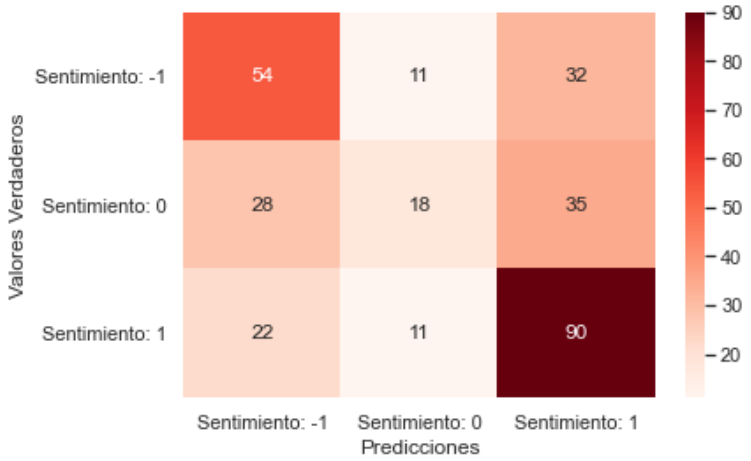


Figura 25. Matriz de confusión del modelo SVM.

La matriz de confusión anterior se puede interpretar de la siguiente manera:

- La intercepción entre valores verdades y predichos para el sentimiento positivo (1), indica que logro predecir correctamente 90 tweets. Mientras que 11 que deberían ser positivos se predijeron como naturales y 22 que deberían ser positivos se predijeron como negativos.
- La intercepción entre valores verdades y predichos para el sentimiento neutral (0), indica que logro predecir correctamente 18 tweets. Mientras que 35 que deberían ser neutrales se predijeron como positivos y 28 que deberían ser neutrales se predijeron como negativos.
- La intercepción entre valores verdades y predichos para el sentimiento negativo (-1), indica que logro predecir correctamente 54 tweets. Mientras que 11 que deberían ser negativos se predijeron como naturales y 32 que deberían ser negativos se predijeron como positivos.

Evaluación del modelo Random Forest Classifier

Este modelo realiza la clasificación a partir de bosques con árboles de decisión aleatorios para crear predicciones con subconjuntos de datos y promedios. Cada subconjunto, para el estudio actual, cada agrupación de datos con los sentimientos de los tweets representa un árbol de predicción (Scikit Learn, 2020). El score y la prueba obtenidos para estos modelos son los correspondientes a la figura 26.

```
SCORE entrenamiento: 0.9842192691029901
Error en entrenamiento: 0.01578073089700993
SCORE prueba: 0.5249169435215947
Error en prueba: 0.4750830564784053
```

Figura 26. Score y error del modelo Random Forest Classifier tanto en prueba como en entrenamiento.

Los porcentajes de la precisión, recall y F1 se muestran en la figura 27. También los datos de la muestra.

```
[[47 16 34]
 [18 25 38]
 [18 19 86]]
              precision    recall  f1-score   support

          -1       0.57      0.48      0.52        97
           0       0.42      0.31      0.35        81
           1       0.54      0.70      0.61       123

    accuracy                           0.52       301
   macro avg       0.51      0.50      0.50       301
weighted avg       0.52      0.52      0.51       301
```

Figura 27. Resultado de la matriz de confusión con la evaluación del modelo Random Forest Classifier.

La figura 28 corresponde a la matriz de confusión graficada con su respectivo mapa de calor para el modelo de Random Forest Classifier.

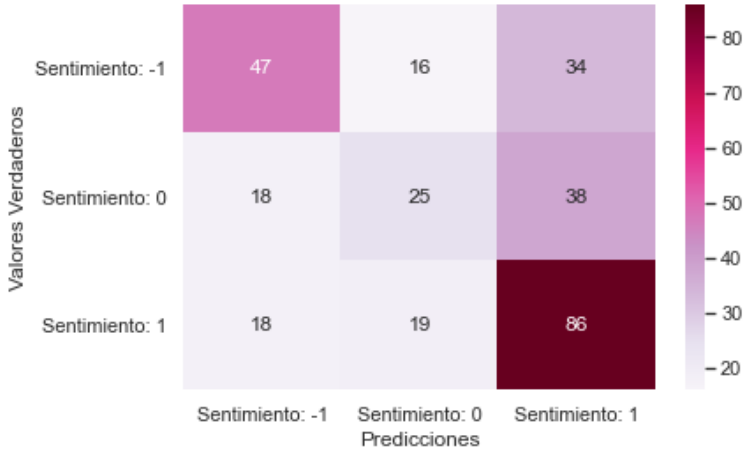


Figura 28. Matriz de confusión del modelo Random Forest Classifier.

La matriz de confusión anterior se puede interpretar de la siguiente manera:

- La intercepción entre valores verdades y predichos para el sentimiento positivo (1), indica que logro predecir correctamente 86 tweets. Mientras que 19 que deberían ser positivos se predijeron como naturales y 18 que deberían ser positivos se predijeron como negativos.
- La intercepción entre valores verdades y predichos para el sentimiento neutral (0), indica que logro predecir correctamente 25 tweets. Mientras que 38 que deberían ser neutrales se predijeron como positivos y 18 que deberían ser neutrales se predijeron como negativos.

- La intercepción entre valores verdades y predichos para el sentimiento negativo (-1), indica que logro predecir correctamente 47 tweets. Mientras que 16 que deberían ser negativos se predijeron como naturales y 34 que deberían ser negativos se predijeron como positivos.

Evaluación del modelo Decision Tree Classifier

Este modelo de aprendizaje también trabaja con el método de aprendizaje supervisado y aplica la ejecución de reglas simples para la clasificación de los datos a partir de sus características (Scikit Learn, 2020). Los porcentajes de score y pruebas de este modelo se pueden consultar en la figura 29.

```
SCORE entrenamiento: 0.9842192691029901
Error en entrenamiento: 0.01578073089700993
SCORE prueba: 0.45182724252491696
Error en prueba: 0.548172757475083
```

Figura 29. Score y error del modelo Decision Tree Classifier tanto en prueba como en entrenamiento.

Los porcentajes de precisión, recall y F1 obtenidos están en la figura 30.

```
[[46 25 26]
 [24 27 30]
 [32 28 63]]
              precision    recall  f1-score   support

          -1       0.45      0.47      0.46        97
           0       0.34      0.33      0.34        81
           1       0.53      0.51      0.52       123

    accuracy                           0.45       301
   macro avg       0.44      0.44      0.44       301
weighted avg       0.45      0.45      0.45       301
```

Figura 30. Resultado de la matriz de confusión con la evaluación del modelo Decision Tree Classifier.

Finalmente, al igual que en los otros modelos se grafica la matriz de confusión, como lo muestra la figuras 31, con su respectivo mapa de calor con el cruce de predicciones como valores verdaderos.

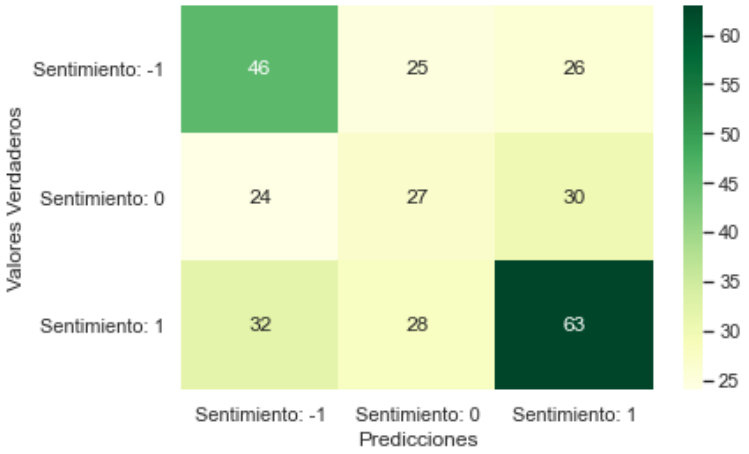


Figura 31. Matriz de confusión del modelo Decision Tree Classifier.

La matriz de confusión anterior se puede interpretar de la siguiente manera:

- La intercepción entre valores verdades y predichos para el sentimiento positivo (1), indica que logro predecir correctamente 63 tweets. Mientras que 28 que deberían ser positivos se predijeron como naturales y 32 que deberían ser positivos se predijeron como negativos.
- La intercepción entre valores verdades y predichos para el sentimiento neutral (0), indica que logro predecir correctamente 27 tweets. Mientras que 30 que deberían ser neutrales se predijeron como positivos y 24 que deberían ser neutrales se predijeron como negativos.
- La intercepción entre valores verdades y predichos para el sentimiento negativo (-1), indica que logro predecir correctamente 46 tweets. Mientras que 25 que deberían ser negativos se predijeron como naturales y 26 que deberían ser negativos se predijeron como positivos.

Para mayor detalle del procesamiento de la información consultar el '*Anexo 6 - Procesamiento'*.

Interpretación de resultados

Revisando los resultados de los cinco modelos evaluados, se pudo identificar que el modelo SVM presenta una mayor exactitud para la clasificación de los sentimientos de los tweets con los

datos de prueba, obteniendo un 53.82% de exactitud. Mientras que el modelo Decision Tree Classifier con el 45.18% fue el que menos exactitud presentó para clasificar los sentimientos de los tweets con los datos de prueba. Respecto a los datos de entrenamiento los modelos que mejor se comportaron fueron el Decision Tree Classifier y el Random Forest Classifier con el 98.42%, mientras que el de porcentaje más bajo de predicción fue GaussianNB con el 47.17%.

En cuanto a la precisión de la clasificación de los tweets positivos el mejor fue GaussianNB con un porcentaje del 68%, mientras que el que menor precisión tuvo fue el Decision Tree Classifier con el 53%. Ahora, respecto a los tweets negativos los de mayor precisión fueron el Logistic Regression y el Random Forest Classifier con un 57%, por otro lado, el de menor precisión fue Decision Tree Classifier con un 45%. En cuanto a los tweets neutrales el de mejor precisión fue el SVM con un 45%, el Logistic Regression fue el de menor precisión con un 26%.

Respecto al Recall, el mejor modelo para predecir los tweets positivos fue el SVM con 73%, mientras que el modelo que peor predicción tuvo al respecto fue el GassianNB con el 41%. Para los tweets negativos el mejor modelo fue GussianNB con un 73% y el que tuvo menor predicción fue el Decision Tree Classifier con un 47%. Ahora para los tweets neutrales el de mejor predicción fue el Decision Tree Classifier con un 33% y el de más baja predicción fue el Logistic Regression con un 21%.

Respecto a las predicciones de la medida F1-score para los tweets positivos el mejor modelo fue el SVM con un 64% mientras que el GussianNB fue el de peor predicción con un 51%. En los tweets negativos el de predicción más alta fue el Logistic Regression con un 61% mientras que el de más baja predicción Decision Tree Classifier con un 61%. En los tweets neutrales el de mejor

predicción fue Random Forest Classifier con el 35% y el de menor predicción fue el Logistic Regression con un 23%.

La nube de palabras presentada en la figura 32, permitió visualizar su ubicación en el centro y con mayor tamaño aquellas más frecuentes dentro de los tweets que fueron clasificados y los cuales hicieron parte del estudio de análisis de datos. La palabra más representativa según la nube fue ‘vacun’ que es la raíz de la palabra vacuna haciendo clara referencia al tema tratado en el proyecto y su relación con el Covid-19. Otras palabras relevantes fueron: colombia, persona, gobierno, segunda dosis entre otras.

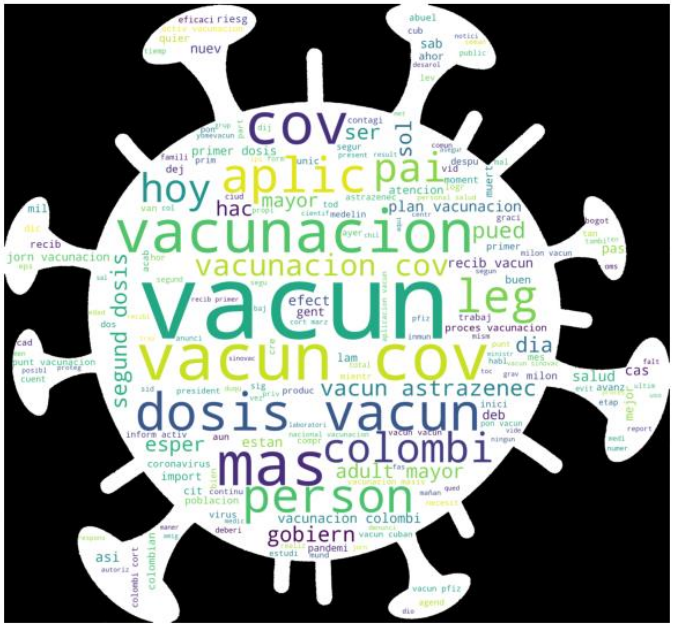


Figura 32. Nube con las palabras clave y con más frecuencia en los tweets.

La figura 33 visualiza las 20 palabras más frecuentes dentro de los 1504 tweets analizados.

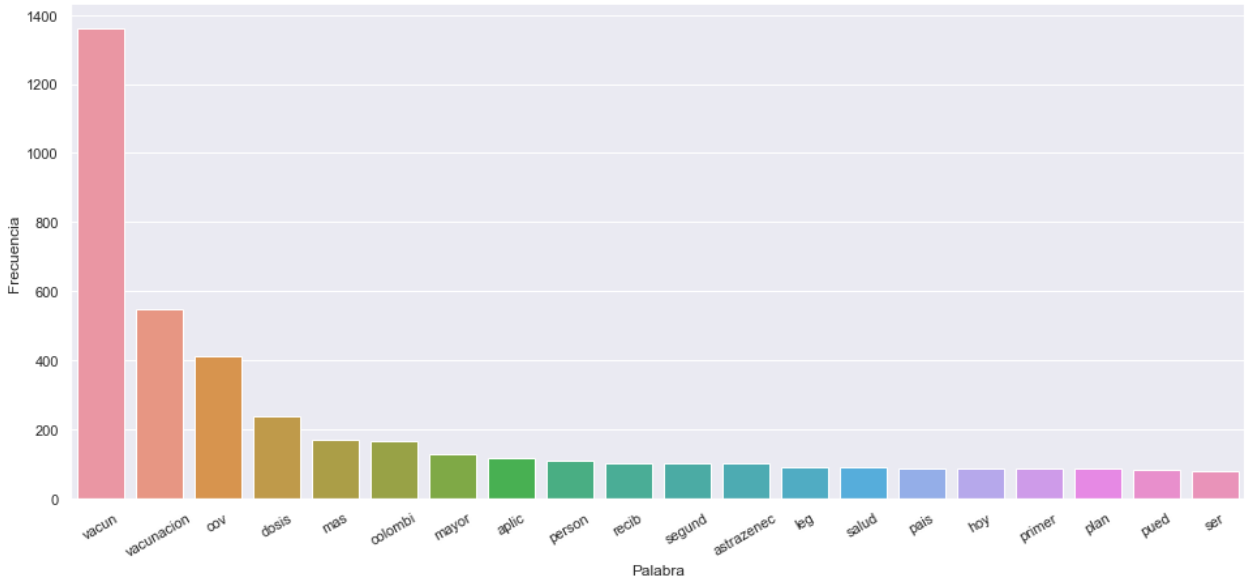


Figura 33. Listado de las veinte palabras más frecuentes y su cantidad.

Conclusiones

Se recolectaron 260.166 tweets que representaron una gran cantidad de información relacionada al tema de la vacunación del Covid 19 en Colombia. El periodo de captura fue desde el 15 de marzo hasta el 25 de abril de 2021. Al implementar los procesos para el análisis de sentimiento y filtrar el contenido por ubicación (Colombia) y temática (vacunación del Covid 19), se limitó la muestra de datos a 1504 tweets que fueron empleados para entrenar y ejecutar los modelos de predicción y las actividades de Machine Learning. De esta cantidad 609 fueron tweets de publicaciones originales de los usuarios que representaron el 40.46% de la muestra, mientras que 896 fueron retweets qué fueron publicaciones replicadas por otros usuarios diferentes a los autores, estos representaron el 59.53%.

