Estimación de Éxito de Videos de YouTube Mediante Enfoques de Machine Learning: Caso DelcaVideography

Estudiantes
Mariana Agudelo Zuluaga
Andrés Mauricio Cano Campiño
Esteban Castro Castaño
Juan David Gallego Montoya
Vanessa Osorio Urrea

Universidad EAFIT
Escuela de Ciencias Aplicadas e Ingeniería
José Lisandro Aguilar Castro
Paula María Almonacid Hurtado
Edison Valencia Díaz

Medellín, Colombia
Junio 10 de 2023

1. Introducción

Según el historial de visualizaciones y demás datos recolectados para un canal de YouTube, es posible generar modelos de clasificación que permitan identificar cuáles son las variables más sensibles o significativas para apalancar el éxito de este. Particularmente se hace un análisis de DelcaVideography, un canal educativo donde se encuentran tutoriales gratuitos dedicados al diseño gráfico, audiovisual y fotográfico.

> Para optimizar el aprendizaje de mi audiencia, aplico técnicas y procedimientos claros y fáciles de entender y, por supuesto, muy divertidos. Allí se pueden encontrar contenidos que incluyen desde diseño de logos, ilustraciones, arte abstracto, arte digital, imagen corporativa, tips o consejos de diseño, hasta reseñas de versiones de programas, diseño conceptual, diseño paramétrico, entre otros (DelcaVideography, 2022).

YouTube no ha dejado de ganar usuarios desde su lanzamiento y actualmente se trata de la plataforma de video abierto por excelencia, gracias al amplio y diverso contenido que ofrece, en su gran mayoría, de libre acceso. En concreto, el servicio de video bajo demanda financiado con publicidad (AVoD) contaba en 2021 con más de 2.500 millones de usuarios en todo el mundo, lo que supuso un incremento de 300 millones con respecto al año anterior (Statista, 2022).

YouTube realizó su análisis anual de las audiencias latinoamericanas en la plataforma, y esta vez registró un gran aumento. Por ejemplo, en Colombia alcanzó a más de 20 millones de personas en 8 meses, contando solo quienes son mayores de 18 años (Cruz, 2020).

Tanto en la búsqueda principal como en la selección de sugerencias, YouTube encuentra videos para los usuarios en lugar de usuarios para los videos. El algoritmo no promociona ni expone los videos, sino que los muestra a cada usuario cuando esta visita la plataforma. El objetivo del sistema de descubrimiento y búsqueda es unir a cada usuario con los videos que mejor respondan a sus intereses, generando así recomendaciones personalizadas (YouTube Creators, 2022).

Los videos de YouTube se clasifican según dos categorías: personalización del usuario y rendimiento del video.

La personalización de usuario se da según:

- Los videos que escogen ver
- Los videos que ignoran
- Los videos que rechazan
- La frecuencia con la que ven un canal o un tema

El rendimiento de los videos se da según el nivel de importancia que dan a estos los usuarios. Esto puede referirse, entre otras variables, a la duración media de la visualización, a cuánto del contenido se ha visto y a los “me gusta” que reciba el video. Lo anterior ayuda al algoritmo a acotar el mejor conjunto de videos para cada uno de los usuarios.

Por su lado, la cantidad de suscriptores, a pesar de ser una variable que puede influir en la cantidad de visualizaciones de un video, no debería ser el centro de atención del creador de contenido. Según Youtube (YouTube Creators, 2022), será más importante concentrarse en conocer a la audiencia y el tipo de contenido que les gusta, más que en el número de suscriptores y saber cómo funciona el algoritmo.

Otros factores que afectan la cantidad de visualizaciones son (YouTube Creators, 2022):

- *Interés en el tema:* Cuántas personas muestran interés en un tema concreto, hay unos temas que generan más interés que otros. Adicionalmente los intereses en los temas pueden cambiar con el paso del tiempo, es decir, pueden perder o ganar popularidad.
- *Competencia:* Puede que un video en particular tenga buenas métricas en cuanto al tiempo de visualización y el tiempo promedio de visualización, sin embargo, si un video del mismo tema tiene mejores métricas, este tiene mayor probabilidad de ser sugerido a los usuarios.
- *Estacionalidad:* El tráfico puede cambiar dependiendo de la época del año. También la etapa de la vida que están atravesando los espectadores puede influir en el éxito de los videos.

Buscando comprender las variables que afectan el posible éxito de un video se analizaron los datos provenientes de DelcaVideography, un canal de YouTube con 112 mil suscriptores, más de 9 millones de vistas y 8 años de antigüedad.

El resto del documento se encuentra organizado de la siguiente forma. En la sección 2 se aborda el marco teórico. En la sección 3 se describe el desarrollo metodológico, incluyendo el entendimiento del problema, el análisis exploratorio de los datos y la selección de modelos. También se plantea el ciclo de vida y la arquitectura de los datos. Finalmente, en la sección 4 y 5, se exponen las conclusiones del trabajo.

2. Marco conceptual

Ver videos se ha convertido en una actividad muy popular. En el pasado la mayoría de los servicios de transmisión eran unidireccionales, los perfiles de los usuarios no eras accesibles y por ende no era posible construir contenido (publicidad o programas) dirigido a un público específico. Con el incremento de las transmisiones por internet, la habilidad para conocer y consolidar información demográfica de los usuarios tales como su género o edad, basados en su historial de uso, hace viable la efectividad en la exposición a la publicidad y a los servicios de contenido ofrecidos (Nananukul, 2022).

Existen diversidad de casos de estudio en los cuales se ha identificado la posibilidad de generar videos cada vez más relacionados con los intereses de los usuarios. A modo de ejemplo se expone el artículo “Wild birds in YouTube videos: Presence of specific species contributes to increased views” (Kikuchi et al., 2022). En este caso, por ejemplo, se encontró que la presencia de especies específicas de aves incrementaba el número de visualizaciones de los videos, así mismo, se identifica que la duración de los videos, así como el número de días después de su carga en YouTube también incide en el éxito de él (Kikuchi et al., 2022).

Teniendo como objetivo identificar las variables que hacen que un canal sea más exitoso, es necesario tener en cuenta que, como se menciona en (Zappin et al., 2022), la monetización se logra casi exclusivamente a través de la publicidad reproducida en el contenido publicado por los creadores en sus canales. La publicidad de los videos aparece en tres momentos diferentes:

- Antes de iniciar la reproducción del video
- Durante el video
- En banners dentro del video.

Como también se menciona en (Zappin et al., 2022), la generación de publicidad en los videos es una variable de gran relevancia para los creadores de contenido, y es a la vez uno de los apalancadores del crecimiento de YouTube. Considerando como la plataforma determina si un video aplica para ser monetizado, esta tiene políticas establecidas para cada uno de los creadores de contenido y canales. Específicamente debe cumplir los siguientes criterios: i) tener más de 1000 suscriptores, ii) tener más de 4000 horas de tiempo de visualización durante el último año y iii) seguir las condiciones de la comunidad de la plataforma (Zappin et al., 2022). No obstante, sigue siendo potestad de YouTube monetizar un video y, a pesar de que pareciera haber indicios de las razones de la exposición a los usuarios de un video, se menciona también en (Zappin et al., 2022) que no se ha publicado oficialmente una guía sobre lo que puede determinar claramente el éxito de un video, dando así cabida al tipo de análisis realizado en este trabajo.

2.1 Resultados previos

En literatura científica encontramos pocas referencias que desarrollen temas analíticos desde el punto de vista de los creadores de contenidos, una de las más relevantes fue el documento de Eitan Altam y Tania Jiménez, en donde enfocan su trabajo en las medidas de retención de la audiencia que ofrece YouTube a los creadores de contenido y les pueden servir para potenciar el rendimiento de sus videos. Su propuesta se basa en el enfoque de red EGO, que expone el interés de estudiar una pequeña subred en lugar de abracar grandes cantidades de videos. Dentro de las variables clave que exponen está la Retención promedio, que es el tiempo promedio que un espectador tarda viendo un video entre todos los espectadores. También están el Porcentaje promedio visto (APV), la Duración promedio de visualización (AVD) y la Duración del video. Para esta última variable, exponen el argumento que reducir el tiempo del video daría como resultado un aumento en la fracción promedio de espectadores. También mencionan la Retención de audiencia relativa (RAR), que es una medida que expone YouTube para comparar de forma justa la retención de videos de diferentes duraciones. Adicional, proponen una medida nueva a la que llaman Porcentaje Leal, la cual representaría la fracción de la audiencia que permanece hasta el final del cuerpo del video (Altman & Jiménez, 2019).

Por otro lado, SEO significa Search Engine Optimization (Optimización para motores de búsqueda), el cual se trata del conjunto de estrategias y técnicas de optimización que se hacen en una página web para que aparezca orgánicamente en buscadores de Internet como Google, Yahoo o Youtube. La correcta aplicación del SEO puede derivar en incrementos en el tráfico y la visibilidad de las marcas en Internet (Mousinho, 2021). Asociado a este concepto y validando con el dueño del canal DelcaVideography, en la comunidad de YouTubers existe un contenido al que pueden acceder los interesados, pagando una suscripción, para realizar el curso llamado CRECETUBE. En el cual, su creador Romuald Fons, mediante contenidos basados en SEO y el entendimiento de los reportes

analíticos de cada canal, ofrece alternativas para crecer en YouTube, profesionalizar los canales, aumentar visualizaciones y suscriptores (Fons & BIGSEO, n.d.).

De acuerdo a la descripción que ofrecen de los contenidos, CRECETUBE se basa en usar los principios del Método Científico, como lo son: Observación, Experimentación, Medición, Mejora, Repetición. El modelo del curso lo fundamentan en 3 pilares (Fons & BIGSEO, n.d.):

1. Detectar: Las oportunidades de optimizar contenido y distinguir a la competencia.
2. Medir: Analizar las métricas es clave para el crecimiento y exposición de un canal de YouTube.
3. Mejorar: Aplicar las estrategias necesarias y concretas a cada área del canal, y verificar el incremento de visualizaciones, suscripciones e ingresos.

Respecto al análisis de los datos y el monitoreo de los contenidos publicados por cada creador de contenidos, hacen énfasis en que "Cuando Mides y Mejoras los Factores Medibles, Tu Canal Crece". Para lo cual mencionan variables de gran relevancia como (Fons & BIGSEO, n.d.):

- CTR
- Retención de Audiencia
- Posicionamiento Orgánico
- Tráfico Recomendado
- Duración Media de Visualización
- Número de Suscriptores

2.2 Modelos

2.2.1 Método lineal - Regresión Logística

Para este problema de clasificación que es un modelo supervisado se utilizará la regresión logística, a continuación, se presenta una pequeña descripción de la técnica empleada:

El objetivo primordial que resuelve esta técnica es el de modelar cómo influye en la probabilidad de aparición de un suceso, habitualmente dicotómico, la presencia o no de diversos factores y el valor o nivel de estos. También puede ser usada para estimar la probabilidad de aparición de cada una de las posibilidades de un suceso con más de dos categorías (Molinero, 2001).

La regresión logística puede ser representada de la siguiente manera:

$$logit(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 \ + \beta_2 X_2 \ + \cdots + \beta_n X_n$$

Donde $X_1, X_2, \dots, X_n$ representan las variables de entrada al modelo y $\beta_0, \beta_1, \beta_2, \dots \beta_n$ representan los coeficientes de regresión de las variables de entrada.

Como se menciona en (Nananukul, 2022) regresión logística tiene varias características que son adecuadas para implementar el modelo de inferencia.

- Para la regresión lineal regular, las probabilidades podrían llegar a ser mayores que uno y menores que cero. Esos valores son inadmisibles para representar probabilidad.
- La función logística proporciona una varianza más realista de la salida, especialmente cuando hay dos resultados. En contraste con la regresión regular donde la varianza de la salida es constante a través de los valores de las entradas, la varianza de la salida de la función logística se acerca a cero mientras p se acerca a uno o cero.

