“ANÁLISIS DE SENTIMIENTOS UTILIZANDO APRENDIZAJE AUTOMÁTICO DE MENCIONES EN TWITTER PARA LA SECRETARÍA DE MOVILIDAD DE BOGOTÁ”

Luis Eduardo Quiñonez Romero

Luisa Fernanda Carbonell García

Andrés Gabriel Peralta Alean

Nota de Aceptación

Certificamos que el presente Trabajo de Grado Satisface, en alcances y calidad, todos los requisitos que demanda un Trabajo de Grado de Maestría.

Director

Jurado

Jurado

Aprobado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana Cali, para optar el título de Magister en Ciencia de Datos.

HERNÁN CAMILO ROCHA NIÑO Ph. D.
Decano Facultad de Ingeniería y Ciencias

JUAN CARLOS MARTÍNEZ ARIAS
Director Posgrados de Ingeniería y Ciencias

24/08/2023

Acta de Correcciones al Documento de Trabajo de Grado

Santiago de Cali, 24 de Agosto 2023

Autores: Luis Eduardo Quiñonez Romero, Luisa Fernanda Carbonell García, Andrés Gabriel Peralta Alean

Título del Trabajo de Grado: ""ANÁLISIS DE SENTIMIENTOS UTILIZANDO APRENDIZAJE AUTOMÁTICO DE MENCIONES EN TWITTER PARA LA SECRETARÍA DE MOVILIDAD DE BOGOTÁ"

Director: María Constanza Pabón

Como indica el artículo 2.13 de las Directrices para Trabajo de Grado de Maestría, he verificado que el estudiante indicado arriba ha implementado todas las correcciones que los Jurados del Proyecto de Trabajo de Grado definieron que se efectuaran, como consta en el Acta de Evaluación correspondiente.

Firma del Director del Trabajo de Grado

FICHA RESUMEN
PROYECTO APLICADO – MAESTRÍA EN CIENCIA DE DATOS

TÍTULO: "ANÁLISIS DE SENTIMIENTOS UTILIZANDO APRENDIZAJE AUTOMÁTICO DE MENCIONES EN TWITTER PARA LA SECRETARÍA DE MOVILIDAD DE BOGOTÁ"

1. ÁREA DE TRABAJO: Análisis de sentimientos, machine learning
2. TIPO DE PROYECTO (Aplicado, innovación, investigación): Aplicado
3. ESTUDIANTE(S): Andrés Gabriel Peralta Alean, Luis Eduardo Quiñones, Luisa Fernanda Carbonell García
4. CORREO ELECTRÓNICO: lequinonesr@javerianacali.edu.co, luisafernanda1@javerianacali.edu.co, angaperal@javerianacali.edu.co
5. DIRECCIÓN Y TELÉFONO: Calle 123 A 11 B – 55 apto 303 | 3132523388. Bogotá
6. DIRECTOR: María Constanza Pabón Burbano
7. VINCULACIÓN DEL DIRECTOR: Docente Facultad de Ingeniería y Ciencias
8. CORREO ELECTRÓNICO DEL DIRECTOR: mcpabon@javerianacali.edu.co
9. CO-DIRECTOR (Si aplica): No aplica
10. GRUPO O EMPRESA QUE LO AVALA (Si aplica):
11. OTROS GRUPOS O EMPRESAS: N/A
12. PALABRAS CLAVE: twitter, web scraping, análisis de sentimientos, aprendizaje supervisado, movilidad.
13. FECHA DE INICIO: Octubre 2022
14. DURACIÓN ESTIMADA (En meses): 12
15. RESUMEN:

La evolución de la sociedad moderna ha llevado a la instauración de urbes de gran densidad en donde difícilmente se logra mantener un balance entre las zonas de trabajo, estudio y las zonas residenciales, lo que genera desplazamientos considerables para la población media dentro de su ciclo de cotidianidad. La ciudad de Bogotá no es la excepción a estas condiciones, por lo que la Secretaría de Movilidad asume un reto en la implementación de medidas que agilicen el transporte de los ciudadanos.

Un desafío subyacente en la implementación de estas medidas es la medición de su efectividad, donde la percepción de los usuarios juega un papel fundamental en la evolución de los planes de movilidad y la identificación de necesidades y ajustes de las iniciativas actuales.

En ese sentido las redes sociales operan como compiladores masivos de percepciones sobre

la gestión realizada, generando que la Secretaría enfoque esfuerzos de comunicación sobre la red social twitter, queriendo contar con un mecanismo automatizado que permita identificar las tendencias en cuanto a las percepciones de los usuarios.

Para tal fin se pretende desarrollar un análisis de sentimientos con un modelo de clasificación de aprendizaje supervisado, el cual permita, mediante una aplicación en línea usando uno o varios modelos entrenados, identificar y clasificar conjuntos de *tweets*.

FICHA RESUMEN
PROYECTO APLICADO – MAESTRÍA EN CIENCIA DE DATOS

TÍTULO: "ANÁLISIS DE SENTIMIENTOS UTILIZANDO APRENDIZAJE AUTOMÁTICO DE MENCIONES EN TWITTER PARA LA SECRETARÍA DE MOVILIDAD DE BOGOTÁ"

1. ÁREA DE TRABAJO: Análisis de sentimientos, machine learning
2. TIPO DE PROYECTO (Aplicado, innovación, investigación): Aplicado
3. ESTUDIANTE(S): Andrés Gabriel Peralta Alean, Luis Eduardo Quiñones, Luisa Fernanda Carbonell García
4. CORREO ELECTRÓNICO: lequinonesr@javerianacali.edu.co, luisafernanda1@javerianacali.edu.co, angaperal@javerianacali.edu.co
5. DIRECCIÓN Y TELÉFONO: Calle 123 A 11 B – 55 apto 303 | 3132523388. Bogotá
6. DIRECTOR: María Constanza Pabón Burbano
7. VINCULACIÓN DEL DIRECTOR: Docente Facultad de Ingeniería y Ciencias
8. CORREO ELECTRÓNICO DEL DIRECTOR: mcpabon@javerianacali.edu.co
9. CO-DIRECTOR (Si aplica): No aplica
10. GRUPO O EMPRESA QUE LO AVALA (Si aplica):
11. OTROS GRUPOS O EMPRESAS: N/A
12. PALABRAS CLAVE: twitter, web scraping, análisis de sentimientos, aprendizaje supervisado, movilidad.
13. FECHA DE INICIO: Octubre 2022
14. DURACIÓN ESTIMADA (En meses): 12
15. RESUMEN:

La evolución de la sociedad moderna ha llevado a la instauración de urbes de gran densidad en donde difícilmente se logra mantener un balance entre las zonas de trabajo, estudio y las zonas residenciales, lo que genera desplazamientos considerables para la población media dentro de su ciclo de cotidianidad. La ciudad de Bogotá no es la excepción a estas condiciones, por lo que la Secretaría de Movilidad asume un reto en la implementación de medidas que agilicen el transporte de los ciudadanos.

Un desafío subyacente en la implementación de estas medidas es la medición de su efectividad, donde la percepción de los usuarios juega un papel fundamental en la evolución de los planes de movilidad y la identificación de necesidades y ajustes de las iniciativas actuales.

En ese sentido las redes sociales operan como compiladores masivos de percepciones sobre la

gestión realizada, generando que la Secretaría enfoque esfuerzos de comunicación sobre la red social twitter, queriendo contar con un mecanismo automatizado que permita identificar las tendencias en cuanto a las percepciones de los usuarios.

Para tal fin se pretende desarrollar un análisis de sentimientos con un modelo de clasificación de aprendizaje supervisado, el cual permita, mediante una aplicación en línea usando uno o varios modelos entrenados, identificar y clasificar conjuntos de *tweets*.

ANÁLISIS DE SENTIMIENTOS UTILIZANDO APRENDIZAJE AUTOMÁTICO DE MENCIONES EN TWITER PARA LA SECRETARIA DE MOVILIDAD DE BOGOTÁ

Andrés Gabriel Peralta Alean
Código: 8972774

Luis Eduardo Quiñones Romero
Código 8972069

Luisa Fernanda Carbonell García
Código: 8973333

Proyecto Aplicado para optar al título de
Magister en Ciencia de Datos

Directora
María Constanza Pabón

FACULTAD DE INGENIERÍA Y CIENCIAS
MAESTRÍA EN CIENCIA DE DATOS
SANTIAGO DE CALI, MAYO 22 DE 2023

TABLA DE CONTENIDOS

TABLA DE ILUSTRACIONES

INTRODUCCIÓN

La creciente necesidad de oportunidades de estudio y trabajo, entre otros, ha generado la concentración de habitantes en las principales poblaciones del mundo, lo que genera desafíos y retos en la planeación, ejecución y convivencia, así como en la definición de la infraestructura para soportar la creciente demanda de recursos, el balance de construcción entre viviendas, oficinas, comercio y otros, la ubicación de centros educativos de diferentes niveles, etc.

Sin embargo, hay un tema que tiene un impacto constante en la calidad de vida de los ciudadanos: el tiempo dedicado a la movilización diaria. A pesar de los continuos esfuerzos realizados por la Secretaría de Movilidad de Bogotá para desarrollar iniciativas que mejoren esta situación, se trata de una tarea difícil debido a la alta densidad poblacional actual y la prácticamente imposible ampliación de la infraestructura vial en la ciudad.

Aunque la medición del impacto de la movilidad en las grandes urbes, y en especial en la ciudad de Bogotá se ha constituido en uno de los retos más grandes, las iniciativas para mejorarla también representan una apuesta de alto valor, pues su resultado puede llevar a la profundización de iniciativas exitosas y al abandono oportuno de otras menos impactantes. En tal sentido la percepción de los usuarios en la ciudad de Bogotá, juega un rol definitivo pues al final el objetivo es mejorar la calidad de vida de los ciudadanos.

Las redes sociales son una excelente fuente de recopilación de percepciones, lo cual ha hecho que la Secretaría de Movilidad estimule el uso de Twitter como mecanismo de manifestación de percepciones y quejas.

Este proyecto brinda a la Secretaría de Movilidad de Bogotá, un mecanismo automático de medición de percepción de la ciudadanía respecto a la movilidad de la ciudad y los efectos que sobre esta tienen las diferentes medidas que se tomen. Para tal fin, se espera comparar el desempeño de diferentes algoritmos de aprendizaje supervisado para la clasificación de menciones en twitter entre percepciones positivas, negativas y neutras de la movilidad en la ciudad de Bogotá.

1. Definición del problema

1.1. Planteamiento del problema

Los desafíos de la movilidad de Bogotá transitan por el camino de la transformación. A mayo de 2022 Bogotá cuenta con alrededor de 2.400.000 de vehículos[1], de estos el 50% son automóviles, 20% motos, 14% camionetas, dejando a un 7% para el servicio público o taxis, según las cifras de la página Bogotá cómo vamos.

La dificultad de lograr una movilidad eficiente en la ciudad de Bogotá, debido al incremento masivo de vehículos, constituye un problema en constante aumento. Esto se debe a que la ciudad carece tanto de la infraestructura necesaria como de las capacidades requeridas para gestionar el flujo de semejante cantidad de automóviles. Si a esta ecuación le añadimos el costo asociado a la necesidad de cambiar los hábitos de los ciudadanos en paralelo a la implementación de nuevas normativas e iniciativas, el escenario adquiere un nivel de interés aún mayor.

Gradualmente, se ha observado cómo las ciudades europeas han logrado mejorar la circulación vehicular al establecer vías libres de automóviles y carriles exclusivos para bicicletas, y Bogotá no es una excepción. Con un promedio de casi 900 mil desplazamientos en bicicleta anualmente, la cifra de bicicletas aumentó en 1,5 millones para el año 2019. Además, surge la necesidad de prevenir aglomeraciones debido a la pandemia de COVID-19, situación que ha propiciado un incremento aún mayor en este número.

Evidentemente, la presencia de riesgos en las vías es innegable. A pesar de los esfuerzos por reducir la velocidad promedio, persisten lamentables incidentes mortales que involucran a diversos actores viales, tales como peatones, conductores, pasajeros, ciclistas y motociclistas. Estos trágicos sucesos resaltan la urgente necesidad de implementar medidas más efectivas para garantizar la seguridad en las vías y proteger la vida de todos los usuarios. La concienciación, la educación vial y el diseño adecuado de la infraestructura cobran un rol fundamental en la búsqueda de una circulación más segura y responsable en nuestras calles y carreteras.

Con base en estos hechos, la Alcaldía Mayor de Bogotá ha venido liderando diversas iniciativas y proyectos con el fin de mejorar la interacción y el flujo de movilidad de los habitantes de la ciudad como una gestión de la demanda de transporte. Por ejemplo, las tres principales regulaciones e

[1] https://bogotacomovamos.org/preocupa-crecimiento-de-parque-automotor/

iniciativas que han impactado a los ciudadanos de Bogotá son “pico y placa regional”, “ciclorutas” y “alquiler de bicicletas”, entre otros.

Desde luego para la secretaria es indispensable contar con medios que habiliten la evaluación del impacto que se tiene sobre las iniciativas implementadas, para este fin existe un conjunto de indicadores definidos, como velocidad promedio y tiempos de recorrido, entre otros. Sin embargo, un elemento que predomina para este caso particular es la percepción de los usuarios (ciudadanos) respecto a las condiciones generales de movilidad en la ciudad. Sin embargo, esto representa un reto pues instaurar mecanismos de recopilación de percepciones constantes es difícil de mantener, teniendo en cuenta esto y la alta afluencia de usuarios con que cuenta una red social como twitter, se encuentra una alternativa viable y de alto valor en la obtención automatizada de textos en twitter como elemento representativo de percepciones respecto a la movilidad en Bogotá, al filtrar los comentarios asociados a cuentas y Hastags(#) específicos. Adicionalmente para poder contar con un modelo de medición cuantitativo y estandarizado es de alta relevancia contar con la capacidad de clasificar las diferentes percepciones entre positivas, negativas y neutras en altos volúmenes de comentarios de los ciudadanos en la red social.

1.2. Formulación del problema

Ante la necesidad del distrito de mitigar los graves problemas de transporte, se plantea la pregunta de formulación ¿Cómo usar la información contenida en Twitter para identificar la efectividad de las medidas tomadas por la Secretaría Distrital de Movilidad según la percepción del público? Y en ese sentido se plantean las preguntas de sistematización ¿Qué mecanismo es el más eficiente para obtener descargas masivas de los textos de Twitter asociados a los temas de movilidad? ¿Cómo realizar la calificación de los textos de los tweets evitando sesgos? ¿Cuál es el mecanismo de aprendizaje automático que mejor desempeño tiene para la clasificación de sentimientos de los textos? ¿Cómo combinar diferentes técnicas de clasificación para mejorar el desempeño de la clasificación?

2. Objetivos del proyecto

2.1. Objetivo general

Construir un modelo de análisis de sentimientos que permita evaluar la percepción del público general respecto a la movilidad en Bogotá a través de un análisis de menciones de los usuarios de Twitter.

2.2. Objetivos específicos

- Preparar la información de los tweets que hagan referencia a la movilidad en Bogotá en estructuras de datos computacionales procesables por algoritmos de aprendizaje supervisado.
- Calificar manualmente un conjunto significativo de registros entre la muestra obtenida como positivos, negativos o neutros.
- Evaluar y comparar resultados entre diferentes técnicas de aprendizaje supervisado, incluyendo optimización de hiperparámetros, teniendo como variable de respuesta la clasificación del texto entre negativo, positivo o neutro.
- Construir interfaz de uso para la ejecución las técnicas ante nuevos conjuntos de datos.

3. Marco teórico y antecedentes

El problema de procesamiento de texto a nivel computacional es una herramienta que ha sido ampliamente tratada y desarrollada a lo largo de diferentes industrias, generando así un amplio rango de posibilidades de implementación. Particularmente son muy comunes los casos de análisis de sentimiento, mecanismos de recomendación, y en los casos más avanzados, el uso de chat bots o inclusive, las asistentes personales. En este caso se hizo uso de estas técnicas con el fin de identificar percepciones respecto a la movilidad en Bogotá que sirvan como base para identificar el efecto de diferentes medidas en la ciudad.

3.1. Marco teórico

3.1.1. Aprendizaje supervisado [1]

El aprendizaje supervisado es una técnica a través de la cual se conocen a priori los grupos o etiquetas de los elementos o entidades a revisar. De forma general existen mecanismos para clasificación o predicción. En los primeros la variable de respuesta es categórica, mientras que en los segundos se trabaja en el tipo de la variable de salida numérica. Dentro de los diferentes algoritmos que se encuentran para la clasificación están:

- **Naive Bayes:**

El algoritmo Naive Bayes es una técnica de clasificación supervisada basada en el teorema de Bayes con una suposición de independencia entre los predictores. En términos simples, un clasificador Naive Bayes asume que la presencia de una característica particular en una clase no está relacionada con la presencia de ninguna otra característica [1]. Esta suposición se llama "ingenua" debido a que, en la realidad, las características pueden no ser independientes entre sí.

El algoritmo de Naive Bayes es adecuado para realizar análisis de sentimientos debido a su eficiencia para manejar grandes volúmenes de datos, su habilidad para manejar múltiples clases

y su eficacia en la clasificación de texto, lo que es especialmente útil para tareas como el análisis de tweets [2].

El algoritmo Naive Bayes utiliza el teorema de Bayes para calcular la probabilidad de que un evento ocurra dado que otro evento ya ha ocurrido. En el contexto de la clasificación de textos, este evento puede ser la aparición de una palabra o frase dada la categoría (positiva, negativa, neutra) del texto.

El algoritmo es ampliamente utilizado en la detección de spam, el filtrado de noticias, el reconocimiento de voz y, como es relevante en nuestro caso, el análisis de sentimientos de los tweets. Sin embargo, un desafío importante con el algoritmo Naive Bayes es que si se encuentra con una palabra desconocida (una que no está en el conjunto de datos de entrenamiento), puede calcular una probabilidad de 0 y ser incapaz de hacer una predicción correcta [3].

Ventajas:

1. Eficiencia computacional: debido a su suposición de independencia, Naive Bayes es computacionalmente eficiente, lo que lo hace adecuado para manejar grandes volúmenes de datos [76].
2. Sencillez: Naive Bayes es simple y fácil de implementar, lo que facilita su uso en la clasificación de sentimientos [7].
3. Tolerancia al ruido: Naive Bayes puede manejar conjuntos de datos ruidosos o con características irrelevantes, lo que es útil para el análisis de sentimientos de textos de Twitter, donde el lenguaje puede ser informal y contener errores gramaticales [80].

Desventajas:

1. Suposición de independencia: la suposición de independencia puede ser una simplificación excesiva, ya que las palabras en un texto a menudo están relacionadas [81].
2. Falta de captura de contexto: Naive Bayes no captura el contexto ni las relaciones semánticas entre las palabras, lo que puede afectar su precisión [82].
3. Sensibilidad a desequilibrios en los datos: el algoritmo puede verse afectado por desequilibrios en la distribución de las clases en el conjunto de datos de entrenamiento [29].

- **Random forest:**

Es un algoritmo de aprendizaje automático supervisado que se utiliza para clasificación, regresión y otras tareas. Este algoritmo se basa en la técnica de los árboles de decisión y crea un conjunto de árboles de decisión, cada uno construido a partir de una muestra de bootstrap del conjunto de

datos de entrenamiento [4].

El algoritmo Random Forest genera un gran número de árboles de decisión independientes durante el entrenamiento y genera una predicción basándose en el modo de las predicciones realizadas por cada árbol en el conjunto. Este proceso de "promedio" puede ayudar a mejorar el rendimiento predictivo y a controlar el sobreajuste [5].

En el caso del análisis de sentimientos de tweets, Random Forest puede ser útil debido a su habilidad para manejar grandes volúmenes de datos, su capacidad para manejar características de alta dimensión y su resistencia al ruido y a los valores atípicos [6].

Si bien el Random Forest es una técnica sumamente potente, también presenta ciertas limitaciones. En particular, puede resultar relativamente lento durante la etapa de entrenamiento en casos particulares de configuración de hiperparametros, particularmente cuando se trabaja con un elevado número de árboles [7]. Además, a pesar de su eficacia con datos no lineales, en los cuales las relaciones entre variables no son directas, su desempeño puede ser menos efectivo en situaciones donde las relaciones son lineales y podrían ser representadas con una simple línea recta debido a una alta correlación directa entre las variables.