Se clasificaron 612 tweets como positivos, 486 como negativos y 407 como neutrales, con un porcentaje del 40.66%, 32.29% y 27.04% respectivamente.

Al aplicar la metodología de aprendizaje supervisado y los algoritmos de machine learning el modelo que mejor exactitud tuvo al momento de la predicción fue el Support Vector Machines para el procesamiento de text mining, se logró establecer que el modelo en una evaluación general e inicial contó con un 54% en su primera ejecución. Entre tanto el modelo que menor exactitud tuvo fue el Decision Tree Classifier con un 45%. La diferencia de exactitud entre el modelo con menor porcentaje respecto al de mayor porcentaje fue del 9%.

Con la aplicación de las técnicas y metodologías de machine learning para el análisis de sentimientos, se pudo llegar a una clasificación acertada respecto a una muestra de datos frente a una temática específica en una red social como twitter. Teniendo en cuenta que se aplicó la librería SnowballStemmer perteneciente al paquete nltk.tokenize que es utilizado para el procesamiento de

lenguaje natural en Python con el fin de obtener la raíz de las palabras en el idioma español, se identificó que no logra estandarizar todas las palabras por lo tanto se recomienda para trabajos futuros la posibilidad de implementar un algoritmo que complemente este proceso, con el fin obtener las palabras lo más estandarizadas posibles.

Referencias

Alamoodi, A., Zaidan, B., Zaidan, A., Albahri, O., Mohammed, K., Malik, R., Almahdi, E., Chyad, M., Tareq, Z., Albahri, A., Hameed, H., & Alaa, M. (2021). Sentiment analysis and its applications in fighting COVID-19 and infectious diseases: A systematic review. 13.

Bian, Y., Cui, K., Wang, L., Zheng, G., Guo, H., Yang, J., Jiang, M., & Lu, A. (2014). IEEE International Conference on Bioinformatics and Biomedicine - Application of Acupuncture on Coronary Heart Disease Treatment: A Text Mining Study. 4.

Bian, Y., Zhou, H., Guo, J., Wang, Y., Zheng, G., Guo, H., Tan, Y., Ren, X., Dong, R., Zhang, J., Cui, Z., Lu, A., Jiang, M., & Wang, Y. (2014). IEEE International Conference on Bioinformatics and Biomedicine, Study of acupuncture therapy on hypertension based on text ming. 4.

Bisong, E. (2019). *Building Machine Learning and Deep Learning Models on Google Cloud Platform*.

Caputo, A., Giacchetta, A., & Langher, V. (2016). AIDS as social construction: text mining of AIDSrelated information in the Italian press. 7.

Chakraborty, K., Bhatia, S., Bhattacharyya, S., Platos, J., Bag, R., & Hassanien, A. (2020). Sentiment Analysis of COVID-19 tweets by Deep Learning Classifiers—A study to show how popularity is affecting accuracy in social media. *ELSEVIER*, 14.

Gonzalez Peña, D., Lourenço, A., López Fernández, H., Reboiro Jato, H., & Fdez Riverola, F. (2014, SEPTIEMBRE). Web scraping technologies in an API world - Briefings in Bioinformatics.

Kabir, Y., & Madria, S. (2020, JULIO 11). CoronaVis: A Real-time COVID-19 Tweets Data Analyzer and Data Repository. 10.

KARAMI, A., LUNDY, M., WEBB, F., & DWIVEDI, Y. (2020). Twitter and Research: A Systematic Literature Review Through Text Mining. *IEEE ACCESS*.

Martinez, J. (2016). *Primer Taller de Análisis de Sentimiento en Twitter con R. DB GUIDANCE*. https://www.youtube.com/watch?v=nOIZnYLlPBo

Mathkour, H., Hashimi, H., & Hafez, A. (2015). Computers in Human Behavior - Selection criteria for text mining approaches. *ELSEVIER*, 5.

Narvaiza Cortes, W., & Medina Valverde, O. A. (2020). Analítica de datos no estructurados para dar soporte a la toma decisiones en el área de comercialización de la Empresa Representaciones Batericar S.A.C. utilizando la metodología ICAV y la plataforma de Microsoft. *ALICIA*, 86.

Nielsen, F. (2011, MARZO 15). A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. 6.

Python Software Foundation. (2009). *RE— Regular expression operations.* Retrieved 2021, from https://docs.python.org/3/library/re.html

SAS. (2021). *Inteligencia artificial, que es y por qué es importante.* Retrieved 2021, from https://www.sas.com/es_co/insights/analytics/what-is-artificial-intelligence.html

Scikit Learn. (2020). *Decision Trees*. Decision Trees. Retrieved 2021, from https://scikit-learn.org/stable/modules/tree.html#tree

Scikit Learn. (2020). *Naive Bayes*. Naive Bayes. https://scikit-learn.org/stable/modules/naive_bayes.html

Scikit Learn. (2020). *RandomForestClassifier*. Retrieved 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=randomforestclassifier#sklearn.ensemble.RandomForestClassifier

Scikit Learn. (2020). *Support Vector Machines*. Support Vector Machines. Retrieved 2020, from https://scikit-learn.org/stable/modules/svm.html#svm-classification

Twitter Inc. (2021). *Developers - Documentation - Search Tweets*. https://developer.twitter.com/en/docs/twitter-api/tweets/search/introduction

Twitter Inc. (2021). *Documentation - Data dictionary: Standard v1.1 - Tweet Data Dictionary*. Retrieved 2021, from https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet

Twitter Inc. (2021). *Twitter API v2: Early Access*. Retrieved 2021, from https://developer.twitter.com/en/docs/twitter-api/early-access

Valcárcel Asencios, V. (2004). Data mining y el descubrimiento del conocimiento. 5.

Wasserman, S., & Faust, K. (2013). Análisis de redes sociales. Métodos y aplicaciones.

Anexos

Anexo 1 - Creación de base de datos MongoDB en la nube.

Anexo 2 - Obtención Keys y Tokens de Twitter.

Anexo 3 - Extracción de datos.

Anexo 4 - Versionamiento, administración y repositorio de código fuente.

Anexo 5 - Preprocesamiento.

Anexo 6 - Procesamiento.

Anexo 1 - Creación de base de datos MongoDB en la nube

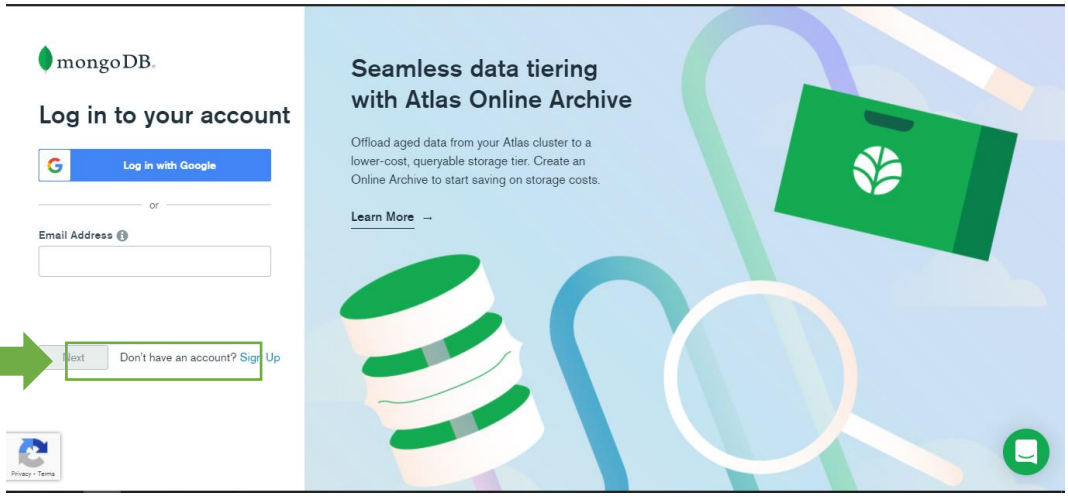
Registro en la plataforma

Debido a que no se contaba con una cuenta en la plataforma de MongoDB Atlas, se creó el usuario de la siguiente manera.

1. Ingreso a la dirección de MongoDB Atlas.

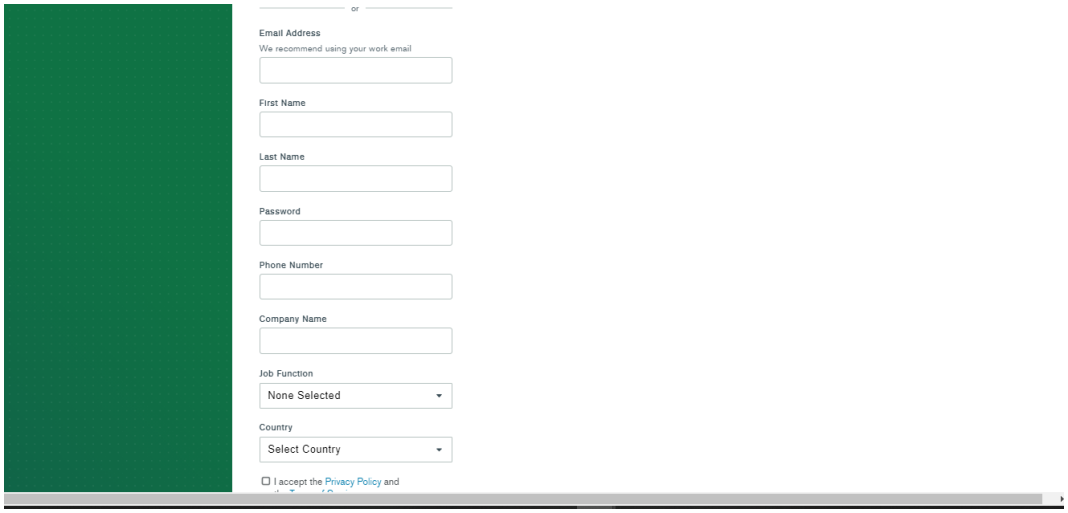
   https://account.mongodb.com/account/login?signedOut=true

2. Se seleccionó la opción de **Sign Up**:

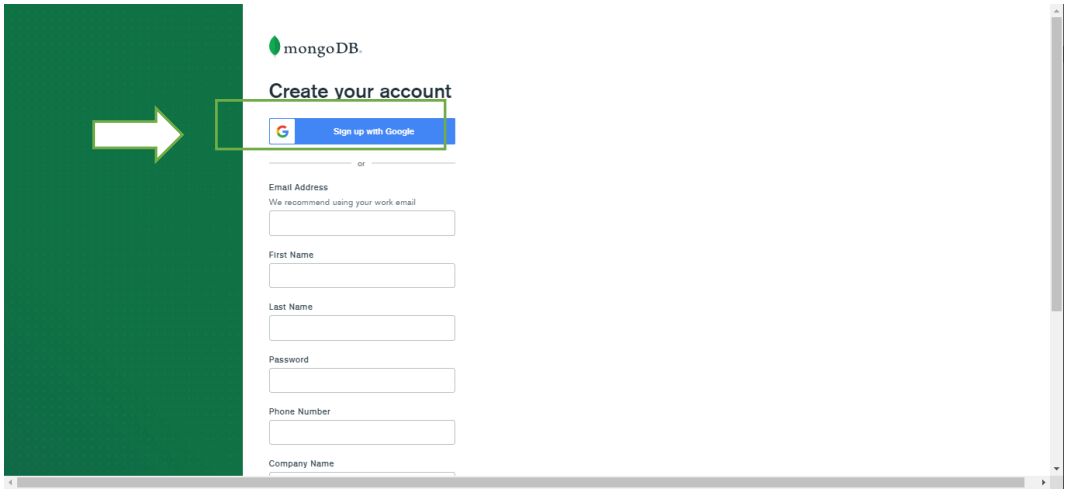


Existen dos opciones:

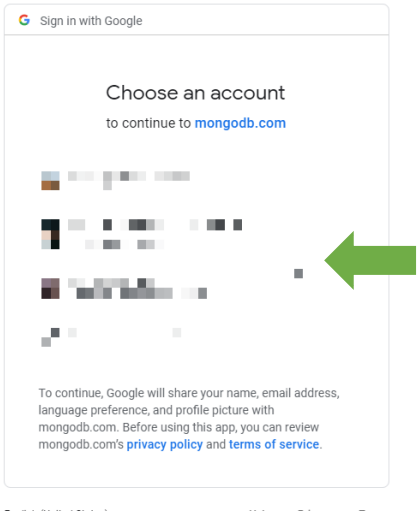
   a. Completar el formulario de registro con la información con el nombre de la persona y medios de contacto.



b. Seleccionar el botón de Ingresar con una cuenta de Google y seleccionar la cuenta de correo electrónico para el registro.



Al seleccionar la cuenta solicitó establecer una contraseña en la plataforma y después de ello el proceso de registro finalizó.



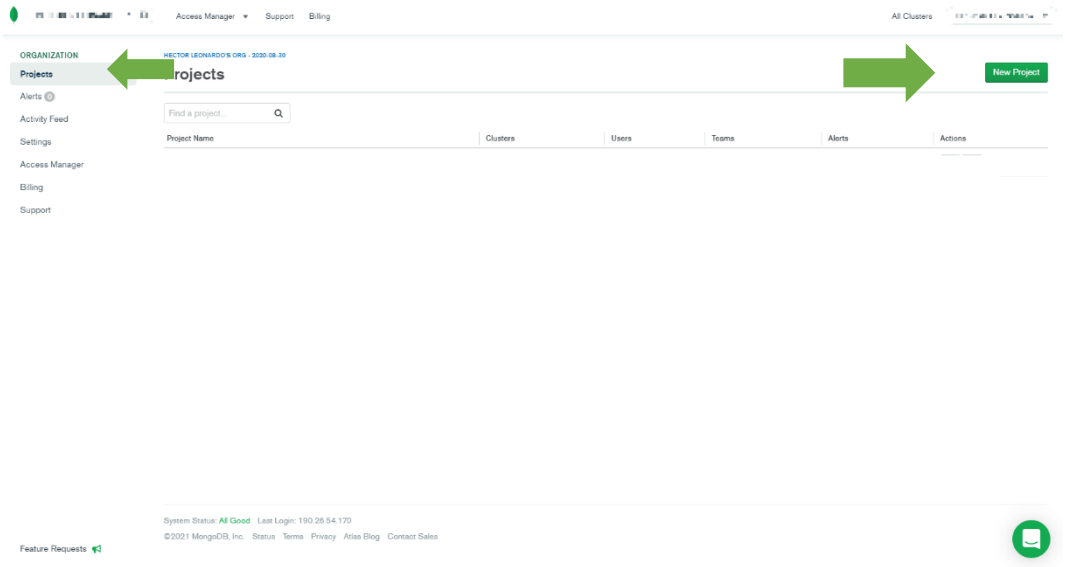
Creación del proyecto

Para crear la base de datos, previamente se necesitó crear un proyecto para posteriormente crear el Cluster contenedor de la base de datos.

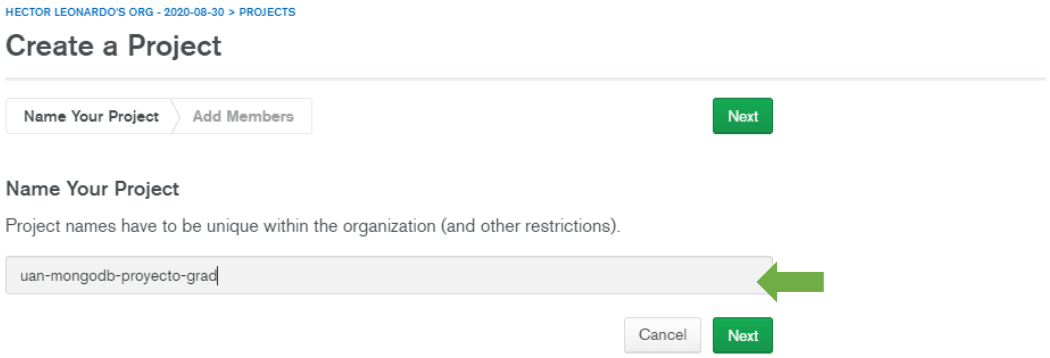
Para la creación del proyecto se deben seguir los siguientes pasos:

1. En el menú del panel derecho de **Organization** se seleccionó la primera opción de **Projects**.

Una vez allí se utilizó el botón en la parte superior derecha de **New Project**.