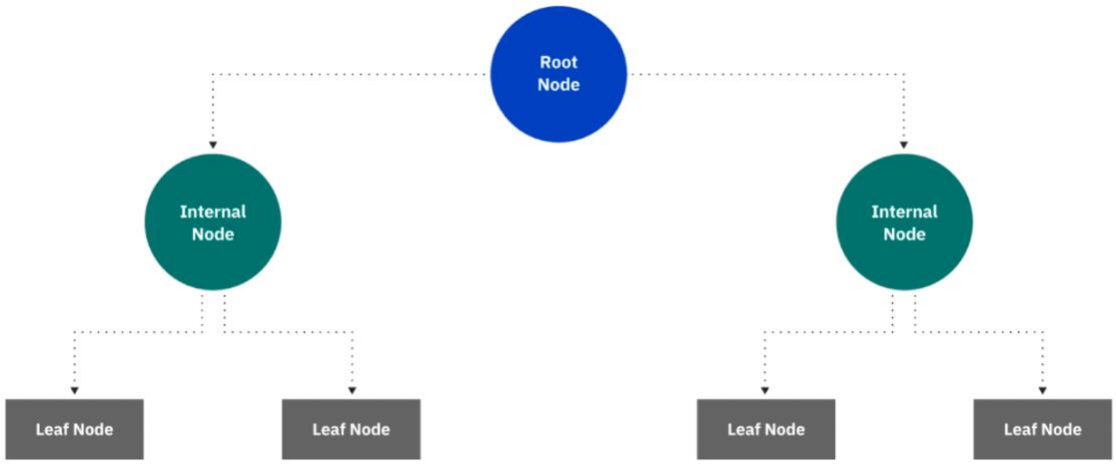
2.2.2 Métodos basados en árboles

Los modelos de aprendizaje automático basados en árboles ofrecen ventajas que los hacen altamente deseables (Pinelis & Ruppert, 2020). Por un lado, al ser no paramétricos, no hacen suposiciones sobre la distribución de los datos, lo que les brinda flexibilidad en diversos escenarios (Pinelis & Ruppert, 2020). Además, son invariantes a la escala de las características, lo que significa que los cambios en la escala no afectan las predicciones (Pinelis & Ruppert, 2020). Por último, su estructura de parámetros es más manejable en comparación con técnicas como las redes neuronales, lo que simplifica el proceso de optimización (Pinelis & Ruppert, 2020).

En este proyecto se emplean árboles de decisión, Random Forest y XGBoost.

i) Árbol de decisión

Un árbol de decisión es un algoritmo de aprendizaje supervisado no paramétrico, que se utiliza tanto para tareas de clasificación como de regresión. Tiene una estructura de árbol jerárquica, que consta de un nodo raíz, ramas, nodos internos y nodos hoja (IBM, n.d.-a)



Fuente: (IBM, n.d.-a)

El árbol siempre comienza en un nodo raíz sin ramas entrantes. Las ramas salientes del nodo raíz, alimentan los nodos de decisión, es decir los nodos internos. De acuerdo con la información disponible, estos nodos forman subconjuntos homogéneos que se conocen como nodos hoja o nodos terminales, los cuales “representan todos los resultados posibles dentro del conjunto de datos” (IBM, n.d.-a).

El aprendizaje del árbol funciona identificando los puntos de división óptimos dentro de él.

> Este proceso de división se repite de forma recursiva de arriba hacia abajo hasta que todos o la mayoría de los registros se hayan clasificado bajo etiquetas de clase específicas. Que todos los puntos de datos se clasifiquen o no como conjuntos homogéneos depende en gran medida de la complejidad del árbol de decisión (IBM, n.d.-a).

En los árboles más pequeños, es más fácil tener nodos de hojas puros, es decir puntos de datos de una sola clase. En contraste, cuando el árbol se hace más grande es más difícil mantener esa pureza, lo que generalmente hace que haya muy pocos datos de subárboles determinados y cuando esto sucede es conocido como fragmentación de datos, lo cual puede crear un overfitting. Por esta razón, para los árboles de decisión, es preferible crear árboles pequeños, "lo cual es consistente con el principio de parsimonia en la Navaja de Occam. Es decir, "las entidades no deben multiplicarse más allá de la necesidad"" (IBM, n.d.-a).

Para evitar la complejidad y sobreajuste, se usa la técnica de poda, en donde se eliminan las ramas que se dividen en características de poca importancia y son convertidas en una hoja terminal.

Adicionalmente, también se conoce que cuando los datos de entrenamiento son pocos o incoherentes, se produce un overfitting (Lior Rokach and Oded Maimon, 2008).

Otra forma para que los árboles mantengan su precisión, es mediante el uso de Bosques aleatorios.

Existen varios tipos de árboles de decisión, entre los más populares se encuentran (IBM, n.d.-a):

- **ID3:** Este desarrollo se atribuye a A Ross Quinlan. ID3 es la abreviatura de "Iterative Dichotomiser 3". Este algoritmo aprovecha la entropía y la ganancia de información como métricas para evaluar las divisiones de candidatos.
- **C4.5:** Es considerado una iteración posterior de ID3, también desarrollada por Quinlan. "Puede utilizar la ganancia de información o las proporciones de ganancia para evaluar los puntos de división dentro de los árboles de decisión".
- **CART:** CART es una abreviatura de "árboles de clasificación y regresión" ("classification and regression trees"), introducido por Leo Breiman. El algoritmo normalmente utiliza "la impureza de Gini para identificar el atributo ideal para la división. La impureza de Gini mide la frecuencia con la que se clasifica incorrectamente un atributo elegido al azar. Cuando se evalúa usando la impureza de Gini, un valor más bajo es más ideal".

Generalmente, los árboles de decisión son superados por otros algoritmos, sin embargo, son útiles para minería de datos y descubrimiento o conocimiento de la información.

ii) Random Forest

Un Random Forest (bosque aleatorio) es un algoritmo de aprendizaje automático que se utiliza tanto para la clasificación como para la regresión. Combina múltiples árboles de decisión independientes,

cada uno entrenado con una muestra aleatoria del conjunto de datos original, y luego combina las predicciones de cada árbol para obtener una predicción final.

El Random Forest es muy popular debido a su capacidad para manejar conjuntos de datos grandes y complejos, y su habilidad para evitar el sobreajuste (overfitting)(Breiman, 2001a). Al usar una combinación de árboles de decisión, el Random Forest reduce la varianza y el sesgo inherentes a un solo árbol, mejorando así la precisión general del modelo

> Es una modificación sustancial de bagging que construye una amplia colección de árboles no correlacionados y luego los promedia. En muchos problemas, el rendimiento de los bosques aleatorios es muy similar al del boosting, y son más simples de entrenar y ajustar. Como consecuencia, los bosques aleatorios son populares y se implementan en una variedad de paquetes (Hastie et al., 2001a).

El Random Forest tiene otras ventajas, como la capacidad de manejar tanto variables numéricas como categóricas, identificar la importancia relativa de las características utilizadas en el modelo y proporcionar una medida de la confianza en las predicciones.

Para poder explicar el Random Forest, es muy útil especificar que la idea esencial del *bagging* es promediar muchos modelos que son ruidosos pero que son aproximadamente insesgados para reducir la varianza. Los árboles son buenos candidatos para el bagging ya que capturan interacciones complejas en los datos y si tienen la suficiente profundidad, pueden tener un sesgo relativamente bajo (Hastie et al., 2001a). Debido a la naturaleza ruidosa de los árboles, estos se benefician significativamente al ser promediados. Además, en el bagging, cada árbol generado sigue una distribución idéntica $(i.d.)$, lo que implica que el promedio de B árboles es equivalente en expectativa a cualquier árbol visto individualmente. Esto implica que el sesgo de los árboles en el bagging es igual al de los árboles individuales generados mediante remuestreo (bootstrap), y la única posibilidad de mejora radica en reducir la varianza. Esto contrasta con el boosting, donde los árboles se construyen de manera adaptativa para eliminar el sesgo y, por lo tanto, no siguen una distribución idéntica.

El promedio de B variables aleatorias independientes e idénticamente distribuidas $(i.i.d.)$, cada una con una varianzaσ^2, tiene una varianza de $\frac{1}{B}\sigma^2$. Sin embargo, si las variables son simplemente idénticamente distribuidas $(i.d.)$ pero no necesariamente independientes, y tienen una correlación positiva ρ entre ellas, la varianza del promedio es

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

A medida que B aumenta el segundo término desaparece, pero el primero permanece, lo que significa que el tamaño de la correlación entre los pares de árboles obtenidos mediante bagging limita los beneficios del promedio (Breiman, 2001).

Para lograr la reducción de la varianza que se obtiene en el bagging, reduciendo la correlación entre los árboles, es necesario que los árboles crezcan mediante la selección aleatoria de las variables de entrada. Esto se logra mediante el siguiente algoritmo descrito en (Hastie et al., 2001a).

1) Para $b = 1\ hasta\ B$:

   a) Extraer una muestra bootstrap Z^* de tamaño N de los datos de entrenamiento.
   b) Construir un árbol de bosques aleatorios T_b a partir de los datos bootstrap, repitiendo recursivamente los siguientes pasos para cada nodo terminal del árbol, hasta que se alcance el tamaño mínimo de nodo n_{min}.
      i) Seleccionar m variables al azar de entre las p variables.
      ii) Elegir el mejor punto de división/variable entre las m.
      iii) Dividir el nodo en dos nodos hijo.

2) Obtener el conjunto de árboles $\{T_b\}_1^B$ como resultado.

   Para hacer una predicción en un nuevo punto x:

   Regresión: $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.
   Clasificación: Sea $\hat{C}_b(x)$ la predicción de clase del $b - ésimo$ árbol de bosques aleatorios. Entonces $\hat{C}_{rf}^B(x) = voto\ mayoritario\ \{\hat{C}_b(x)\}_1^B$

Al reducirse m también se reduce la correlación entre cualquier par de árboles del ensamble y se reduce la varianza del promedio.

> No todos los estimadores se benefician de perturbar los datos de esta manera. Parece que los estimadores altamente no lineales, como los árboles, obtienen mayores beneficios. En el caso de los árboles generados mediante remuestreo (bootstrap), la correlación ρ tiende a ser baja (0.05 o menos es común, mientras que la varianza σ^2 no difiere mucho de la varianza del árbol original. Por otro lado, el bagging no altera las estimaciones lineales, como la media muestral (y, por lo tanto, tampoco su varianza); la correlación entre las medias generadas mediante remuestreo es aproximadamente del 50% (Hastie et al., 2001b)

La optimización de hiper-parámetros es importante dado que según la literatura (Hastie et al., 2001a)., en el caso en el que el Random Forest se utiliza para clasificación el valor por default de m es $\sqrt{p}$ y el tamaño mínimo de los nodos es de 1. No obstante, cómo se puede ver en este caso, se utiliza el modelo en otras condiciones y más adelante se observarán los resultados.

Otra condición importante del Random Forest es el uso de muestras excluidas del remuestreo. Por cada observación $z_i = (x_i, y_i)$ se construye Random Forest promediando aquellos árboles que corresponden a las muestras de remuestreo en las que z_i no aparece. Por lo anterior, a diferencia de muchos otros estimadores no lineales, los Random Forests pueden ajustarse en una sola secuencia realizando la validación cruzada durante el proceso. Una vez que el error se estabiliza el entrenamiento puede ser finalizado (Hastie et al., 2001a).

En cada división de cada árbol del Random Forest, se calcula la mejora del criterio de división, lo cual representa la importancia asignada a la variable utilizada en dicha división. Esta importancia se acumula individualmente para cada variable a lo largo de todos los árboles del bosque.Comparandolo con el boosting, este puede llegar a ignorar por completo algunas variables, mientras que el Random Forest no lo hace. Además, en un Random Forest existe una selección de variables candidatas para la división, lo que aumenta la probabilidad de que cualquier variable en particular sea incluida en el modelo. En cambio, en el boosting no se realiza dicha selección.

Los Random Forests utilizan las muestras excluidas del remuestreopara construir una medida de importancia de variable diferente, que aparentemente mide la capacidad predictiva de cada variable. Durante el crecimiento del árbol, se asignan las muestras excluidas del remuestreo a dicho árbol y se registra la precisión de las predicciones. Luego, se realizan permutaciones aleatorias de los valores de la variable $j-\acute{e}sima$ en las muestras y se recalcula la precisión. La disminución promedio en precisión como resultado de estas permutaciones se utiliza como medida de la importancia de la variable j en el Random Forest (Breiman, 2001b)

Cuando el número de variables es grande, pero la fracción de variables relevantes es pequeña, es probable que los bosques aleatorios no funcionen bien con m pequeños. Cuando aumenta el número de variables relevantes, el rendimiento de los bosques aleatorios es sorprendentemente robusto. Esta robustez se debe en gran medida a la relativa insensibilidad del coste de la clasificación errónea al sesgo y la varianza de las estimaciones de probabilidad en cada árbol.

iii) XGBoost

El boosting de árboles con gradientes es una técnica destacada en el campo del aprendizaje automático, conocida por ofrecer resultados de referencia tanto en problemas de clasificación como en problemas de regresión (Chen & Guestrin, 2016a). XGBoost se destaca como una implementación altamente eficiente de esta técnica, utilizando una secuencia de árboles de decisión que se mejoran entre sí para construir un modelo robusto y preciso (Chen & Guestrin, 2016a). Su impacto en desafíos de aprendizaje automático y minería de datos, como los organizados por Kaggle y KDDCup, es notable y demuestra su relevancia en la comunidad (Chen & Guestrin, 2016a).