Ventajas:

1. Precisión. Los bosques aleatorios son conocidos por su alta precisión en la clasificación de datos. La robustez y precisión del modelo provienen de la combinación de múltiples árboles de decisión. Cada árbol se entrena en un subconjunto aleatorio de los datos y utiliza un subconjunto aleatorio de características. Al agregar (por ejemplo, promediando) las predicciones de estos árboles, el bosque aleatorio puede capturar diversas representaciones de los datos. Esta diversidad en la representación ayuda a mitigar el riesgo de sobreajuste, lo que a su vez mejora la generalización del modelo
2. Robustez a los datos atípicos: Esto debido a que cada árbol se entrena de forma independiente y se promedia el resultado, los errores individuales de los árboles se compensan, lo que ayuda a reducir los errores en las predicciones o clasificaciones.
3. Manejo de gran cantidad de datos: Random forest o los árboles aleatorios son eficientes en términos de tiempo en entrenamiento y predicción, esto debido a que los árboles se pueden entrenar de forma paralela y la clasificación se puede dar de manera eficiente por la mayoría de los votos.
4. Selección alegoría de características: Seleccionan subconjuntos de características para cada árbol, disminuyendo la probabilidad de correlaciones entre los árboles.
5. Estimación de importancia de características: Random forest proporciona una medida de la importancia de cada característica, permitiendo seleccionar las más relevantes para la clasificación.

Desventajas:

1. Falta de interpretación: Pueden ser difíciles de interpretar debido a la complejidad del modelo y a la cantidad de árboles que lo componen. Esto pude dificultar la identificación de las características más importantes y la compresión de cómo se toman las decisiones en la clasificación.
2. Requiere más memoria: Por la cantidad de árboles que procesa tiene limitaciones con equipos sencillos, en especial si se procesa gran cantidad de información.
3. Sensible a la selección de hiper parámetros: La selección incorrecta de hiper parámetros puede generar errores en la clasificación.
4. No es adecuado para datos secuenciales: por su misma lógica de aleatoriedad, no tiene en cuenta secuencias temporales al clasificar o identificar características.

- **Regresión Logística:**

La Regresión Logística es un algoritmo de aprendizaje automático supervisado que se utiliza para problemas de clasificación. Este algoritmo es una extensión del modelo de regresión lineal, que se utiliza para modelar la probabilidad de un evento binario. Dicho evento puede ser pasar/fallar, ganar/perder, vivo/muerto, etc., que son representados en términos de 1s y 0s [8].

La regresión logística funciona al usar una función logística para modelar la probabilidad del evento de interés. Esta probabilidad se convierte luego en un logit (o log-odds), que puede ser modelado como una combinación lineal de predictores [9].

En el caso de análisis de sentimientos, la regresión logística puede ser útil por su simplicidad y eficiencia, especialmente si los datos pueden ser linealmente separables en términos de sentimiento (por ejemplo, positivo o negativo). Además, este algoritmo también permite una fácil interpretación de los resultados, lo cual es útil para entender qué características son más predictivas del sentimiento [10].

Sin embargo, la regresión logística puede ser limitada en términos de su capacidad para manejar relaciones complejas o no lineales entre las características y el sentimiento. Además, puede ser sensible a valores atípicos y requerir una cuidadosa selección y transformación de características [11].

La regresión logística se puede aplicar tanto a problemas de clasificación binaria, donde se busca predecir una variable dependiente que toma dos valores (por ejemplo, sí o no), como a problemas de clasificación multiclase, donde se busca predecir una variable dependiente con más de dos categorías. [7]

Los coeficientes estimados en la regresión logística proporcionan información sobre la relación entre las variables independientes y la variable dependiente. Los coeficientes positivos indican una relación positiva con la probabilidad de pertenecer a una clase específica, mientras que los coeficientes negativos indican una relación negativa. [9]

Ventajas:

1. Interpretabilidad: La regresión logística proporciona coeficientes que indican la dirección y magnitud de la influencia de cada variable predictora en la probabilidad de ocurrencia del evento. Esto permite una interpretación más clara y directa de los efectos de las variables en la salida. [54]
2. Eficiencia computacional: La regresión logística es un modelo relativamente simple y rápido de entrenar en comparación con algunos otros modelos de aprendizaje supervisado más complejos, como las redes neuronales profundas. Esto la hace adecuada para conjuntos de datos grandes o problemas en los que el tiempo de entrenamiento es un factor importante. [54]
3. Diagnóstico de ajuste del modelo: La regresión logística ofrece diversas métricas y técnicas de diagnóstico diferenciales que son útiles para evaluar la calidad del ajuste del modelo. Entre estas métricas se encuentran el valor p, la desviación residual. Estas herramientas permiten no solo medir la capacidad predictiva del modelo, sino también identificar posibles áreas de mejora en caso de ser necesario [54].

Desventajas:

1. Sensibilidad a valores atípicos y datos mal clasificados: La regresión logística puede ser sensible a la presencia de valores atípicos o datos mal clasificados en el conjunto de entrenamiento. Estos puntos anómalos pueden influir en la estimación de los coeficientes y afectar la calidad del ajuste del modelo. [8]
2. Requiere variables relevantes: La regresión logística puede verse afectada por la inclusión de variables irrelevantes o redundantes en el modelo. Estas variables pueden introducir ruido y afectar negativamente el rendimiento del modelo, por lo que se requiere un proceso de selección de características cuidadoso. [8]
3. Linealidad en el log-odds: La regresión logística asume que existe una relación lineal entre las variables predictoras y el logaritmo de odds de la variable respuesta. Es decir, aunque puede modelar relaciones no lineales entre las variables predictoras y la probabilidad de un evento, estas relaciones son transformadas a través del log-odds. Si la verdadera relación no sigue este patrón lineal en el log-odds, la regresión logística puede no capturarla adecuadamente, afectando así la precisión del modelo.
4. Suposición de independencia de las observaciones: La regresión logística asume que las observaciones son independientes entre sí. Sin embargo, en algunos casos, esta suposición puede no ser válida, especialmente si los datos están correlacionados o si las observaciones se recopilan en clústeres o grupos.
5. No maneja bien datos faltantes: La regresión logística puede tener dificultades para manejar datos faltantes en las variables predictoras. Si hay valores faltantes, es necesario realizar una imputación

o tratamiento adecuado antes de aplicar la regresión logística, de lo contrario, la precisión del modelo puede verse afectada.

3.1.2. Medidas de desempeño

Las medidas de desempeño son una parte crucial en la evaluación de los modelos de aprendizaje automático. A continuación, se describen cuatro medidas de desempeño comúnmente utilizadas: Accuracy, Recall, Precision y F1 Score [28].

- **Accuracy (Exactitud):** Es la proporción de predicciones correctas con respecto al total de predicciones realizadas. Es una medida intuitiva de la eficacia de un clasificador. Sin embargo, su principal desventaja es que puede ser engañosa en conjuntos de datos desequilibrados, donde la mayoría de las instancias pertenecen a una clase [29].
- **Precision (Precisión):** Es la proporción de predicciones positivas correctas respecto al total de predicciones positivas realizadas. Se utiliza para medir la calidad de las predicciones positivas del clasificador [30].
- **Recall (Sensibilidad o Exhaustividad):** Es la proporción de predicciones positivas correctas respecto al total de instancias positivas reales. Se utiliza para medir la capacidad del clasificador para encontrar todas las instancias positivas [32].
- **F1 Score:** Es la media armónica de Precision y Recall. Dado que considera tanto Precision como Recall, F1 Score es una medida de desempeño útil cuando se tiene un conjunto de datos desequilibrado y es importante mantener un equilibrio entre estos dos parámetros [32].

Cada una de estas medidas tiene sus ventajas y desventajas, y su elección depende del contexto específico y los requisitos del problema. Por ejemplo, si se considera más importante evitar los falsos positivos, se puede dar más peso a la Precisión. Si es más importante evitar los falsos negativos, se puede dar más peso al Recall.

3.1.3. Balanceo de datos

En aprendizaje automático, es común encontrarse con conjuntos de datos desequilibrados donde una clase está representada significativamente más que otra. Esto puede ser problemático ya que

muchos algoritmos de aprendizaje automático asumen que las clases están balanceadas. Para resolver este problema, se utilizan técnicas de balanceo de datos como undersampling, oversampling y SMOTE (Synthetic Minority Over-sampling Technique) [33].

- **Undersampling (Submuestreo):** Esta técnica implica reducir aleatoriamente el número de instancias de la clase mayoritaria para equilibrar la distribución de las clases. Aunque el undersampling puede mejorar el rendimiento del clasificador en la clase minoritaria, tiene el riesgo de perder información potencialmente útil [34].
- **Oversampling (Sobremuestreo):** Esta técnica implica replicar aleatoriamente las instancias de la clase minoritaria para equilibrar la distribución de las clases. Aunque el oversampling puede aumentar el rendimiento del clasificador en la clase minoritaria, también puede llevar a un sobreajuste debido a la repetición de instancias [35].
- **SMOTE:** Esta es una técnica de sobremuestreo que crea "sintéticamente" nuevas instancias de la clase minoritaria. En lugar de simplemente replicar las instancias existentes, SMOTE utiliza un algoritmo basado en los vecinos más cercanos para crear nuevas instancias que son combinaciones de las instancias minoritarias cercanas. Esto puede mejorar el rendimiento del clasificador sin aumentar el riesgo de sobreajuste tanto como el oversampling tradicional [36].

Es importante mencionar que no hay una "mejor" técnica para el balanceo de datos. La elección de la técnica de balanceo debe basarse en el problema específico, la naturaleza del conjunto de datos y el algoritmo de aprendizaje automático utilizado.

3.1.4. Optimización de hiper parámetros

En el aprendizaje automático, un hiper parámetro es un parámetro cuyo valor se establece antes del proceso de aprendizaje y no se aprende a partir de los datos. Los hiper parámetros pueden tener un gran impacto en el rendimiento de los modelos y su elección correcta puede ser crítica para obtener buenos resultados [46].

La optimización de hiper parámetros se refiere al proceso de búsqueda del conjunto de hiper parámetros que produce el mejor rendimiento del modelo. Dado que este proceso implica una búsqueda en el espacio de hiper parámetros, puede ser computacionalmente costoso y desafiante, especialmente cuando el número de hiper parámetros es grande [47].

Existen varias técnicas para la optimización de hiper parámetros:

- **Búsqueda en grilla (Grid Search):** Es una técnica simple y exhaustiva que prueba todas las posibles combinaciones de hiperparámetros dentro de un rango predefinido [48].
- **Búsqueda aleatoria (Random Search):** A diferencia de la búsqueda en grilla, la búsqueda aleatoria selecciona aleatoriamente combinaciones de hiperparámetros para probar. Aunque menos exhaustiva, puede ser más eficiente que la búsqueda en grilla, especialmente cuando el número de hiperparámetros es grande [49].
- **Optimización Bayesiana**: Esta es una técnica más avanzada que utiliza métodos de optimización basados en el teorema de Bayes para buscar en el espacio de hiperparámetros. Puede ser más eficiente que las técnicas de búsqueda en grilla y búsqueda aleatoria, especialmente en problemas con muchos hiperparámetros [50].

Es importante destacar que no hay una "mejor" técnica de optimización de hiper parámetros. La elección de la técnica debe basarse en el problema específico, la naturaleza del modelo de aprendizaje automático y las restricciones computacionales.

3.1.5. Validación cruzada

La validación cruzada es una técnica estadística ampliamente utilizada en el aprendizaje automático para evaluar la habilidad de un modelo para generalizar a un conjunto independiente de datos. Ayuda a entender cómo el modelo se desempeñará en nuevos datos y también a seleccionar los hiper parámetros óptimos para el modelo [55].

En la validación cruzada, el conjunto de datos se divide en 'k' subconjuntos, o 'folds'. El modelo se entrena 'k' veces, cada vez usando 'k-1' folds para el entrenamiento y el fold restante para la prueba. La medida de rendimiento del modelo se calcula como el promedio de las medidas de rendimiento obtenidas en las 'k' iteraciones [56].

Existen varios métodos de validación cruzada:

- **K-Fold Cross Validation:** Es el método más comúnmente utilizado. En este método, el conjunto de datos se divide en 'k' subconjuntos de igual tamaño [57].
- **Stratified K-Fold Cross Validation:** Es similar a la validación cruzada de k-folds, pero en este método, se asegura que cada fold tenga la misma proporción de observaciones con una determinada etiqueta de clasificación [58].

- **Leave-One-Out Cross Validation (LOOCV):** Es un caso especial de validación cruzada de k-folds en el que 'k' es igual al número total de observaciones en el conjunto de datos. En cada iteración, una observación se utiliza para la prueba y el resto para el entrenamiento [59].

La elección del método de validación cruzada y el número de folds depende del tamaño y la naturaleza del conjunto de datos.

3.1.6. Regularización L1

La regularización es una técnica utilizada en el aprendizaje automático y la estadística para prevenir el sobreajuste, es decir, que el modelo se adapte demasiado a los datos de entrenamiento y pierda capacidad para generalizar a datos no vistos. La regularización L1, también conocida como Lasso (Least Absolute Shrinkage and Selection Operator), es una forma común de regularización [7].

La regularización L1 agrega un término a la función de costo basado en el valor absoluto de los coeficientes del modelo, de tal forma que la minimización de la función de costo ahora se convierte en un balance entre minimizar el error del modelo y minimizar la magnitud de los coeficientes [52].

Una característica interesante y útil de la regularización L1 es que puede llevar a algunos de los coeficientes del modelo a cero, lo que implica que algunas características se descartan completamente del modelo. Esto hace que la regularización L1 sea útil para la selección de características y para crear modelos más interpretables y más eficientes [53].

En la regresión logística, la regularización L1 puede ser especialmente útil cuando se tienen muchas características y se sospecha que solo un subconjunto es realmente relevante para la predicción. Al imponer una penalización a los coeficientes en la función de costo, la regularización L1 puede ayudar a seleccionar las características más importantes y a prevenir el sobreajuste [54].

3.1.7. Regularización L2

La regularización L2, comúnmente llamada Ridge, es una técnica esencial en el aprendizaje automático diseñada para combatir el sobreajuste. A diferencia de la regularización L1, que se enfoca en el valor absoluto de los coeficientes, la regularización L2 penaliza el cuadrado de la magnitud de estos coeficientes [57]. Este enfoque tiene la intención de castigar valores de coeficientes excesivamente grandes, que pueden ser indicativos de sobreajuste, especialmente en situaciones donde se tiene un alto número de características y se quiere prevenir la multicolinealidad [58].

A diferencia de L1, L2 tiende a disminuir la magnitud de los coeficientes sin anularlos por

completo. Esto implica que todas las características introducidas se consideran en el modelo, pero se limita su impacto potencialmente desmedido [59]. En contextos como la regresión logística, donde se pretende conservar todas las características, pero garantizar un modelo equilibrado, la regularización L2 es particularmente valiosa [60].

3.1.8. Regularización ElasticNet

ElasticNet es una técnica de regularización que combina las propiedades de las regularizaciones L1 y L2. A través de un parámetro de mezcla, se determina el balance entre Ridge (L2) y Lasso (L1). Así, ElasticNet es especialmente útil cuando se tienen muchas características o cuando las características están altamente correlacionadas entre sí [61].

A diferencia de Lasso, que podría seleccionar una característica entre un grupo de características correlacionadas, ElasticNet tiende a seleccionar todas las características correlacionadas, lo que puede ser deseable en ciertos contextos donde se necesita conservar información relacionada [162]. Al mismo tiempo, se aprovecha el efecto de Ridge para garantizar que los coeficientes no crezcan de manera desmedida [103].

En el contexto de la regresión logística, cuando se enfrenta a datos de alta dimensionalidad o a conjuntos de datos donde las características presentan multicolinealidad, ElasticNet puede ser una herramienta valiosa. Combina la capacidad de Lasso de realizar selección de características con la robustez de Ridge para manejar la multicolinealidad [104].

3.1.9. Reducción de dimensionalidad

La reducción de dimensionalidad es el proceso de transformar datos de alta dimensión en datos de baja dimensión, de manera que se conserve la mayor cantidad posible de información relevante. Se utiliza ampliamente en el aprendizaje automático, pero también se ve en el aprendizaje no supervidado y la ciencia de datos, especialmente cuando se trata con conjuntos de datos de alta dimensión. La reducción de dimensionalidad puede ayudar a mejorar el rendimiento de los algoritmos de aprendizaje automático, a reducir el ruido y la redundancia en los datos, y a hacer que los datos sean más manejables y comprensibles [60].

SelectKBest es una técnica de reducción de dimensionalidad que selecciona las 'k' mejores características basándose en una determinada medida de importancia. Esta técnica es útil para la selección de características y puede ser utilizada como un paso de preprocesamiento para reducir la dimensionalidad de los datos antes de aplicar un algoritmo de aprendizaje automático [61].

El método que SelectKBest utiliza para determinar la importancia de las características depende de la tarea de aprendizaje automático. Para la clasificación, se pueden utilizar pruebas estadísticas como la prueba chi-cuadrado o la prueba ANOVA de un solo factor. Para la regresión, se pueden utilizar medidas de correlación como la correlación de Pearson [62].

Es importante mencionar que SelectKBest es un método univariante de selección de características, lo que significa que evalúa la importancia de cada característica de forma independiente. Por lo tanto, puede no ser capaz de capturar las interacciones entre características. Para tareas donde las interacciones entre características son importantes, pueden ser más adecuadas técnicas de reducción de dimensionalidad multivariante como el Análisis de Componentes Principales (PCA) o el Análisis de Discriminantes Lineales (LDA) [63].

3.1.10. Procesamiento de lenguaje natural [2]

Rama de la inteligencia artificial que permite estructurar textos en lenguaje humano de forma que sea procesable por computadora. Se pretende buscar el mecanismo por el cual se programen computadores para procesar y analizar grandes cantidades de datos del lenguaje natural (forma en que se comunican las personas). Para el fin mencionado previamente existen diferentes mecanismos y técnicas, entre las que se encuentran:

- **Tokenización:**

La tokenización es un paso fundamental en el procesamiento del lenguaje natural y en la minería de textos. Este proceso implica dividir un cuerpo de texto en unidades más pequeñas, conocidas como "tokens" [16].

Un token puede ser una palabra, una frase, una oración o cualquier otro fragmento de texto que tenga algún significado en el contexto de análisis. En la mayoría de los casos, especialmente en el análisis de texto y en la minería de datos, los tokens suelen ser palabras o términos.

En el contexto del análisis de sentimientos, la tokenización es un paso crucial, ya que convierte el texto libre (como los tweets) en una forma estructurada que puede ser analizada y procesada por los algoritmos de aprendizaje automático [17].

La tokenización también puede implicar la eliminación de palabras vacías (palabras que no aportan mucho significado a las frases, como "y", "es", "en", etc.), la normalización de las palabras (como convertir todo a minúsculas o eliminar la puntuación) y la lematización o la derivación (procesos que reducen las palabras a su raíz o base) [18].

Sin embargo, es importante destacar que la tokenización puede ser un proceso desafiante en algunos casos, especialmente cuando se manejan idiomas con reglas gramaticales complejas, jerga, abreviaturas o errores ortográficos, que son comunes en los textos de las redes sociales como Twitter [19].

- **Palabras de parada o stop words**:

Las "Stop Words", también conocidas como palabras de parada, son palabras comúnmente utilizadas en un idioma que se filtran durante el procesamiento del lenguaje natural (NLP). Estas palabras suelen ser preposiciones, pronombres y conjunciones, como "y", "es", "en", "de", "la", "el", etc. [20].

La razón principal para eliminar estas palabras es que suelen ser muy comunes y no aportan mucho significado a una frase en términos de contenido único. Al eliminar las palabras de parada, los datos se vuelven menos ruidosos y más centrados en las palabras que realmente importan para el análisis [20].

En el contexto del análisis de sentimientos, las palabras de parada son generalmente eliminadas para que los algoritmos de aprendizaje automático puedan centrarse en palabras que probablemente indiquen un sentimiento, como "feliz", "triste", "enojado", etc. [22].

Sin embargo, es importante notar que hay casos en los que las palabras de parada pueden ser útiles o incluso necesarias. Por ejemplo, en algunas tareas de NLP como el modelado de temas o cuando se necesita preservar el contexto completo de las oraciones, podría ser mejor conservar las palabras de parada [23].