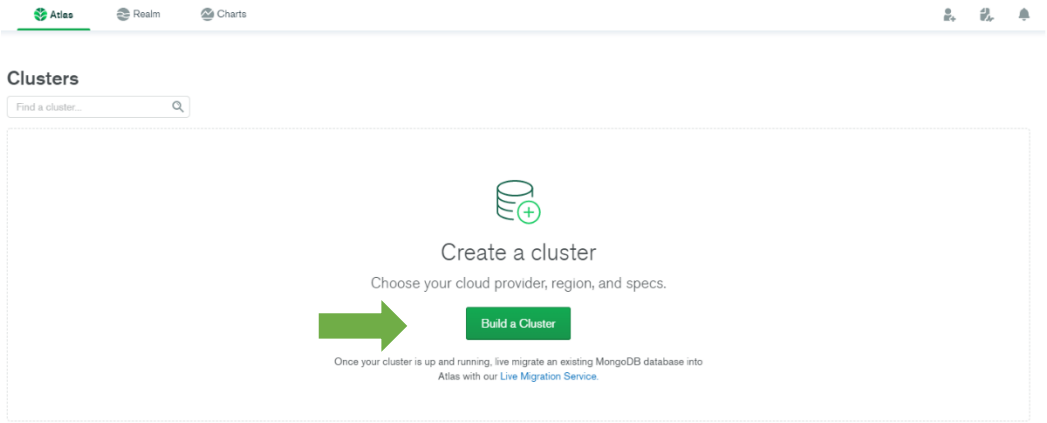


2. Al ingresar el nombre del proyecto se dio clic en el botón **Next** para su creación. El nombre del proyecto establecido fue uan-mongodb-proyecto-grado.



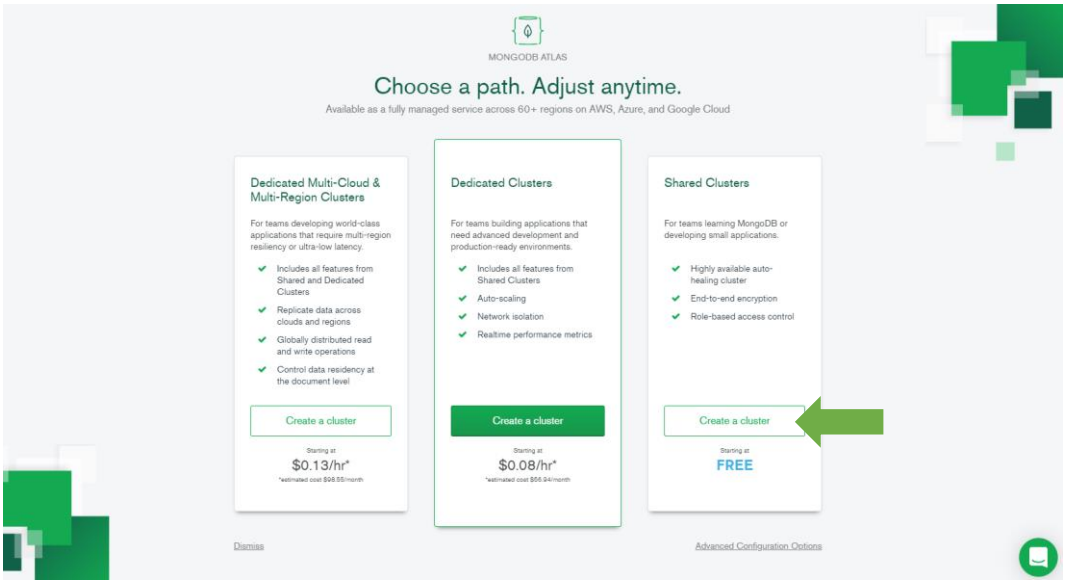
Creación del Cluster

Una vez creado el proyecto, en el panel central de la página se seleccionó el botón **Build a Cluster**. Acto seguido se procedió a configurar el cluster de acuerdo a las opciones que ofrece el proveedor para su uso.



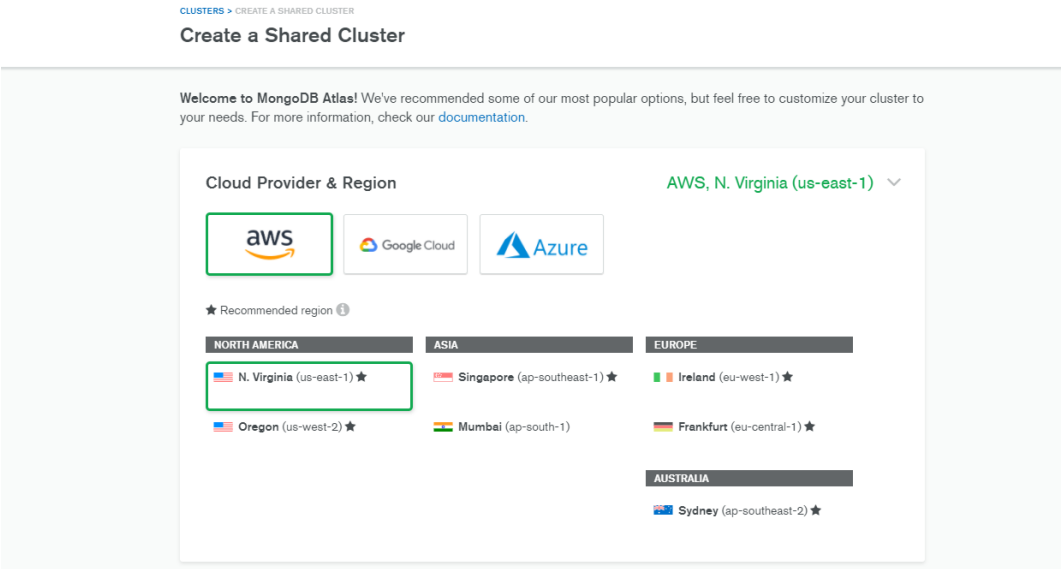
De las tres opciones ofrecidas por el proveedor se seleccionó la versión gratis (Free) y para continuar con la configuración se dio clic en el botón de **Create a cluster**. Para el proyecto esta

fue adecuada dado el alcance que tiene y su enfoque orientado a equipos que desean aprender más sobre su entorno de desarrollo.

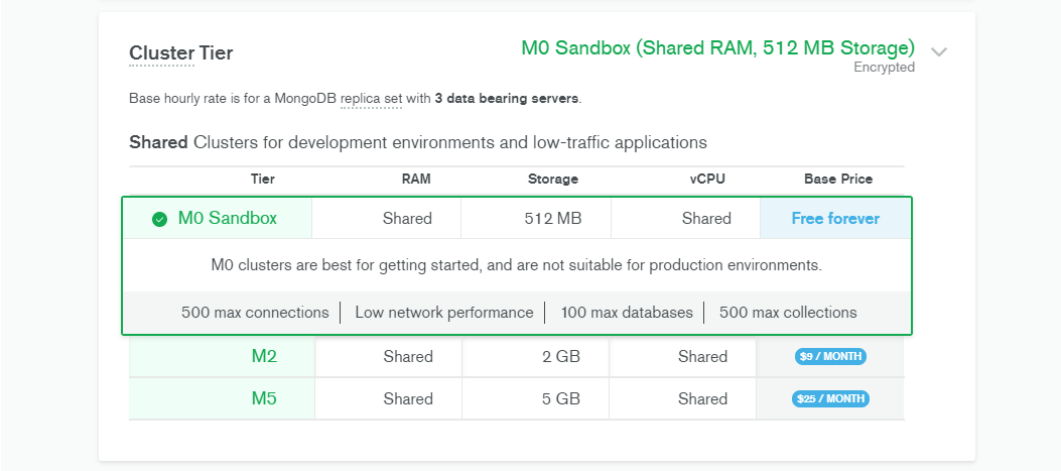


Una vez confirmado el tipo de servidor, se procedió a seleccionar los ítems de configuración. A continuación, se especifican las características del cluster de almacenamiento que hace parte de la arquitectura tecnológica propuesta en el proyecto.

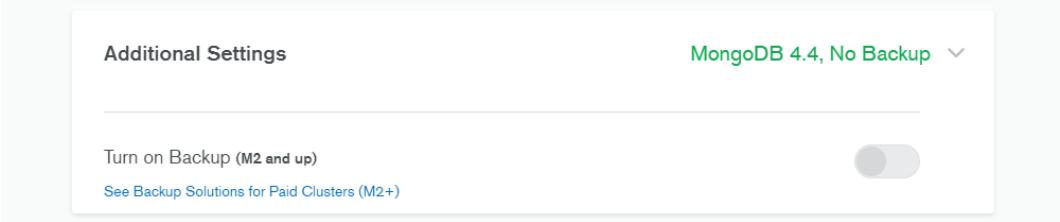
1. Selección del proveedor en la nube para el servidor: En este caso un servidor AWS Cloud ubicado en la ciudad de Virginia del Norte, Estados Unidos.



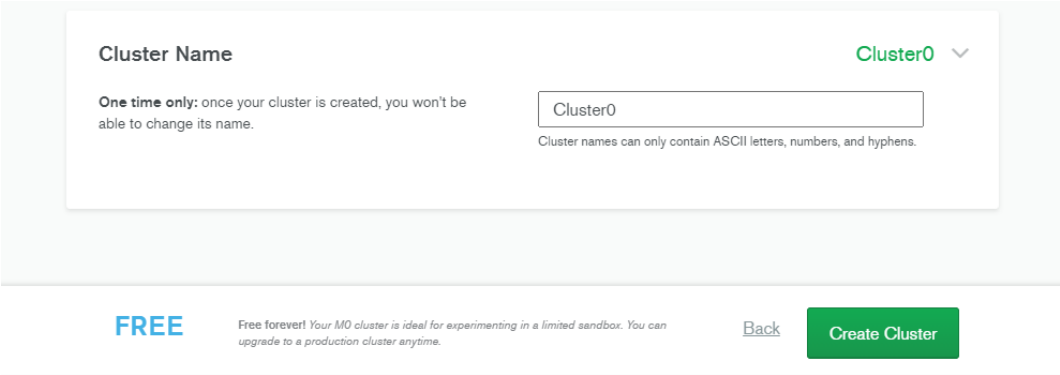
2. La capacidad de almacenamiento ofrecida para este tipo de servidor es de 512 MB, capacidad que se consideró apta para el volumen de registros de Tweets que se consideraron a almacenar.



3. La plataforma ofrece la opción de gestionar Backups, pero estos servicios son de pago. De este modo se decidió no contar con esta opción y gestionar los backups de los datos en la base de datos de forma diferente.



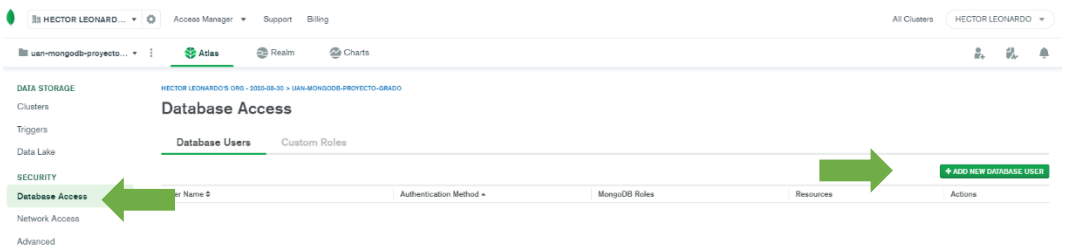
4. Finalmente se estableció el nombre para el cluster quedando este identificado como **una-mongodb-server-project**.



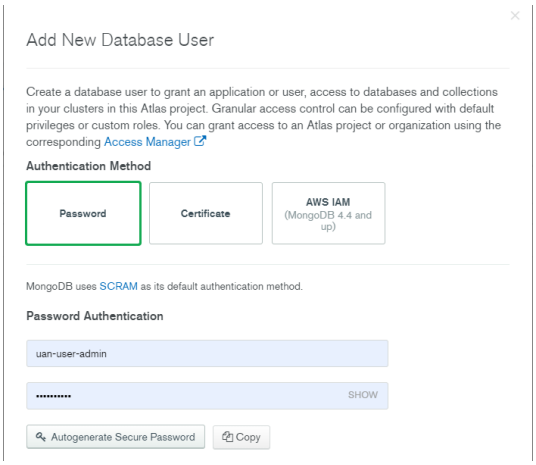
Creación del acceso a la base de datos (Database Access)

Para la creación del usuario de administración de la base de datos MongoDB en la nube se ejecutaron los siguientes pasos:

1. Selección de la opción **Database Access** ubicada en el menú lateral izquierdo de la plataforma, seguido de esto el uso del botón **Add new Database User**.

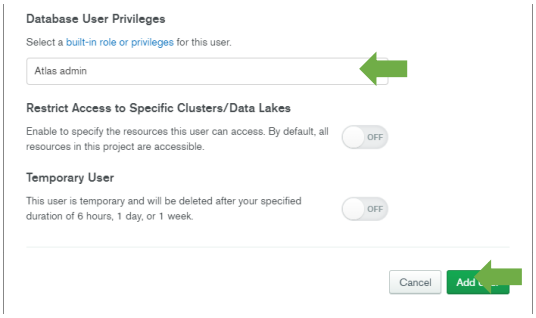


2. Establecimiento de las credenciales del usuario. Nombre en este caso **uan-user-admin** y contraseña. Existen tres formas de establecer la seguridad del usuario: Contraseña, certificado o el servicio de identificación de AWS. En este caso se definió la contraseña como atributo de seguridad del usuario.

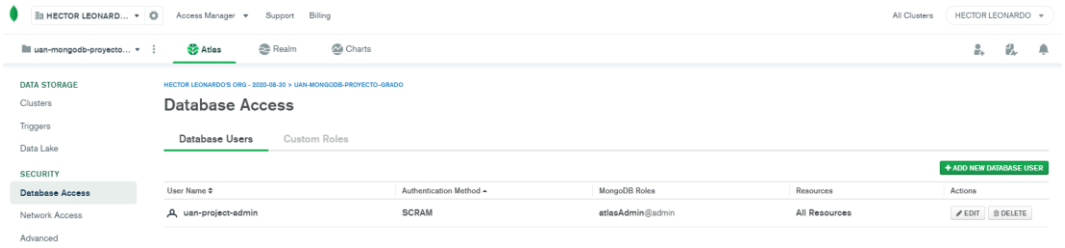


3. Definición de privilegios: Como este es el usuario administrador de la base de datos se seleccionó el rol de **Atlas admin**.

4. Finalmente, para confirmar la configuración del usuario administrador se hizo mediante el botón **Add user**.



Al regresar al tablero de control se confirmó que el usuario se había creado con éxito.

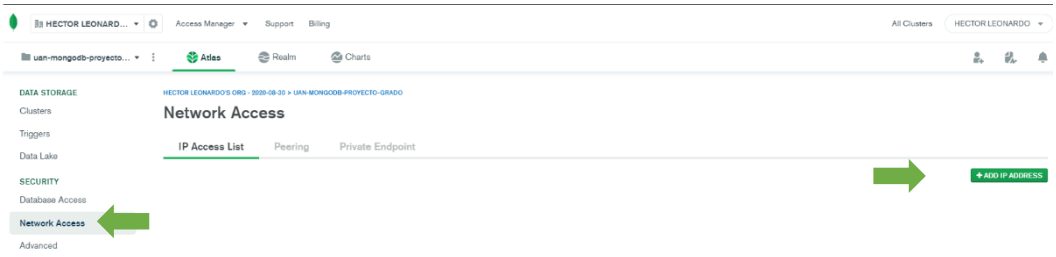


Creación de acceso de red (Network Access)

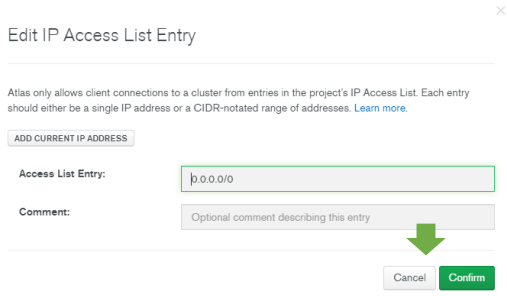
Ya establecido el usuario de administración y acceso a la base de datos. Se procedió a crear el acceso externo desde la red al servidor. Es de suma importancia realizar la configuración de este acceso, de lo contrario no se podrá tener acceso al servidor desde otros dispositivos o aplicaciones así se cuente con la cadena de conexión.