La técnica destaca por su excepcional escalabilidad en diversos escenarios. El sistema muestra un rendimiento diez veces más rápido que las soluciones populares en una sola máquina y es capaz de escalar a conjuntos de datos masivos en configuraciones distribuidas o con limitaciones de memoria (Chen & Guestrin, 2016b). Esto se logra gracias a innovaciones como un algoritmo de aprendizaje de árboles que maneja eficientemente datos dispersos y un procedimiento de esquema de cuantiles ponderados respaldado teóricamente, que permite un manejo adecuado de los pesos de las instancias en el aprendizaje aproximado de los árboles (Chen & Guestrin, 2016b). Además, la computación paralela y distribuida acelera el proceso de aprendizaje, lo que resulta en una exploración más rápida del modelo (Chen & Guestrin, 2016b). Un aspecto destacado es que XGBoost aprovecha recursos de cómputo externos al núcleo, permitiendo a los científicos de datos procesar eficientemente conjuntos de datos de cientos de millones de ejemplos en una estación de trabajo (Chen & Guestrin, 2016a).

Dadas n observaciones, existen variables independientes x_i, y cada una de estas variables tiene m características, por lo tanto $x_i \in \mathbb{R}^m$ (Parsa et al., 2020). Para cada una de esas variables, existen

variables dependientes $y_i, y_i \in \mathbb{R}$. Un modelo de ensamble de árboles predice la variable dependiente y^i utilizando las variables independientes y K funciones aditivas (Parsa et al., 2020):

$$y^i = \sum_{k=1}^{K} f_k(x_i),\ f_k \in F$$

Aquí, f_k representa una estructura de árbol independiente con puntajes de las hojas, y F es el espacio de árboles. El objetivo es minimizar la ecuación (Parsa et al., 2020):

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

donde l es una función de pérdida, Ω es un término para penalizar la complejidad del modelo, y (Parsa et al., 2020):

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w_i\|^2$$

T es el número de hojas, y w_i es la puntuación de la i-ésima hoja. Al resolver las ecuaciones (1) - (3), se obtienen los valores óptimos para w_j^* y los valores correspondientes son (Parsa et al., 2020):

$$w_j^* = -\frac{\sum_{i \in I_j} \partial_{\hat{y}^{t-1}} l(y_i, \hat{y}^{t-1})}{\sum_{i \in I_j} \partial^2_{\hat{y}^{t-1}} l(y_i, \hat{y}^{t-1}) + \lambda}$$

$$\mathcal{L}^{\sim t}(q) = -\frac{1}{2}\sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} \partial_{\hat{y}^{t-1}} l(y_i, \hat{y}^{t-1})\right)^2}{\sum_{i \in I_j} \partial^2_{\hat{y}^{t-1}} l(y_i, \hat{y}^{t-1}) + \lambda} + \gamma T$$

Dado que, en la práctica, es difícil calcular este valor para todas las posibles estructuras de árboles, en su lugar se utiliza la siguiente fórmula (Parsa et al., 2020):

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i \in I_L} \partial_{\hat{y}^{t-1}} l(y_i, \hat{y}^{t-1})\right)^2}{\sum_{i \in I_L} \partial^2_{\hat{y}^{t-1}} l(y_i, \hat{y}^{t-1}) + \lambda} + \frac{\left(\sum_{i \in I_R} \partial_{\hat{y}^{t-1}} l(y_i, \hat{y}^{t-1})\right)^2}{\sum_{i \in I_R} \partial^2_{\hat{y}^{t-1}} l(y_i, \hat{y}^{t-1}) + \lambda} - \frac{\left(\sum_{i \in I} \partial_{\hat{y}^{t-1}} l(y_i, \hat{y}^{t-1})\right)^2}{\sum_{i \in I} \partial^2_{\hat{y}^{t-1}} l(y_i, \hat{y}^{t-1}) + \lambda}\right] - \gamma$$

Donde $I = I_L \cup I_R$

Aquí, I_L representa los conjuntos de instancias de los nodos izquierdos después de la división, mientras que I_R representa los conjuntos de instancias de los nodos derechos después de la división (Parsa et al., 2020). Como ventaja adicional, XGBoost, como algoritmo de árbol de decisión, no se ve afectado por la multicolinealidad (Parsa et al., 2020). Por lo tanto, incluso si dos variables capturan el mismo fenómeno en un sistema, ambas pueden ser conservadas.

Varios parámetros deben ser optimizados para evitar el sobreajuste y exceso de complejidad en modelo. El número de iteraciones determina la cantidad de árboles utilizados, mientras que la profundidad máxima controla el nivel de detalle de cada árbol (Parsa et al., 2020). La selección aleatoria de observaciones se realiza mediante el parámetro de submuestra, y la velocidad de

aprendizaje ajusta el impacto de cada árbol mediante la reducción de pesos (Parsa et al., 2020). Además, el modelo permite el submuestreo de columnas. Los términos de regularización L2 y L1, denominados "lambda" y "alpha" respectivamente, contribuyen a hacer que el modelo sea más conservador (Parsa et al., 2020).

XGBoost es un paquete de código abierto compatible con varias funciones de clasificación, así como funciones objetivo definidas por el usuario (Parsa et al., 2020). Está disponible en lenguajes como Python, R y Julia.

iv) K-Means

El autor (Parsa et al., 2020), define K-means como un método para agrupar un conjunto de datos en K clusters, donde K es un parámetro predefinido. El objetivo del algoritmo es agrupar datos sin etiquetar y minimizar la suma de las distancias al cuadrado entre cada punto de datos y su centroide asignado, lo que se conoce como la "inercia" del clustering.

El algoritmo K-means, que MacQueen (1967) describe en su paper, es una estrategia iterativa que comienza con una inicialización aleatoria de los K centroides y luego realiza dos pasos iterativos hasta que se alcanza la convergencia:

- Asignación de los puntos de datos a los clusters más cercanos según la distancia Euclidiana.
- Actualización de los centroides de cada cluster como la media de los puntos de datos asignados a él en la etapa anterior.

El proceso se repite hasta que la inercia del clustering no cambia significativamente entre iteraciones, lo que indica que se ha alcanzado una solución estable (MacQueen, 1967).

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^j - c_j \right\|^2$$

Donde:

J es la función objetivo
K = número de clusters
N = número de observaciones
C = Centroide

Según la literatura consultada, el algoritmo de K-Means es el algoritmo de agrupamiento particional más utilizado, por las siguientes razones (Celebi et al., 2013):

1. Es conceptualmente simple y fácil de implementar.
2. Es muy versátil, ya que la mayoría de sus parámetros pueden modificarse.
3. Su complejidad temporal es lineal en N, D, y K (en general, D << N y K << N).
4. Su complejidad de almacenamiento es lineal en N, D, y K.
5. Se garantiza convergencia a una tasa cuadrática.
6. Finalmente, es invariable ante el ordenamiento de los datos.

A pesar de estas características, este algoritmo también presenta algunas desventajas importantes, como lo son (Celebi et al., 2013)

1. Sólo puede detectar cúmulos hiperesféricos compactos que estén bien separados. Esto se puede mejorar, usando una distancia más robusta, como por ejemplo la de Mahalanobis.
2. Debido a que en su planteamiento básico usa la distancia Euclideana al cuadrado, es muy sensible al ruido y a los puntos atípicos.
3. Por su naturaleza de gradiente descendente, normalmente converge a un mínimo local de la función de criterio. Y por esta misma razón, es muy sensible a la selección de los centros iniciales.

De acuerdo con esto, las desventajas 2 y 3 pueden solucionarse usando métodos de inicialización adaptativos (IM por sus siglas en inglés), dentro de los que se encuentran ejemplos de Métodos de inicialización de complejidad de temporal Lineal, Loglineal, Cuadrática. También métodos de división binaria, división binaria de búsqueda dirigida, método global de K-means, métodos basados en metaheurísticas como el recocido simulado y los algoritmos genéticos, entre otros (Celebi et al., 2013).

3. Desarrollo metodológico

3.1 Entendimiento del problema, pregunta de negocio o hipótesis

Para desarrollar este proyecto, nos enfocaremos en el caso específico de DelcaVideography, un canal educativo creado en Medellín, Colombia en el año 2014.

A diciembre de 2022 contaba con 112,000 suscriptores, se encontraba en la posición 723 del ranking de canales de YouTube en Colombia y en el puesto 1,432 del ranking de canales educativos en YouTube (*Social Blade*, n.d.)

Por cada contenido que se publica, se invierten 8 horas en producción en promedio, para ser visto por aproximadamente entre el 0.1% y el 0.3% de sus suscriptores. Adicionalmente, el ingreso del canal depende en gran medida del éxito de sus videos.

Por las razones anteriores el dueño del canal DelcaVideography plantea la necesidad de responder a la pregunta ¿cómo saber si un video que publicará será exitoso o no?

Para esto, con base en el criterio experto, iniciamos con definir qué se considera un video exitoso, así:

Un video es exitoso si su porcentaje de clics de las impresiones es mayor a 3 (publicaciones entre el 16 de febrero de 2021 y 30 de agosto de 2022).

$$\begin{cases} \text{exitoso} & \text{si } i > 3 \\ \text{No exitoso} & \text{en otro caso} \end{cases}$$

Esta pregunta, decidimos resolverla con un modelo de clasificación, el cual detallaremos más adelante.

Si bien no se conocen todas las variables que influyen en el algoritmo de recomendación de YouTube, se sabe que las primeras horas de publicación de un video son claves para su éxito futuro (teniendo en cuenta las estadísticas del canal y la bibliografía consultada). Por esto, el hecho de apoyar en la validación del éxito de los videos publicados con base en el porcentaje de visualización ayudará al canal a tener más impresiones (recomendaciones) por parte de YouTube y por lo tanto mayores ingresos para el canal.

3.2 Entendimiento y Preparación de los datos

La información usada proviene de los videos publicados entre el 16 de febrero de 2021 y el 30 de agosto de 2022. Para extraer una parte de la información se desarrolló un web scraping para extraer información pública de los videos. El resto de las variables corresponden a estadísticas privadas y que solo pueden ser accedidas con la información del creador de contenido.

Con el objetivo de afinar y seleccionar las variables relevantes, se hace una ingeniería de descriptores siguiendo los siguientes pasos:

1) **Eliminación de variables duplicadas:** al trabajar con dos fuentes de datos diferentes se obtienen variables duplicadas. En este caso se eliminan *'T√≠tulo del v√≠deo', 'Hora de publicaci√≥n del v√≠deo', 'Mostrado en el feed', 'Vistos (frente a saltados) (%)', 'fecha_publicacion', 'likes', 'dislikes', 'viewCount', 'tus_ingresos_estimados_(usd)', 'impresiones' y 'porcentaje_de_clics_de_las_impresiones_(%)'*.

2) **Eliminación de variables nulas numéricas**: se eliminan las variables que por su naturaleza no tienen información y/o no es posible imputarles un valor. Corresponden a *'Clics por tarjeta mostrada (%)', 'Me gusta (vs. No me gusta) (%)', 'Clics por elemento de pantalla final mostrado (%)' y 'Clics en teaser por teaser de tarjeta mostrado (%)'*.

3) **Eliminación de variables tipo objeto:** Con el objetivo de realizar un análisis de correlaciones y contar solo con variables numéricas, se eliminan las variables tipo objeto.

Con la aplicación de los pasos descritos anteriormente, fue posible reducir de 61 a 16 las variables que serán utilizadas en los modelos. Para una lista completa de las variables inicialmente consideradas, se puede remitir al **Anexo 1**.

Finalmente, se eliminan filas con registros nulos y se pasa de 108 a 100 videos.