- **Bag of words, bigramas y trigramas**

El modelo "Bag of Words" (BoW) es una representación simplificada utilizada en el procesamiento del lenguaje natural y la recuperación de información. En este modelo, un texto (como una frase o un documento) se representa como el conjunto de sus palabras, sin tener en cuenta el orden o la gramática, pero manteniendo la multiplicidad [12].

Aunque el modelo BoW es muy eficiente y fácil de implementar, tiene ciertas limitaciones. En particular, al no considerar el orden de las palabras, el BoW puede perder el contexto y la semántica de las frases, lo que es especialmente importante para el análisis de sentimientos [13].

Para solucionar este problema, se utilizan los modelos de bigramas y trigramas, que consideran pares de palabras consecutivas y tríos de palabras consecutivas, respectivamente. Los bigramas y trigramas pueden capturar más contexto que el modelo BoW solo, permitiendo analizar el sentimiento de frases más efectivamente [14].

Por ejemplo, la frase "*don’t like*" tiene un significado negativo, pero el modelo BoW la interpretaría como dos palabras independientes, "*don’t*" y "*like*", que podrían interpretarse erróneamente como un sentimiento positivo debido a la palabra "*like*". En cambio, un bigrama captaría la frase "*don’t like*" como una sola entidad, preservando su significado negativo [15].

En conclusión, BoW, bigramas y trigramas son técnicas fundamentales en el procesamiento del

lenguaje natural y pueden ser muy efectivas para el análisis de sentimientos, especialmente cuando se utilizan juntas.

- **Steming y lematización**:
  El Stemming y la Lematización son dos técnicas comúnmente utilizadas en el procesamiento del lenguaje natural para reducir las palabras a su raíz o forma base. Ambos son técnicas de normalización de texto que ayudan a reducir el ruido y la variabilidad en los datos de texto, pero difieren en su enfoque y resultados [24].

  El stemming implica cortar las palabras a su raíz o "tallo" eliminando los prefijos y sufijos. Por ejemplo, las palabras "corriendo", "corredor" y "corre" podrían ser todas reducidas a su raíz, "corr". Este proceso es heurístico y no tiene en cuenta el contexto de la palabra en la oración. Aunque el stemming es eficiente, puede dar lugar a "stems" que no son palabras reales y pueden perder información semántica [25].

  Por otro lado, la lematización reduce las palabras a su forma base o "lema" teniendo en cuenta el contexto y la parte de la palabra en la oración (verbo, sustantivo, adjetivo, etc.). Por ejemplo, "corriendo" se lematizaría a "correr", que es la forma base del verbo. La lematización es más sofisticada y precisa que el stemming, pero también es computacionalmente más costosa [26].

  En el análisis de sentimientos, tanto el stemming como la lematización pueden ser útiles para reducir la variabilidad de los datos y concentrar el análisis en el significado principal de las palabras. La elección entre stemming y lematización puede depender del contexto específico, la complejidad del lenguaje y la capacidad de procesamiento disponible [27].

- **Expresiones regulares:**
  Las Expresiones Regulares, también conocidas como regex o regexp, son una herramienta poderosa para el procesamiento de texto. Son una secuencia de caracteres que forma un patrón de búsqueda, principalmente utilizada para la búsqueda de patrones en cadenas de texto [37].

  El propósito principal de las expresiones regulares es identificar cadenas de texto que coincidan con un patrón específico. Los patrones pueden incluir caracteres literales, agrupaciones, rangos, cuantificadores, entre otros [38].

Las expresiones regulares son muy útiles en varias tareas de procesamiento de texto, incluyendo la búsqueda y sustitución de texto, la validación de entradas de usuario, el parseo de archivos de texto y muchas otras tareas de análisis de texto [39].

En el contexto del análisis de sentimientos, las expresiones regulares pueden ser útiles para la limpieza y preprocesamiento de datos. Por ejemplo, se pueden usar para eliminar los enlaces URL o las menciones a usuarios de los tweets antes de analizar el sentimiento [22].

Sin embargo, las expresiones regulares también pueden ser difíciles de manejar debido a su complejidad y la variedad de sintaxis utilizada en diferentes lenguajes y bibliotecas. Además, aunque son muy poderosas para el procesamiento de texto, no pueden manejar bien las dependencias contextuales o las estructuras de lenguaje más complejas [41].

- **Vectorización de textos:**
  La vectorización de textos es el proceso de convertir datos textuales en representaciones numéricas que pueden ser utilizadas por los algoritmos de aprendizaje automático. Existen varias técnicas para la vectorización de textos, entre las que destacan CountVectorizer y TfidfVectorizer [26].

  - CountVectorizer: Es una técnica de vectorización de textos que simplemente cuenta la frecuencia de cada palabra en un documento. Es uno de los métodos más simples y comunes para transformar el texto en un formato que pueda ser procesado por los algoritmos de aprendizaje automático. Sin embargo, una desventaja de esta técnica es que da el mismo peso a todas las palabras, independientemente de su importancia relativa en el documento [43].
  - TfidfVectorizer (Term Frequency-Inverse Document Frequency): Es una técnica de vectorización de textos que considera no solo la frecuencia de las palabras en un documento (Term Frequency), sino también su rareza en el corpus general de documentos (Inverse Document Frequency). De esta forma, da más peso a las palabras que son comunes en un documento específico pero raras en el corpus general, que son probablemente las palabras más relevantes para el significado del documento. TfidfVectorizer puede dar mejores resultados que CountVectorizer cuando los documentos son largos y contienen muchas palabras comunes [44].

Ambas técnicas producen una matriz de términos por documento, que puede ser utilizada como entrada para la mayoría de los algoritmos de aprendizaje automático. Sin embargo, es importante mencionar que estas técnicas no capturan la semántica de las palabras ni la estructura de las frases o los documentos, por lo que pueden no ser adecuadas para tareas de procesamiento del lenguaje natural más complejas [45].

3.2. Antecedentes / Trabajos relacionados

Sistema Deep Learning para el análisis de sentimientos en opiniones de productos para la ordenación de resultados de un buscador semántico [64]

En este trabajo se presentan detalles, resultado de experimentación de algoritmos de aprendizaje profundo para la clasificación de sentimientos sobre la opinión de productos. En él se utiliza como fuente de datos el Corpus TASS en donde se ofrecen 6800 tweets para entrenamiento, los cuales vienen acompañados de una polaridad supervisada. Se usó como técnica de línea base Naive Bayes Multinomial, Regresión Logística, Maquinas de soporte vectorial y Bosques aleatorios. Luego como técnicas a evaluar mejoras se utilizaron redes neuronales (LSTM), usando 30 epochs y función de activación RELU. Como resultado se obtiene una mejora al implementar los mecanismos de Aprendizaje Profundo en indicadores como recall y precisión, adicionalmente se observa que se enfrentaron a retos como el desbalance de los datos en términos del número de registros dentro de cada grupo de la respuesta (sentimiento).

Un análisis de sentimiento en Twitter con Machine Learning: Identificando el sentimiento sobre las ofertas de #BlackFriday [65]

En esta iniciativa se pueden tomar diferentes desarrollos desde la captura de datos hasta la utilización de diferentes técnicas de clasificación de sentimientos tomadas de *twitter* respecto a opiniones generadas a partir de ofertas en black Friday. Los datos fueron obtenidos mediante el API expuesta por Twitter desde la que se descargó un total de 2.204 *tweets* usando como referencia el #BlackFriday. Para la clasificación de textos se utilizó una librería preconstruida “MonkeyLearn”, la cual permite la clasificación de textos en español en sentimientos positivos, negativos y neutros. Como conclusiones se obtienen la clasificación de sentimientos para diferentes compañías generadoras de promociones en el Black Friday.

Análisis de sentimientos en Twitter: Una implementación sobre Cloudera [66]

En este trabajo se observan diferentes aproximaciones desde la obtención de los textos de twitter, hasta el procesamiento de computación distribuida con la herramienta hadoop. En este caso también se usó la API para la obtención de los textos de los tweets asociados al proyecto de ley antidespido de argentina, obteniendo 240.000 tweets. Es así que se enfrentaron a un requerimiento amplio de capacidad computacional, razón por la cual se requirió el uso de componentes de computación distribuida, la técnica usada para la clasificación de los sentimientos fue la de diccionario léxico. Como conclusión se obtiene que el método es altamente dependiente de los diccionarios y el poder de clasificación ante ironías no es muy alto,

adicionalmente presenta problemas de clasificación ante la presencia de lenguaje no formal o previamente no contemplado.

Caso de estudio de análisis de sentimientos en Twitter: Tratado de libre comercio de América del Norte [67]

En este trabajo se observan diferentes técnicas de minería y procesamiento de texto sobre el tratado de libre comercio en América del Norte, logrando una precisión de clasificación del texto del 87%. Se presentan comparaciones de los resultados al realizar entrenamiento con diferentes algoritmos de machine learning. De aquí también se puede revisar la revisión de estadísticas de exactitud utilizadas para la comparación. Finalmente se observa un mecanismo de votación de algoritmos para la clasificación en tiempo real.

Implementación de un modelo de análisis de sentimientos con respecto a la JEP basado en minería de datos en twitter [68]

En este caso se realiza la descarga de datos utilizando el API de twitter mediante la librería de Python tweepy enfocados en aquellos casos en que se mencionara la palabra “JEP”, obteniendo 25.000 tweets en formato csv. Para el análisis de la información se usaron unigramas y bigramas. Este trabajo busca la identificación de sentimientos de twitter respecto a comentarios asociados a la JEP (Jurisdicción Especial para la Paz). Dentro de los algoritmos explorados se encontraron Maquinas de vectores de soporte, Random forest, y Naive Bayes. Las medidas de evaluación para comparar el desempeño de los modelos fueron precisión, recall, F1 Score y exactitud, obteniendo el mejor desempeño con el algoritmo de Random Forest.

Técnicas de análisis de sentimientos aplicadas a valoración de opiniones en el lenguaje español [69]

Se obtiene la fuente de datos de opiniones extraídas de la página www.cinesagentinos.com.ar, estas opiniones son reseñas de distintas películas que los usuarios generan sin estructura establecida, adicional a esto se cuenta con una calificación entre 1 y 5 estrellas. El número de registros obtenidos fue de 52.309. las métricas de comparación de los modelos fueron accuracy, precisión, recall y F1 score. Entre las técnicas evaluadas se encuentran, naive bayes, random. La técnica que mejores resultados genero fu BETO

Resultados del estudio de revisión y actualización del plan maestro de movilidad de Bogotá [70]

Documento que compila el conjunto de metas e iniciativas planteadas del plan de movilidad para la ciudad de Bogotá. Se resalta el plan distrital de desarrollo, identificando como problemática principal “una experiencia de viaje poco satisfactoria, principalmente para poblaciones más vulnerables “. También se identifican causas y efectos, directos e indirectos en cada caso. Como una de las causas principales se evidencia el vertiginoso aumento en el número de viajes requeridos en la ciudad. Finalmente se identifica conjunto de alternativas que

permitirían mejorar la condición problemática inicialmente identificada

4. Preparación de la información

4.1. Proceso de descarga

El proceso de recopilación de datos de Twitter se llevó a cabo utilizando snscrape, una potente biblioteca de Python diseñada para la extracción de datos de las redes sociales mediante la técnica del webscraping [70].

Se estableció una lista de consultas de búsqueda específicas a explorar en Twitter. En este caso, las consultas fueron @sectormovilidad, @bogotatransito, @felipe_ramir, #MuéveteBienInformado, y #GerenciaEnVía. Además, se definió un rango de fechas de búsqueda, desde el 1 de enero de 2010 hasta el 2 de agosto de 2022.

Utilizando snscrape, el programa comenzó a buscar tweets que correspondieran a cada consulta dentro del rango de fechas establecido. Se implementó un sistema de seguimiento para proporcionar actualizaciones regulares a lo largo del proceso de búsqueda y recopilación.

Para cada tweet que coincidía con las consultas y estaba dentro del rango de fechas, snscrape recolectó una serie de datos. Estos datos incluían detalles sobre el tweet, como la fecha y el contenido, y también información sobre el usuario que lo publicó, como su nombre de usuario, número de seguidores y ubicación. Cada uno de estos tweets se almacenó para su posterior análisis.

Este proceso se repitió para cada consulta en la lista, y al final, todos los tweets recopilados se unieron en un único conjunto de datos.

Además, se extrajo información adicional de los tweets, como las coordenadas de latitud y longitud de cualquier geolocalización adjunta al tweet, y el nombre del lugar, si estaba disponible.

Al final de este proceso, se recopilaron un total de 888,927 tweets basándonos en las consultas y el rango de fechas especificado. Vale la pena señalar que se probó el uso de la API oficial de Twitter inicialmente, pero se descubrió que esta restringía el número de tweets que se podían descargar, lo que motivó el uso de la biblioteca snscrape y su método de webscraping.

A continuación, se presenta una descripción de los datos obtenidos en la descarga

Field	Type	Description
place_name	String	Tweet place name
tweet_id	Int	Identification number of each tweet.
text	String	Content of each tweet.

favorite_count	Int	Times a tweet has been marked as favorite.
retweet_coun	Int	Times a tweet has been re-tweeted
consulta	String	Source of the tweet
source	String	Source device of the tweet
longitude	Int	Tweet longitude
latitude	Int	Tweet latitude
datetime	Date	Tweet creation date and time
Conversation_id	Int	Identification number of each conversation.
username	String	Original Tweet's author
displayname	String	Original Tweet's public author´s name
user_description	String	Tweet's user type
user_followers_count	Int	Number of Tweet's user followers
user_friends_count	Int	Number of Tweet's user friends
user_statuses_count	Int	Number of Tweet's user status
user_favorites_count	Int	Number of Tweet's user favorites
user_listed_count	Int	Number of Tweet's user listed
user_media_count	Int	Number of Tweet's user media
user_location	Int	Tweet's user location
reply_count	Int	Number of user Tweet's replies
like_count	Int	Number of user Tweet's likes
quote_count	Int	Number of user Tweet's quote
lang	String	Original Tweet's Language
coordinates	Int	Original Tweet's latitude and longitude

Tabla 1: Campos obtenidos de cada Tweet

4.2. Calificación Manual textos

De 888,927 tweets descargados se realizó calificación manual de 1,228 tweets seleccionados aleatoriamente, la cantidad original era 1,300 (fue la cantidad propuesta como objetivo por limitaciones de tiempo y alcance) pero fue necesario descartar 72 por no ser de usuarios. Estos fueron evaluados manualmente por dos personas del grupo; en los casos en que existiera desacuerdo en la calificación, un tercero daba su concepto y se generaba un acuerdo por mayoría. En la ilustración 1 se puede observar el resultado de las diferentes calificaciones por cada evaluador

		Calificador 1			
		positivo	**neutro**	**negativo**	**Total general**
Caslificador 2	positivo	117	1		**118**
	neutro	2	278	4	**284**
	negativo		6	820	**826**
	Total general	**119**	**285**	**824**	**1228**

Ilustración 1: resultados calificación manual

Según estos resultados se puede obtener el estimador Cohen kappa [105] para la validación de acuerdo con un valor de 0.98, lo que en términos de la referencia implica un acuerdo casi perfecto, por lo que puede concluirse que la evaluación manual es válida por acuerdo entre los evaluadores

Utilizamos para este proceso de calificación un código phyton de la librería Pandas que convierte

en integer el id de cada tweet, realizaba la localización en una sencilla base de datos Excel y se visualizaba el tweet con sus emoticogens, o @ y otros.

```
[2]: import pandas as pd
     df=pd.read_csv('twitts_2.csv')

•[235]: tweet_id ="TID-1552995244299505665"

        tweet_id = int(tweet_id[4:])
        tweet_id

        texto = df.loc[df['tweet_id']==tweet_id, 'text'].iloc[0]

        print(texto)

        @BluRadioCo Todo lo quieren prohibir, en vez de regularizar, controlar, por l
        os derechos de movilidad y trabajo, la señora @ClaudiaLopez y su @SectorMovil
        idad, quieren prohibir, patinetas eléctricas, bicicletas eléctricas, en que m
        undo vive @MinTransporteCo se necesita carriles especiales
```

Ilustración 2: ejemplo de código para lectura y calificación de Tweets

Luego de correr este código con cada uno de los 1245 Tweets se consolidó en una matriz de excel y se realizó un análisis de los resultados. Un gran número (828), dio como comentarios negativos, otro menor (280) dio positivo y neutro (100).

Ahora el reto es poder identificar cuál de los modelos es el mejor para poder clasificar estos tweets, teniendo en cuenta que existe una gran diferencia entre los datos de los comentarios negativos, neutros y positivos.

4.3. Limpieza de datos

El capítulo "Limpieza de datos" se centra en la preparación de los datos recopilados para su análisis posterior. El objetivo es refinar los datos de los tweets, eliminando la información no relevante y normalizando el texto para que pueda ser procesado de manera más eficiente.

Para realizar esta tarea, se implementó una función llamada clean_txt que opera sobre el texto de cada tweet. Esta función implementa varios pasos para limpiar y normalizar el texto.

Primero, la función convierte todo el texto a minúsculas. Esta etapa es fundamental para la normalización del texto, ya que garantiza que todas las palabras se traten de la misma manera, independientemente de cómo se hayan escrito originalmente [18].

A continuación, la función elimina todas las menciones a otros usuarios de Twitter del tweet (es decir, cualquier texto que comienza con @). Estas menciones no son útiles para el análisis del

contenido del tweet, ya que no proporcionan información sobre el tema o el sentimiento del tweet.

Consideramos que generalmente los enlaces a otras páginas, puntuación o salto de línea no aportan información suficientemente representativa para el análisis del sentimiento del tweet, inclusive en ocasiones estos elementos podrían interferir con la clasificación, por ello, la función elimina estos elementos.

En esta etapa no se eliminan Stop Words pues se considera de forma preliminar que este elemento particular podría aportar información suficiente para mejorar la clasificación (ej: “y del carro que”, solo quedaría “carro”), sin embargo, este elemento fue evaluado y contrastado en diferentes escenarios con el fin de evaluar su impacto.

El resultado de aplicar esta función es un texto limpio y normalizado que está listo para el análisis posterior.

5. Selección de modelos entre técnicas con optimización de hiper parámetros

5.1. Descripción metodología

Se exploraron seis tipos de representaciones de texto, cada uno con un enfoque único en el tratamiento de las "stop words" (palabras vacías), el uso de la lematización y el "stemming" [18)].

Las representaciones de texto incluidas en el estudio son las siguientes:

1. Sin eliminar "stop words"
2. Sin eliminar "stop words", con lematización
3. Sin eliminar "stop words", con "stemming"
4. Eliminando "stop words"
5. Eliminando "stop words", con lematización
6. Eliminando "stop words", con "stemming"

La necesidad de probar con y sin "stop words" se fundamenta en que, en algunos casos, al eliminar estas palabras comunes, se pueden dejar los textos de los tweets sin palabras o con una sola palabra. Dado que los mensajes de Twitter son inherentemente cortos, las "stop words" pueden tener un papel relevante en la representación del sentimiento del texto. Este fenómeno ha sido discutido en la literatura, donde se ha observado qué en ciertos contextos, las "stop words" pueden aportar información significativa que contribuye a la interpretación del sentimiento [26]. Tomando en cuenta lo anterior, para los casos en que se eliminan las stop words se trabajará con una conjunto de tamaño diferente pues en algunos casos al eliminar las *stop words* el tweet queda

vacío, por lo que se genera un subconjunto de datos de dimensiones: 822 negativos, 253 neutros y 117 positivos (total de 1,192). La implementación de la eliminación de las “Stop Words” se realizó mediante el uso de la librería nltk en el conjunto de stopwords en español. Cabe anotar que para los conjuntos de datos en los que se eliminaron las “Stop Words” fue necesario eliminar algunos registros pues el texto resultante era vacío en la Ilustración 3: Número de registros según conjunto de datos, se observa el número de registros en cada caso (esto eventualmente tendrá impacto en los resultados de las matrices de confusión de distintos orígenes de datos)

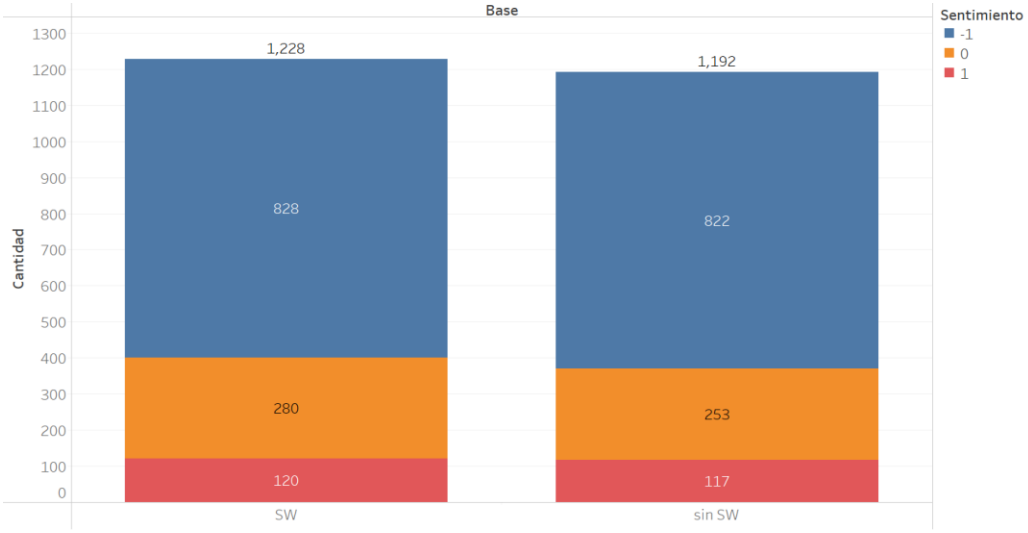


Ilustración 3: Número de registros según conjunto de datos

Además, se decidió probar con y sin lematización y "stemming". La lematización reduce las palabras a su forma base o raíz de diccionario, como el lema que puedes encontrar en un diccionario, mientras que el "stemming" recorta las palabras a su raíz sin garantizar que el resultado sea una palabra real. Ambas técnicas son formas comunes de normalización de texto en el procesamiento de lenguaje natural [108]. Aunque estas técnicas pueden simplificar el texto y reducir la dimensionalidad de los datos, también pueden eliminar cierta cantidad de información semántica y contextuales que pueden ser vitales para la detección de sentimientos [109].