Este fue el procedimiento realizado para dicho fin.

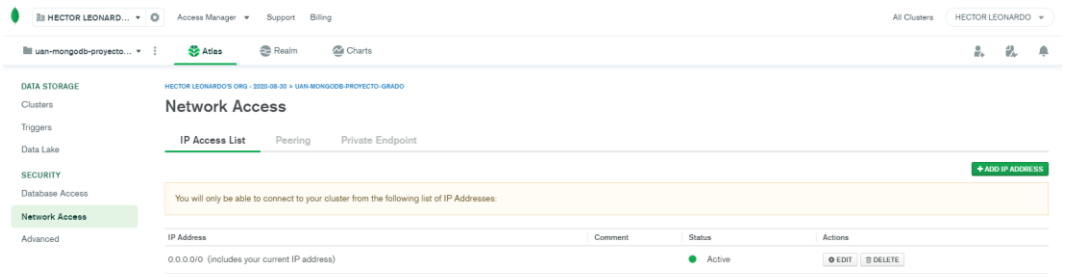
1. En el menú de **Data Storage** se seleccionó la opción de **Network Access**. Acto seguido se seleccionó el botón **Add IP Address**.



2. Para la configuración del acceso permite dos diferentes tipos de accesos. Estableciendo una IP fija de acceso o un acceso desde cualquier dirección IP que cuente con la cadena de conexión al servidor. Para confirmar la configuración de acceso basto con seleccionar la opción **Confirm**.

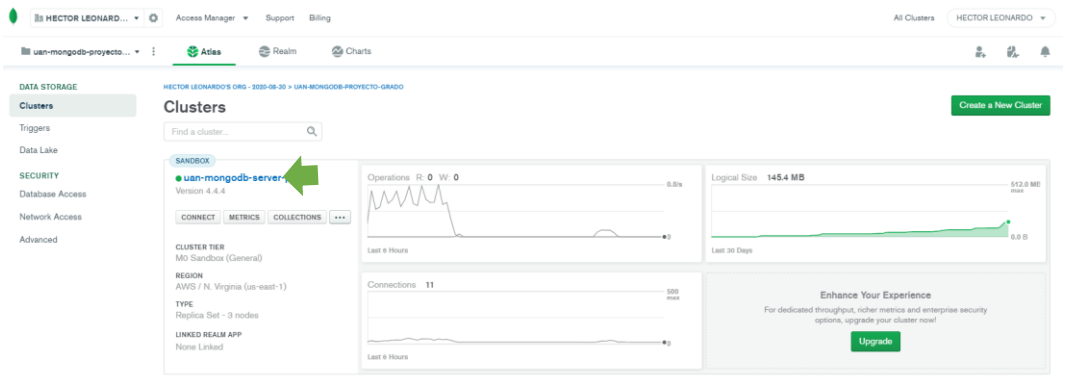


De vuelta al tablero de control se confirmó que el canal de acceso por red a la base de datos se encontraba activo y correctamente configurado.

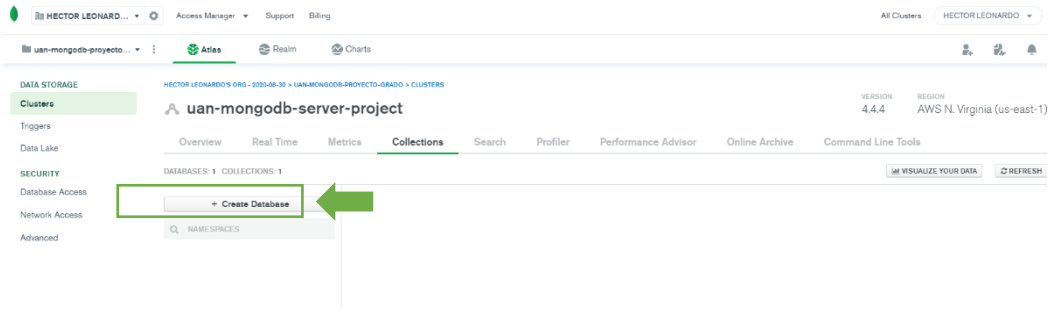


Creación de la base de datos y colección

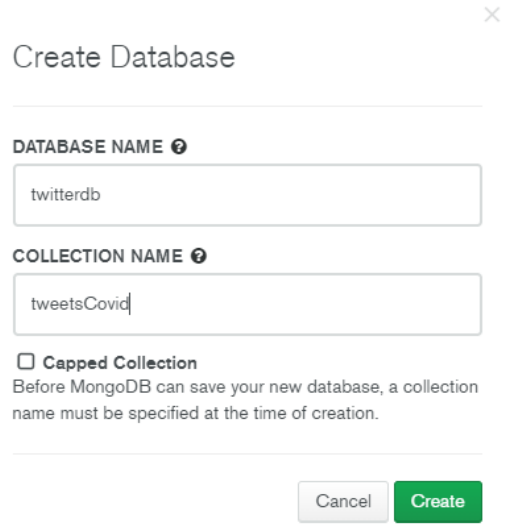
Para la creación de la base de datos fue necesario dirigirse al tablero de control y dar clic sobre el nombre del cluster.



Una vez allí se utilizó el botón **Create Database**.

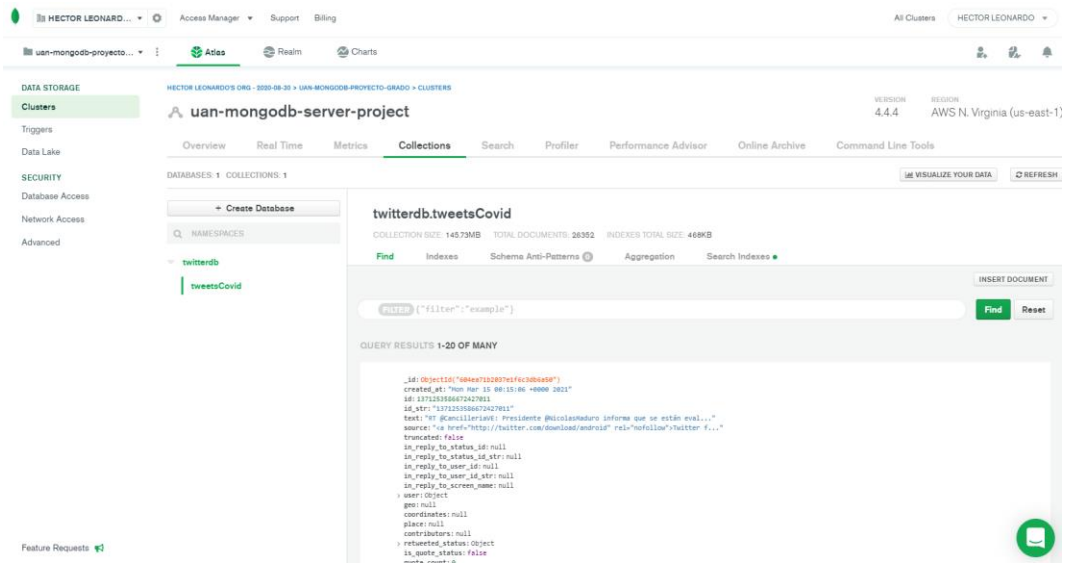


Se definió el nombre de la base de datos identificándola como **twitterdb** y estableciendo también el nombre de la colección donde se almacenaron los Tweets, colección definida como **tweetsCovid**.



Finalmente se puede apreciar la base de datos en funcionamiento. Para el momento en que fue tomada la siguiente imagen ya se habían realizado trabajos de ingesta de datos mediante la API Rest y API Streaming de Twitter y el código en Python ejecutado desde las máquinas locales de los integrantes de este proyecto.

Este es un vistazo final de la configuración y la ejecución de la base de datos MongoDB creada en la nube.



Anexo 2 - Solicitar cuenta de desarrollo en Twitter

Para la obtención de estas claves y el uso de la API es necesario seguir lo siguientes pasos:

1. Crear una cuenta en Twitter.
2. Dirigirse al sitio web https://developer.twitter.com/ y dar clic en el vínculo “Developer Portal” para crear una cuenta de desarrollador en Twitter para el uso de las APIs.

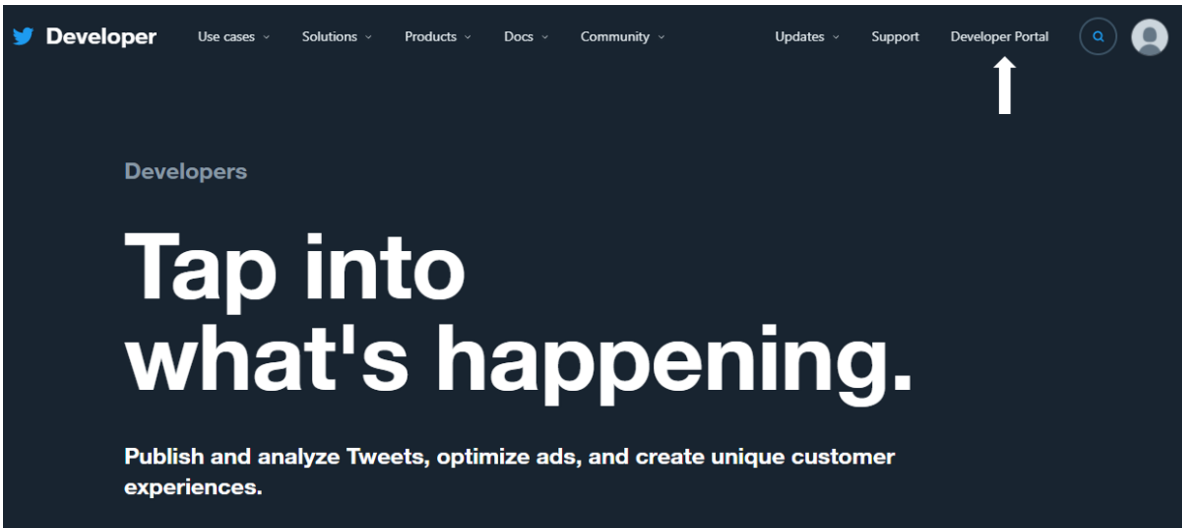


Figura 1. Obtención cuenta desarrollador en Twitter.

3. Llenar el formulario y justificar el uso y los fines que se quieren con la API de Twitter. En este caso se procede a informar que su uso será enteramente académico con el fin de ser utilizado para el proyecto de grado de la Especialización de Gobierno de Datos de la Universidad Antonio Nariño.
4. Revisar el correo electrónico con el cual se registró la cuenta, esto para recibir la confirmación del uso de la API.

Creación de proyecto y uso de las keys y tokens

1. Una vez se confirme el acceso a la plataforma de desarrollo de Twitter basta con dirigirse de nuevo a la opción de "Developer Portal" y crear un nuevo proyecto para el uso de las Keys y Tokens.
2. Al crear el nuevo proyecto, sobre el panel central de la página de desarrolladores de Twitter, se encontrará la opción de **Keys and tokens** y usando el botón de **Regenerate** para cada opción respectivamente genera los códigos que serán usados en el código en Python que se explicará posteriormente en este capítulo.

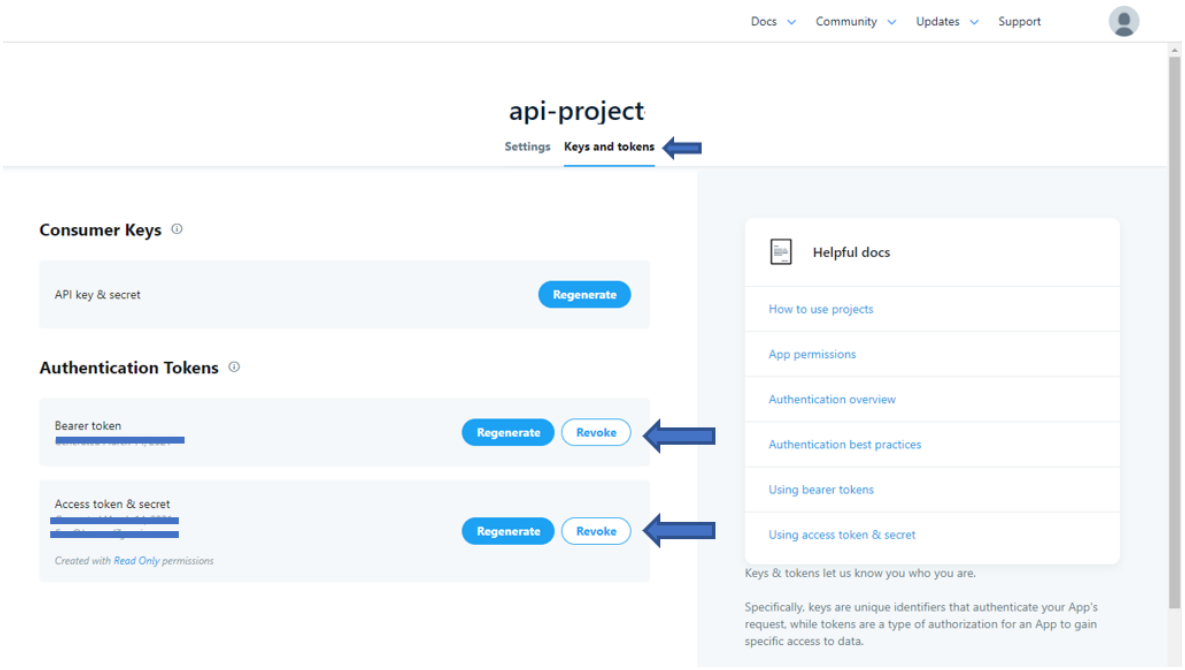


Figura 2. Generación de keys y tokens para la API de Twitter.

Anexo 3 - Captura de datos de Twitter en Python

A continuación, se ilustran los pasos realizados para el proceso de extracción de datos de Twitter, mediante el uso de la API Streaming y la inserción de información en la base de datos MongoDB, implementando las librerías de Python **tweepy** y **pymongo**:

Creación del Proyecto:

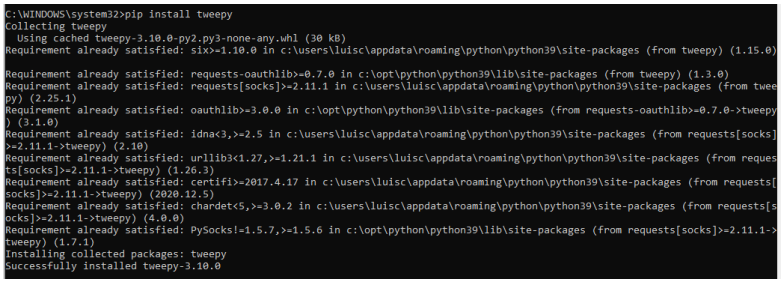
El notebook **1. Extracción y almacenamiento de datos**

Importación de librerías Python necesarias para crear el código correspondiente a la extracción de tweets:

Instalación librería tweepy.

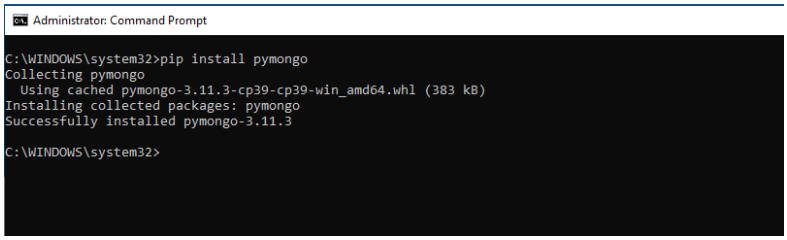
pip install tweepy

Abrir una consola de comando en modo administrador:



Instalación librería **pymongo**, para establecer la conexión con la base de datos MongoDB.

pip install pymongo



Una vez importadas las librerías, se creó la función getAauth() para la autenticación en la API de Twitter:

```python
import tweepy
#Funcion que se utilizara para realizar el proceso de Autenticacion en la API de Twitter
def getAauth():
    consumer_key = 'XXXXXXXXX'
    consumer_secret = 'XXXXXXXXXXXXXX'
    access_token = 'XXXXXXXXXXXXXXXXXXXXXX'
    access_token_secret = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    return auth
```

En este mismo archivo se creó la función conMongoDb() para establecer la cadena de conexión a la Base de Datos MongoDB, retornará un String con la cadena de conexión:

```python
def connMongoDb():
    #host='mongodb+srv://uan-project-admin:ProjectUan2021@uan-mongodb-server-proj.kcyc6.mongodb.net/test'
    #Server 2
    #host='mongodb+srv://uan-project-admin:ProjectUan2021@uan-mongodb-server-proj.1vlyc.mongodb.net/test'
    #Server 3
    host = 'mongodb+srv://uan-project-admin:ProjectUan2021@uan-mongodb-server-proj.l6k2h.mongodb.net/test'
    return host
```

Seguidamente se visualiza el código utilizado para la extracción de los tweets a partir de la API y el almacenamiento respecto a la base de datos de MongoDB.