Las variables obtenidas son:

Variable	Descripción

Suscriptores ganados	Número total de veces que los usuarios se han suscrito a este canal en la región y en el intervalo de fechas seleccionados
Tiempo de visualizaci√≥n (horas)	Horas totales estimadas de visualización de tu contenido por parte de tu audiencia.
Me gusta	Número de *Me gusta* del video
Clics en teaser de tarjeta	Media de clics de un teaser por impresión. Sirve para medir la frecuencia con la que los usuarios hacen clic en un teaser después de verlo.
Tarjetas mostradas	Número de veces que se ha mostrado una tarjeta. Una sola tarjeta puede tener una impresión por visualización como máximo.
Clics en elementos de pantalla final	Número de veces que se ha hecho clic en un elemento de la pantalla final.
suscriptores	Número de suscriptores del canal
Densidad_Publicitaria	$(\frac{Duración\ Video\ en\ Minutos}{Cantidad\ de\ Publicidades})^{-1}$
Porcentaje de clics de las impresiones (%)	Visualizaciones por impresiones mostradas. Sirve para medir la frecuencia con la que los usuarios han visto un video después de ver una impresión.
Duracion Minutos	Duración en minutos del video
Comentarios a√±adidos	El número de comentarios que se han publicado en el video o canal. Puede incluir los comentarios eliminados y los mensajes del chat directo.
Porcentaje medio visto (%)	Porcentaje medio visto de un video por la audiencia.
rating	Calificación de 1 a 5 dada a un video por su relación *Me gusta - No me gusta*
dia_semana	0: lunes, 1: martes, 2: miércoles, 3: jueves, 4: viernes, 5: sábado, 6: domingo
consecutivo_tema	0: Archivo CorelDraw, 1: Características CorelDraw, 2: Consejos CorelDraw, 3: Curso CorelDraw
No me gusta	Número de *No me gusta* del video

3.3 Análisis exploratorio de datos

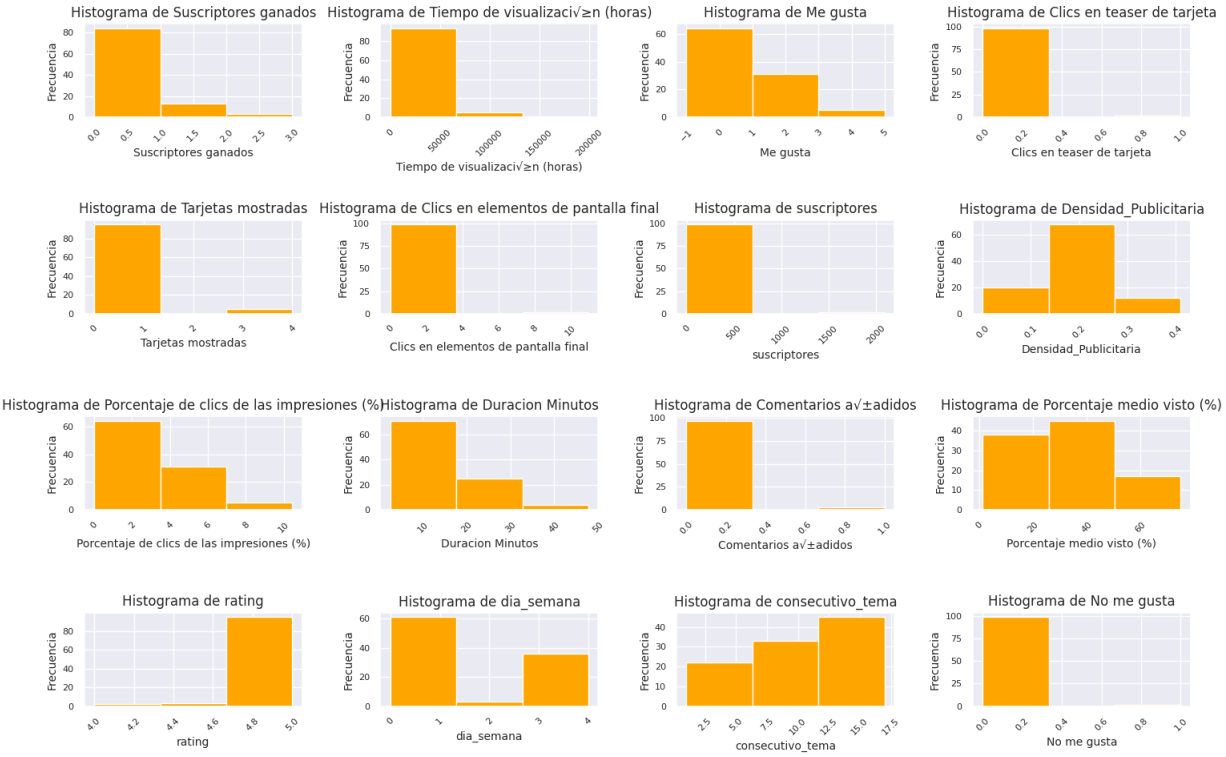
Inicialmente se consolida la información de los datos obtenido a priori, si bien una vista inicial no permite obtener conclusiones, sí permite empezar a abordar el problema identificando diferencias de escala entre las variables y demás.

	count	mean	std	min	25%	50%	75%	max
Suscriptores ganados	100	0,21	0,56	0	0	0	0	3
Tiempo de visualizaci√≥n (horas)	100	14317,03	33172,3	0	0,24	0,62	15754,75	199551
Me gusta	100	0,54	0,97	-1	0	0	1	5
Clics en teaser de tarjeta	100	0,02	0,14	0	0	0	0	1
Tarjetas mostradas	100	0,16	0,71	0	0	0	0	4
Clics en elementos de pantalla final	100	0,22	1,15	0	0	0	0	11
suscriptores	100	43,47	211,73	-4	4	8,5	20,75	2090
Densidad_Publicitaria	100	0,18	0,1	0	0,17	0,21	0,24	0,41
Porcentaje de clics de las impresiones (%)	100	3,17	1,94	0	1,91	2,92	4,2	10,45
Duracion Minutos	100	14,77	7,75	2,57	9,02	13,32	18,46	47,87
Comentarios a√±adidos	100	0,03	0,17	0	0	0	0	1
Porcentaje medio visto (%)	100	32,27	18,02	1,31	20,41	32,25	41,21	75,08
rating	100	4,91	0,14	4	4,89	4,94	5	5
dia_semana	100	1,71	1,04	0	1	1	3	4
consecutivo_tema	100	10,68	5	1	7	10	16	17
No me gusta	100	0,01	0,1	0	0	0	0	1

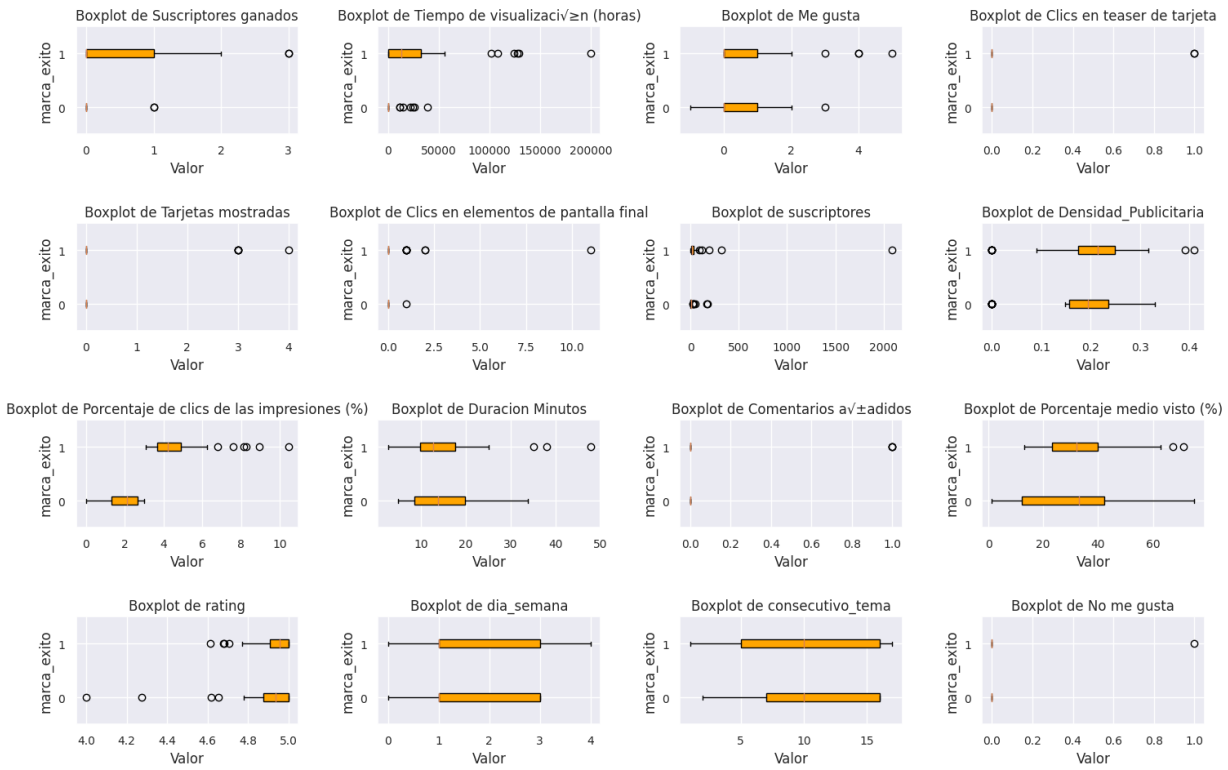
Se puede observar como las escalas de las variables ‘Tiempo de visualización en horas', ‘RPM (USD)’ y ‘Suscriptores’ tienen una escala muy diferente a las demás variables, lo cual lleva a proponer que,

para el procesamiento adecuado de los modelos, se realice la correspondiente estandarización de datos.

Continuando con el análisis, se construyen histogramas para todas las variables consideradas en el ejercicio. En ningún caso se puede ver que alguna de ellas siga un comportamiento siquiera parecido a una distribución normal. Lo anterior da cabida a la aplicación de técnicas no paramétricas en el análisis y ejecución de los modelos

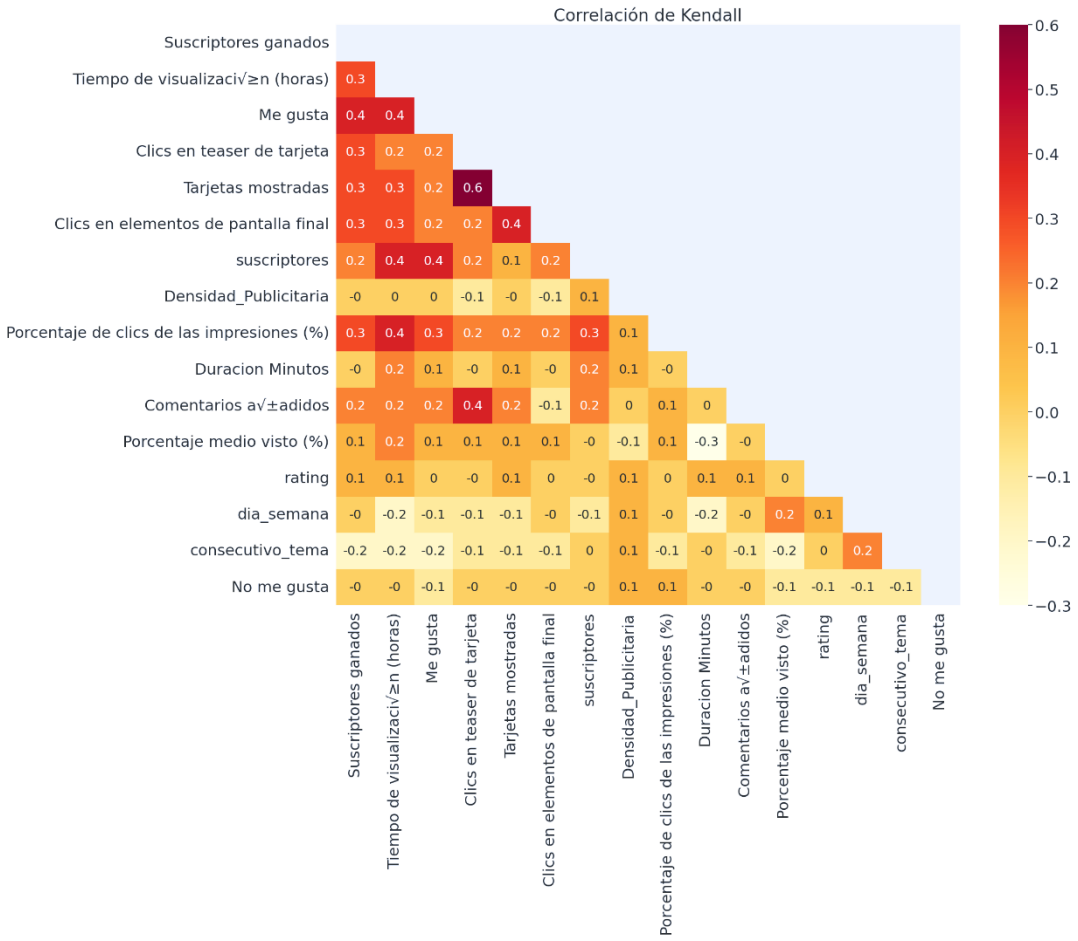


Dentro del análisis descriptivo de los datos es posible encontrar como los boxplot de las variables analizadas respecto a la variable *'marca_éxito'* parecen mostrar en algunos casos un comportamiento diferente según el label de dicha marca. Inicialmente esto ofrece la posibilidad de encontrar variables que ayuden a identificar de manera más precisa el éxito de los videos en el futuro. Entre las variables que presentan dicho comportamiento marcadamente están: *suscriptores ganados, tiempo de visualización (horas), clics de las impresiones, duración en minutos y porcentaje medio visto.*