Por lo tanto, la elección de incluir estas técnicas en la representación de texto depende en gran medida del dominio de los datos y del compromiso entre la precisión del modelo y la interpretabilidad del texto. Por ejemplo, para el caso de los datos de Twitter, que a menudo incluyen errores ortográficos, abreviaturas y jerga, la lematización y el "stemming" pueden ser menos efectivos y potencialmente contraproducentes [100].

Adicionalmente de los métodos de simplificación de texto mencionados anteriormente, también se consideraron dos técnicas de representación de datos para la evaluación: CountVectorizer y TF-IDF (Term Frequency-Inverse Document Frequency). Aunque los detalles de estos modelos se explican en el marco teórico, es importante destacar por qué se eligieron para la evaluación y

cómo contribuyen a la búsqueda del mejor modelo para clasificar el sentimiento del texto.

CountVectorizer y TF-IDF son dos técnicas comunes para convertir el texto en una forma que puede ser procesada por los modelos de aprendizaje automático. Ambas técnicas transforman el texto en vectores de características, pero difieren en cómo cuantifican la importancia de cada palabra.

CountVectorizer simplemente cuenta la frecuencia de cada palabra en el texto, lo que puede ser útil para identificar las palabras más comunes. Sin embargo, este método no tiene en cuenta la importancia de las palabras en el contexto del documento completo o del conjunto de documentos [12].

Por otro lado, TF-IDF no sólo considera la frecuencia de las palabras, sino también su rareza en el conjunto de documentos. Las palabras que aparecen frecuentemente en un documento, pero raramente en otros documentos se consideran más importantes, lo que puede ser útil para identificar palabras clave y temas [93].

La elección entre CountVectorizer y TF-IDF puede tener un impacto significativo en los resultados del análisis de sentimientos. Por ejemplo, TF-IDF puede ser más eficaz para identificar el sentimiento en textos largos y complejos, mientras que CountVectorizer puede ser suficiente para textos cortos y simples como los tweets. Por lo tanto, es esencial evaluar ambos métodos para encontrar el que mejor se adapte a nuestros datos [104].

Al combinar estas dos técnicas de representación de datos con los seis métodos de simplificación de texto mencionados anteriormente, se obtienen doce modelos diferentes para la evaluación. Esta amplia gama de modelos nos permite explorar una variedad de enfoques para el análisis de sentimientos y aumenta nuestras posibilidades de encontrar el modelo más eficaz para nuestros datos [105].

A parte de las técnicas de simplificación y representación de texto, también se evaluaron tres técnicas de aprendizaje automático: Naive Bayes, Random Forest y Regresión Logística. Estos algoritmos se seleccionaron debido a su popularidad y eficacia en tareas de clasificación de texto, como el análisis de sentimientos [104].

Para cada uno de estos algoritmos, se evaluaron dos versiones del modelo: una con los parámetros por defecto y otra optimizada mediante la técnica de Grid Search. Grid Search es un método de optimización de hiperparámetros que explora sistemáticamente una cuadrícula de posibles combinaciones de hiperparámetros para encontrar la que produce el mejor rendimiento del modelo [107].

La evaluación de los modelos con parámetros por defecto proporciona una línea de base para comparar la eficacia de los algoritmos. Por otro lado, la evaluación de los modelos optimizados

permite explorar el potencial de mejora de los algoritmos cuando se ajustan sus hiperparámetros [108].

Al combinar estas tres técnicas de aprendizaje automático con las doce representaciones de texto mencionadas anteriormente, se obtienen un total de 36 modelos diferentes para la evaluación. Sin embargo, al considerar tanto los modelos con parámetros por defecto como los optimizados, el número total de modelos evaluados se duplica a 72. Esta amplia gama de modelos nos permite explorar una variedad de enfoques para el análisis de sentimientos y aumenta nuestras posibilidades de encontrar el modelo más eficaz para nuestros datos [109].

El desbalance en los datos es un desafío común en el análisis de sentimientos, y nuestro conjunto de datos no es una excepción, con 828 textos calificados como negativos, 280 como neutros y 120 como positivos. Para abordar este desafío y evaluar su impacto en la eficacia de los modelos, se consideraron seis escenarios diferentes para cada uno de los modelos mencionados anteriormente.

1. Conjunto de datos original: Este escenario sirve como línea de base para la evaluación. Aunque los datos están desbalanceados, es importante evaluar cómo se desempeñan los modelos en las condiciones originales, ya que esto refleja la distribución real de los sentimientos en los datos [28].
2. Oversampling replicando muestras: En este escenario, se aumentó el número de textos neutros y positivos (llevando siempre al número de muestras negativas 828 para las bases con *stop words* y 822 en los casos en los que se eliminaron) replicando muestras existentes para equilibrar la distribución de los sentimientos. El oversampling puede ayudar a mejorar la eficacia de los modelos en conjuntos de datos desbalanceados, pero también puede llevar al sobreajuste si se replican demasiado las mismas muestras [33].
3. Oversampling con ampliación de hiperparámetros: Este escenario es similar al anterior, pero además se amplió el rango de los hiperparámetros para evaluar si el rango original era adecuado. La optimización de hiperparámetros es un aspecto crucial en el aprendizaje automático, y la ampliación del rango puede ayudar a explorar más a fondo el espacio de hiperparámetros y encontrar una combinación más eficaz [27].
4. Oversampling con SMOTE: SMOTE (Synthetic Minority Over-sampling Technique) es una técnica de oversampling que genera muestras sintéticas en lugar de replicar las existentes. SMOTE puede ser más eficaz que el oversampling simple para evitar el sobreajuste y mejorar la eficacia de los modelos en conjuntos de datos desbalanceados [33].
5. Grid Search con función de coste personalizada: En este escenario, se utilizó una función de coste personalizada (ilustración 3) en el Grid Search que castiga más los casos más graves de error de clasificación, como calificar un texto negativo como positivo y

viceversa. Esta función de coste refleja la importancia de evitar estos errores graves en el análisis de sentimientos [104].

		Predicción		
		Neg	Neu	Pos
Real	Neg	1	-1	-2
	Neu	-1	1	-1
	Pos	-2	-1	1

Ilustración 4: valores función coste personalizada

6. Reducción de la dimensionalidad con Chi-cuadrado: En este escenario, se redujo la dimensionalidad de la muestra del modelo con oversampling al 80% de los atributos más relevantes usando la técnica de Chi-cuadrado. La reducción de la dimensionalidad puede ayudar a mejorar la eficacia y la eficiencia de los modelos al eliminar las características menos relevantes [61].

Cada uno de estos escenarios aporta una perspectiva única sobre cómo diferentes técnicas pueden afectar la eficacia de los modelos de análisis de sentimientos. Al evaluar todos estos escenarios, podemos obtener una visión más completa de cómo manejar el desbalance en los datos y cómo optimizar los modelos para nuestro conjunto de datos específico.

Para cada uno de los escenarios mencionados anteriormente, se utilizó una división de los datos en un 80% para entrenamiento y un 20% para pruebas. Esta división es una práctica común en el aprendizaje automático y tiene como objetivo proporcionar una evaluación justa y robusta del rendimiento del modelo. El conjunto de entrenamiento se utiliza para ajustar el modelo, mientras que el conjunto de pruebas se utiliza para evaluar cómo se desempeña el modelo en datos no vistos previamente. Esta evaluación puede proporcionar una estimación realista de cómo se desempeñará el modelo en la práctica [56].

Tanto para CountVectorizer como para TF-IDF, se utilizaron n-gramas de 1 a 3. Los n-gramas son secuencias contiguas de n palabras en el texto. Al considerar n-gramas en lugar de palabras individuales, podemos capturar más contexto y patrones de palabras, lo que puede ser especialmente útil para el análisis de sentimientos, donde el significado a menudo depende del contexto [56].

Para el Grid Search, se utilizaron 5 folds. Esto significa que los datos se dividieron en 5 partes, o "folds", y el Grid Search se realizó utilizando un procedimiento de validación cruzada de 5 folds. En cada iteración del procedimiento, 4 folds se utilizan para el entrenamiento y 1 fold se utiliza para la validación. La validación cruzada es una técnica robusta para la evaluación y la optimización

de hiperparámetros que puede proporcionar una estimación más precisa del rendimiento del modelo y evitar el sobreajuste [32].

La métrica principal que se utilizará para evaluar los modelos será el F1 score promedio. El F1 score es una métrica que combina la precisión y la sensibilidad (recall) en un solo número, proporcionando una medida de la eficacia del modelo que tiene en cuenta tanto los falsos positivos como los falsos negativos. El F1 score es especialmente útil en situaciones donde las clases están desbalanceadas, como en nuestro caso [32].

El F1 score promedio se calcula tomando la media de los F1 scores para cada clase. Esta métrica proporciona una visión general de la eficacia del modelo en todas las clases, sin dar más peso a ninguna clase en particular. Esto es importante en nuestro caso, ya que queremos que nuestro modelo sea eficaz en la detección de todos los sentimientos, no sólo los negativos que son la mayoría en nuestro conjunto de datos.

No se utiliza el F1 score promedio ponderado porque daría más peso a los textos negativos, que son la mayoría en nuestro conjunto de datos. Aunque los textos negativos son importantes, también queremos que nuestro modelo sea eficaz en la detección de textos neutros y positivos. Al utilizar el F1 score promedio en lugar del promedio ponderado, nos aseguramos de que todas las clases se consideren igualmente importantes en la evaluación del modelo [39].

En los capítulos siguientes, se abordará cada técnica (Naive Bayes, Random Forest y Regresión Logística) de manera individual, explorando todos los escenarios mencionados anteriormente. Para cada técnica, se registrarán las características de la simplificación de palabras, el modelo de representación y el escenario escogido. También se registrarán métricas de rendimiento clave, incluyendo la precisión, el recall, y el F1 score promedio [32].

En cada técnica (capítulo), se seleccionarán los cinco modelos con el mejor rendimiento en términos del F1 score promedio. Para estos modelos, se proporcionarán métricas detalladas, incluyendo la matriz de confusión y el reporte completo de clasificación. Estas métricas proporcionarán una visión más detallada del rendimiento del modelo, permitiendo una evaluación más completa de su eficacia [28].

Finalmente, entre estos modelos, se elegirá el que ofrezca el mejor rendimiento a la luz de las necesidades del negocio. Esta elección se basará no sólo en las métricas de rendimiento, sino también en la interpretabilidad del modelo y su adecuación a las características específicas de los datos de Twitter [73].

En resumen, este análisis diversificado de representaciones de texto busca proporcionar una visión integral de cómo diferentes técnicas de simplificación y representación pueden afectar la eficacia del análisis de sentimientos. Los resultados de este análisis contribuirán a la optimización continua de los hiperparámetros de nuestros modelos de análisis de sentimientos [102].

5.2. Herramientas utilizadas

Scikit-learn, conocida también como sklearn, es una reconocida biblioteca de Python ampliamente empleada en el ámbito de aprendizaje automático y análisis de datos [67]. Como producto de código abierto, posee un extenso conjunto de algoritmos tanto de aprendizaje supervisado como no supervisado, junto con herramientas valiosas para la evaluación y selección de modelos, así como funciones destinadas al preprocesamiento de datos. Su diseño tiene la intención de ser intuitivo para el usuario, sin dejar de lado su eficiencia en rendimiento [89].
En el contexto de nuestro proyecto de análisis de sentimientos en comentarios de Twitter se utilizan diferentes funciones de esta librería, entre las que se encuentran: CountVectorizer, MultinomialNB, GridSearchCV, TfidfVectorizer, confusion_matrix entre otras.

Natural Language Toolkit, más conocido por sus siglas NLTK, es una de las herramientas que se destaca en el presente proyecto. Esta biblioteca para el procesamiento del lenguaje natural (PLN) en Python, proporciona un conjunto de métodos eficientes y prácticos para preprocesar y analizar texto en lenguaje natural. NLTK es extraordinariamente útil en nuestro proyecto debido a su amplio conjunto de características y su versatilidad.

La librería NLTK nos ofrece una serie de funcionalidades necesarias para la realización de nuestro análisis de sentimientos. Dentro de sus ventajas, podemos destacar la facilidad para realizar tareas como la tokenización, la eliminación de palabras vacías (stop words), el etiquetado gramatical (POS tagging), el análisis sintáctico, y la detección de entidades nombradas. Además, NLTK provee recursos léxicos como WordNet, y corpora de entrenamiento que facilitan la construcción y validación de modelos de aprendizaje automático.

En el contexto de nuestro proyecto, NLTK facilita la fase inicial de preprocesamiento del texto, lo que es fundamental para luego aplicar modelos de aprendizaje automático en el análisis de sentimientos. Con su ayuda, podemos convertir los datos en bruto de las menciones en Twitter en información estructurada y útil para nuestro estudio. Su flexibilidad y amplio alcance hacen de NLTK una elección sobresaliente y esencial para el desarrollo de nuestro proyecto.

5.3. Definición línea base

Como punto de comparación contra todo modelo desarrollado dentro del presente proyecto se define un modelo de línea base, esto con el fin de entender el potencial de mejora de los diferentes modelos evaluados.

El modelo elegido para este fin es el Naive Bayes con datos de origen sin eliminar *stop words*, sin balanceo, con modelo de representación countVectorizer y con los parámetros por defecto (es

decir aplha=1). Los resultados obtenidos al entrenar un modelo con estas características son:

- F-1 promedio = 31.07%
- Recall promedio = 35.45%
- Precisión promedio = 64.17%
- Accuracy= 66.26%

5.4. Naive Bayes

5.4.1. Hiper Parámetros

Alpha es el parámetro de suavizado aditivo, también conocido como suavizado de Laplace. Este parámetro ayuda a manejar el problema de la probabilidad cero en el cálculo de las probabilidades condicionales. En el contexto del análisis de sentimientos, es posible que ciertas palabras aparezcan en una clase de sentimiento, pero no en otra. Sin suavizado, la probabilidad condicional de estas palabras sería cero, lo que podría llevar a predicciones incorrectas. Al agregar el suavizado aditivo, se asegura que ninguna palabra tenga una probabilidad cero, lo que puede mejorar la eficacia del modelo [3].

Fit_prior es un parámetro booleano que determina si se deben aprender las probabilidades previas de las clases o si se deben utilizar probabilidades previas uniformes. En el contexto del análisis de sentimientos, las probabilidades previas pueden proporcionar información valiosa sobre la distribución general de los sentimientos en los datos. Sin embargo, si las probabilidades previas están muy desbalanceadas, pueden sesgar las predicciones del modelo hacia la clase mayoritaria. Al ajustar este parámetro, podemos explorar el equilibrio entre incorporar el conocimiento previo y evitar el sesgo [85].

Por lo tanto, la variación de estos parámetros en el Grid Search puede ayudar a encontrar el equilibrio adecuado entre el manejo de las probabilidades cero, la incorporación de conocimientos previos y la prevención del sesgo, lo cual es crucial para la eficacia del análisis de sentimientos.

Selección rango hiper parámetros

Con el fin de elegir el rango adecuado de hiper parámetros para la búsqueda dentro del *GridSearch* se comparan resultados entre el un rango de parámetros inicial(normal) y uno ampliado:

- Rango Normal:
  - 'alpha': [0.1, 0.5, 1.0, 1.5, 2.0, 5.0],
  - 'fit_prior': [True, False]
- Rango Ampliado
  - 'alpha': [0.001, 0.1, 0.2, 0.3 ,0.5, 0.7 ,1.0, 1.5, 2.0, 3.0, 7.5 ,5.0, 10, 15],
  - 'fit_prior': [True, False]

Para evaluar si la diferencia es significativa entre los resultados obtenidos con el rango de hiper parámetros normal y ampliado se realizó una prueba no paramétrica de signos de Wilcoxon sobre

los valores F1-Score obtenidos ante diferentes fuentes (con stop words, sin stop words, con lematización, etc), tipo de vectorización (count y TF-IDF) y tipo de optimización (GridSearch y simple). Esta prueba es una alternativa no paramétrica a la prueba t de Student para muestras pareadas y se utiliza cuando no se puede asumir que los datos siguen una distribución normal [7]. La prueba de Wilcoxon compara las medianas de dos conjuntos de datos pareados para determinar si difieren significativamente.

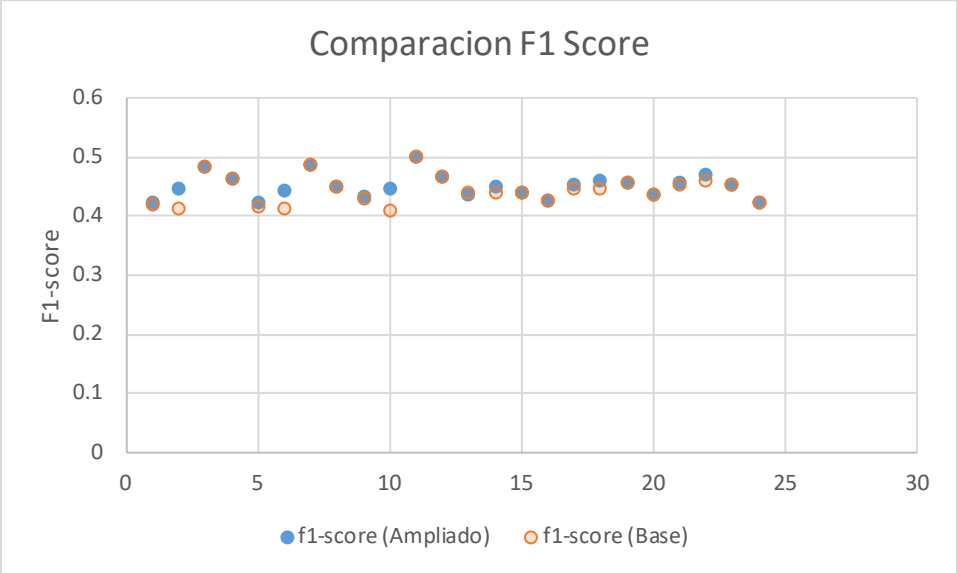


Ilustración 5: Comparación parámetros base Vs ampliado Naive Bayes

Para los datos observados en la ilustración 4 y teniendo las hipótesis:

- H0: Las medianas del F1-promedio son iguales
- H1: Las medianas del F1-promedio NO son iguales

Los resultados de la prueba fueron los siguientes: W-statistic: 93.0, P-value: 17%. La estadística W es el valor de la suma de los rangos de las diferencias, mientras que el valor p es la probabilidad de obtener un resultado al menos tan extremo como el observado, bajo la hipótesis nula de que las medianas de los dos conjuntos de datos son iguales.