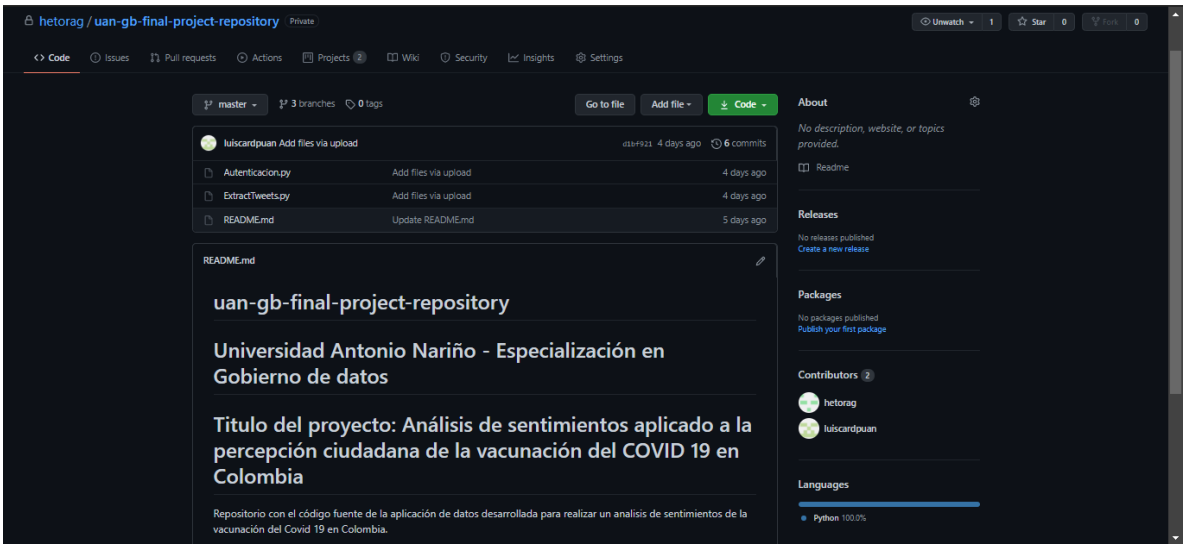
```
from pymongo import MongoClient
import time
#Clase encargada de la extraccion de Tweest mediante al API Streaming y la inservion en la Base de Datos MongoDB
class Extraertweets(tweepy.StreamListener):
    x = 0;
    #Funcion que inprime el mensaje
    def on_connect(self):
        print('Conexion establecida correctamente a la API Streaming!')
    # Funncion que recibe el estado de la conexion con la informacion del tweets que se esta extrayendo
    def on_status(self, status):
        try:
            #print (status._json)
            client = MongoClient(connMongoDb())
            db = client.twitterdb
            db.tweetsCovid.insert(status._json)
            self.x = self.x + 1
            print('Insercion Tweets Nro: ' + str(self.x))
        except:
            return False
#Listado de palabra para realizar el filtro de los tweets
#WORDS = ['vacunacion','vacunación','#vacunacion', '#YoMevacuno', '#YoNoMevacuno','vacuna','#vacunación en Colombia','inyecc
WORDS = ['vacuna','covid','coronavirus']
#Iniciar la Aplicacion
if __name__ == "__main__":
    print('***** Incio programa caprura de Tweets *****')
    i = 1
    while True:
        try:
            print('Intento de extraccion de Tweest Nro. : '+str(i))
            api = tweepy.API(getAauth(), wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
            stream = Extraertweets()
            atreamingApi = tweepy.Stream(auth=api.auth, listener=stream)
            atreamingApi.filter(track=WORDS)
        #Si se presenta algun error en la ejcucion de la captura e insercion de los tweets no se finaliza el programa y lanz
        except:
            i = i + 1
            print('Se genero un error en el intento de extraccion de Tweets Nro: ' + str(i)+' Se enviara una nueva solicitud
            #Esperar 30 segundo para realizar la nueva invocacion a la API
            time.sleep(120)
            continue
    print('***** Fin programa para caprura de Tweets *****')

```

Ejecución del programa:

```
***** Incio programa caprura de Tweets *****
Intento de extraccion de Tweest Nro. : 1
Conexion establecida correctamente a la API Streaming!
<ipython-input-3-c63c4d0be642>:15: DeprecationWarning: insert is deprecated. Use insert_one or insert_many instead.
  db.tweetsCovid.insert(status._json)
Insercion Tweets Nro: 1
Insercion Tweets Nro: 2
Insercion Tweets Nro: 3
Insercion Tweets Nro: 4
Insercion Tweets Nro: 5
Insercion Tweets Nro: 6
Intento de extraccion de Tweest Nro. : 1
Conexion establecida correctamente a la API Streaming!
Intento de extraccion de Tweest Nro. : 1
Conexion establecida correctamente a la API Streaming!
Intento de extraccion de Tweest Nro. : 1
Conexion establecida correctamente a la API Streaming!
Intento de extraccion de Tweest Nro. : 1
```

Anexo 4 - Versionamiento de código fuente

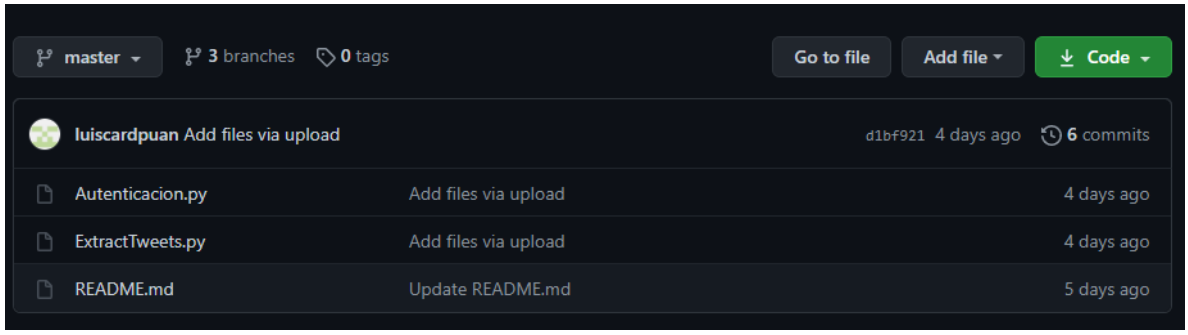
Para el versionamiento para el código fuente se utilizó la herramienta GITHUB que es un repositorio que permite administrar la codificación del proyecto y todos los demás componentes asociados al proyecto.



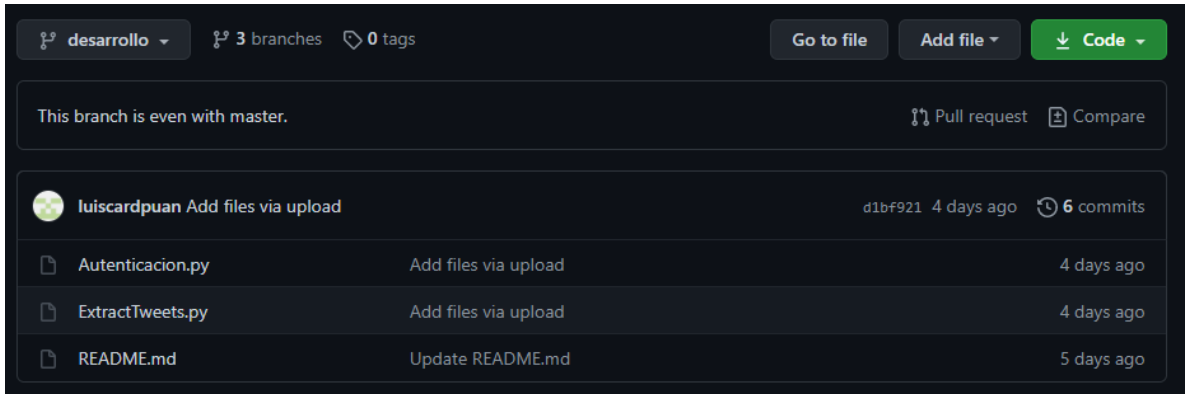
Con el fin de salvaguardar y administrar todo el contenido creado durante el desarrollo del proyecto, se utilizó esta plataforma GITHUB en la nube la cual permite administrar las fuentes y realizar versionamiento del código agrupando esté en ramas diferentes con el fin de realizar un correcto gobierno sobre el material creado.

Se crearon tres ramas (branch) en el proyecto definidos de la siguiente manera:

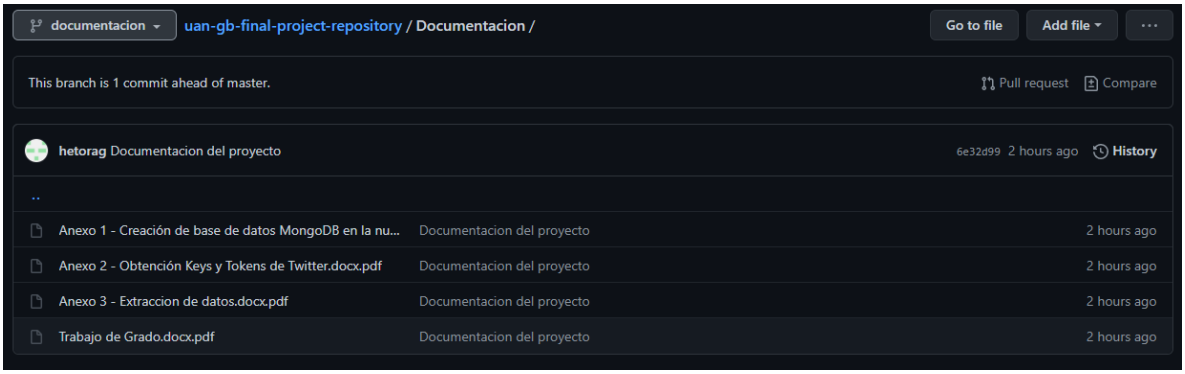
1. master: Esté branch contiene las clases del código fuente definitivo para hacer uso del mismo en los temas de extracción, preprocesamiento y procesamiento de datos.



2. desarrollo: Este branch contiene las clases de código fuente que se encuentran en desarrollo o modificadas, con el fin de realizar las pruebas necesarias sin alterar las clases definitivas. Una vez cumplan con los requerimientos y las pruebas respectivas se procederá a integrarlas (realizar merge) con el branch **master** como versión definitiva de código hasta el momento.



3. Documentacion: Este branch contiene toda la documentación generada a lo largo del proyecto. De acuerdo con la evolución de esta, el contenido en este branch será actualizado constantemente para contar con la versión más reciente de la documentación, así como se hace con el código fuente.


documentacion
uan-gb-final-project-repository / Documentacion /
Go to file
Add file
This branch is 1 commit ahead of master.
Pull request
Compare
hetorag Documentacion del proyecto
6e32d99 2 hours ago
History
..
Anexo 1 - Creación de base de datos MongoDB en la nu... Documentacion del proyecto 2 hours ago
Anexo 2 - Obtención Keys y Tokens de Twitter.docx.pdf Documentacion del proyecto 2 hours ago
Anexo 3 - Extraccion de datos.docx.pdf Documentacion del proyecto 2 hours ago
Trabajo de Grado.docx.pdf Documentacion del proyecto 2 hours ago

Anexo 5 - Preprocesamiento

A continuación, se indican los pasos realizados en la etapa de preprocesamiento. Importación de librerías.

In [2]:
```
import csv
import pymongo
from dateutil.parser import parse
from pymongo import MongoClient
import pandas as pd
import re
import numpy as np
```

Definición de función **clean**(), para eliminar emoticones y saltos de línea.

In [3]:
```
#Quita Emoticones y Saltos de Linea
def clean(text):
    text_clean = ''
    if text == '' or pd.isnull(text):
        text_clean=''
    else:
        auxiliar = text.replace('\n', '').replace('\r', '')
        a = [text[j] for j in range(len(auxiliar))]
        for aux in a:
            if ord(aux) in range(32, 126) or ord(aux) in range(128, 254) :
                text_clean = text_clean + aux
    return text_clean.strip() #Quita Espacios al principio y al final
```

Definición de función **remove_url()**, para eliminar direcciones url y referencias a cuentas de tweet.

```

#Quita URL y @
def remove_url(tweet):
    text_clean = ''
    if tweet == '' or pd.isnull(tweet):
        text_clean=''
    else:
        #Reemplazar Saltos de Linea y retorno de carro
        auxiliar = tweet.replace('\n', '').replace('\r', '')
        #Quitar URL
        auxiliar = re.sub("http\S+", "", auxiliar)
        #Quitar las @
        auxiliar = re.sub("@\S+", "", auxiliar)
        text = auxiliar
        a = [text[j] for j in range(len(text))]
        text_clean = ''
        for aux in a:
            if ord(aux) in range(32, 126) or ord(aux) in range(128, 254) :
                text_clean = text_clean + aux
        #Quitar Doble Espacio
        text_clean = ' '.join(text_clean.split())
    return text_clean.strip() #Quita Espacios al principio y al final
```

Definición de función que retorna la cadena de conexión de los servidores de MongoDB.