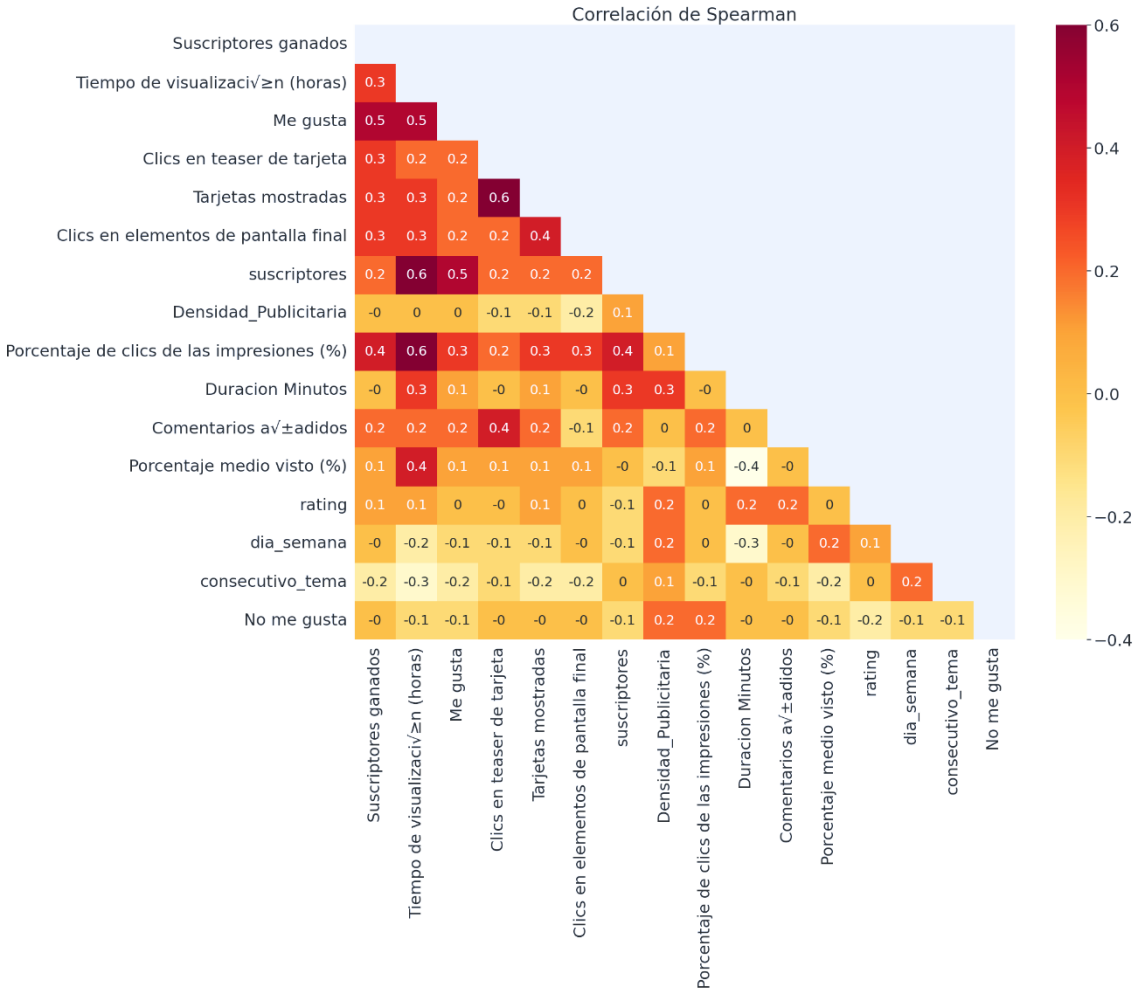
Otro de los elementos evaluados en el análisis descriptivo de los datos son las correlaciones de los datos. En este caso, como no se está tratando con datos que aparentemente siguen una distribución normal, se utilizaron las correlaciones de Kendall y Spearman para este análisis. La correlación de Kendall es una medida no paramétrica de correlación, es decir, no maneja ningún supuesto sobre la distribución de los datos y mide la fuerza y la dirección de la relación entre dos variables ordinales. El coeficiente de correlación de Kendall puede variar en un rango de -1 a 1. Un valor de 1 indica una correlación positiva perfecta, un valor de -1 indica una correlación negativa perfecta, y un valor de 0 indica que no hay correlación (Minitab, 2023).

Se puede observar como la correlación más alta entre las variables del problema estudiado es de 0.6, es decir, las correlaciones son moderadas en términos generales dentro del dataset. Adicionalmente la correlación más alta o baja no se da entre las variables que inicialmente se podrían catalogar como las más significativas para el problema.



Buscando profundizar en el análisis se exploró también el cálculo de la correlación de Spearman la cual es una medida no paramétrica de correlación de rangos que mide la fuerza y dirección de la relación monótona entre dos variables. Recibe su nombre en honor a Charles Spearman, quien la desarrolló en 1904. La monotonicidad significa que las variables tienden a aumentar o disminuir juntas, pero no necesariamente a una tasa constante. Por ejemplo, existe una relación monótona entre la altura y el peso, pero la relación no es lineal. El coeficiente de correlación de Spearman puede variar en un rango de -1 a 1. Un valor de 1 indica una correlación positiva perfecta, un valor de -1 indica una correlación negativa perfecta y un valor de 0 indica que no hay correlación (Spearman, 1904).

Con esta otra medida de correlación se sigue sin poder observar una fuerte relación entre las variables a estudiar. Si bien hay más variables que tienen una correlación moderada (Suscriptores y porcentaje de clic de las impresiones con el tiempo de visualización en horas) Sigue sin ser concluyente este análisis de correlaciones e invita a explorar el problema con otros métodos que puedan dar más luces sobre la relación e importancia de las variables



3.4 Selección de variables por aprendizaje evolutivo

Para el caso de este trabajo se han seleccionado los algoritmos genéticos para abordar el problema de selección de variables. Los algoritmos genéticos son una técnica de búsqueda y optimización inspirada en la selección natural y la teoría de la evolución de Darwin. Se utilizan para encontrar soluciones aproximadas a problemas de optimización y búsqueda en problemas de gran escala o complejos.

En los algoritmos genéticos una población inicial de soluciones se crea aleatoriamente y luego se somete a operadores genéticos (crossover, mutación o selección) para producir nuevas soluciones en cada iteración. El proceso se repite durante varias generaciones hasta que se encuentra una solución satisfactoria o se alcanza un criterio de parada predefinido (Melanie, 1999).

Para el presente problema se determinó como genoma un array de booleanos, donde cada cromosoma esta dado por 'Falso' o 'Verdadero', lo cual determinará si una característica será incluida (Verdadero) o excluida (Falso).

Así pues, este algoritmo en específico genera una población inicial de N individuos con genomas de manera aleatoria, seguido se generan X regresiones logísticas con cada genoma y se toma la evaluación de accuracy promedio de estas regresiones como la evaluación de desempeño del

individuo, seguido se realiza un crossover de los mejores ‘padres’ cortando el genoma en un punto al azar, y generando así la misma evaluación de métrica para este nuevo ‘hijo’, si es mejor que el peor de los individuos en la población, este ‘hijo’ pasa a reemplazar el peor.

Por otra parte, se inicia una evaluación de que se dé un evento al azar, esto para determinar si se da una mutación o no del mejor individuo o generación de un nuevo individuo por completo, para evitar que la población se quede en un óptimo local y no mejore con las generaciones.

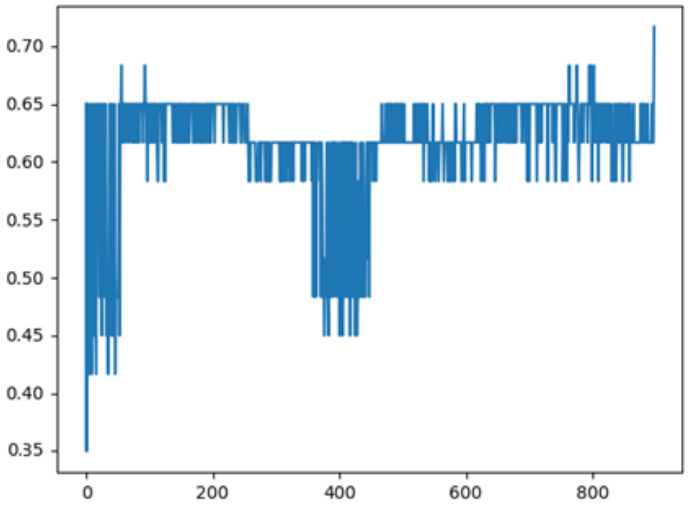
Finalmente se determinan 2 criterios de parada, para el entrenamiento del modelo, primero un numero límite de generaciones por si la iteración nunca llega o no es posible llegar a igualar o superar una métrica objetivo, dicho esto, esa métrica objetivo (para nuestro ejercicio determinado como Accuracy mayor o igual a 0.75) será el segundo criterio de parada, y una vez el mejor individuo de la población alcance dicha medida se tomara como la combinación de características que maximiza la métrica para la regresión logística.

Ya que el objetivo del modelo evolutivo es encontrar las características optimas a seleccionar para maximizar la métrica de accuracy de un modelo de árbol de decisión para los datos de YouTube, los cuales son datos etiquetados entre ‘exitosos’ y ‘no exitosos’, en otras palabras, una etiqueta binaria.

Gracias a que las etiquetas de los datos están balanceadas, y tienen igual importancia, por lo tanto, la medida de accuracy puede ser usada de manera certera para medir el modelo, así mismo la ecuación que puede ser usada para el cálculo de la métrica está dada por:

$$Accuracy = \frac{Número\ de\ predicciones\ correctas}{Total\ de\ predicciones}$$

Al corer este proceso vemos que se llega a una solución incumbente después de 1000 generationes con un Accuracy del 0.7 y un genoma [True, True, True, True, False, False, True, True, True, True, False, False, False, False, False] el cual hace referencia a usar las variables: ['Suscriptores ganados', 'Tiempo de visualizacion (horas)', 'Me gusta', 'Clics en teaser de tarjeta', 'suscriptores', 'Densidad_Publicitaria', 'Duracion Minutos', 'Comentarios a√±adidos']



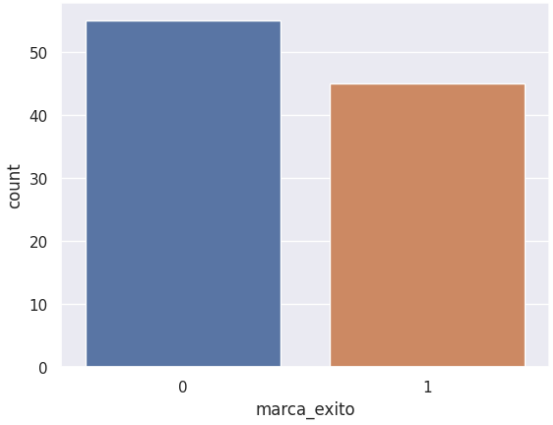
En dicha grafica vemos el accuracy en el eje Y y las generaciones en el eje X, podemos observar cómo los mejores individuos de la población comienzan en una métrica de aproximadamente 0.35 pero mejoran a través de las generaciones, aunque en ciertos momentos se evidencia como la población cae en óptimos locales durante múltiples generaciones, finalmente logra salir de estos por medio de mutaciones y crossovers hasta alcanzar una métrica final del 0.70.

Aun así, este no mejora el resultado obtenido por los métodos aplicados, esto puede darse por la definición de mutaciones y crossovers las cuales pueden mejorarse para alcanzar resultados iguales o mejores a los resultados finales.

4. Selección de modelos, entrenamiento y evaluación de métricas

Partiendo de los datos con la etiqueta de videos exitosos y no exitosos explicada anteriormente y construida con el criterio experto del dueño del canal, concluimos que estamos frente a un problema de aprendizaje supervisado. Tenemos unos datos de entrada que son las variables descritas en la sección anterior y la salida que es la marca de 1 y 0 de éxito de los videos.

Para la variable dependiente de marca de éxito, las etiquetas de los datos se encuentran balanceadas, el 45% de los videos son exitosos (55% no exitosos).

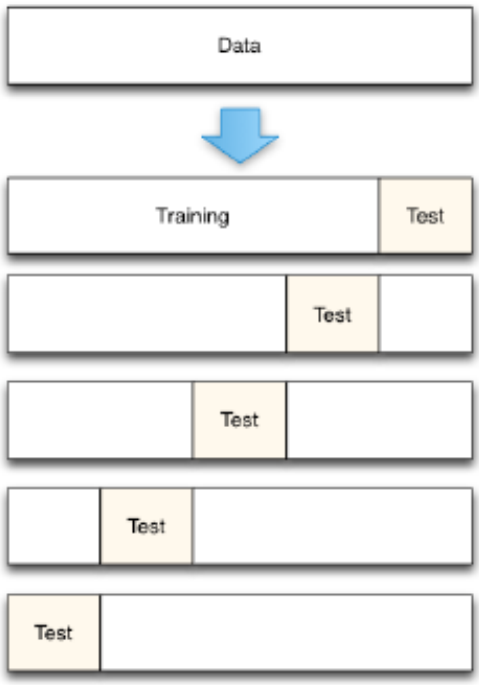


En este caso el objetivo de los modelos es la clasificación, permitiendo la implementación de los modelos mencionados en el marco teórico. Estos son: el algoritmo K means, Random Forest, regresión logística y árbol de decisión.

En la parte de entrenamiento de los modelos, quedamos con un dataset de 100 observaciones con lo cual usamos el 30% para testeo (30 observaciones) y 70% para entrenamiento (70 observaciones).