Dado que el valor p es mayor que el umbral comúnmente aceptado de 5%, no podemos rechazar la hipótesis nula. Esto sugiere que no hay una diferencia significativa entre las medianas de los resultados obtenidos con el rango de hiper parámetros normal y ampliado, por ello se concluye que no vale la pena ampliar o detallar aún más el rango de hiper parámetros.

Sin embargo, a pesar de estos resultados, se decidió continuar con la evaluación de los modelos con el rango de hiper parámetros ampliado. Esta decisión se tomó para tener una mayor garantía de obtener la mejor clasificación del sentimiento del texto, ya que un rango de hiper parámetros más amplio puede permitir explorar una mayor variedad de configuraciones de modelo [53].

5.4.2. Selección y evaluación de modelos

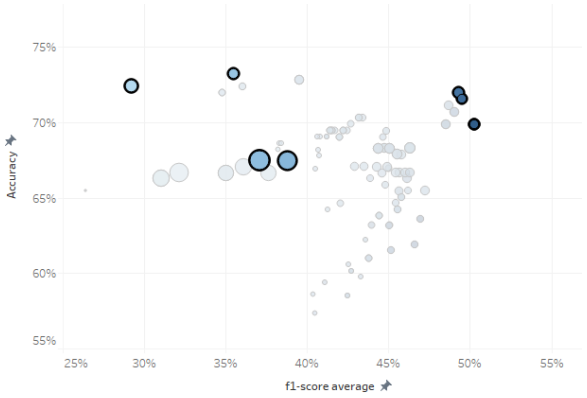
A continuación, se observa un tablero donde en la parte superior se tiene una gráfica de dispersión (F1 VS accuracy), con el fin de identificar trade offs entre estas dos medidas, para la cual cada punto representa el resultado de un modelo corrido, en el eje horizontal está el valor de F1 score promedio, en el eje vertical el accuracy, el tamaño del punto representa la precisión promedio y el color el recall. En la tabla inferior (detalle) se observan los valores detallados del conjunto de modelos seleccionados, donde para cada medida se observa una escala independiente de color según la cual los valores más altos serán los verdes, los más bajos estarán en color rojo y valores intermedios se observan en amarillo. Para la columna fuente la notación representa lo siguiente:

- X_sw: Sin eliminar "stop words"
- X_sw_lema: Sin eliminar "stop words", con lematización
- X_sw_steam: Sin eliminar "stop words", con "stemming"
- X_no_sw: Eliminando "stop words"
- X_no_sw_lema: Eliminando "stop words", con lematización
- X_no_sw_steam: Eliminando "stop words", con "stemming"

Model

Naive Bayes

F1-VS accuracy



detalle

index	best_model	balance_type	fuente	reduccion dimencionalidad	Rango parametros	tipo_score	Optimizacion hiper parametros	vectorizer	f1-score	accuracy	recall	precision
104	MultinomialNB()	oversampling	X_no_sw_steam	none	ampliado	f1-score	none	count_vectorized	50.2%	69.9%	54.6%	52.1%
140	MultinomialNB()	oversampling	X_no_sw_steam	20%	ampliado	f1-score	none	count_vectorized	49.5%	71.5%	52.4%	52.3%
142	MultinomialNB()	oversampling	X_no_sw_lema	20%	ampliado	f1-score	none	count_vectorized	49.3%	72.0%	50.8%	53.4%
159	MultinomialNB(alpha=0.5)	Sin Balanceo	X_sw_steam	none	base	f1-score	GridSearch	count_vectorized	38.8%	67.5%	39.6%	65.4%
171	MultinomialNB(alpha=0.1)	Sin Balanceo	X_sw_steam	none	base	f1-score	GridSearch	tfidf_vectorized	37.1%	67.5%	38.7%	67.4%
163	MultinomialNB(alpha=1.5)	Sin Balanceo	X_no_sw	none	base	f1-score	GridSearch	count_vectorized	35.5%	73.2%	37.4%	53.1%
174	MultinomialNB()	Sin Balanceo	X_no_sw	none	base	f1-score	none	tfidf_vectorized	29.2%	72.4%	34.0%	57.4%

Ilustración 6: Resultados Naive Bayes

Según lo observado en la Ilustración 6: Resultados Naive Bayes, se toman tres conjuntos de datos, los tres modelos con mejor relación f1-score y Accuracy (arriba y a la derecha), dos modelos que son los de mejor precisión (mayor tamaño del círculo) y los dos modelos con Accuracy más alto (a costa de tener bajo f1-score). Estos modelos serán evaluados a detalle a continuación.

En la parte inferior se observan las ilustraciones 6 y 7 detallando los mejores modelos resultantes. En la ilustración 3 se observa el reporte del modelo y en la 4 la matriz de confusión. Cada modelo se identifica por un ID.

best_model	vectorizer	fuente	balance_ty..	reduction_t..	id	sentimiento	Indicador f1-score	precision	recall
MultinomialNB()	count_vectorized	X_no_sw_lema	up_sampling	20%	142	-1	84.44%	80.85%	88.37%
						0	38.46%	60.00%	28.30%
						1	25.00%	19.23%	35.71%
		X_no_sw_steam	up_sampling	20%	140	-1	84.59%	81.62%	87.79%
						0	35.44%	53.85%	26.42%
						1	28.57%	21.43%	42.86%
				none	104	-1	83.00%	82.29%	83.72%
						0	38.55%	53.33%	30.19%
						1	29.17%	20.59%	50.00%
	tfidf_vectorized	X_no_sw	default	none	174	-1	83.90%	72.27%	100.00%
						0	3.70%	100.00%	1.89%
						1	0.00%	0.00%	0.00%
MultinomialNB(alpha=0.1)	tfidf_vectorized	X_sw_steam	default	none	171	-1	81.12%	68.83%	98.76%
						0	11.94%	33.33%	7.27%
						1	18.18%	100.00%	10.00%
MultinomialNB(alpha=0.5)	count_vectorized	X_sw_steam	default	none	159	-1	81.44%	69.60%	98.14%
						0	11.43%	26.67%	7.27%
						1	23.53%	100.00%	13.33%
MultinomialNB(alpha=1.5)	count_vectorized	X_no_sw	default	none	163	-1	85.07%	74.35%	99.42%
						0	10.34%	60.00%	5.66%
						1	11.11%	25.00%	7.14%

Ilustración 7: Reporte detalle Naive Bayes

							Prediccion					
							Cantidad			porc		
best_model (resumen_confusion.csv)	vectorizer	fuente (resume..	balance_ty..	reduction_t..	id	Real	-1	0	1	-1	0	1
MultinomialNB()	count_vect..	X_no_sw_lema	up_sampling	20%	142	-1	152,0	9,0	11,0	63.60%	3.77%	4.60%
						0	28,0	15,0	10,0	11.72%	6.28%	4.18%
						1	8,0	1,0	5,0	3.35%	0.42%	2.09%
		X_no_sw_steam	up_sampling	20%	140	-1	151,0	11,0	10,0	63.18%	4.60%	4.18%
						0	27,0	14,0	12,0	11.30%	5.86%	5.02%
						1	7,0	1,0	6,0	2.93%	0.42%	2.51%
				none	104	-1	144,0	13,0	15,0	60.25%	5.44%	6.28%
						0	25,0	16,0	12,0	10.46%	6.69%	5.02%
						1	6,0	1,0	7,0	2.51%	0.42%	2.93%
	tfidf_vecto..	X_no_sw	default	none	174	-1	172,0	0,0	0,0	71.97%	0.00%	0.00%
						0	52,0	1,0	0,0	21.76%	0.42%	0.00%
						1	14,0	0,0	0,0	5.86%	0.00%	0.00%
MultinomialNB(alpha=0.1)	tfidf_vecto..	X_sw_steam	default	none	171	-1	159,0	2,0	0,0	64.63%	0.81%	0.00%
						0	51,0	4,0	0,0	20.73%	1.63%	0.00%
						1	21,0	6,0	3,0	8.54%	2.44%	1.22%
MultinomialNB(alpha=0.5)	count_vect..	X_sw_steam	default	none	159	-1	158,0	3,0	0,0	64.23%	1.22%	0.00%
						0	51,0	4,0	0,0	20.73%	1.63%	0.00%
						1	18,0	8,0	4,0	7.32%	3.25%	1.63%
MultinomialNB(alpha=1.5)	count_vect..	X_no_sw	default	none	163	-1	171,0	1,0	0,0	71.55%	0.42%	0.00%
						0	47,0	3,0	3,0	19.67%	1.26%	1.26%
						1	12,0	1,0	1,0	5.02%	0.42%	0.42%

Ilustración 8: matriz confusión Naive Bayes

El primer modelo por revisar es el 174, el cual se descarta por el hecho de no clasificar ni un solo texto como positivo. Los siguientes modelos por revisar son los 171 y 159, los cuales destacan por tener precisión del 100% en los *tweets* positivos, lo que indica que estos modelos lo que califiquen como positivos con alta probabilidad serán positivos realmente, sin embargo, su recall para positivos es muy bajo lo que indica que dejan muchos comentarios positivos por fuera, por esta razón se descartan.

De otro lado, se puede notar que el modelo con ID 142, que utiliza el vectorizador CountVectorizer, no elimina las "stop words" y aplica lematización, y utiliza un balanceo de datos mediante "oversampling" con una reducción atributos del 20%, presenta un rendimiento aceptable. Este modelo logra un F1-score de 84.44% para los tweets negativos, 38.46% para los neutros y 25.00% para los positivos (F1-score promedio de 49.3). Aunque el rendimiento para los tweets neutros y positivos es más bajo, este modelo logra cierto equilibrio entre las tres clases.

El modelo con ID 140, que también utiliza el vectorizador CountVectorizer, no elimina las "stop words" y aplica "stemming", y utiliza un balanceo de datos mediante "up sampling" con una reducción del 20%, también muestra un rendimiento razonable, con un F1-score promedio de 49.59%.

Finalmente se observa el modelo con ID 104, para el cual se tiene el mejor F1-promedio de 50.2% es el que tiene el mejor recall sobre los comentarios positivos, elemento importante a tener en cuenta pues según esto es el que menos comentarios de este tipo deja escapar.

Al considerar tanto el rendimiento de la clasificación como la matriz de confusión, el modelo con ID 104 parece ser el más equilibrado, ya que logra un rendimiento superior en todas las clases. Por lo tanto, este modelo se selecciona como el modelo finalista para Naive Bayes.

En términos de comparación con el modelo de línea base se observa una mejora de 19% en términos del indicador principal, el F1-score promedio.

Es importante destacar que la selección del modelo finalista no sólo se basa en las métricas de rendimiento, sino también en su capacidad para equilibrar el rendimiento entre las diferentes clases, lo cual es crucial en el análisis de sentimientos, donde todas las clases (negativa, neutra y positiva) son importantes [105]

5.5. Random forest

5.4.2 Hiper Parámetros

n_estimators es el número de árboles en el bosque. Este parámetro es importante porque un número mayor de árboles puede mejorar la precisión del modelo al reducir la varianza, pero también puede aumentar el tiempo de computación y el riesgo de sobreajuste. En el contexto del análisis de sentimientos, donde los datos pueden ser ruidosos y complejos, puede ser beneficioso tener un número mayor de árboles para capturar diferentes aspectos de los datos [146].

max_depth es la profundidad máxima de los árboles. Este parámetro controla el grado de complejidad de los árboles y, por lo tanto, la capacidad del modelo para capturar interacciones complejas en los datos. En el análisis de sentimientos, donde las palabras pueden tener diferentes significados dependiendo del contexto, puede ser útil permitir una mayor profundidad para capturar estas interacciones [77].

min_samples_split es el número mínimo de muestras requeridas para dividir un nodo interno. Este parámetro puede ayudar a controlar el sobreajuste al evitar que el modelo divida nodos que contienen muy pocas muestras. En el análisis de sentimientos, donde los datos pueden estar desbalanceados, este parámetro puede ser especialmente importante para garantizar que el modelo no se ajuste demasiado a las clases minoritarias [148].

Por lo tanto, la variación de estos parámetros en el GridSearch puede ayudar a encontrar el equilibrio adecuado entre la precisión del modelo, el tiempo de computación, y la prevención del sobreajuste, lo cual es crucial para la eficacia del análisis de sentimientos.

Selección rango hiper parámetros

Para la técnica Random forest el rango de parámetros normal y ampliado fue:

- Rango normal
    - 'n_estimators': [100, 200, 300],
    - 'max_depth': [None, 10, 20, 30],
    - 'min_samples_split': [2, 5, 10]

- Rango Ampliado
    - 'n_estimators': [75 ,100, 150 ,200, 250, 300, 325],
    - 'max_depth': [None, 5 ,10, 15 ,20, 25 ,30, 35],
    - 'min_samples_split': [1 ,2, 3 ,5, 7 ,10, 15]

Siguiendo un procedimiento similar al realizado para Naive Bayes, se obtienen el conjunto de datos a comparar observados en la Ilustración 8

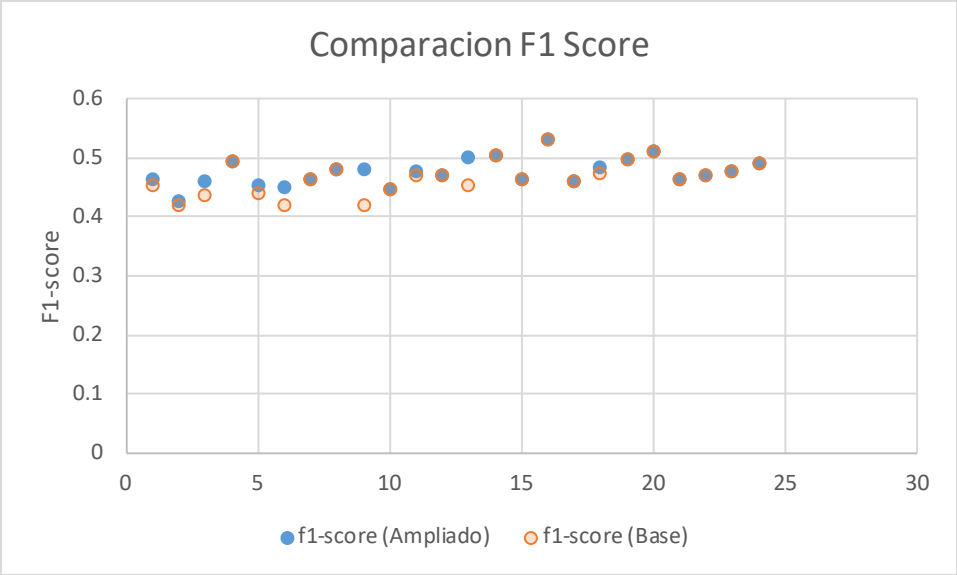


Ilustración 9: Comparación parámetros base Vs ampliado Random Forest

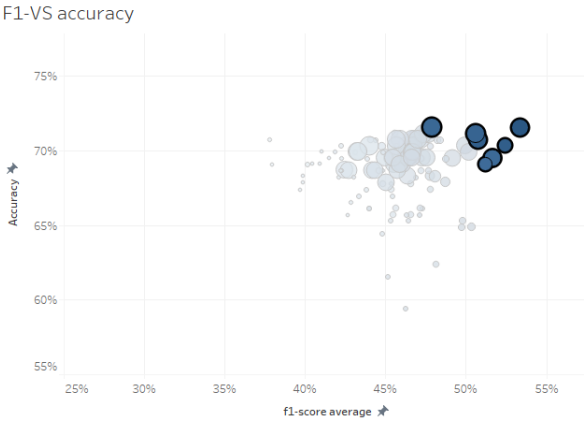
Según estos valores los resultados de la prueba de Wilcoxon fueron los siguientes: W-statistic: 127.0, P-value: 0.527. Al igual que en el caso de Naive Bayes, el valor p es mayor que el umbral de 0.05, lo que sugiere que no hay una diferencia significativa entre las medianas de los resultados obtenidos con el rango de hiperparámetros normal y ampliado.

Por lo tanto, aunque los resultados indican que no es necesario ampliar o detallar más el rango de hiperparámetros para Random Forest, se decidió continuar con el rango de hiperparámetros ampliado para asegurar una mayor precisión en los resultados de la clasificación del sentimiento del texto [53].

5.4.3 Selección y evaluación de modelos

Model

Random Forest



detalle

index	best_model	balance_type	fuente	reduccion dimencionalidad	Rango parametros	tipo_score	Optimizacion hiper parametros	vectorizer	f1-score	accuracy	recall	precision
288	RandomForestClassifier()	oversampling	X_sw	none	ampliado	f1-score	none	tfidf_vectorized	53.3%	71.5%	51.3%	75.6%
36	RandomForestClassifier()	oversampling	X_sw	20%	ampliado	f1-score	none	tfidf_vectorized	52.4%	70.3%	50.7%	69.3%
41	RandomForestClassifier(min_s..	oversampling	X_sw_lema	20%	ampliado	f1-score	GridSearch	tfidf_vectorized	51.7%	69.5%	49.5%	74.1%
292	RandomForestClassifier()	oversampling	X_sw_lema	none	ampliado	f1-score	none	tfidf_vectorized	51.2%	69.1%	49.3%	68.3%
39	RandomForestClassifier(n_esti..	oversampling	X_sw_steam	20%	ampliado	f1-score	GridSearch	tfidf_vectorized	50.8%	70.7%	50.7%	75.7%
289	RandomForestClassifier(min_s..	oversampling	X_sw	none	ampliado	f1-score	GridSearch	tfidf_vectorized	50.6%	71.1%	50.1%	76.0%
278	RandomForestClassifier()	oversampling	X_sw_steam	none	ampliado	f1-score	none	count_vectorized	47.9%	71.5%	49.7%	76.5%

Ilustración 10: Resultados Random Forest

Los tres mejores modelos evaluados están dados por el index 288, 36 y 41. El modelo que presentó mejor f1-Score (promedio entre precisión y Recall) fue la combinación entre up_sampling (oversamplig), eliminando stopword o palabras no significativas, sin ninguna reducción y sin ninguna búsqueda exhaustiva de hiperparámetros, vectorizado con TfidfVectorizer permitiendo llegar a un 53.3% de f1-Score y una accuracy de 71,5%.