In [8]:
```
def connMongoDb(indice):
    if indice==1:
        host='mongodb+srv://uan-project-admin:ProjectUan2021@uan-mongodb-server-proj.kcyc6.mongodb.net/test'
    #Server 2
    if indice==2:
        host='mongodb+srv://uan-project-admin:ProjectUan2021@uan-mongodb-server-proj.1vlyc.mongodb.net/test'
    #Server 3
    if indice==3:
        host = 'mongodb+srv://uan-project-admin:ProjectUan2021@uan-mongodb-server-proj.l6k2h.mongodb.net/test'
    return host
```

Código fuente para obtener información de los tweets almacenados en los servidores de MongoDB, y por cada uno crear un dataframe:

In [9]:

```
rows = []
for i in [1,2,3]:
    rgx = re.compile('.*Colom.*', re.IGNORECASE)  # compile the regex
    myclient = pymongo.MongoClient(connMongoDb(i))
    mydb = myclient["twitterdb"]
    mycol = mydb["tweetsCovid"]
    myquery = {"user.location":rgx,"text":{"$exists":True}}
    mydoc = mycol.find(myquery)
    for tw in mydoc:
        myList = []
        myList.append(parse(tw["created_at"]))#FechaCreacion
        myList.append(tw["id_str"])#Id_tweest
        myList.append(tw["truncated"])#Truncado
        myList.append(tw["user"]["id_str"])#Id_Usuario
        myList.append(clean(tw["user"]["name"]))#Nombre_Usuario
        myList.append(clean(tw["user"]["location"]))#Ubicacion
        myList.append(clean(tw["text"]))#Texto
        if 'quoted_status' in tw:
          myList.append(parse(tw["quoted_status"]["created_at"]))#Q_FechaCreacion
          myList.append(tw["quoted_status"]["id_str"])#Q_Id_tweest
          myList.append(tw["quoted_status"]["truncated"])#Q_Truncado
          myList.append(tw["quoted_status"]["user"]['id_str'])#Q_Id_Usuario
          myList.append(clean(tw["quoted_status"]["user"]['name']))#Q_Nombre_Usuario
          myList.append(clean(tw["quoted_status"]["user"]['location']))#Q_Ubicacion
          if tw["quoted_status"]["truncated"]==True:
            myList.append(clean(tw["quoted_status"]["extended_tweet"]["full_text"]))#Q_Texto
          else:
              myList.append(clean(tw["quoted_status"]["text"]))#Q_Texto
        else:
           myList.append('')#Q_FechaCreacion
           myList.append('')#Q_Id_tweest
           myList.append('')#Q_Truncado
           myList.append('')#Q_Id_Usuario
           myList.append('')#Q_Nombre_Usuario
           myList.append('')#Q_Location
           myList.append('')#Q_Texto
        if 'retweeted_status' in tw:
          myList.append(parse(tw["retweeted_status"]["created_at"]))#R_FechaCreacion
          myList.append(tw["retweeted_status"]["id_str"])#R_Id_tweest
          myList.append(tw["retweeted_status"]["truncated"])#R_Truncado
          myList.append(tw["retweeted_status"]["user"]['id_str'])#R_Id_Usuario
          myList.append(clean(tw["retweeted_status"]["user"]['name']))#R_Nombre_Usuario
          myList.append(clean(tw["retweeted_status"]["user"]['location']))#R_Ubicacion
          if tw["retweeted_status"]["truncated"]==True:
            myList.append(clean(tw["retweeted_status"]["extended_tweet"]["full_text"]))#R_Texto
          else:
              myList.append(clean(tw["retweeted_status"]["text"]))#R_Texto
        else:
           myList.append('')#R_FechaCreacion
           myList.append('')#R_Id_tweest
           myList.append('')#R_Truncado
           myList.append('')#R_Id_Usuario
           myList.append('')#R_Nombre_Usuario
           myList.append('')#R_Location
           myList.append('')#R_Texto
        rows.append(myList)
    #Server 2
    if i==1:
        df1 = pd.DataFrame(rows, columns = ['FechaCreacion','Id_tweest','Truncado','Id_Usuario','Nombre_Usuario','Ubicacion'
    #Server 2
    if i==2:
        df2 = pd.DataFrame(rows, columns = ['FechaCreacion','Id_tweest','Truncado','Id_Usuario','Nombre_Usuario','Ubicacion'
    #Server 3
    if i==3:
        df3 = pd.DataFrame(rows, columns = ['FechaCreacion','Id_tweest','Truncado','Id_Usuario','Nombre_Usuario','Ubicacion'

```

Código fuente para consolidar los dataframes con la información que se generó en el punto anterior a partir de este se creó el archivo .csv **Base_Original_1.csv**; se generó otro dataframe seleccionando las columnas **'FechaCreacion', 'Id_tweest', 'Ubicacion', 'Texto', 'R_Texto'** que fueron objeto del análisis, se aplicó una primera limpieza general aplicando la función **remove_url**, se eliminaron registros duplicados, se agregó una columna adicional **'Tipo'** para clasificar el texto del tweets como **tweets** o **retweet**, se suprimieron los registros que no contienen la palabra "**vacun**"

y finamente se agregó la columna '**Index**' para numerar los registros, con esta infoarmacion se creó el archivo .csv Base_Original.csv, que fue utilizado para la clasificación anual de los tweet.

```
#Consolidacion de Dataframes
df=pd.concat([df1, df2, df3], axis=0)

#Descargar Archivo CSV consolidado
df.to_csv(r'C:\opt\ProyectoGrado\notebooks\Entrenamiento\Archivos\Data\Base_Original_1.csv', index = False, header=True,enco

df_aux=df.loc[:, ['FechaCreacion','Id_tweest','Ubicacion','Texto','R_Texto']]

#remover URL y @NombreUsuario a la Columna R_Texto
df_aux['R_Texto'] = df_aux['R_Texto'].apply(lambda x: remove_url(x))

#remover URL y @NombreUsuario
df_aux['Texto'] = df_aux['Texto'].apply(lambda x: remove_url(x))
#Eliminar filas vacias y que contengan un caracter de la columna Texto
df_aux = df_aux.drop(df_aux[(df_aux['Texto'].str.len()<=1)].index)

#Quitar duplicados de la columna Texto
df_aux=df_aux.drop_duplicates(subset=['Texto'])

#Agregar columana Tipo para clasificar el tweet
df_aux['Tipo'] = np.where(df_aux['Texto'].str.contains("RT "), 'Retweet', 'Tweet')

#Actualizar la Columna R_Texto las que esten vacias se agregara el Text
#df_aux['R_Texto'] = np.where(df_aux['R_Texto'].str.len()<=0, df_aux['Texto'], df_aux['R_Texto'])
df_aux['R_Texto'] = np.where(df_aux['Tipo']=='Tweet', df_aux['Texto'], df_aux['R_Texto'])

#Eliminacion de rgistros que no tengan la palabta vacun
df_aux = df_aux.drop(df_aux[~df_aux['Texto'].str.contains("vacun")].index)

#Agregar la columna Index con el indice
df_aux=df_aux.rename_axis('Index').reset_index()
df_aux['Index']=df_aux.reset_index()

df_aux.to_csv(r'C:\opt\ProyectoGrado\notebooks\Entrenamiento\Archivos\Data\Base_Original.csv', index = False, header=True,en

```

La siguiente imagen ilustra la estructura del dataframe que se consolido

Out[16]:

	FechaCreacion	Id_tweest	Truncado	Id_Usuario	Nombre_Usuario	Ubicacion	Texto	Q_FechaCreacion	Q_Id
0	2021-03-15 00:15:08+00:00	1371253596411588620	False	406689798	Dr Chapid	Colombia	RT @MrDoctorOficial: ¿CÓMO ME RECUPERÉ DE #COV...		
1	2021-03-15 00:15:09+00:00	1371253598638768128	False	20945781	Natalia Baquero	Bogotá, Colombia	RT @solsilvanazb: Respetado @MinSaludCol si ha...	2021-03-15 00:00:49+00:00	137124999166
2	2021-03-15 00:15:28+00:00	1371253678330494979	True	1300202339064778752	DAI COFFEE	Bogotá - Colombia	WEBINAR EXCLUSIVO PARA VENDEDORES DE #CAFÉ TOS...		
3	2021-03-15 00:15:44+00:00	1371253746903175169	False	2359763479	jbarrios107	Colombia	RT @JuanGnusmas: Los test serológicos miden la...		
4	2021-03-15 01:53:20+00:00	1371278308487348228	False	2429579405	Bernardo Soto García	Bogotá, Colombia	RT @LiteraturayMas_: Los científicos del momen...		
...	...	...	...	...	...	...	...	...	
3818	2021-04-25 02:12:46+00:00	1386141103238299648	True	731212165	Juan G. Valencia	Barranquilla, Colombia	@jaimepumarejo @SecSaludBAQ Una vacunación sin...		

La siguiente imagen ilustrar el dataframe que se usó para crear el archivo **Base_Original.csv**

Out[22]:

	Index	FechaCreacion	Id_tweest	Ubicacion	Texto	R_Texto	Tipo
0	0	2021-03-15 01:57:03+00:00	1371279244534374401	Bogota Colombia	RT [Video] Engañan a otra abuela y no le aplic...	[Video] Engañan a otra abuela y no le aplican ...	Retweet
1	1	2021-03-15 01:57:18+00:00	1371279307088261120	Barranquilla, Colombia	RT y la vacuna pa cuando? @	y la vacuna pa cuando?	Retweet
2	2	2021-03-15 01:58:38+00:00	1371279641009356800	colombia, Córdoba-Monteria	RT Como Gobernador solícito q revisen bien las...	Como Gobernador solícito q revisen bien las ci...	Retweet
3	3	2021-03-15 01:58:38+00:00	1371279642255110146	Valledupar, Cesar, Colombia	RT Los que andaban criticando a por la vacunac...	Los que andaban criticando a por la vacunación...	Retweet
4	4	2021-03-15 01:58:55+00:00	1371279713080115204	Bogotá, D.C., Colombia	RT ATENCIÓN : Irlanda recomienda suspender por...	ATENCIÓN : Irlanda recomienda suspender por "p...	Retweet
...	...	...	...	...	...	...	...
1500	1500	2021-04-25 02:12:37+00:00	1386141065057501188	Colombia	Busca en la página del ministerio los puntos d...	Busca en la página del ministerio los puntos d...	Tweet
1501	1501	2021-04-25 02:12:46+00:00	1386141103238299648	Barranquilla, Colombia	Una vacunación sin agendamiento es lo ideal, d...	Una vacunación sin agendamiento es lo ideal, d...	Tweet
1502	1502	2021-04-25 18:40:13+00:00	1386389601980346368	Colombia	Bien por el saludo... Favor no descuidar el pr...	Bien por el saludo... Favor no descuidar el pr...	Tweet
1503	1503	2021-04-25 18:40:15+00:00	1386389609651769344	Bogotá - Colombia	Interesante opinión sobre el plan vacunación ¿...	Interesante opinión sobre el plan vacunación ¿...	Tweet
1504	1504	2021-04-25 18:40:17+00:00	1386389617377644544	Ibagué, Colombia	RT Oites Tola, esperar la segunda dosis de la ...	Oites Tola, esperar la segunda dosis de la vac...	Retweet

1505 rows × 7 columns

Nota: Los archivos .csv fueron creados en una ruta local del equipo.

Anexo 6 - Procesamiento

A continuación, se indican los pasos realizados en la etapa de Procesamiento. Importación de librerías.

```
import csv
import re
import pandas as pd
import numpy as np
import nltk
#nltk.download('punkt')
#nltk.download('stopwords')
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
from nltk.tokenize import word_tokenize
from nltk.stem import SnowballStemmer
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.svm import SVC
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

import pylab as pl
from tabulate import tabulate
from sklearn.datasets.samples_generator import make_blobs
import warnings
warnings.simplefilter(action="ignore", category=FutureWarning)
```

Creación de funciones para limpieza y procesamiento de datos. Función **clean_tweets** para realizar limpieza de tweet, quitar caracteres especiales, números, dobles espacios, urls, referencias a cuentas de usuarios etc.

```
#Funcion para limpiar Tweets
def clean_tweets(tweet):
    text_clean = ''
    if tweet == '' or pd.isnull(tweet):
        text_clean=''
    else:
        #Reemplazar Saltos de Linea y retorno de carro
        auxiliar = tweet.replace('\n', '').replace('\r', '')
        #Convertir texto a munuscula
        auxiliar = auxiliar.lower()
        #Quitar URL
        auxiliar = re.sub("http\S+", "", auxiliar)
        #Quitar las @
        auxiliar = re.sub("@\S+", "", auxiliar)
        text = auxiliar
        a = [text[j] for j in range(len(text))]
        text_clean = ''
        #Quitar caracteres especiales, numeros, emoticones, etc dejar solo dejar letras, espacios, letra Ñ y letras tildadas
        #ord(aux) in range(97, 123) Permitir numeros
        for aux in a:
            if ord(aux) in range(97, 123) \
                    or ord(aux) in range(48, 58) \
                    or ord(aux) in range(32, 33) \
                    or ord(aux) in range(241,242) \
                    or ord(aux) in [225,233,237,243,250,64]:
                text_clean = text_clean + aux
        #Eliminar numeros
        text_clean = re.sub("\d+", '', text_clean)
        # Quitar doble espacio
        text_clean = ' '.join(text_clean.split())
    return text_clean
```

Función **removeStopwords** para remover palabras más comunes de la lengua y otras palabras que no se consideran relevantes para el proceso de análisis:

```python
#Funcion para remover StopWords
def removeStopwords(text):
    spanish_stopwords = stopwords.words('spanish')#Configurar las palabra mas comunes en español
    spanish_stopwords.extend(("años", "año"))#Se agregan otras palabras como años y año
    spanish_stopwords
    text=' '+text+' '
    for word in spanish_stopwords:
            token = ' ' + word + ' '
            text = text.replace(token, ' ')
    return text.strip()
```

Función **steaming** para recortar palabras a su raíz, se parametrizo la función para configurar las palabras en español.

```python
#Funcion para Steming, recortar las palabras a su raíz
def steaming(text):
    token_words=word_tokenize(text)
    ps = SnowballStemmer('spanish')
    token_words
    aux_text=''
    for word in token_words:
        #print(word," : ",ps.stem(word))
        aux_text=aux_text+' '+ps.stem(word)
    return aux_text.strip()

```

Función **removeWord** para eliminar letras repetidas, ya que al realizar la clasificación manual del tweet se identificó que los usuarios utilizaban palabras como **aburridooooo**, esta función eliminaba las letras repetidas y reemplaza letras tildes por letras sin tildes, por ejemplo, la palabra aburrido se reemplazaba por **aburido**, aunque se reemplazó la letra r no se perdió el contexto de la palabra ya que se pretendía estandarizar el texto.

```python
#Funcion para remover letras repetidas y reemplazar letras con tilde
def removeWord(texto):
    NewWord = " "
    palabra=""
    index = 0
    for char in texto:
        if char != NewWord[index]:
            NewWord += char
            index += 1
    NewWord = NewWord.replace('á', 'a').replace('é', 'e').replace('í', 'i').replace('ó', 'o').replace('ú', 'u')
    return NewWord.strip()
```

Función **tokenizar_tweets**, se utilizó básicamente para eliminar aquellas letras que tiene una longitud menor a 2, ya que estas letras en la oración no aportan ningún valor, al momento de realizar la clasificación manual de los tweets, se identificaron letras como q, por lo tanto, se interpretó que el usuario que escribió el tweet quería expresar la palabra que.

```python
def tokenizar_tweets(texto):
    new_texto=texto
    # Tokenización por palabras individuales
    new_texto = new_texto.split(sep=' ')
    # Se eliminan tokens con una longitud  2
    new_texto = [token for token in new_texto if len(token) > 2]
    aux_text=''
    for word in new_texto:
        #print(word," : ",ps.stem(word))
        aux_text=aux_text+' '+word
    return aux_text.strip()

```

El siguiente paso después importar las librerías necesarias para el procesamiento y definir las funciones anteriormente descritas se creó un dataframe para asignar como origen de datos el archivo .csv **Base_original.csv**, el cual contiene la información de los tweets que se clasificaron manualmente.