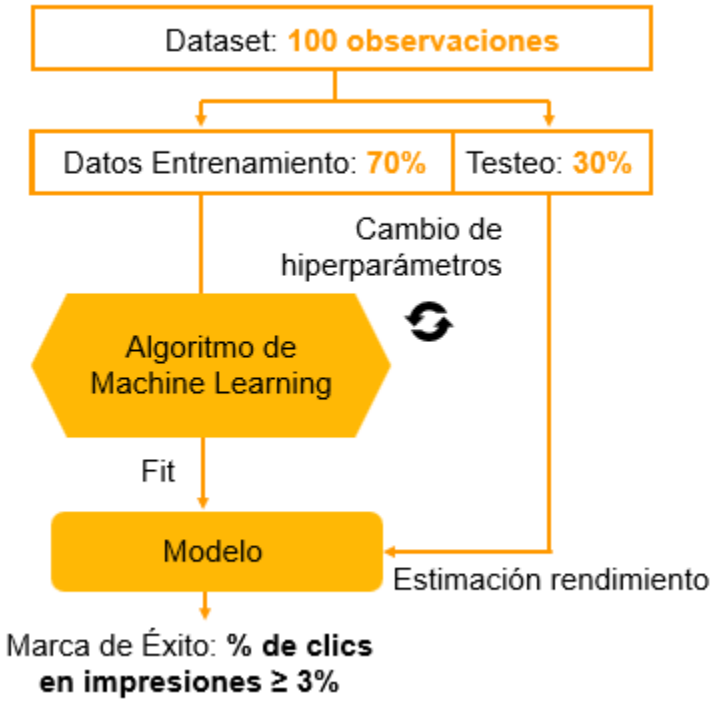
También se utilizó la validación cruzada con los porcentajes de testeo y entrenamiento definidos previamente, con el objetivo de entrenar y testear con diferentes submuestras del conjunto de datos. En este caso generamos 5 folds mediante la función StratifiedKFold del submódulo de model_selection de la librería de Sklearn. Esta función garantiza que cada muestra de datos contenga las clases balanceadas, esto es fundamental para resolver un problema de clasificación.

Esta validación cruzada nos genera una lista de scores con lo cual podemos ver el mínimo, máximo y el score mediano que ese obtiene con el conjunto de datos.



Fuente: (mathworks, 2022)

Vale la pena aclarar que cada modelo se entrenó con sus parámetros por default, y luego se reentrenaron los modelos, pero ajustando los hiperparámetros de cada algoritmo para encontrar la mejor combinación. Este proceso nos permite de forma iterativa mejorar considerablemente el modelo.



Fuente: Elaboración propia

4.1 Regresión Logística

Se procede a ajustar los modelos con los conjuntos de datos de entrenamiento con la librería sklearn y validar con el conjunto de testeo mediante métricas como la matriz de confusión, f1-score, recall y accuracy. Para el caso de este modelo lineal los datos fueron normalizados mediante el StandarScaler que lo hace es llevar todo a una misma escala mediante la media y la desviación estándar de cada campo del conjunto de datos.

En el caso de la regresión logística sin optimizar hiperparámetros obtenemos unos scores de:

- Accuracy train: 0.81
- Accuracy test: 0.76

Con la validación cruzada el accuracy estaría entre 0.64 y 0.78 con un k-fold de 5, con una mediana de 0.64.

Optimizando hiperparámetros

- Penalidad**:** ["l1","l2", "elasticnet", "none"]
- Constante**:** entre 0 y 3 100 datos
- Solver (forma de optimización)**:** ["liblinear", "sag", "saga"]

El mejor estimador es**:** Best estimador: {'solver': 'saga', 'penalty': 'l1', 'C': 1.34}

- Testing accuracy train: 0.81
- Testing accuracy test: 0.83

Con esta regresión logística con regulación L1 (LASSO) y constante de 1.34 mejoramos considerablemente las métricas del modelo.

	precision	recall	f1-score	support
0	0.83	0.95	0.89	21
1	0.83	0.56	0.67	9
accuracy			0.83	30
macro avg	0.83	0.75	0.78	30
weighted avg	0.83	0.83	0.82	30

La precisión, definida como la proporción de verdaderos positivos entre todos los valores clasificados como positivos. En este caso, la precisión del modelo para la 'marca éxito' 0 es del 83% y para la 'marca éxito' 1 es del 83%

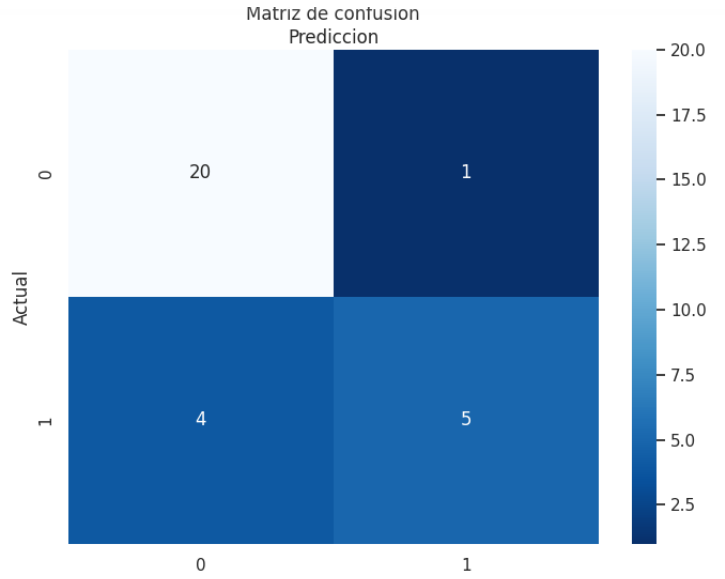
El Recall, que es la proporción de verdaderos positivos entre todos los valores verdaderos de la clase indica para la 'marca éxito' 0 un valor del 95%, lo que significa que de todos los valores verdaderos de la 'marca éxito' 0 en el conjunto de datos, el modelo ha identificado el 95% de ellos. Análogamente, para la 'marca éxito' 1 el recall es de 56%

El F1-score combina la precisión y el recall del modelo en una sola métrica. El valor F1-score para la 'marca éxito' 0 es del 89% y para la 'marca éxito' 1 es del 67%. Cuanto más alto sea el valor de F1-score, mejor será el rendimiento general del modelo.

Support es el número de instancias en el conjunto de datos que pertenecen a cada clase. En este caso, hay 21 instancias de la 'marca éxito' 0 y 9 instancias de la 'marca éxito' 1.

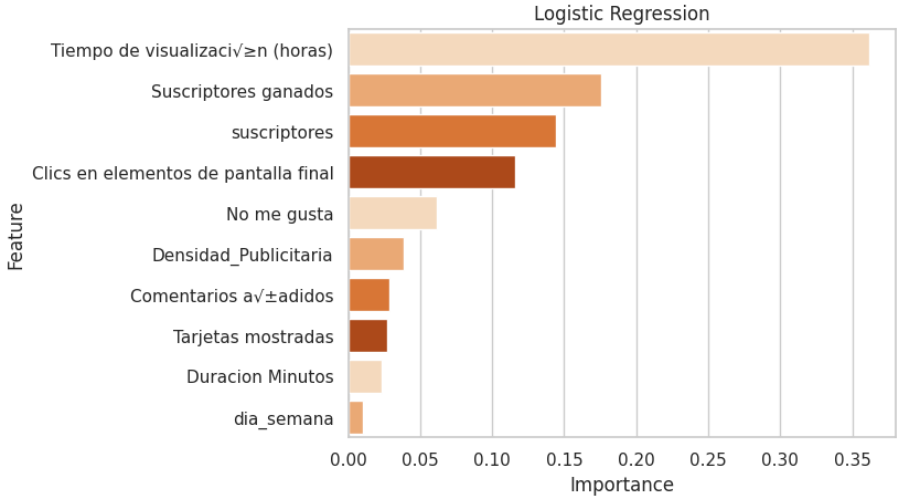
Estos resultados indican que el modelo tiene un buen rendimiento en términos de precisión para ambas clases, mientras que su rendimiento en términos de recall es mejor para la 'marca éxito' 0. El valor de F1-score indica un rendimiento mejor para los videos no exitosos.

Analizando la matriz de confusión:

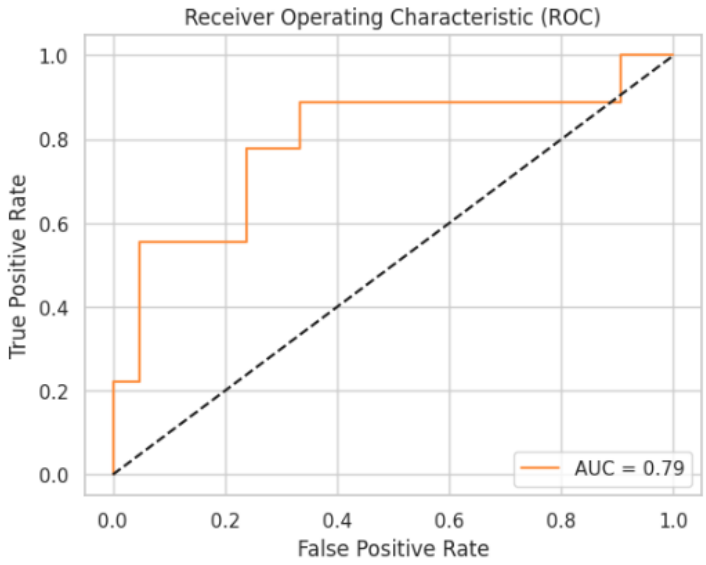


En el caso de la matriz de confusión para la predicción se tiene que la ‘marca éxito’ 0 clasifica adecuadamente 20 (66.6%) negativos mientras que para la clase de videos exitosos clasifica correctamente 5 videos (16.6%), globalmente se tiene un accuracy del 83%, es una métrica con un buen desempeño de clasificación.

Si analizamos el valor absoluto del peso de cada parámetro sobre el total de parámetros se obtiene la siguiente importancia de variables, la cual va muy acorde con lo que realmente ocurre en el día a día de los creadores de contenido:



Para esta métrica, Un AUC de 0.79 indica que el modelo tiene un buen rendimiento para discriminar entre las clases positivas y negativas. Cuanto más cercano esté el AUC a 1, mejor será el rendimiento del modelo. En este caso, un AUC de 0.79 indica que el modelo tiene una tasa de verdaderos positivos relativamente alta y una tasa de falsos positivos relativamente baja, lo que sugiere que es capaz de hacer predicciones precisas en general.



4.2 Árbol de decisión

En el caso del árbol de decisión sin optimizar hiperparametros obtenemos unos scores de:

- Accuracy train: 1
- Accuracy test: 0.66

Con la validación cruzada el accuracy estaría entre 0.42 y 0.92 con un k-fold de 5, con una mediana de 0.71.
Optimizando hiperparámetros

- max_depth: 1 a 25
- Criterio: ["gini", "entropy", "log_loss"]
- min_samples_split: 1 a 10
- min_samples_leaf: 1 a 10

El mejor estimador es**:** Best estimador: {'min_samples_split': 2, 'min_samples_leaf': 6, 'max_depth': 23, 'criterion': 'entropy'}

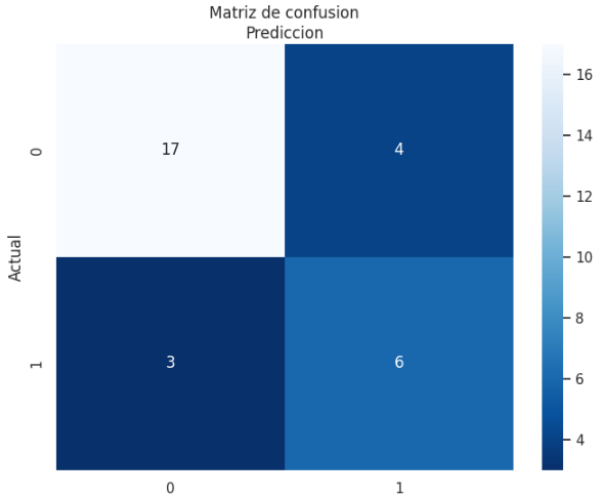
- Testing accuracy train: 0.85
- Testing accuracy test: 0.76

Con este modelo decision tree con optimzación de hiperparámetros se reduce el problema de overfitting que se presentaba en el modelo sin optimizar que generaba un accuracy 1 de los datos de entrenamiento.
A continuación, se presentan las métricas del modelo:

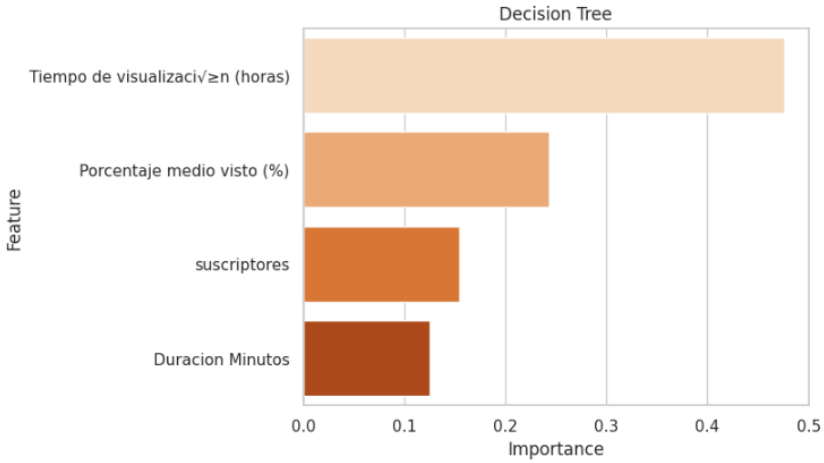
	precision	recall	f1-score	support
0	0.85	0.81	0.83	21
1	0.60	0.67	0.63	9
accuracy			0.77	30
macro avg	0.72	0.74	0.73	30
weighted avg	0.77	0.77	0.77	30

En general, estas métricas indican que el modelo tiene una buena precisión, pero podría haber un margen de mejora en la capacidad de identificar correctamente los videos exitosos (clase "1"), ya que el recall y el F1-score son un poco más bajos en comparación con la clase "0".