El segundo puntaje de f1-Score (52,4%) solo tuvo una modificación en la reducción de un 20% con relación al anterior. El tercer puesto con 51, 7% si presentó cambios significativos en el modelo, generando una clasificación con stopword lemantizados, reducción del 20%, y con una exploración exhaustiva de hiperparámetros (GridSearch).

best_model	vectorizer	fuente	balance_ty..	reduction_t..	id	sentimiento	Indicador f1-score	precision	recall
RandomForestClassifier()	count_vectorized	X_sw_steam	up_sampling	none	278	-1	82.52%	76.60%	89.44%
						0	53.57%	52.63%	54.55%
						1	6.45%	100.00%	3.33%
	tfidf_vectorized	X_sw	up_sampling	20%	36	-1	81.36%	74.61%	89.44%
						0	50.94%	52.94%	49.09%
						1	12.50%	100.00%	6.67%
				none	288	-1	82.42%	76.88%	88.82%
						0	52.25%	51.79%	52.73%
						1	23.53%	100.00%	13.33%
		X_sw_lema	up_sampling	none	292	-1	80.00%	75.98%	84.47%
						0	44.83%	42.62%	47.27%
						1	22.22%	66.67%	13.33%
RandomForestClassifier(min_samples_split=3, n_estimators=200)	tfidf_vectorized	X_sw	up_sampling	none	289	-1	83.00%	77.42%	89.44%
						0	53.10%	51.72%	54.55%
						1	12.50%	100.00%	6.67%
RandomForestClassifier(min_samples_split=3, n_estimators=250)	tfidf_vectorized	X_sw_lema	up_sampling	20%	41	-1	80.91%	74.74%	88.20%
						0	49.52%	52.00%	47.27%
						1	27.78%	83.33%	16.67%
RandomForestClassifier(n_estimators=300)	tfidf_vectorized	X_sw_steam	up_sampling	20%	39	-1	81.14%	75.13%	88.20%
						0	47.27%	47.27%	47.27%
						1	12.50%	100.00%	6.67%

Ilustración 11: Reporte detalle Rendom Forest

best_model (resumen_confusion.csv)	vectorizer	fuente (res..	balance_ty..	reduction_t..	id	Real	Prediccion					
							Cantidad			porc		
							-1	0	1	-1	0	1
RandomForestClassifier()	count_vectoriz..	X_sw_steam	up_sampling	none	278	-1	144,0	17,0	0,0	58.54%	6.91%	0.00%
						0	25,0	30,0	0,0	10.16%	12.20%	0.00%
						1	19,0	10,0	1,0	7.72%	4.07%	0.41%
	tfidf_vectorized	X_sw	up_sampling	20%	36	-1	144,0	17,0	0,0	58.54%	6.91%	0.00%
						0	28,0	27,0	0,0	11.38%	10.98%	0.00%
						1	21,0	7,0	2,0	8.54%	2.85%	0.81%
				none	288	-1	143,0	18,0	0,0	58.13%	7.32%	0.00%
						0	26,0	29,0	0,0	10.57%	11.79%	0.00%
						1	17,0	9,0	4,0	6.91%	3.66%	1.63%
		X_sw_lema	up_sampling	none	292	-1	136,0	24,0	1,0	55.28%	9.76%	0.41%
						0	28,0	26,0	1,0	11.38%	10.57%	0.41%
						1	15,0	11,0	4,0	6.10%	4.47%	1.63%
RandomForestClassifier(min_samples_split=3, n_estimators=200)	tfidf_vectorized	X_sw	up_sampling	none	289	-1	144,0	17,0	0,0	58.54%	6.91%	0.00%
						0	25,0	30,0	0,0	10.16%	12.20%	0.00%
						1	17,0	11,0	2,0	6.91%	4.47%	0.81%
RandomForestClassifier(min_samples_split=3, n_estimators=250)	tfidf_vectorized	X_sw_lema	up_sampling	20%	41	-1	142,0	18,0	1,0	57.72%	7.32%	0.41%
						0	29,0	26,0	0,0	11.79%	10.57%	0.00%
						1	19,0	6,0	5,0	7.72%	2.44%	2.03%
RandomForestClassifier(n_estimators=300)	tfidf_vectorized	X_sw_steam	up_sampling	20%	39	-1	142,0	19,0	0,0	57.72%	7.72%	0.00%
						0	29,0	26,0	0,0	11.79%	10.57%	0.00%
						1	18,0	10,0	2,0	7.32%	4.07%	0.81%

Ilustración 12: matriz confusión Random Forest

Entrando a detalle vemos que en la clasificación el modelo con id 288 presenta el mejor resultado en términos de F1 promedio, con valor de 53.3%, lo que representa una mejora de 22.23% respecto a la línea base y a pesar de que no cuenta con un recall muy alto para positivos, entre este conjunto de modelos tiene uno de los mayores puntajes para esta métrica, por ello se selecciona ese como el mejor de la técnica random forest.

5.6. Regresión logística

5.6.1. Hiper parámetros

C es el parámetro de regularización inversa en la Regresión Logística. Este parámetro controla la cantidad de regularización aplicada al modelo, con valores más bajos que resultan en una mayor regularización. La regularización puede ayudar a prevenir el sobreajuste al penalizar los coeficientes de modelo grandes, lo que puede ser especialmente útil en el análisis de sentimientos, donde los datos pueden ser de alta dimensión debido a la gran cantidad de palabras únicas [149].

Penalty especifica el tipo de regularización a aplicar: L1 (Lasso) o L2 (Ridge). L1 tiende a producir soluciones dispersas, lo que significa que puede reducir algunos coeficientes a cero y, por lo tanto, seleccionar características. L2, por otro lado, no reduce los coeficientes a cero, pero puede reducir su magnitud. Ambas formas de regularización pueden ser útiles en el análisis de sentimientos, donde algunos términos pueden ser irrelevantes para la clasificación del sentimiento [117].

Solver especifica el algoritmo a utilizar para la optimización. Diferentes solvers pueden tener diferentes ventajas en términos de velocidad de convergencia y robustez, y algunos solvers sólo son compatibles con ciertos tipos de regularización. Por lo tanto, es importante probar diferentes solvers para encontrar el que funcione mejor con nuestros datos [99].

Multi_class especifica la estrategia a utilizar para el problema de clasificación multiclase. En nuestro caso, tenemos tres clases de sentimientos (negativo, neutro y positivo), por lo que necesitamos una estrategia multiclase. Las opciones son 'ovr' (one-vs-rest), donde se entrena un clasificador binario para cada clase, y 'multinomial', donde se entrena un solo clasificador multiclase. Ambas estrategias pueden ser útiles en diferentes contextos y vale la pena explorarlas en nuestros datos [109].

Por lo tanto, la variación de estos parámetros en el Grid Search puede ayudar a encontrar el equilibrio adecuado entre la precisión del modelo, la prevención del sobreajuste, la selección de características y la eficiencia computacional, lo cual es crucial para la eficacia del análisis de sentimientos.

Selección rango hiper parámetros

Para la técnica Regresión logistica el rango de parámetros normal y ampliado fue:

- Rango normal
    - 'C': [0.1, 1, 5 ,10],
    - 'penalty': ['l1', 'l2', 'elasticnet', None],
    - 'solver' : ['lbfg', 'newton-cg', 'sag', 'saga'],
    - 'multi_class' : ['auto', 'ovr', 'multinomial'],
- Rango Ampliado
    - 'C': [0.05 ,0.1, 1, 3 ,5 , 7 ,10, 15],
    - 'penalty': ['l1', 'l2', 'elasticnet', None],
    - 'solver' : ['lbfg', 'newton-cg', 'sag', 'saga'],
    - 'multi_class' : ['auto', 'ovr', 'multinomial'],

Siguiendo un procedimiento similar al realizado para Naive Bayes, se obtienen el conjunto de datos a comparar observados en la Ilustración 12

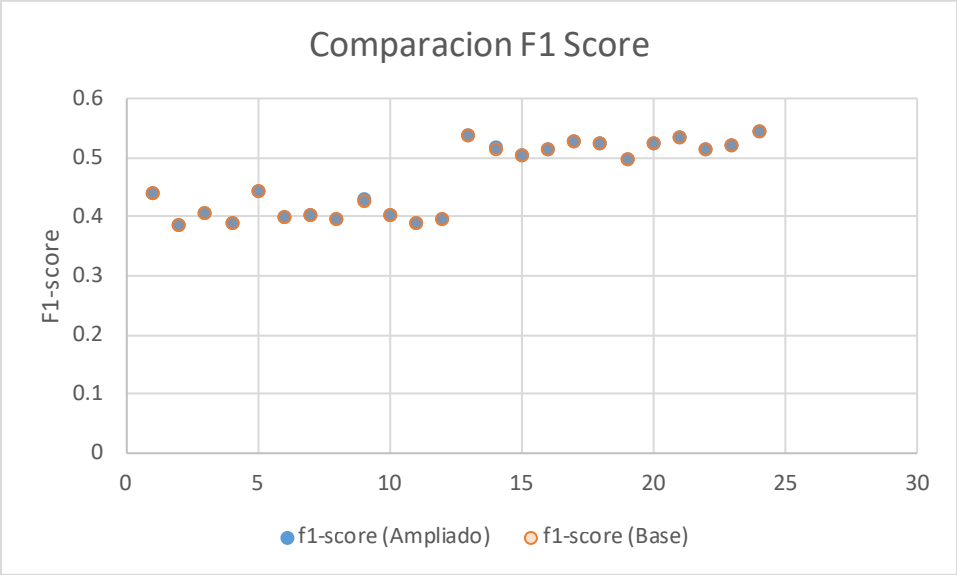


Ilustración 13: Comparación parámetros base Vs ampliado Regresión Logística

En este caso, los resultados de la prueba de Wilcoxon arrojaron un W-statistic de 120.0 y un P-value de 0.832. Este valor p, superior al umbral estándar de 0.05, indica que no hay una diferencia significativa entre las medianas de los resultados obtenidos con el rango de hiper parámetros normal y ampliado.

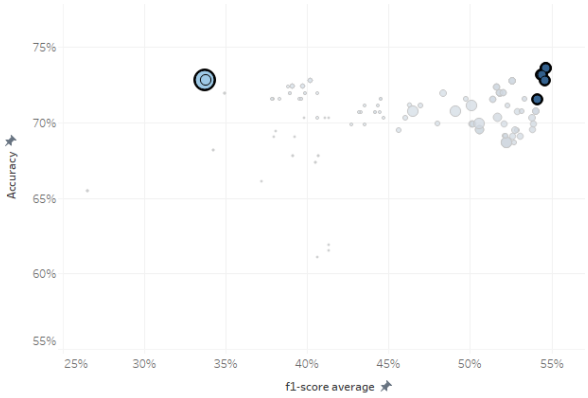
Por lo tanto, dado que los resultados sugieren que no es esencial extender o detallar más el rango de hiper parámetros para la Regresión Logística, se optó por mantener el rango de hiper parámetros ampliado. Esta decisión se basa en el objetivo de garantizar una mayor precisión en los resultados de la clasificación del sentimiento del texto, tal como se hizo en los casos de Naive Bayes y Random Forest [53].

5.6.2. Selección y evaluación de modelos

Model

Regresion Logistica

F1-VS accuracy



detalle

index	best_model	balance_type	fuente	reduccion dimencionalidad	Rango parametros	tipo_score	Optimizacion hiper parametros	vectorizer	f1-score	accuracy	recall	precision
314	LogisticRegression()	oversampling	X_sw_steam	none	ampliado	f1-score	none	tfidf_vectorized	54.7%	73.6%	51.7%	68.4%
240	LogisticRegression()	SMOTE	X_sw	none	ampliado	f1-score	none	tfidf_vectorized	54.6%	72.8%	51.4%	68.0%
75	LogisticRegression(C=5, n_jobs..	Sin Balanceo	X_sw_steam	none	base	f1-score	GridSearch	count_vectorized	54.4%	73.2%	51.9%	66.6%
62	LogisticRegression()	oversampling	X_sw_steam	20%	ampliado	f1-score	none	tfidf_vectorized	54.4%	73.2%	51.5%	67.8%
242	LogisticRegression()	SMOTE	X_sw_steam	none	ampliado	f1-score	none	tfidf_vectorized	54.1%	71.5%	51.1%	67.9%
94	LogisticRegression()	Sin Balanceo	X_no_sw_lema	none	base	f1-score	none	tfidf_vectorized	33.8%	72.8%	36.3%	74.3%
92	LogisticRegression()	Sin Balanceo	X_no_sw_steam	none	base	f1-score	none	tfidf_vectorized	33.7%	72.8%	36.3%	90.9%

Ilustración 14:Resultados Regresión Logistica

best_model	vectorizer	fuente	balance_ty..	reduction_t..	id	sentimiento	Indicador f1-score	precision	recall
LogisticRegression()	tfidf_vectorized	X_no_sw_lema	default	none	94	-1	84.31%	72.88%	100.00%
						0	3.64%	50.00%	1.89%
						1	13.33%	100.00%	7.14%
		X_no_sw_steam	default	none	92	-1	84.11%	72.57%	100.00%
						0	3.70%	100.00%	1.89%
						1	13.33%	100.00%	7.14%
		X_sw	SMOTE	none	240	-1	83.75%	75.25%	94.41%
						0	44.94%	58.82%	36.36%
						1	35.00%	70.00%	23.33%
		X_sw_steam	SMOTE	none	242	-1	83.15%	75.90%	91.93%
						0	43.30%	50.00%	38.18%
						1	35.90%	77.78%	23.33%
			up_sampling	20%	62	-1	83.98%	75.62%	94.41%
						0	48.35%	61.11%	40.00%
						1	30.77%	66.67%	20.00%
				none	314	-1	84.30%	75.74%	95.03%
						0	48.89%	62.86%	40.00%
						1	30.77%	66.67%	20.00%
LogisticRegression(C=5, n_jobs=-1, penalty=None, solver='sag')	count_vectorized	X_sw_steam	default	none	75	-1	84.59%	77.04%	93.79%
						0	47.92%	56.10%	41.82%
						1	30.77%	66.67%	20.00%

Ilustración 15: Reporte detalle Regresión Logística

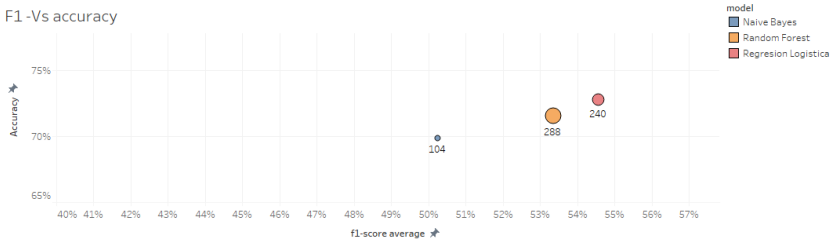
best_model (resumen_confusion.csv)	vectorizer	fuente (resume..	balance_ty..	reduction_t..	id	Real	Prediccion					
							Cantidad			porc		
							-1	0	1	-1	0	1
LogisticRegression()	tfidf_vecto..	X_no_sw_lema	default	none	94	-1	172,0	0,0	0,0	71.97%	0.00%	0.00%
						0	52,0	1,0	0,0	21.76%	0.42%	0.00%
						1	12,0	1,0	1,0	5.02%	0.42%	0.42%
		X_no_sw_steam	default	none	92	-1	172,0	0,0	0,0	71.97%	0.00%	0.00%
						0	52,0	1,0	0,0	21.76%	0.42%	0.00%
						1	13,0	0,0	1,0	5.44%	0.00%	0.42%
		X_sw	SMOTE	none	240	-1	152,0	9,0	0,0	61.79%	3.66%	0.00%
						0	32,0	20,0	3,0	13.01%	8.13%	1.22%
						1	18,0	5,0	7,0	7.32%	2.03%	2.85%
		X_sw_steam	SMOTE	none	242	-1	148,0	13,0	0,0	60.16%	5.28%	0.00%
						0	32,0	21,0	2,0	13.01%	8.54%	0.81%
						1	15,0	8,0	7,0	6.10%	3.25%	2.85%
			up_sampling	20%	62	-1	152,0	7,0	2,0	61.79%	2.85%	0.81%
						0	32,0	22,0	1,0	13.01%	8.94%	0.41%
						1	17,0	7,0	6,0	6.91%	2.85%	2.44%
				none	314	-1	153,0	6,0	2,0	62.20%	2.44%	0.81%
						0	32,0	22,0	1,0	13.01%	8.94%	0.41%
						1	17,0	7,0	6,0	6.91%	2.85%	2.44%
LogisticRegression(C=5, n_jobs=-1, penalty=None, solver='sag')	count_vect..	X_sw_steam	default	none	75	-1	151,0	10,0	0,0	61.38%	4.07%	0.00%
						0	29,0	23,0	3,0	11.79%	9.35%	1.22%
						1	16,0	8,0	6,0	6.50%	3.25%	2.44%

Ilustración 16: matriz de confusión Regresión Logística

Para el caso de la regresión logística se observa que el modelo con mayor F1 score promedio es el 314, sin embargo, está muy cerca del modelo 240 con una diferencia de apenas 0.1%. En este punto se toma como segunda medida el recall de los positivos el para el que el modelo 240 es superior.

Según esto el mejor modelo para regresión logística tiene un F1 score promedio de 54.6%, lo que implica una mejora de 23.53% sobre el modelo de línea base.

5.7. Resultados consolidados



modelos

index	best_model	balance_type	fuente	reduccion dimencionalidad	Rango parametros	tipo_score	Optimizacion hiper parametros	vectorizer	f1-score	accuracy	recall	precision
240	LogisticRegression()	SMOTE	X_sw	none	ampliado	f1-score	none	tfidf_vectorized	54.6%	72.8%	51.4%	68.0%
104	MultinomialNB()	oversampling	X_no_sw_steam	none	ampliado	f1-score	none	count_vectorized	50.2%	69.9%	54.6%	52.1%
288	RandomForestClassifier()	oversampling	X_sw	none	ampliado	f1-score	none	tfidf_vectorized	53.3%	71.5%	51.3%	75.6%

Reporte detalle

best_model	id	sentimiento	Indicador f1-score	Indicador precision	Indicador recall
LogisticRegression()	240	-1	83.75%	75.25%	94.41%
		0	44.94%	58.82%	36.36%
		1	35.00%	70.00%	23.33%
MultinomialNB()	104	-1	83.00%	82.29%	83.72%
		0	38.55%	53.33%	30.19%
		1	29.17%	20.59%	50.00%
RandomForestClassifier()	288	-1	82.42%	76.88%	88.82%
		0	52.25%	51.79%	52.73%
		1	23.53%	100.00%	13.33%

Matriz de confusion

best_model (resumen_confusion.csv)	id	Real	Prediccion Cantidad -1	Cantidad 0	Cantidad 1	porc -1	porc 0	porc 1
LogisticRegression()	240	-1	152,0	9,0	0,0	61.79%	3.66%	0.00%
		0	32,0	20,0	3,0	13.01%	8.13%	1.22%
		1	18,0	5,0	7,0	7.32%	2.03%	2.85%
MultinomialNB()	104	-1	144,0	13,0	15,0	60.25%	5.44%	6.28%
		0	25,0	16,0	12,0	10.46%	6.69%	5.02%
		1	6,0	1,0	7,0	2.51%	0.42%	2.93%
RandomForestClassifier()	288	-1	143,0	18,0	0,0	58.13%	7.32%	0.00%
		0	26,0	29,0	0,0	10.57%	11.79%	0.00%
		1	17,0	9,0	4,0	6.91%	3.66%	1.63%

Ilustración 17: resultados comparativos modelos finalistas[2]

Como se observa en la ilustración 12 en el diagrama de dispersión, los modelos de regresión logística y random forest dominan al modelo naive bayes en términos del F1 promedio y accuracy. Teniendo este ultimo la única ventaja de ser aquel con mayor recall de los sentimientos positivos, sin embargo, a pesar de esta ventaja, se rezaga el modelo naive bayes por tener gran desventaja en las métricas principales (eventualmente podría ser usado como complemento en un modelo de ensamble).

Finalmente al comparar los modelos de regresión logística con random forest se observa una ventaja en random forest respecto a la precisión promedio, sin embargo esa ventaja no compensa la superioridad en términos de la métrica Principal F1-score promedio, por ello el modelo considerado como mejor a la luz de los objetivos de la presente iniciativa es el 240 de regresión logística con fuente de datos sin eliminar stop words, sin realizar steaming o lematización, con balanceo mediante SMOTE y con vectorización de tipo TF-IDF.

5.8. Resultado modelo preconstruido

A pesar de que en las evaluaciones desarrolladas en los capítulos anteriores se obtuvieron mejoras importantes a partir de la línea base, el mejor modelo tiene un nivel moderado en términos de F1 score promedio. Por ello se decide evaluar una opción adicional, incluyendo el uso de una librería preconstruida, en este caso pysentimiento, de donde se obtienen los siguientes resultados:

[2] Como se mencionó en el capítulo 5.1 los tamaños de las matrices de confusión no son iguales debido a que en los orígenes de datos en que se eliminaron “Stop Words” se eliminaron registros.

	precision	recall	f1-score	support
-1	0.90	0.63	0.74	828
0	0.34	0.69	0.45	280
1	0.31	0.21	0.25	120
accuracy			0.60	1228
macro avg	0.51	0.51	0.48	1228
weighted avg	0.71	0.60	0.62	1228

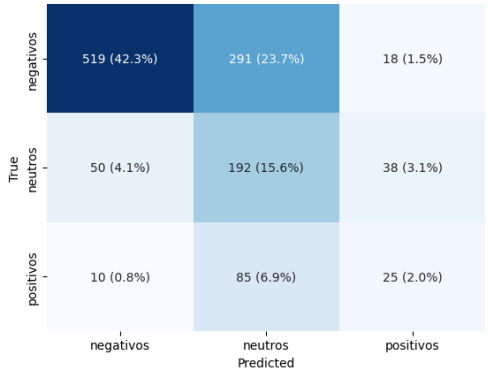


Ilustración 18: Matriz de confusión pysentimiento

Como se puede observar en términos del F1 score promedio hay una disminución de 6.6% respecto al mejor modelo encontrado anteriormente, por ello el modelo preconstruido se considera como de desempeño inferior en la tarea de clasificación de sentimientos de textos de Twitter en temas de movilidad en la ciudad de Bogotá.