La siguiente imagen ilustra el código fuente y la estructura del datafreme:

```python
#Lectura del archivo CSV - Contine la informacion de todos los tweets que se recolectaron
twest=pd.read_csv(r'C:\opt\ProyectoGrado\notebooks\Entrenamiento\Archivos\Data\BaseEntrenamiento\Base_Original.csv', encodin
tw=twest
tw
```

3	3	2021-03-15 01:58:38+00:00	1.371280e+18	Valledupar, Cesar, Colombia	RT Los que andaban criticando a por la vacunac...	Los que andaban criticando a por la vacunación...	Retweet	-1
4	4	2021-03-15 01:58:55+00:00	1.371280e+18	Bogotá, D.C., Colombia	RT ATENCIÓN : Irlanda recomienda suspender por...	ATENCIÓN : Irlanda recomienda suspender por "p...	Retweet	-1
...	...	...	...	...	...	...	...	...
1500	1500	2021-04-25 02:12:37+00:00	1.386140e+18	Colombia	Busca en la página del ministerio los puntos d...	Busca en la página del ministerio los puntos d...	Tweet	1
1501	1501	2021-04-25 02:12:46+00:00	1.386140e+18	Barranquilla, Colombia	Una vacunación sin agendamiento es lo ideal, d...	Una vacunación sin agendamiento es lo ideal, d...	Tweet	1
1502	1502	2021-04-25 18:40:13+00:00	1.386390e+18	Colombia	Bien por el saludo... Favor no descuidar el pr...	Bien por el saludo... Favor no descuidar el pr...	Tweet	1
1503	1503	2021-04-25 18:40:15+00:00	1.386390e+18	Bogotá - Colombia	Interesante opinión sobre el plan vacunación ¿...	Interesante opinión sobre el plan vacunación ¿...	Tweet	0
1504	1504	2021-04-25 18:40:17+00:00	1.386390e+18	Ibagué, Colombia	RT Oites Tola, esperar la segunda dosis de la ...	Oites Tola, esperar la segunda dosis de la vac...	Retweet	0

1505 rows × 8 columns

Seguidamente se aplicaron las funciones de limpieza y estandarización a la columna **R_Text**, que fue objeto de estudio para aplicación del análisis de sentimientos:

```python
#Aplica la Funcion clean_tweets a la columnas R_Texto para Limpieza de los tweets
tw['R_Texto'] = tw['R_Texto'].apply(clean_tweets)

```

```python
#Aplica la Funcion removeWord para eliminar las letras repetidas
#tw = tw.drop_duplicates(subset=['R_Texto'])
tw['R_Texto'] = tw['R_Texto'].apply(removeWord)

```

```python
#Aplicar la funcion de removeStopwords,
tw['R_Texto']=tw['R_Texto'].apply(removeStopwords)
```

```python
#Recortar las plabras a su raiz
tw['R_Texto']=tw['R_Texto'].apply(steaming)

```

```python
tw['R_Texto'] = tw['R_Texto'].apply(lambda x: tokenizar_tweets(x))
tw

```

Luego de ejecutar el código anterior se puede evidenciar como se estandarizo el texto de la columna **R_Text**:

	Index	FechaCreacion	Id_tweest	Ubicacion	Texto	R_Texto	Tipo	Sentiment
0	0	2021-03-15 01:57:03+00:00	1371279244534374401	Bogota Colombia	RT [Video] Engañan a otra abuela y no le aplic...	vide engañ abuel aplic vacun medelin jering vaci	Retweet	-1
1	1	2021-03-15 01:57:18+00:00	1371279307088261120	Barranquilla, Colombia	RT y la vacuna pa cuando? @	vacun	Retweet	-1
2	2	2021-03-15 01:58:38+00:00	1371279641009356800	colombia, Córdoba-Monteria	RT Como Gobernador solícito q revisen bien las...	gobern solicit revis bien cifr estan public re...	Retweet	0
3	3	2021-03-15 01:58:38+00:00	1371279642255110146	Valledupar, Cesar, Colombia	RT Los que andaban criticando a por la vacunac...	andab critic vacunacion enter mas pais vacun	Retweet	-1
4	4	2021-03-15 01:58:55+00:00	1371279713080115204	Bogotá, D.C., Colombia	RT ATENCIÓN : Irlanda recomienda suspender por...	atencion irland recomiend suspend precaucion v...	Retweet	-1
...	...	...	...	...	...	...	...	...
1500	1500	2021-04-25 02:12:37+00:00	1386141065057501188	Colombia	Busca en la página del ministerio los puntos d...	busc pagin ministeri punt vacunacion autoriz	Tweet	1
1501	1501	2021-04-25 02:12:46+00:00	1386141103238299648	Barranquilla, Colombia	Una vacunación sin agendamiento es lo ideal, d...	vacunacion agend ideal deberi prolong inici asi	Tweet	1
1502	1502	2021-04-25 18:40:13+00:00	1386389601980346368	Colombia	Bien por el saludo... Favor no descuidar el pr...	bien salud favor descuid proces vacunacion	Tweet	1
1503	1503	2021-04-25 18:40:15+00:00	1386389609651769344	Bogotá - Colombia	Interesante opinión sobre el plan vacunación ¿...	interes opinion plan vacunacion chil estan dif...	Tweet	0
1504	1504	2021-04-25 18:40:17+00:00	1386389617377644544	Ibagué, Colombia	RT Oites Tola, esperar la segunda dosis de la ...	oit tol esper segund dosis vacun duch tod enja...	Retweet	0

1505 rows × 8 columns

Teniendo la información estandarizada, se realizaron cálculos generales con el fin de comprender la información contenida en el dataframe, la siguiente figura muestra la clasificación de la cantidad de tweet y retweet.

```
#Grafica con clasificacion de las etiquetas de los tweets de la columna Sentiment
import seaborn as sb
sb.catplot('Tipo',data=tw,kind="count")
```

Según la imagen anterior se puede apreciar el número de tweet y retweet del dataframe, a continuación, se ilustra el numero asignado a cada categoría y el porcentaje que le corresponde de acuerdo con el total de registros clasificados.

```
tw.groupby("Tipo")["Tipo"].count()
```

```
Tipo
Retweet    896
Tweet      609
Name: Tipo, dtype: int64
```

```
tw.groupby("Tipo")["Tipo"].count()/len(tw)
```

```
Tipo
Retweet    0.595349
Tweet      0.404651
Name: Tipo, dtype: float64
```

Teniendo una aproximación sobre la estructura de la base anterior, se creó el dataframe **df_tw**, al cual se asignaron las columnas **R_Text** y **Sentiment**, como se ilustra en la siguiente figura, este se utilizará para los cálculos finales.

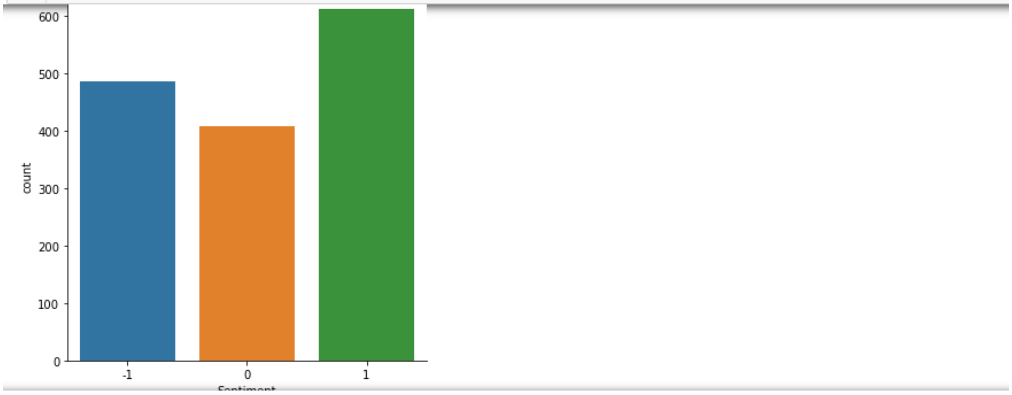
```
#Creacion del dataframe df_tw con las columnas R_Text y Sentiment:
df_tw=tw.loc[:, ['R_Texto','Sentiment']]
df_tw=df_tw.reset_index(drop=True)
df_tw
```

	R_Texto	Sentiment
0	vide engañ abuel aplic vacun medelin jering vaci	-1
1	vacun	-1
2	gobern solicit revis bien cifr estan public re...	0
3	andab critic vacunacion enter mas pais vacun	-1
4	atencion irland recomiend suspend precaucion v...	-1
...	...	...
1500	busc pagin ministeri punt vacunacion autoriz	1
1501	vacunacion agend ideal deberi prolong inici asi	1
1502	bien salud favor descuid proces vacunacion	1
1503	interes opinion plan vacunacion chil estan dif...	0
1504	oit tol esper segund dosis vacun duch tod enja...	0

1505 rows × 2 columns

Se utilizó el dataframe creado en el punto anterior para medir el número de tweet clasificados como positivos, negativos y neutrales:

```
#Grafica con clasificacion de las etiquetas de los tweets de la columna Sentiment
import seaborn as sb
sb.catplot('Sentiment',data=df_tw,kind="count")
```



De igual forma se estableció el número total de tweets clasificados, según la categoría y el porcentaje de acuerdo con el total de registros:

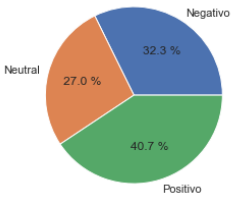
```
#Calcular el numero de Tweets calsificados como; Negativo=-1, Positivo=1, Neutral=0, en la columna Sentiment
df_tw.groupby("Sentiment")["Sentiment"].count()
```

```
Sentiment
-1    486
 0    407
 1    612
Name: Sentiment, dtype: int64
```

```
#Calcular el porcentaje de el Sentimiento que se especifica en la columna Sentiment: Negativo=-1, Positivo=1, Neutral=0
df_tw.groupby("Sentiment")["Sentiment"].count()/len(df_tw)
```

```
Sentiment
-1    0.322924
 0    0.270432
 1    0.406645
Name: Sentiment, dtype: float64
```

```
sentimiento = df_tw.groupby("Sentiment")["Sentiment"].count()/len(df_tw)
etiquetas = ["Negativo","Neutral","Positivo"]
plt.pie(sentimiento, labels=etiquetas, autopct="%0.1f %%")
plt.show()
```



El siguiente paso consistió en calcular la frecuencia de las palabras más usadas, para este proceso se creó una función **dic_frecuecia_palabras**, que recibe la lista de palabras y retorna un diccionario con la frecuencia de cada una, este diccionario se convierte a un datafreme, para mostrar las 20 palabras más frecuentes.

```
#Se crea una funcion que retorna un diccionario con la lista de frecuencia de las palbras
def dic_frecuecia_palabras(listaPalabras):
    listaPalabras = listaPalabras.split()
    frecuenciaPalab = [listaPalabras.count(w) for w in listaPalabras]
    return dict(list(zip(listaPalabras, frecuenciaPalab)))
```

```
#Se crea una lista de palabras a partir de la variable data que contiene el corpus
lista = df_tw['R_Texto'].tolist() #palabras Normales
#Crear lista de plabras
comment_words=''
for x in range(0,len(lista)):
    tokens = lista[x]
    comment_words += "".join(tokens)+" "
comment_words=comment_words.strip()#Quitar espacios al principio y al final
```

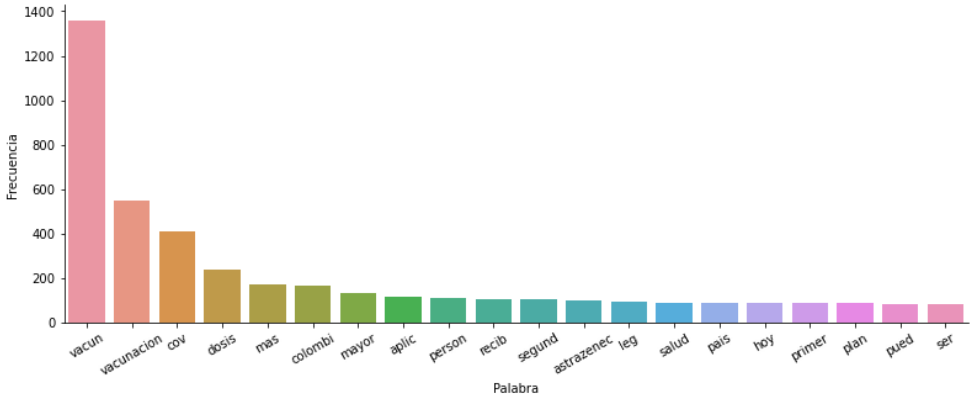
Se crea el dataframe y se muestran las 20 palabras con mayor frecuencia.

```
#Aplicar la funcion dic_frecuecia_palabras para optener la frecuencia de las palabras y crear un datasetframe
cadenaPalabras=comment_words
dic=dic_frecuecia_palabras(comment_words)
#Convertir el Diccionario en un dataframe
df_f = pd.DataFrame(list(dic.items()),columns = ['Palabra','Frecuencia'])
#Mostrar las 20 palabras con mayor frecuencia las que mas se usan en los tweets
aux1=df_f.sort_values(by=['Frecuencia'], ascending=False).head(20)
aux1
```

	Palabra	Frecuencia
4	vacun	1360
34	vacunacion	549
84	cov	411
26	dosis	239
36	mas	171
164	colombi	164
169	mayor	129
3	aplic	116
67	person	107
286	recib	102
125	segund	102
43	astrazenec	100

La siguiente imagen es una ilustración mediante grafico de barras de la frecuencia de las palabras calculadas anteriormente

```
g = sns.catplot(x="Palabra",
                y="Frecuencia",
                kind="bar",
                data=aux1)
g.set_xticklabels(rotation=30)
g.fig.set_size_inches(12,4)
```



El siguiente paso que se realizo fue aplicar el análisis de sentimientos, aplicando la metodología de análisis supervisado y los algoritmos de machine learning (**LogisticRegression, GaussianNB, SVC, RandomForestClassifier y DecisionTreeClassifier**) para medir comportamiento de los modelos seleccionados, con el fin de establecer si presentan un desempeño adecuado en las predicciones.

Para la aplicación de los modelos indicados anteriormente, se estableció el corpus con los datos de análisis, para crear la matriz de frecuencias, y a partir de esta establecer el porcentaje de entrenamiento y prueba de los datos, para evaluar los modelos indicados. A continuación, se ilustra el código fuente aplicado a esta fase.

Es importante tener en cuenta que para la aplicación de los algoritmos de maching learning, se evaluaron los modelos con los valores predeterminados para cada algoritmo, evitando realizar sobreajustes, ya que la finalidad es medir el desempeño de cada método con sus parámetros por defecto.

Creación del corpus

```
#Crear el Corpus que corresponde a la informacion de la columna R_Text - para aplicar el analisis de sentimiento
data=df_tw['R_Texto']
data
```

```
0        vide engañ abuel aplic vacun medelin jering vaci
1                                                   vacun
2       gobern solicit revis bien cifr estan public re...
3            andab critic vacunacion enter mas pais vacun
4       atencion irland recomiend suspend precaucion v...
                              ...
1500         busc pagin ministeri punt vacunacion autoriz
1501      vacunacion agend ideal deberi prolong inici asi
1502           bien salud favor descuid proces vacunacion
1503    interes opinion plan vacunacion chil estan dif...
1504    oit tol esper segund dosis vacun duch tod enja...
Name: R_Texto, Length: 1505, dtype: object
```

Creación de matriz de frecuencia, esta matriz no contiene la columna del sentimiento "**Sentiment**", esta columna se agregará a la matriz en el siguiente paso.

```
#Creacion de la matriz de Frecuencias en un DataFrame
#Se Filtran las palabras que tienen una frecuencia mayor o igual a 10 es decir que se repitan 10 o mas veces
cv=CountVectorizer(min_df=10)
data=cv.fit_transform(data)
#Crear un DataFrame con las palabrar y la veces que aparecen
m_fre_tw=pd.DataFrame(data.toarray(),columns=cv.get_feature_names())
m_fre_tw
```

	abril	abuel	abuelit	aca	acab	aceler	acompañ	activ	actual	acuerd	...	vam	van	ver	vez	via	vid	vide	virus	viv	yomevacun
0	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1500	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1501	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1502	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1503	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1504	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

1505 rows × 354 columns

Se agrega la columna “Sentiment” a la matriz de frecuencias

```
#Se optiene la columna Sentiment, que contiene la clasificacion del sentimiento y se agrega a la Matriz
df_sent=df_tw.loc[:, ['Sentiment']]
#Se realiza el join entre los dos dataframe
df=df_sent.join(m_fre_tw)
df

```

	Sentiment	abril	abuel	abuelit	aca	acab	aceler	acompañ	activ	actual	...	vam	van	ver	vez	via	vid	vide	virus	viv	yomevacun
0	-1	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
1	-1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	-1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	-1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1500	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1501	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1502	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1503	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1504	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

1505 rows × 355 columns

Luego de tener la matriz de frecuencia, se dividió está en 80% de los datos para entrenamiento y 20% para prueba, para en entrenamiento de los modelos seleccionados.

```
#Asignar todas las columnas que representan cada una de las palabras, se elimina la columnas del Sentimiento
X = df.drop('Sentiment', axis=1)
#Corresponde a la variable dependiente que es la que se quiere Medir el Sentimiento
y = df['Sentiment']

#Dividir la base en Entrenamiento 80% y Test 20%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state=1, stratify=y)
```

```
#Numero de Tweets Positivos=1, Negativos=-1 Neutrales=0
y_train.value_counts()
```

```
 1    489
-1    389
 0    326
Name: Sentiment, dtype: int64
```

```
y_test.value_counts()
```

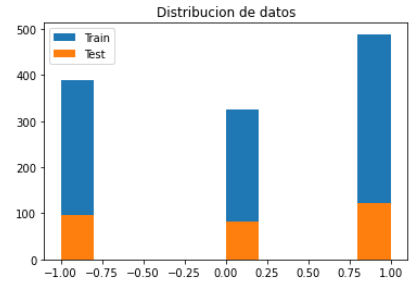
```
 1    123
-1     97
 0     81
Name: Sentiment, dtype: int64
```

```
print('datos de Entrenamiento: X_train')
print(X_train.shape)
print('datos de test: X_test')
print(X_test.shape)
```

```
datos de Entrenamiento: X_train
(1204, 354)
datos de test: X_test
(301, 354)
```

Seguidamente se ilustra en una gráfica el porcentaje de distribución de los datos de entrenamiento y de prueba.

```
pl.hist(y_train, label="Train")
pl.hist(y_test, label="Test")
pl.legend()
pl.title("Distribucion de datos")
pl.show()
```



Finalmente, se realiza la aplicación de los algoritmos de maching learning para evaluar cada uno de los modelos.