La precisión global del modelo que se observa en la matriz de confusión, es decir, la proporción de instancias clasificadas correctamente en general. En este caso, la precisión es 0.77, lo que significa que el modelo clasifica correctamente el 77% de los videos en total.



Por su parte, este modelo tiene en cuenta las siguientes variables en el feature importance:



4.3 Random Forest

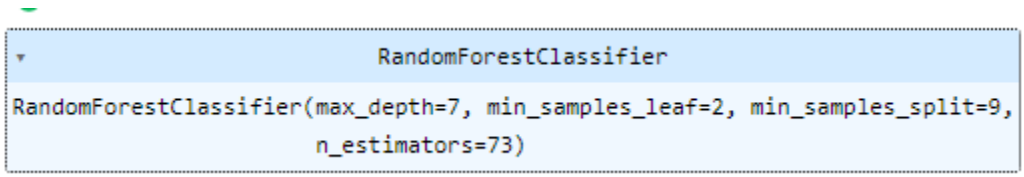
Se procede a ajustar los modelos con los conjuntos de datos de entrenamiento con la librería sklearn y validar con el conjunto de testeo mediante métricas como la matriz de confusión, f1-score, recall y accuracy. Para el caso de este modelo los datos fueron normalizados mediante el StandarScaler que lo hace es llevar todo a una misma escala mediante la media y la desviación estándar de cada campo del conjunto de datos.

Para poder utilizar este modelo, inicialmente se hace una optimización de hiper-parámetros a través de optimización bayesiana. Esta es una estrategia de diseño secuencial para la optimización global de funciones de caja negra que no asume ninguna forma funcional. Se utiliza generalmente para optimizar funciones costosas de evaluar. El hecho de que no necesite información previa del problema facilita que pueda aprender sin necesidad de realizar cambios. También, que no dependa de información a priori, generalmente proporcionada por un modelo o un experto, evita que exista un sesgo inicial que impida aprender a mejorar su función objetivo (Garnett, 2023).

Utilizando la librería *hyperopt* de Python, se crea una función objetivo para la optimización bayesiana que es la maximización de la función objetivo *accuracy* de un modelo de Random Forest. Esta optimización en este caso cuenta con un criterio de parada que son mil repeticiones y que busca optimizar la función objetivo en función de los siguientes hiper-parámetros:

- **n_estimators:** Este hiper-parámetro determina el número de árboles de decisión que se construirán en el bosque.
- **max_depth**: Este hiper-parámetro especifica la profundidad máxima permitida para cada árbol de decisión en el bosque. Una mayor profundidad puede permitir que los árboles capturen relaciones más complejas en los datos, pero también puede llevar a un sobreajuste.
- **min_samples_split:** Este hiper-parámetro establece el número mínimo de muestras requeridas para que un nodo interno se divida durante la construcción de un árbol. Si el número de muestras en un nodo es menor que este valor, no se realizará una división adicional.
- **min_samples_leaf:** Este hiper-parámetro establece el número mínimo de muestras requeridas para que un nodo hoja sea considerado válido. Si después de una división, alguna de las hojas resultantes tiene un número de muestras menor que este valor, se revertirá la división.

Luego de realizar las iteraciones correspondientes se obtienen los siguientes valores para los hiper-parámetros:

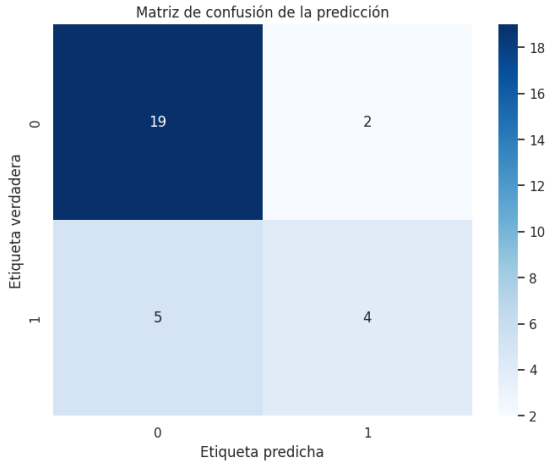


Con estos hiper-parámetros, más la utilización de cross validation con k=5 se corre el modelo y se obtienen los siguientes resultados:

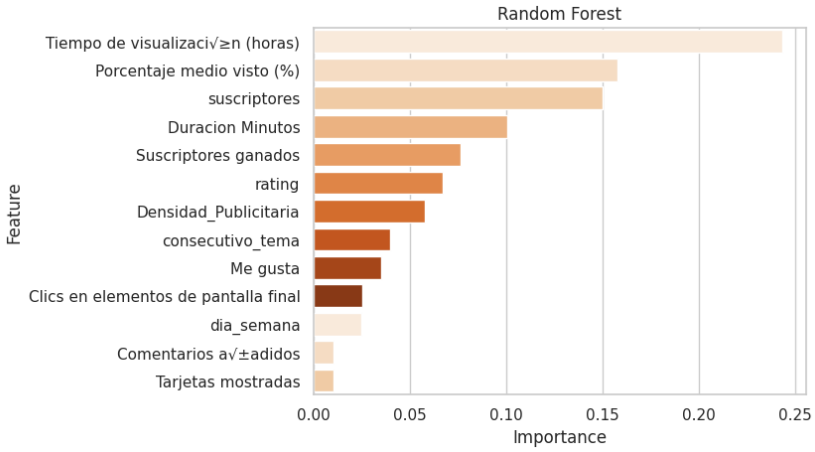
```
Accuracy: 0.7666666666666667
Precisión: 0.6667
Recall: 0.4444
F1-score: 0.5333
Error cuadrático medio (MSE): 0.2333
Área bajo la curva ROC (ROC-AUC): 0.8889
```

Se aclara que en diferentes corridas se obtienen diferentes hiper-parámetros y diferentes resultados. Experimentalmente se logró identificar que el mayor accuracy obtenido fue del 80%, sin embargo, nunca se superó este resultado.

La matriz de confusión obtenida muestra que donde mayor cantidad de errores se presentaron en la clasificación con los datos de testeo es en la identificación de videos exitosos que fueron predichos como no exitosos.

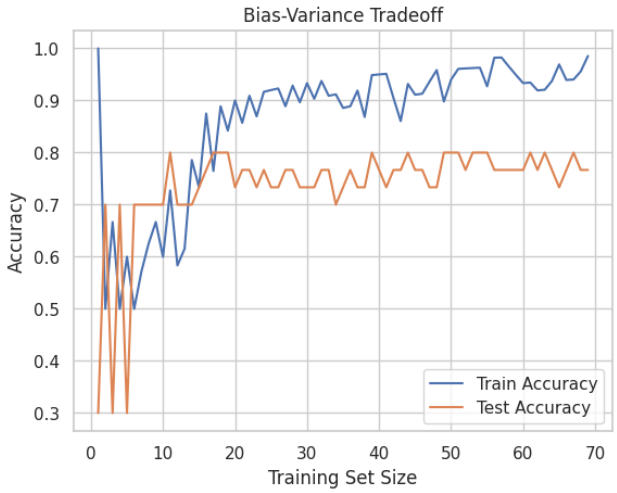


Este modelo da importancia a las variables que se muestran a continuación. Se resalta que las primeras cuatro variables coinciden con las de importancia en el resultado obtenido con el Decision Tree.



Dado que el modelo en cuestión tiene un alto puntaje de precisión en el conjunto de entrenamiento (0.98), pero un puntaje de precisión más bajo en el conjunto de prueba (0.78), podemos inferir lo siguiente:

i. Sesgo (Bias): El modelo muestra un bajo sesgo, ya que es capaz de ajustarse bastante bien a los datos de entrenamiento, lo que se refleja en la alta precisión en el conjunto de entrenamiento. Sin embargo, el sesgo no es el problema principal aquí, ya que el modelo tiene la capacidad de ajustarse y capturar las relaciones en los datos de entrenamiento.
ii. Varianza: El modelo muestra una varianza relativamente alta, ya que existe una discrepancia significativa entre el rendimiento en el conjunto de entrenamiento y el conjunto de prueba. La diferencia en las puntuaciones de precisión sugiere que el modelo está sobreajustando los datos de entrenamiento y no generaliza bien a nuevos datos.



4.4 K-means

Se aplicó un enfoque de agrupamiento mediante el uso de algoritmos de clustering, cuyo objetivo principal es agrupar un conjunto de datos en clusters o grupos homogéneos en base a sus características. Esta técnica permite obtener información valiosa sobre las relaciones y tendencias en los datos, así como identificar valores atípicos y simplificar la complejidad de estos.

A continuación, se presenta una breve descripción de K-Means:

K-Means es un método utilizado para agrupar un conjunto de datos en K clusters, donde K es un parámetro predefinido. El objetivo del algoritmo es minimizar la suma de las distancias al cuadrado entre cada punto de datos y su centroide asignado, lo que se conoce como la "inercia" del clustering.

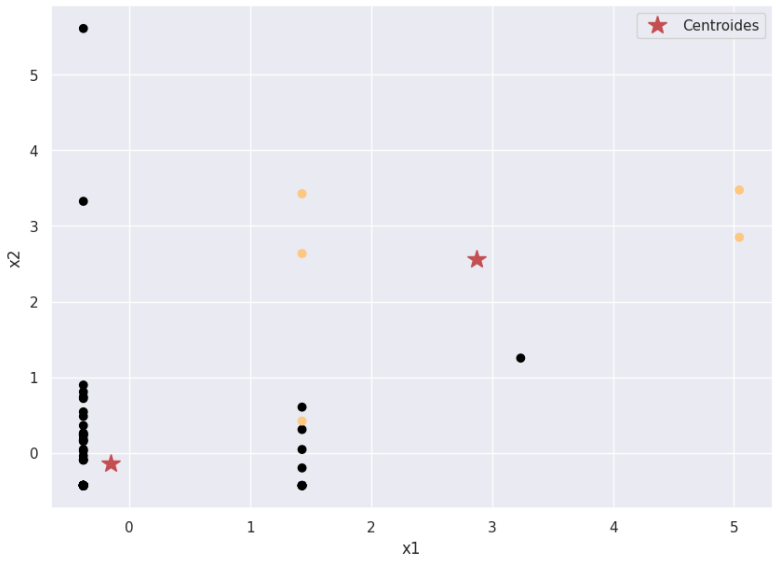
El algoritmo K-Means descrito por MacQueen en su artículo es una estrategia iterativa que consta de los siguientes pasos hasta alcanzar la convergencia:

- Asignación de los puntos de datos a los clusters más cercanos mediante el cálculo de la distancia Euclidiana.

- Actualización de los centroides de cada cluster como la media de los puntos de datos asignados a él en la etapa anterior.

Este proceso se repite hasta que la inercia del clustering no cambie significativamente entre iteraciones, lo que indica que se ha alcanzado una solución estable.

Así mismo para dar una mejora a dicho proceso y comparar si el desempeño mejora hasta ser un posible modelo se da una intervención no paramétrica a la distancia entre los puntos a la hora de definir centroides, utilizando la distancia de Mahalanobis en sustitución a la distancia euclídea.



Como se puede apreciar de manera gráfica, los datos no se prestan para este tipo de cauterización, también esto queda evidenciado en las métricas obtenidas tanto para la aplicación euclídea como de mahalanobis.

Para evaluar el modelo consideramos métricas como el El Silhouette score y el Davies-Bouldin score, ambas son válidas para evaluar la calidad de la agrupación en un modelo de clustering K-means.

El Silhouette score mide la similitud de cada punto con su propio cluster en comparación con otros clusters. El valor del Silhouette score varía entre -1 y 1, donde un valor más cercano a 1 indica que los puntos están bien agrupados y separados entre sí. En este caso, el valor de 0.58 indica una no tan clara división entre los clusters, este en la intervención de mahalanobis da un 0.14.

El Davies-Bouldin score mide la relación media entre la distancia intra-cluster y la distancia inter-cluster. Un valor más cercano a cero indica una mejor separación entre los clusters. En este caso, el valor de 1.42 sugiere que los clusters están relativamente bien separados, pero no hay una separación clara y distintiva, este en la intervención de mahalanobis da un 3.29.