6. Evaluación de la percepción de los usuarios sobre medidas implementadas

Una de las preguntas que pretende resolver el presente trabajo es poder determinar el impacto de las iniciativas, sobre la percepción de los usuarios. Para tal fin se planteó la construcción de un modelo de clasificación de textos con la expectativa de lograr clasificar el sentimiento de este con un nivel de F1 score promedio aceptable. A pesar de que los resultados de los modelos evaluados no son los ideales, se logró una mejora considerable desde la línea base y adicionalmente estos resultados superaron a los del modelo preconstruido pysentimiento. A pesar de lo anterior se considera que el nivel de la métricas seleccionadas del modelo elegido no es suficientemente buenas para la evaluación de textos en tres niveles, sin embargo al realizar un ajuste sobre el modelo seleccionado y agrupar los resultados de los textos neutros y Positivos en una categoría “No negativos”, el desempeño mejora considerablemente como se puede observar en la Ilustración 19: Transformación de modelo, llegando a un F1 score de 83.8%.

Matriz confusión mejor modelo encontrado, clasificación 3 categorías

	Neg	Neu	Pus
Neg	152	9	0
Neu	32	20	3
Pos	18	5	7

F1 promedio: 54.6%
Precisión promedio: 68%
Recall promedio: 51.4%
Accuracy: 69.9%

Matriz confusión mejor modelo encontrado, clasificación Negativos Vs no negativos

	Neg	No neg
Neg	152	9
No neg	50	35

F1 : 83.8%
Precisión : 75.3%
Recall: 94.41%
Accuracy: 76.02%

Ilustración 19: Transformación de modelo

Utilizando este modelo se procede a la comparación de la proporción de *tweets* negativos entre fechas asociadas a eventos e iniciativas y fechas en las que no. Para tal fin se tiene como referencia el conjunto fechas referenciadas en la Ilustración 20: fechas eventos e iniciativas. Los eventos resaltados en verde son aquellos que se incluyen en la comparación (eventos de más allá de 2022-08-01 no se incluyen, pues la base del corpus total tiene esta fecha límite)

id	Fecha inicio	Fecha fin	EVENTO
1	6/01/2021		Bogotá celebrará el Día Mundial de la Bicicleta
2	25/01/2021		Empresas privadas se comprometen con Bogotá y la movilidad sostenible
3	26/02/2021		En 2020, a buen ritmo avanzó la apuesta de Bogotá por la Nueva Movilidad
4	15/06/2021		Gremio taxista en Bogotá se capacita para mejorar el servicio a las mujeres
5	12/07/2021	16/07/2021	La Semana de la Seguridad Vial es un espacio que convoca a expertos y líderes mundiales en materia de seguridad vial. La Secretaría Distrital de Movilidad de Bogotá ha liderado este evento para visibilizar la seguridad vial como un asunto de salud pública.
6	1/09/2021		Cambio trámite de derechos de tránsito relacionado con PMT por eventos, se debe realizar el registro de los PMT y pago de éstos, a través de la URL
7	10/09/2021		Distrito reconoce a 200 conductores del SITP por su compromiso con una movilidad segura en Bogotá
8	23/09/2021		Se pone en servicio la nueva intersección semafórica sobre la calle 26 con carrera 98
9	30/09/2021		La XVI Semana de la Seguridad Vial de Bogotá prende motores con 'MotoPicnic'
10	6/10/2021		En mayo, Bogotá registró la cifra más baja de fatalidades de motociclistas de los últimos 3 años
11	30/01/2022	3/02/2022	Semana de la movilidad sostenible
12	22/03/2022		Autonomy mobility world Expo
13	23/06/2022		Mobilize learning lab Bogotá
14	5/08/2022	9/08/2022	Semana del taxista
15	22/09/2022		Dia sin carro y sin moto
16	23/09/2022	27/09/2022	Semana de la Bici
17	5/10/2022		Dia del Motociclista
18	14/10/2022		entregó 80 nuevas motos para fortalecer el Cuerpo de Agentes Civiles de Tránsito y mejorar la movilidad en la ciudad
19	5/11/2022		Día mundial de las víctimas de siniestros viales

Ilustración 20: fechas eventos e iniciativas

Luego de utilizar el modelo seleccionado para realizar el filtro para cada uno de los eventos e iniciativas basado en las fechas, se obtiene el número de registros descritos en la Ilustración 21: Número de registros por evento. Para cada uno de los registros de cada uno de los eventos se realiza la corrida del modelo seleccionado para clasificar el sentimiento del texto entre “negativo” o “no negativo” y luego se realiza una comparación por pares de proporciones sobre los comentarios negativos, para este fin se toma la notación en donde se tendrá cada evento en las filas(A) para ser contrastado contra los demás eventos en las columnas(B). Si pA es la proporción de comentarios negativos en el grupo A y PB la proporción de eventos negativos en el evento B, se plantea: H0: pA =pB, Ha: pA ≠pB , en la ilustración 22 se puede observar los resultados de todas las comparaciones entre eventos y “no evento” (menciones de fechas sin eventos asociados luego de 2021-01-01) en términos de si se puede rechazar o no la hipótesis nula con nivel de significancia de 5%. Adicionalmente en la ilustración 23 se observa la relación entre las proporciones para los casos en donde se evidencio diferencia significativa.

A	
no_evento	245.668
01	596
02	335
03	488
04	370
05	2.582
06	691
07	455
08	562
09	560
10	632
11	1.660
12	517

Ilustración 21: Número de registros por evento

A	B 01	02	03	04	05	06	07	08	09	10	11	12	13
no_evento	con dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	con dif significativa	Sin dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa	Sin dif significativa	con dif significativa
01		con dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa
02			Sin dif significativa	con dif significativa	con dif significativa	Sin dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa	Sin dif significativa	con dif significativa
03				Sin dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	con dif significativa	Sin dif significativa
04					Sin dif significativa	con dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	con dif significativa	Sin dif significativa
05						con dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	con dif significativa	Sin dif significativa
06							con dif significativa	con dif significativa	con dif significativa	con dif significativa	con dif significativa	Sin dif significativa	con dif significativa
07								Sin dif significativa	Sin dif significativa	Sin dif significativa	Sin dif significativa	con dif significativa	Sin dif significativa
08									Sin dif significativa	Sin dif significativa	Sin dif significativa	con dif significativa	Sin dif significativa
09										Sin dif significativa	Sin dif significativa	con dif significativa	Sin dif significativa
10											Sin dif significativa	con dif significativa	Sin dif significativa
11												con dif significativa	Sin dif significativa
12													con dif significativa

Ilustración 22: significancia comparación eventos

A	B 01	02	03	04	05	06	07	08	09	10	11	12	13
no_evento	PA>PB				PA<PB		PA<PB	PA<PB	PA<PB	PA<PB	PA<PB		PA<PB
01		PA<PB	PA<PB	PA<PB	PA<PB	PA<PB	PA<PB	PA<PB	PA<PB	PA<PB	PA<PB	PA<PB	PA<PB
02				PA<PB	PA<PB		PA<PB	PA<PB	PA<PB	PA<PB	PA<PB		PA<PB
03												PA>PB	
04						PA>PB						PA>PB	
05						PA>PB						PA>PB	
06							PA<PB	PA<PB	PA<PB	PA<PB	PA<PB		PA<PB
07												PA>PB	
08												PA>PB	
09												PA>PB	
10												PA>PB	
11												PA>PB	
12													PA<PB

Ilustración 23: resultados resumidos comparación eventos

De las ilustraciones 22 y 23 se pueden obtener diferentes conclusiones

- Al comparar “no realizar” evento contra la mayoría de las iniciativas se encuentra una diferencia significativa entre la proporción de comentarios negativos, encontrando que en la gran mayoría de casos significantes (7/8) se observa una menor proporción de comentarios negativos en las muestras de fechas no asociadas a eventos. Lo anterior a pesar de ser contra intuitivo (la expectativa inicial era que la implementación de medidas mejorara la percepción de los usuarios) puede implicar que en general la red social es más llamativa para comentarios negativos y más que concluir que los iniciativas desarrolladas empeoren la percepción de los usuarios, se puede pensar que los comentarios de Twitter no necesariamente son una buena medida del nivel de percepción de los usuarios, porque de hecho es una muestra sesgada a los usuarios que suelen usar Twitter.
- El evento 12 en general parece generar menos comentarios negativos que el promedio de eventos de forma significativa.
- El evento 6 generó menor proporción de comentarios negativos que los eventos 4 y 5

7. Desarrollo de la interfaz de usuario

Con el fin de brindar la capacidad de revisar los resultados del modelo entrenado se desarrolla interfaz de usuario que permite la evaluación de tweets en nuevos rangos de tiempo y con la posibilidad de incluir nuevas menciones a explorar. Para tal fin se desarrolló una aplicación mediante el uso de la librería de Python dash y la implementación de call backs con el fin de generar interactividad. La aplicación cuenta con acceso a 4 ventanas interactivas y a un tablero embebido desarrollado en Power Bi. A continuación, se presenta la descripción de cada una de las ventas de acceso para el usuario.

7.1. Cargue información de búsqueda

En esta ventana se da la posibilidad al usuario de cargar listas de hashtags o cuentas para rastrear (el usuario cuenta con la posibilidad de agregar o quitar hashtags), de forma que en ventanas posteriores se puedan elegir sobre cuáles de estas se desea descargar las menciones de twitter.

Ilustración 24: Interface para cargue de referencias

7.2. Filtros de búsqueda

En esta ventana se habilita al usuario para limitar la búsqueda en términos de las menciones y el rango temporal, la aplicación tiene la capacidad de informar al usuario sobre cuantas referencias están activas para la búsqueda y sobre el número de días incluidos en el rango de fechas

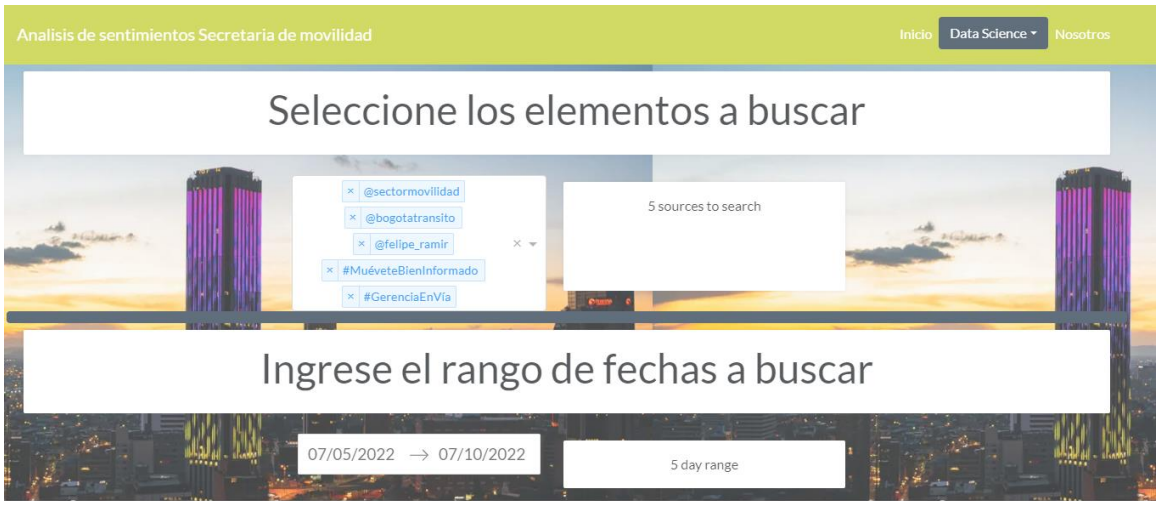


Ilustración 25: Interface selección rango de fechas

7.3. Procesamiento análisis de sentimientos

En esta ventana el usuario visualiza el número de tweets descargados en el paso anterior y tiene la posibilidad de iniciar el proceso de análisis de sentimientos con el modelo entrenado para tal fin, de esta forma al culminar el proceso el usuario podría visualizar el resultado de la calificación de los textos entre sentimientos positivo, negativo y neutro.

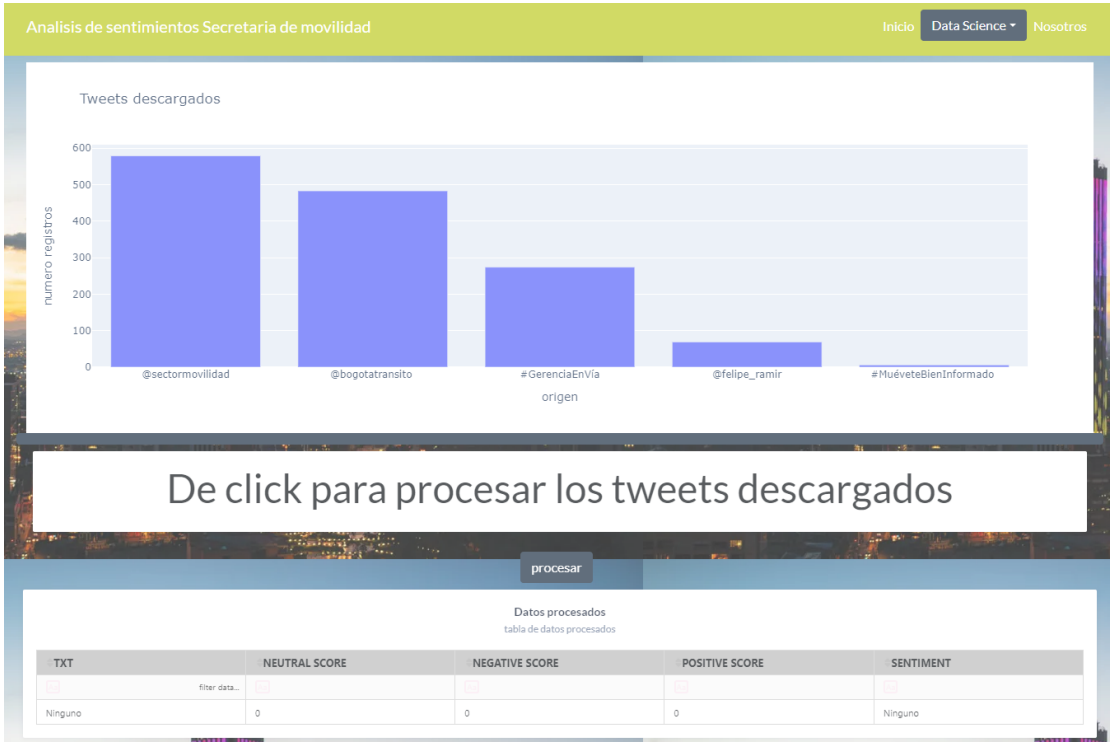


Ilustración 26: Interface resultados descarga

7.4. Selección de datos para el reporte

En esta sección se permite al usuario definir si sobre el reporte en Power Bi quiere ver los datos recién descargados o previere visualizar el corpus de data base de este proyecto. Al realizar la acción la aplicación envía el archivo resultante a un espacio de almacenamiento blob storage en Azure.

Ilustración 27: resultados ejecucion modelo

7.5. Arquitectura de aplicación

A continuación, se presenta la arquitectura de alto nivel de proyecto de la aplicación

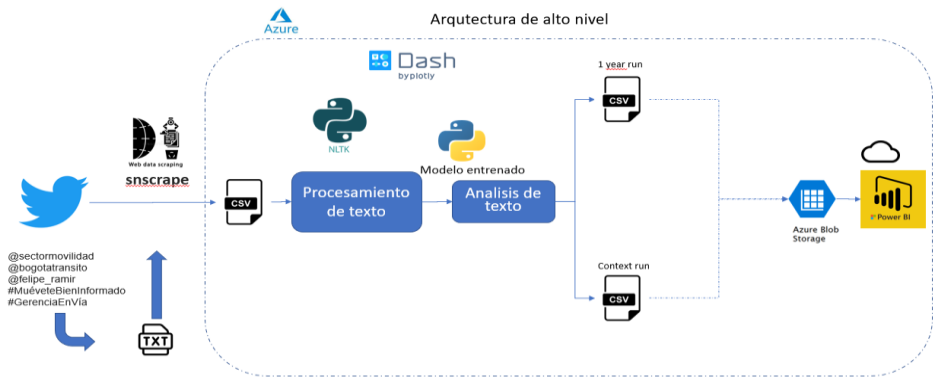
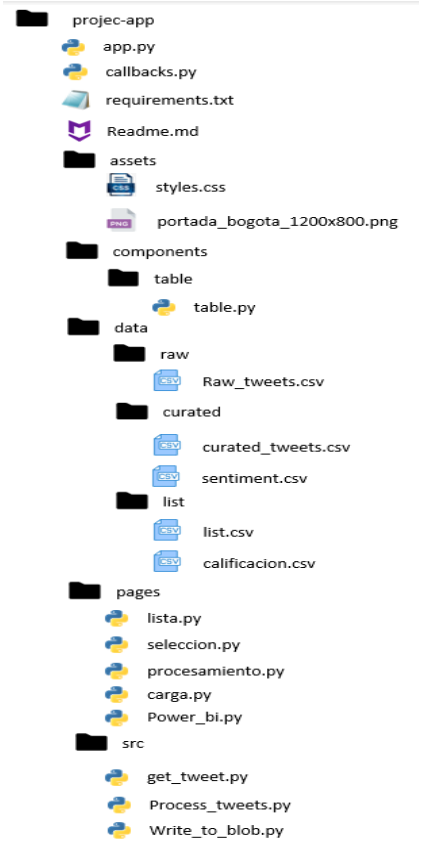


Ilustración 28: arquitectura de alto nivel de la aplicacion


projec-app
app.py
callbacks.py
requirements.txt
Readme.md
assets
styles.css
portada_bogota_1200x800.png
components
table
table.py
data
raw
Raw_tweets.csv
curated
curated_tweets.csv
sentiment.csv
list
list.csv
calificacion.csv
pages
lista.py
seleccion.py
procesamiento.py
carga.py
Power_bi.py
src
get_tweet.py
Process_tweets.py
Write_to_blob.py

8. Conclusiones y trabajos futuros

8.1. Conclusiones

8.1.1. Clasificación de sentimientos

1. Dentro del conjunto de técnicas seleccionadas el mejor desempeño obtenido a la luz de las métricas definidas es el modelo de regresión logística (con parámetros por defecto) con regularización L2, C = 1.0, solver= lbfgs y multi_class = auto. Con un nivel F1-promedio de 54.1%.
2. A pesar de ser el modelo de regresión logística el que obtiene el mejor desempeño en términos del F1- Score y Acuracy, tiene baja capacidad para detectar comentarios positivos lo cual se evidencia con un total de 7.3% (sobre el total del conjunto de prueba) de menciones positivas calificadas como negativas y tan solo un 2.85% de menciones positivas calificadas correctamente. Esto se ve claramente reflejado en un recall de menciones negativos de tan solo 23%. Sin embargo, el modelo encontrado supera el desempeño del modelo pre-entrenado de pysentimiento en términos del F1-Score promedio en 6.6%.
3. Si bien la técnica de Naive Bayes presentó inicialmente el rendimiento más bajo con un F1-score promedio del 50.2%, destacó en áreas donde la regresión logística mostró deficiencias, particularmente en el recall de menciones positivas (50% frente al 23.33% de la regresión logística). Aunque en una primera instancia Naive Bayes fue descartada, su desempeño en ciertos indicadores sugiere que podría ser reconsiderada en futuras evaluaciones, especialmente en el contexto de modelos de ensamble.
4. Dentro del conjunto de modelos con mejor desempeño se encontró que para regresión logística y random forest el conjunto de datos que generó el mejor resultado fue en el que no se eliminaron “Stop words”, ni se aplicó lematización o steaming, lo que valida la hipótesis de que en las Stop words puede haber información relevante a la luz de la clasificación de sentimientos.
5. Para las tres técnicas, el mejor modelo tuvo algún tipo de balanceo de datos, esto está alineado con el hecho de contar con una división de datos desbalanceado entre las clases.
6. Con los resultados actuales no se considera contar con un modelo lo suficientemente robusto para implementar una evaluación de sentimientos en producción.

8.1.2. Obtención de información

1. Dentro de las posibilidades de obtención de los textos e información adicional (ej: número de seguidores, número de likes, etc) se evaluaron tres opciones 1) el uso del API de Twitter, la cual fue descartada por imposibilidad de obtener históricos más allá de una semana con la licencia de desarrollador (existe una licencia de investigador que supera este límite pero no fue posible obtener este acceso) 2) proceso de ingeniería de datos que paulatinamente recopile menciones por procesos que se ejecuten automáticamente como ETL, fue descartado pues requiere un trabajo fuerte de ingeniería de datos y consumos constantes de servidores 3) uso de librería snscrape, la cual obtiene

información a través de técnicas de webscraping. Se determinó que al momento de realizar el proyecto la tercera opción era la más viable a pesar de que en varias ocasiones fue necesario reinstalar la librería por actualizaciones del servicio de Twitter, dado que snscrape no es una librería certificada por Twitter.