Aplicación del algoritmo LogisticRegression

```
#Regresion Logistica
clasificador_lr = LogisticRegression(multi_class='multinomial',solver='lbfgs')
clasificador_lr.fit(X_train, y_train);
print('SCORE entrenamiento: {}'.format(clasificador_lr.score(X_train, y_train)))
print('Error en entrenamiento: {}'.format(1-clasificador_lr.score(X_train, y_train)))
print('SCORE prueba: {}'.format(clasificador_lr.score(X_test, y_test)))
print('Error en prueba: {}'.format(1-clasificador_lr.score(X_test, y_test)))
```

```
SCORE entrenamiento: 0.7342192691029901
Error en entrenamiento: 0.26578073089700993
SCORE prueba: 0.5149501661129569
Error en prueba: 0.48504983388704315
```

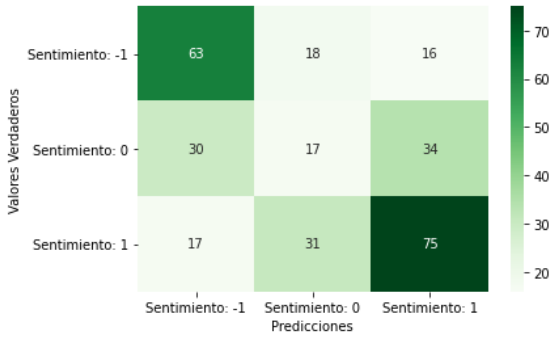
```
y_pred=clasificador_OVR.predict(X_test)
matrizRegresion = confusion_matrix(y_test,y_pred)
print(matrizRegresion)
print(classification_report(y_test,y_pred))
```

```
[[63 18 16]
 [30 17 34]
 [17 31 75]]
              precision    recall  f1-score   support

          -1       0.57      0.65      0.61        97
           0       0.26      0.21      0.23        81
           1       0.60      0.61      0.60       123

    accuracy                           0.51       301
   macro avg       0.48      0.49      0.48       301
weighted avg       0.50      0.51      0.51       301
```

Matriz de confusión.

```
col=['Sentimiento: %s'%(i) for i in list(np.unique(y_test))[0:len(np.unique(y_test))]]
df_m=pd.DataFrame(matrizRegresion, index=col, columns=col)
grafica=sns.heatmap(df_m,cmap='Greens',annot=True,fmt='d')
grafica.set(xlabel='Predicciones',ylabel='Valores Verdaderos')
```

```
[Text(0.5, 15.0, 'Predicciones'),
 Text(32.99999999999999, 0.5, 'Valores Verdaderos')]
```



Aplicación del algoritmo GaussianNB

```
#Gausiano Naive Bayes
clasificador_Gauss = GaussianNB()
clasificador_Gauss.fit(X_train, y_train);
print('SCORE entrenamiento: {}'.format(clasificador_Gauss.score(X_train, y_train)))
print('Error en entrenamiento: {}'.format(1-clasificador_Gauss.score(X_train, y_train)))
print('SCORE prueba: {}'.format(clasificador_Gauss.score(X_test, y_test)))
print('Error en prueba: {}'.format(1-clasificador_Gauss.score(X_test, y_test)))
```

```
SCORE entrenamiento: 0.6112956810631229
Error en entrenamiento: 0.38870431893687707
SCORE prueba: 0.4717607973421927
Error en prueba: 0.5282392026578073
```

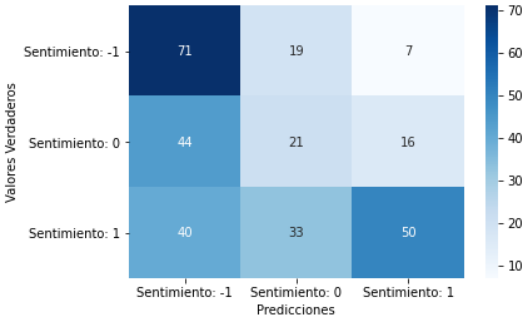
```
y_pred=clasificador_Gauss.predict(X_test)
matrizGaus = confusion_matrix(y_test,y_pred)
print(matrizGaus)
print(classification_report(y_test,y_pred))
```

```
[[71 19  7]
 [44 21 16]
 [40 33 50]]
              precision    recall  f1-score   support

          -1       0.46      0.73      0.56        97
           0       0.29      0.26      0.27        81
           1       0.68      0.41      0.51       123

    accuracy                           0.47       301
   macro avg       0.48      0.47      0.45       301
weighted avg       0.50      0.47      0.46       301
```

Grafica de matriz de Confusión.

```
col=['Sentimiento: %s'%(i) for i in list(np.unique(y_test))[0:len(np.unique(y_test))]]
df_m=pd.DataFrame(matrizGaus, index=col, columns=col)
grafica=sns.heatmap(df_m,cmap='Blues',annot=True,fmt='d')
grafica.set(xlabel='Predicciones',ylabel='Valores Verdaderos')
```

```
[Text(0.5, 15.0, 'Predicciones'),
 Text(32.99999999999999, 0.5, 'Valores Verdaderos')]
```



Aplicación del algoritmo SVM

```
#Aplicar el algoritmo de SVC sobre la muestra de entrenamiento
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)
y_pred = svclassifier.predict(X_test)
print('SCORE entrenamiento: {}'.format(svclassifier.score(X_train, y_train)))
print('Error en entrenamiento: {}'.format(1-svclassifier.score(X_train, y_train)))
print('SCORE prueba: {}'.format(svclassifier.score(X_test, y_test)))
print('Error en prueba: {}'.format(1-svclassifier.score(X_test, y_test)))
```

```
SCORE entrenamiento: 0.8222591362126246
Error en entrenamiento: 0.1777408637873754
SCORE prueba: 0.5382059800664452
Error en prueba: 0.46179401993355484
```

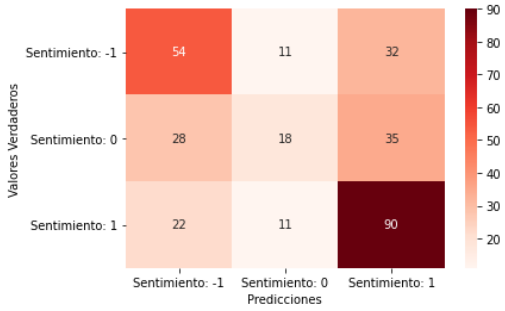
```
matrizSVM=confusion_matrix(y_test,y_pred)
print(matrizSVM)
print(classification_report(y_test,y_pred))
```

```
[[54 11 32]
 [28 18 35]
 [22 11 90]]
              precision    recall  f1-score   support

          -1       0.52      0.56      0.54        97
           0       0.45      0.22      0.30        81
           1       0.57      0.73      0.64       123

    accuracy                           0.54       301
   macro avg       0.51      0.50      0.49       301
weighted avg       0.52      0.54      0.52       301
```

Grafica de matriz de confusión.

```
col=['Sentimiento: %s'%(i) for i in list(np.unique(y_test))[0:len(np.unique(y_test))]]
df_m=pd.DataFrame(matrizSVM, index=col, columns=col)
grafica=sns.heatmap(df_m,cmap='Reds',annot=True,fmt='d')
grafica.set(xlabel='Predicciones',ylabel='Valores Verdaderos')
```

```
[Text(0.5, 15.0, 'Predicciones'),
 Text(32.99999999999999, 0.5, 'Valores Verdaderos')]
```



Como ejercicio adicional se realizaron los cálculos manualmente para validar los valores obtenidos al correr el modelo SVC:

Creación de dataframe a partir de la matriz de confusión generada por el modelo SVC, a este dataframe se le agrego una columna con el total de las filas, así como una fila con el total de cada columna para luego aplicar las fórmulas y obtener los valores de predicción del modelo

```
df_m.loc['TotalColuma']=df_m.sum(axis=0)
df_m['TotalFila'] = df_m.sum(axis=1)
df_m
```

	Sentimiento: -1	Sentimiento: 0	Sentimiento: 1	TotalFila
Sentimiento: -1	54	11	32	97
Sentimiento: 0	28	18	35	81
Sentimiento: 1	22	11	90	123
TotalColuma	104	40	157	301

Se realizaron los cálculos manuales, se evidencia que los resultados son iguales a los generados por el algoritmo SVC:

```
#https://www.youtube.com/watch?v=ZdfloOxXeZ4
print('CALCULOS POR SENTIMIENTO')
print()
print('******************Precision*********************')
print('Sentimiento -1: '+"{:.2%}".format(df_m.iloc[0,0]/df_m.iloc[3,0]))
print('Sentimiento 0: '"{:.2%}".format(df_m.iloc[1,1]/df_m.iloc[3,1]))
print('Sentimiento 1: '+"{:.2%}".format(df_m.iloc[2,2]/df_m.iloc[3,2]))
print('***************Recall - Cobertura*********************')
print('Sentimiento -1: '+"{:.2%}".format(df_m.iloc[0,0]/df_m.iloc[0,3]))
print('Sentimiento 0: '+"{:.2%}".format(df_m.iloc[1,1]/df_m.iloc[1,3]))
print('Sentimiento 1: '+"{:.2%}".format(df_m.iloc[2,2]/df_m.iloc[2,3]))
print('***************Accuracy - Exactitud*********************')
print('Accuracy: '+"{:.2%}".format((df_m.iloc[0,0]+df_m.iloc[1,1]+df_m.iloc[2,2])/df_m.iloc[3,3]))
print('***************Media*********************')
```

```
CALCULOS POR SENTIMIENTO

******************Precision*********************
Sentimiento -1: 51.92%
Sentimiento 0: 45.00%
Sentimiento 1: 57.32%
***************Recall - Cobertura*********************
Sentimiento -1: 55.67%
Sentimiento 0: 22.22%
Sentimiento 1: 73.17%
***************Accuracy - Exactitud*********************
Accuracy: 53.82%
***************Media*********************
```

```
#Precicion Media Ponderada
A=(df_m.iloc[3,0]/df_m.iloc[3,3])*(df_m.iloc[0,0]/df_m.iloc[3,0])
B=(df_m.iloc[3,1]/df_m.iloc[3,3])*(df_m.iloc[1,1]/df_m.iloc[3,1])
C=(df_m.iloc[3,2]/df_m.iloc[3,3])*(df_m.iloc[2,2]/df_m.iloc[3,2])
print(A+B+C)
```

```
0.5382059800664452
```

Aplicación del algoritmo RandomForestClassifier

```
#RandomForest
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
clf=RandomForestClassifier(random_state=123)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
print('SCORE entrenamiento: {}'.format(clf.score(X_train, y_train)))
print('Error en entrenamiento: {}'.format(1-clf.score(X_train, y_train)))
print('SCORE prueba: {}'.format(clf.score(X_test, y_test)))
print('Error en prueba: {}'.format(1-clf.score(X_test, y_test)))

```

```
SCORE entrenamiento: 0.9842192691029901
Error en entrenamiento: 0.01578073089700993
SCORE prueba: 0.5249169435215947
Error en prueba: 0.4750830564784053
```

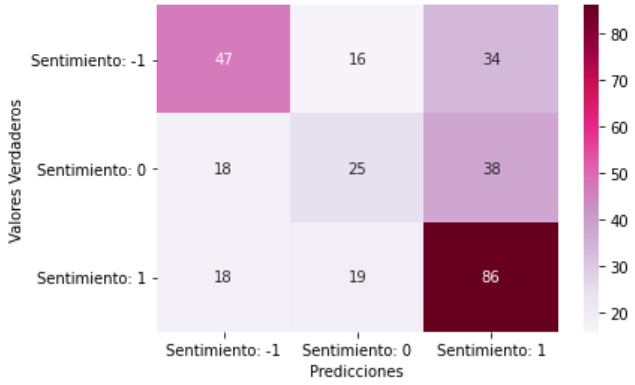
```
matrizRandomForest=confusion_matrix(y_test,y_pred)
print(matrizRandomForest)
print(classification_report(y_test,y_pred))
```

```
[[47 16 34]
 [18 25 38]
 [18 19 86]]
              precision    recall  f1-score   support

          -1       0.57      0.48      0.52        97
           0       0.42      0.31      0.35        81
           1       0.54      0.70      0.61       123

    accuracy                           0.52       301
   macro avg       0.51      0.50      0.50       301
weighted avg       0.52      0.52      0.51       301
```

Grafica de matriz de confusión.

```
col=['Sentimiento: %s'%(i) for i in list(np.unique(y_test))[0:len(np.unique(y_test))]]
df_m=pd.DataFrame(matrizRandomForest, index=col, columns=col)
grafica=sns.heatmap(df_m,cmap='PuRd',annot=True,fmt='d')
grafica.set(xlabel='Predicciones',ylabel='Valores Verdaderos')
```

```
[Text(0.5, 15.0, 'Predicciones'),
 Text(32.99999999999999, 0.5, 'Valores Verdaderos')]
```



Aplicación del algoritmo DecisionTreeClassifier

```
#DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=123)
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print('SCORE entrenamiento: {}'.format(clf.score(X_train, y_train)))
print('Error en entrenamiento: {}'.format(1-clf.score(X_train, y_train)))
print('SCORE prueba: {}'.format(clf.score(X_test, y_test)))
print('Error en prueba: {}'.format(1-clf.score(X_test, y_test)))
```

```
SCORE entrenamiento: 0.9842192691029901
Error en entrenamiento: 0.01578073089700993
SCORE prueba: 0.45182724252491696
Error en prueba: 0.548172757475083
```

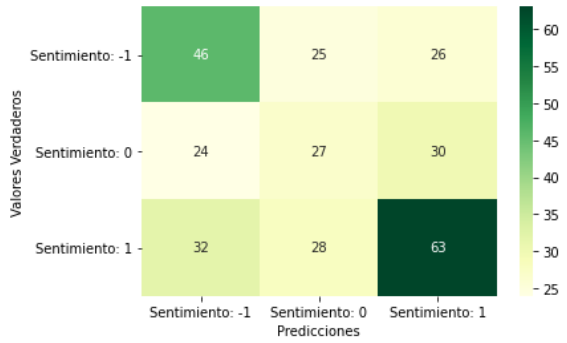
```
matrizDecisionTree=confusion_matrix(y_test,y_pred)
print(matrizDecisionTree)
print(classification_report(y_test,y_pred))
```

```
[[46 25 26]
 [24 27 30]
 [32 28 63]]
              precision    recall  f1-score   support

          -1       0.45      0.47      0.46        97
           0       0.34      0.33      0.34        81
           1       0.53      0.51      0.52       123

    accuracy                           0.45       301
   macro avg       0.44      0.44      0.44       301
weighted avg       0.45      0.45      0.45       301
```

Grafica matriz de confusión.

```
col=['Sentimiento: %s'%(i) for i in list(np.unique(y_test))[0:len(np.unique(y_test))]]
df_m=pd.DataFrame(matrizDecisionTree, index=col, columns=col)
grafica=sns.heatmap(df_m,cmap='YlGn',annot=True,fmt='d')
grafica.set(xlabel='Predicciones',ylabel='Valores Verdaderos')
```

```
[Text(0.5, 15.0, 'Predicciones'),
 Text(32.99999999999999, 0.5, 'Valores Verdaderos')]
```



Finalmente, después de aplicar cada uno de los algoritmos indicados anteriormente, se creó la nueve de palabras, la cual consiste en dibujar en el centro de la imagen la palabra más usada y alrededor de esta las demás palabras, es decir el tamaño de la palabra corresponde a la frecuencia de ocurrencia en el corpus que se utilizó para el análisis de sentimientos.

Código fuente utilizado para generar la nube de palabras

```
#Nube de Palabras
from PIL import Image
from wordcloud import WordCloud
my_mask = np.array(Image.open(r'C:\opt\ProyectoGrado\notebooks\Entrenamiento\virus.png'))
wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                contour_width=10,
                contour_color='black',
                mask=my_mask,
                min_font_size = 10).generate(comment_words)

# crear la imagen de WordCloud
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

Imagen de la nube de palabras.