2. El desarrollo de clasificación de textos en sentimientos, con origen en redes sociales, representan una dificultad al momento de obtener la variable de respuesta para el entrenamiento de los datos, pues a pesar de contar con una base de más de 800k textos, solo fue posible calificar 1228 textos dentro de la clasificación de sentimiento positivo, negativo o neutro, por limitaciones de tiempo y de alcance.

8.1.3. Proceso

1. Dentro del desarrollo de las diferentes evaluaciones se encontraron grandes beneficios en la construcción de librerías de funciones, las cuales permitieron la evaluación de un conjunto amplio de modelos, 120 modelos por técnica para un total de 360 modelos evaluados, generando así confianza en la selección de los mejores modelos dentro de las técnicas evaluadas.
2. Aunque siempre se debe definir la una métrica de evaluación que permita comparar el desempeño de diferentes técnicas, es importante no perder de perspectiva los trade off que se pueden presentar y por lo tanto para modelos finales es importante tener la posibilidad de revisar la matriz de confusión en problemas de clasificación, teniendo en cuenta que entre mayor sea el número niveles a clasificar, esta evaluación se vuelve más compleja.

8.1.4. Evaluación de eventos

1. A pesar de no conseguir los niveles deseados en las medidas de desempeño seleccionadas para una clasificación en los niveles positivo, negativo y neutro, en general los modelos tenían alta capacidad de detectar tweets negativos por lo que se consideran funcionales para comparaciones sobre la proporción de negativos.
2. Al comparar la proporción de comentarios negativos se observa un desempeño inferior (mayor proporción de negativos) en las fechas asociadas a eventos. Lo anterior a pesar de ser contra intuitivo (la expectativa inicial era que la implementación de medidas mejorara la percepción de los usuarios) puede implicar que en general la red social es más llamativa para comentarios negativos y más que concluir que los iniciativas desarrolladas empeoren la percepción de los usuarios, se puede pensar que los comentarios de Twitter no necesariamente son una buena medida del nivel de percepción de los usuarios, porque de hecho es una muestra sesgada a los usuarios que suelen usar Twitter.

8.2. Trabajos futuros

1. Como se mencionó, una de las limitaciones más fuertes fue el hecho de no contar con una gran base de datos para entrenamiento por la imposibilidad de contar con textos

calificados, al ser la calificación un trabajo manual, por ello se recomienda para futuros desarrollos la implementación de mecanismos estándar para la calificación de los sentimientos de los textos.

2. Un hecho particular respecto a la calificación de textos de Twitter es la brevedad de los textos que allí residen, el proyecto actual evaluó cada texto de forma independiente, sin embargo, la mayoría de los comentarios responden o están en cadena con otros comentarios, por ello en futuros trabajos seria valioso evaluar los textos como parte de cadenas y no como elementos independientes.
3. Para la clasificación de textos en el proyecto actual se utilizó únicamente el texto del tweet sin embargo podría integrarse al análisis de clasificación otras variables que son posibles obtener, como número de likes, hora de tweet, entre otros.
4. Dentro del proyecto actual solo se trabajaron dos técnicas de representación Count-vectorizer y TF-IDF. Sin embargo, existen otras técnicas como Glove o word2vec que vale la pena evaluar.
5. Dentro del alcance del presente proyecto solo se contempló la evaluación de las técnicas Naive Bayes, Random Forest y regresión lineal, sin embargo, vale la pena evaluar otros modelos.
6. Dentro de los resultados se identificó que algunos modelos son buenos identificando lo que otros son malos, por ello valdría la pena la evaluación de técnicas de ensamble que permitan obtener los beneficios de múltiples modelos.

9. Referencias bibliográficas

[1] K. P. Murphy, "Machine learning: a probabilistic perspective," MIT press, 2012.

[2] A. McCallum and K. Nigam, "A comparison of event models for Naive Bayes text classification," in AAAI-98 workshop on learning for text categorization, 1998, vol. 752, pp. 41-48.

[3] H. Zhang, "The optimality of naive Bayes," FLAIRS2004 conference, pp. 562-567, 2004.

[4] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5-32, 2001.

[5] A. Liaw and M. Wiener, "Classification and regression by randomForest," R news, vol. 2, no. 3, pp. 18-22, 2002.

[6] H. Chen, A. Liaw, and L. Breiman, "Using random forest to learn imbalanced data," University of California, Berkeley, vol. 110, no. 1, pp. 24-25, 2004.

[7] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning: data mining, inference, and prediction," Springer Science & Business Media, 2009.

[8] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, "Applied logistic regression," John Wiley & Sons, 2013.

[9] A. Agresti, "An introduction to categorical data analysis," John Wiley & Sons, 2007.

[10] R. A. Becker, C. Volinsky, and A. R. Wilks, "Fraud detection in telecommunications: History and lessons learned," Technometrics, vol. 52, no. 1, pp. 20-33, 2010.

[11] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," Decision Support Systems, vol. 62, pp. 22-31, 2014.

[12] S. Scott and S. Matwin, "Text classification using WordNet hypernyms," in Use of WordNet in natural language processing systems: Proceedings of the conference, 1998, vol. 1, pp. 38-44.

[13] M. Sahami and T. D. Heilman, "A web-based kernel function for measuring the similarity of short text snippets," in Proceedings of the 15th international conference on World Wide Web, 2006, pp. 377-386.

[14] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques," in Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, 2002, pp. 79-86.

[15] Y. Jo and A. H. Oh, "Aspect and sentiment unification model for online review analysis," in Proceedings of the fourth ACM international conference on Web search and data mining, 2011, pp. 815-824.

[16] M. Abulaish and T. H. Dey, "A text normalization technique using machine learning approach for text mining," in International Conference on Computational Intelligence and Communication Networks (CICN), 2010, pp. 520-524.

[17] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in Proceedings of the 18th International Conference on Machine Learning (ICML), 2001, pp. 282-289.

[18] S. Bird, E. Klein, and E. Loper, "Natural language processing with Python: analyzing text with the natural language toolkit," O'Reilly Media, Inc., 2009.

[19] D. Jurafsky and J. H. Martin, "Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition," Pearson/Prentice Hall, 2009

[20] J. Ramos, "Using TF-IDF to determine word relevance in document queries," in Proceedings of the first instructional conference on machine learning, 2003, pp. 133-142.

[21] B. Liu, "Sentiment analysis and opinion mining," Synthesis lectures on human language technologies, vol. 5, no. 1, pp. 1-167, 2012.

[22] R. Rehurek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, 2010, pp. 45-50.

[23] M. F. Porter, "An algorithm for suffix stripping," Program, vol. 14, no. 3, pp. 130-137, 1980.

[24] D. Harman, "How effective is suffixing?" Journal of the American society for information science, vol. 42, no. 1, pp. 7-15, 1991.

[25] C. D. Manning, P. Raghavan, and H. Schütze, "Introduction to Information Retrieval," Cambridge university press, 2008.

[26] A. Balahur, R. Steinberger, M. Kabadjov, V. Zavarella, E. Van Der Goot, B. Halkia, ... and M. Turchi, "Sentiment analysis in the news," arXiv preprint arXiv:1309.6202, 2013.

[27] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," in Proceedings of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies, 2007, pp. 3-24.

[28] J. R. Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufmann Publishers, 1993.

[29] T. Fawcett, "An introduction to ROC analysis," Pattern recognition letters, vol. 27, no. 8, pp. 861-874, 2006.

[30] D. M. W. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," Journal of Machine Learning Technologies, vol. 2, no. 1, pp. 37-63, 2011.

[31] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, 2002.

[32] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in International Conference on Intelligent Computing, 2005, pp. 878-887.

[33] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," SIGKDD Explorations, vol. 6, no. 1, pp. 20-29, 2004.

[34] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in IEEE International Joint Conference on Neural Networks (IJCNN), 2008, pp. 1322-1328.

[35] A. V. Aho, "Algorithms for finding patterns in strings," in Handbook of theoretical computer science (vol. A): algorithms and complexity, 1990, pp. 255-300.

[36] M. Friedl, "Mastering Regular Expressions," O'Reilly Media, Inc., 2006.

[37] R. Sedgewick and K. Wayne, "Algorithms, Part II," Coursera, 2011.

[38] T. Mitchell, "Machine Learning," McGraw Hill, 1997.

[39] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in Neural Information Processing Systems, 2013, pp. 3111-3119.

[40] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information Processing & Management, vol. 24, no. 5, pp. 513-523, 1988.

[41] Y. Goldberg, "Neural network methods for natural language processing," Synthesis Lectures on Human Language Technologies, vol. 10, no. 1, pp. 1-309, 2017.

[42] C. E. Rasmussen and C. K. I. Williams, "Gaussian Processes for Machine Learning," MIT Press, 2006.

[43] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," Journal of Machine Learning Research, vol. 13, no. Feb, pp. 281-305, 2012.

[44] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in International Conference on Learning and Intelligent Optimization, 2011, pp. 507-523.

[45] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in Advances in Neural Information Processing Systems, 2011, pp. 2546-2554.

[46] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," arXiv preprint arXiv:1012.2599, 2010.

[47] R. Tibshirani, "Regression shrinkage and selection via the lasso," Journal of the Royal Statistical Society: Series B (Methodological), vol. 58, no. 1, pp. 267-288, 1996.

[48] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," Technometrics, vol. 12, no. 1, pp. 55-67, 80-86 1970.

[49] Y. Shevade and S. Keerthi, "A simple and efficient algorithm for gene selection using sparse logistic regression," Bioinformatics, vol. 19, no. 17, pp. 2246-2253, 2003.

[50] L. Breiman, J. Friedman, R. Olshen, and C. Stone, "Classification and Regression Trees," Wadsworth, 1984.

[51] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in International joint Conference on artificial intelligence, 1995, pp. 1137-1145.

[52] K. K. Dobbin and R. M. Simon, "Sample size planning for developing classifiers using high-dimensional DNA microarray data," Biostatistics, vol. 9, no. 1, pp. 13-27, 2008.

[53] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," Neural computation, vol. 10, no. 7, pp. 1895-1923, 1998.

[54] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," Bioinformatics, vol. 23, no. 19, pp. 2507-2517, 2007.

[55] G. James, D. Witten, T. Hastie, and R. Tibshirani, "An Introduction to Statistical Learning: with Applications in R," Springer Texts in Statistics, 2013.

[56] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," Journal of machine learning research, vol. 3, no. Mar, pp. 1157-1182, 2003.

[57] H. Liu and H. Motoda, "Feature Selection for Knowledge Discovery and Data Mining," Springer, 1998.

[58] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern Classification," John Wiley & Sons, 2012.

[59] G. Fernández and M. Alarcón, "Estudio comparativo de algoritmos de clasificación para el análisis de sentimientos en tweets", Universidade da Coruña, España, 2019. [Online]. Available: https://ruc.udc.es/dspace/handle/2183/25152. [Accessed: 22- May- 2023].

[60] A. Gonzalez-Pardo and D. Camacho, "Un análisis de sentimiento en Twitter con machine learning: Identificando el sentimiento sobre las ofertas de Black Friday", ResearchGate, 2018. [Online]. Available: https://www.researchgate.net/publication/328413334_Un_Analisis_de_Sentimiento_en_Twitter_con_Machine_Learning_Identificando_el_sentimiento_sobre_las_ofertas_de_BlackFriday_A_Sentiment_Analysis_in_Twitter_with_machine_learning_capturing_sentiment_from_. [Accessed: 22- May- 2023].

[61] E. Domínguez Leiva, "Análisis de Sentimientos en Twitter", Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, Argentina, 2019. [Online]. Available: https://ria.utn.edu.ar/xmlui/bitstream/handle/20.500.12272/4704/An%C3%A1lisis_de_Sentimientos_en_Twitter_%28167%29.pdf?sequence=1&isAllowed=y. [Accessed: 22- May- 2023].

[62] J. C. Ortega, "Caso de estudio de análisis de sentimientos en Twitter: Tratado de libre comercio de América", Revista de Ciencias Sociales, vol. 147, no. 5, pp. 26-40, 2018. [Online]. Available:

https://rcs.cic.ipn.mx/rcs/2018_147_5/Caso%20de%20estudio%20de%20analisis%20de%20sentimientos%20en%20Twitter_%20Tratado%20de%20libre%20comercio%20de%20America.pdf. [Accessed: 22- May- 2023].

[63] L. A. López Rojas and D. L. Guarin Guarin, "Técnicas de análisis de sentimientos aplicadas a la valoración de opiniones en el lenguaje español", ResearchGate, 2021. [Online]. Available: https://www.researchgate.net/publication/355887680_Tecnicas_de_Analisis_de_Sentimientos_Aplicadas_a_la_Valoracion_de_Opiniones_en_el_Lenguaje_Espanol. [Accessed: 22- May- 2023].

[64] Secretaría Distrital de Movilidad de Bogotá, "Plan de Movilidad", Bogotá, Colombia, 2021. [Online]. Available: https://www.movilidadbogota.gov.co/web/sites/default/files/Páginas/21-01-2021/cartilla_plan_de_movilidad_0.pdf. [Accessed: 22- May- 2023].

[65] M. Henze, "snscrape: A social media scraping library," GitHub, 2022. [Online]. Available: https://github.com/JustAnotherArchivist/snscrape. [Accessed May. 22, 2023].

[66] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," Transactions of the Association for Computational Linguistics, vol. 5, pp. 135-146, 2017. [Online]. Available: https://www.aclweb.org/anthology/Q17-1010/. [Accessed May. 22, 2023].

[67] S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach", Pearson, 2020.

[68] M. Z. Afzal, A. Maurer, B. Kolb, P. Bühler and A. Dengel, "Naive Bayes as Baseline for Text Classification", arXiv preprint arXiv:1910.10680, 2019.

[69] C. Bishop, "Pattern Recognition and Machine Learning", Springer, 2006.

[70] Y. Goldberg, "A Primer on Neural Network Models for Natural Language Processing", Journal of Artificial Intelligence Research, vol. 57, pp. 345-420, 2016.

[71] S. H. Walker and D. B. Duncan, "Estimation of the Probability of an Event as a Function of Several Independent Variables", Biometrika, vol. 54, no. 1-2, pp. 167-179, 1967.

[72] P. Domingos and M. Pazzani, "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss", Machine Learning, vol. 29, no. 2-3, pp. 103-130, 1997.

[73] I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, 2016

[74] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality", Advances in Neural Information Processing Systems, 2013.

[75] J. VanderPlas, "Python Data Science Handbook: Tools and Techniques for Developers," O'Reilly Media, Inc., 2016.

[76] J. Brownlee, "How to Prepare Text Data for Machine Learning with scikit-learn," Machine Learning Mastery, 2016. [Online]. Available: https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/. [Accessed May 2023].

[77] A. Géron, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems," O'Reilly Media, Inc., 2019.

[78] K. Arulkumaran et al., "A Brief Introduction to Neural Networks," Dpunkt.Verlag, 2017.

[79] S. Raschka and V. Mirjalili, "Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2," Packt Publishing, 2019.

[80] B. Pang and L. Lee, "Opinion mining and sentiment analysis," Foundations and Trends in Information Retrieval, vol. 2, no. 1–2, pp. 1–135, Jan. 2008.

[81] I. Rish, "An empirical study of the naive Bayes classifier," in IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, vol. 3, no. 22, pp. 41-46, 2001.

[82] R. A. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., 1999.

[83] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," Knowledge-Based Systems, vol. 18, no. 4-5, pp. 187–195, Aug. 2005.

[84] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning," arXiv:1811.12808 [cs, stat], Nov. 2018.

[85] X. Wu and X. Zhu, "Mining with noise knowledge: Error-aware data mining," IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, vol. 38, no. 4, pp. 917–932, 2008.

[86] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in Advances in large margin classifiers, vol. 10, no. 3, pp. 61–74, 1999.

[87] Python Software Foundation. (2021). The Python Standard Library - 11. Data Persistence. Retrieved from

[88] Porter, M. F. (1980). An algorithm for suffix stripping. Program, 14(3), 130-137.

[89] Sahami, M., & Heilman, T. D. (2006). A web-based kernel function for measuring the similarity of short text snippets. Proceedings of the 15th international conference on World Wide Web, 377-386.

[90] Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). Mining of massive datasets. Cambridge university press.

[91] Ramos, J. (2003). Using TF-IDF to Determine Word Relevance in Document Queries. Proceedings of the First Instructional Conference on Machine Learning.

[92] Aggarwal, C. C., & Zhai, C. (2012). A survey of text classification algorithms. In Mining text data (pp. 163-222). Springer.

[93] Zhang, W., Yoshida, T., & Tang, X. (2011). A comparative study of TF IDF, LSI and multi-words for text classification. Expert Systems with Applications, 38(3), 2758-2765.*

[94] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb), 281-305.

[95] Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification. Department of Computer Science, National Taiwan University.

[96] Dietterich, T. G. (2000). Ensemble methods in machine learning. In Multiple classifier systems (pp. 1-15). Springer.

[97] Elkan, C. (2001). The foundations of cost-sensitive learning. In Proceedings of the 17th international joint conference on Artificial intelligence (pp. 973-978).

[98] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427-437.

[99] Ng, A. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. In Proceedings of the twenty-first international conference on Machine learning (p. 78).

[100] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267-288.

[101] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

[102] Rifkin, R., & Klautau, A. (2004). In defense of one-vs-all classification. Journal of machine learning research, 5(Jan), 101-141.

[103] Wilcoxon, F. (1945). Individual comparisons by ranking methods. Biometrics bulletin, 1(6), 80-83.

[104] L. de Bruijn, "Inter-Annotator Agreement (IAA). Pair-wise Cohen kappa and group Fleiss' kappa (κ) coefficients for categorical annotations," Towards Data Science, 18-Jul-2020. [En línea]. Disponible en: https://towardsdatascience.com/inter-annotator-agreement-2f46c6d37bf``oaicite: {"number":1,"metadata": {"title":"Inter-Annotator Agreement (IAA). Pair-wise Cohen kappa and group Fleiss'… | by Louis de Bruijn | Towards Data Science","url":"https://towardsdatascience.com/inter-annotator-agreement-2f46c6d37bf3","text":"Inter-Annotator

[105] A. N. Tikhonov, "On the stability of inverse problems," in Dokl. Akad. Nauk SSSR, 1943.

[106] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," in Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 67, no. 2, pp. 301-320, 2005.

[107] I. Y. Chun and S. Keles, "Sparse partial least squares regression for simultaneous dimension reduction and variable selection," in Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 72, no. 1, pp. 3-25, 2010.

[108] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," in Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 68, no. 1, pp. 49-67, 2006.

[109] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," in Foundations and Trends® in Machine Learning, vol. 3, no. 1, pp. 1-122, 2011.

Santiago de Cali, 28 de Junio de 2023

Ingeniero:
Juan Carlos Martínez Arias
Director Posgrados de Ingeniería
Facultad de Ingeniería y Ciencias
Pontificia Universidad Javeriana - Cali

Con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el Trabajo de Grado y posteriormente optar por el título de Magíster en Ciencia de Datos, nos permitimos presentar a su consideración el proyecto de Trabajo de Grado denominado Análisis de Sentimientos Utilizando Aprendizaje Automático de Menciones en Twitter Para la Secretaria de Movilidad de Bogotá, el cual será realizado por el (los) estudiantes Luis Eduardo Quiñones, Luisa Fernanda Carbonell y Andrés Peralta Alean Con códigos 8972069 ,8973333 y 8972774, perteneciente al énfasis en Sistemas y Computación, bajo la dirección de la profesora María Constanza Pabón Burbano.

El suscrito director del Trabajo de Grado autoriza para que se proceda a hacer la evaluación de este Proyecto ante el Tribunal que para el efecto se designe, toda vez que ha revisado cuidadosamente el documento y avala que ya se encuentra listo para ser presentado oficialmente.

Atentamente,

Andrés Peralta Alean
C.C. 10884691 de San Marcos

Luis Eduardo Quiñones

Luis Eduardo Quiñones
C.C 80.084.297 Bogotá

Luisa Fernanda Carbonell García
CC 52538837

María Constanza Pabón Burbano
C.C. 34.559.226