4.5 XGBoost

La implementación de XGBoost se llevó a cabo en varias etapas. En un principio, se utilizó el modelo con los hiperparámetros predeterminados proporcionados por la librería XGBoost. Luego, se procedió a la optimización de los hiperparámetros, primero sin restricciones y luego estableciendo rangos adecuados para el problema en cuestión. Además, se incorporó el uso de validación cruzada para entrenar el modelo de manera más robusta y precisa.

4.5.1 Entrenamiento del modelo con hiperparámetros predeterminados

El modelo es inicialmente entrenado con los hiperparámetros predeterminados. Para ello se utiliza la siguiente información:

- Subsample: referido a la proporción de la muestra que será utilizada para entrenar cada árbol. Estos datos son seleccionados de forma aleatoria. El valor predeterminado es 1, queriendo decir que se usará el conjunto de datos completos para entrenar cada árbol.
- Colsample_bytree: utilizado para controlar la proporción de variables que será usada en cada árbol. El valor predeterminado es 1.
- ETA: es la velocidad de aprendizaje. Controla la contribución de cada árbol al modelo final, determinando cuánto se ajustan los valores de los parámetros en cada iteración del proceso de entrenamiento. El valor predeterminado es 0.3.
- Max_depth: este parámetro controla la longitud del camino más largo desde la raíz a la hoja. Esto es la profundidad. Este parámetro contribuye a regular la complejidad del modelo y a evitar el sobreajuste. El valor con el que se trabaja inicialmente es 6.

Adicional a lo ya mencionado, el parámetro 'eval_metric' permite especificar las métricas de evaluación para el entrenamiento del modelo. En este caso fueron implementadas:

- Log Loss: penaliza las predicciones incorrectas de forma más significativa.
- AUC (Area Under the Curve): evalúa la capacidad del modelo para distinguir entre las clases. Calcula el área bajo la curva ROC y da una medida del rendimiento general del clasificador. Un modelo con AUC cercano a 1 indica un modelo con buen rendimiento.
- Error: se calcula como # casos incorrectos / # casos totales.

XGBoost utiliza una estructura de datos llamada DMatrix. Es una estructura optimizada para almacenar los datos. Aquí, los datos son representados como una matriz dispersa.

Para el entrenamiento del modelo se transforman los datos a la estructura DMatrix, y se introducen los siguientes hiperparámetros:

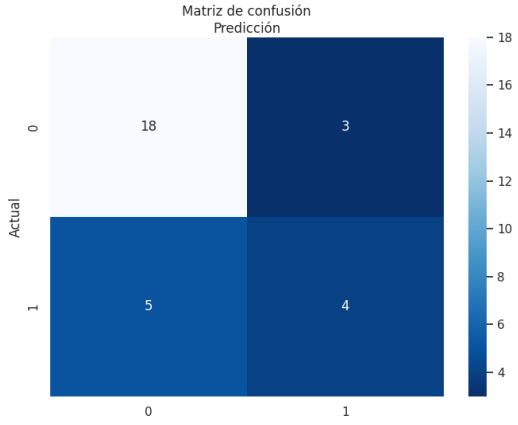
- N_estimators: se refiere al número de árboles que se construirán en el entrenamiento del modelo. Cada árbol se ajusta para corregir los errores del modelo anterior, permitiendo una mejora gradual. Un mayor número proporcionará un modelo más complejo, sacrificando eficiencia computacional y aumentando las posibilidades de sobreajuste. Inicialmente se contemplan 300 épocas.
- Early_stopping_round: permite detener el entrenamiento del modelo si no se observa una mejora en una métrica de evaluación específica en un número determinado de iteraciones

consecutivas. Esto es útil para para detener el entrenamiento cuando se alcanza un punto de convergencia donde la mejora adicional en la métrica de evaluación es mínima. Este hiperparámetro puede ser útil para evitar el sobreajuste.

Con esta configuración se obtiene en los datos de entrenamiento un logloss de 0.05937, un error del 0 y un AUC de 1. Por otro lado, en el conjunto de testeo se evidencian resultados de 0.4972, 0.23 y 0.87, respectivamente. A pesar de que el proceso se detuvo en la iteración 53, estos resultados podrían mostrar un problema de sobreajuste del modelo, al obtener métricas significativamente peores en el conjunto de testeo.

A pesar de que el accuracy se sitúa en 0.73, y el F1-Score general en 0.72, es posible identificar que el modelo final es mejor para la clasificación de videos no exitosos que de videos exitosos. En el caso de los videos exitosos, se obtiene un F1-Score de 0.5, con una precisión de 0.57 y un recall de 0.44. En la matriz de confusión también es posible observar la inhabilidad del modelo para clasificar adecuadamente los videos exitosos.

	precision	recall	f1-score	support
0	0.78	0.86	0.82	21
1	0.57	0.44	0.50	9
accuracy			0.73	30
macro avg	0.68	0.65	0.66	30
weighted avg	0.72	0.73	0.72	30



4.5.2 Optimización de hiperparámetros

Los hiperparámetros ya mencionados fueron optimizados en este paso:

- Subsample: un valor menor a 1 en este hiperparámetro puede tener dos efectos:
    - Reducción de varianza: una muestra aleatoria de menor tamaño en el entrenamiento de cada árbol reduce la variabilidad en la construcción de los árboles, reduciendo la varianza del modelo resultante. Esto ayudará a evitar el sobreajuste y a mejorar la capacidad de generalización del modelo.
    - Aumento en eficiencia computacional: menos tiempo y recursos computacionales son requeridos. Esto es especialmente útil en conjuntos de datos grandes, sin sacrificar rendimiento.
- Colsample_bytree: valores menores a 1 ayudan a reducir la correlación entre árboles, aumentando la capacidad de generalización del modelo y evitando el sobreajuste. De igual forma, puede aumentar la eficiencia computacional.

- ETA: un valor bajo implica que los ajustes en cada iteración son pequeños, esto hará el proceso más lento, pero más preciso. Una tasa de aprendizaje alta acelera el proceso, pero puede llevar al sobreajuste.

- Max_depth: un valor alto permitirá el aprendizaje de relaciones complejas en los datos de entrenamiento, sin embargo, conlleva un riesgo mayor de sobreajuste, especialmente cuando los datos son ruidosos o tienen características irrelevantes. Este parámetro es importante para controlar el trade-off sesgo-varianza.

- N_estimator: un número demasiado bajo puede llevar a un modelo que no sea lo suficientemente complejo para capturar los patrones de los datos. Por el contrario, un número demasiado alto podría llevar al sobreajuste.

Para este paso se usa la librería Hyperopt de optimización de hiperparámetros. Usa el enfoque bayesiano para buscar eficientemente en el espacio de hiperparámetros, y utiliza algoritmos como el árbol de búsqueda parzen (TPE). Se define una función de entrenamiento del modelo y se devuelven los parámetros con mejor accuracy para mil evaluaciones diferentes. Con esto se obtienen los siguientes hiperparámetros:

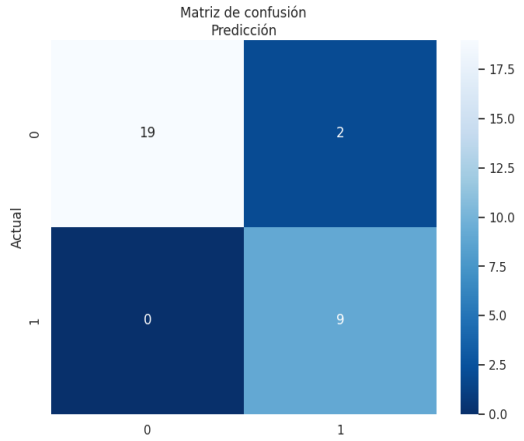
- Colsample_bytree: 0.9062
- ETA: 0.8761
- Max_depth: 8.0
- N_estimators: 631,
- Subsample: 0.7325

En la iteración 631 se logra en los datos de entrenamiento un logloss de 0.03932, un error de 0 y un AUC de 1. Para los datos de testeo se llega a un logloss de 0.386, error de 0.06667 y AUC de 0.92063. En las gráficas a continuación se puede evidenciar esto. Como es de esperarse las métricas del set de

testeo son un poco inferiores a las de testeo. Aún con esto, las métricas obtenidas son buenas.

En estas gráficas se observa que el modelo converge rápidamente a métricas aceptables para los datos de entrenamiento. En los tres casos aquí graficados, se muestra una mejoría entre las métricas de la primera y la última iteración. Sin embargo, no se aprecia una mejoría en cada paso del proceso que justifique el aumento en la complejidad del modelo. Es decir, la complejidad del modelo no necesariamente se ve traducida en un mejor rendimiento.

	precision	recall	f1-score	support
0	1.00	0.90	0.95	21
1	0.82	1.00	0.90	9
accuracy			0.93	30
macro avg	0.91	0.95	0.93	30
weighted avg	0.95	0.93	0.94	30



Al igual que con el primer modelo entrenado, es posible concluir este modelo es mejor para la predicción de videos no exitosos que de videos exitosos. El F1-score para la categoría 0 es 0.95, mientras que para la categoría 1 es 0.90. Sin embargo, este modelo tiene una métrica de accuracy en 0.93, y adecuados resultados en el F1-score de ambas categorías. Estos resultados indican una buena habilidad del modelo para capturar tantos los casos de videos exitosos como no exitosos.

Es necesario revisar el valor de la velocidad de aprendizaje, pues puede llevar a un sobreajuste del modelo.

A modo de experimento, se continua con la optimización de max_depth y n_estimators, manteniendo el resto de los hiperparámetros constantes. Se toman estos dos parámetros al ser de importancia para la complejidad del modelo, y la relación sesgo-varianza. Se determinan los siguientes parámetros:

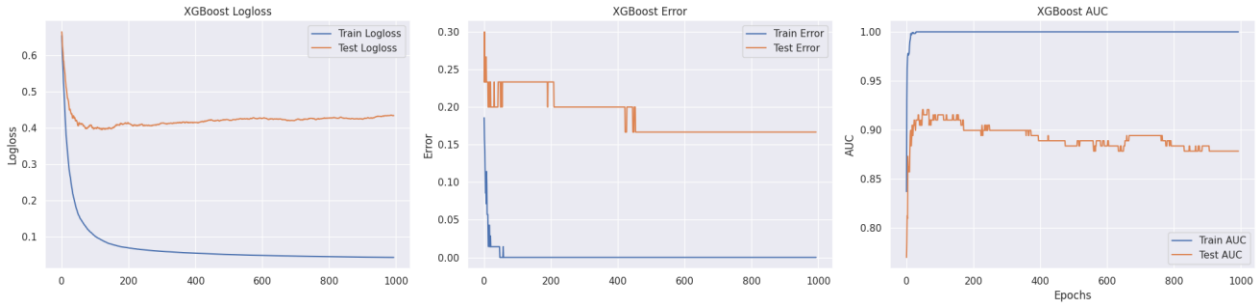
- Subsample: 0.8
- Colsample_bytree: 0.85
- ETA: 0.1

Estos valores se definen buscando evitar el sobreajuste del modelo. Se realizan 1000 evaluaciones para encontrar los mejores hiperparámetros, y con esto se obtiene:

- Max_depth: 7.0
- N_estimators: 994

En la última iteración del modelo se logra un logloss de 0.04255, error de 0 y AUC de 1 para el set de entrenamiento. Por el lado del set de testeo se alcanza un logloss de 0.43403, error de 0.16667 y AUC de 0.87831. En las gráficas a continuación, es posible observar que se logran buenas métricas de evaluación para el set de entrenamiento rápidamente. Sin embargo, no sucede lo mismo para el set

de testeo, e incluso, a medida que se aumenta la complejidad del modelo, aumenta el logloss del modelo y disminuye el AUC.



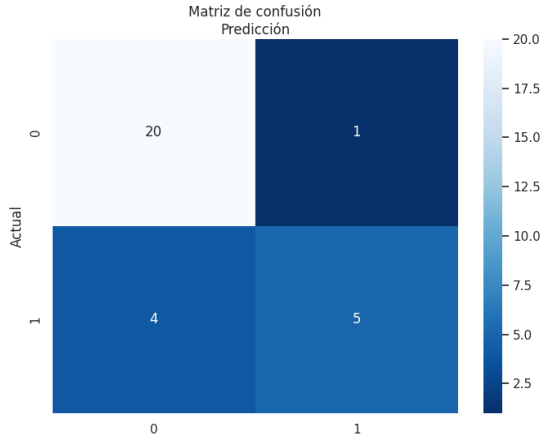
4.5.3 Optimización de hiperparámetros con Cross-Validation

Utilizando GridSearchCV se buscan los mejores valores para max_depth, subsample, colsample_bytree y eta. Para cada uno de ellos se define un espacio de búsqueda y se entrena el modelo con 5 folds. En este proceso se hallan los siguientes números:

- Colsample_bytree: 0.1
- ETA: 0.24
- Max_depth: 5
- Subsample: 0.9

Las métricas de evaluación se muestran a continuación:

	precision	recall	f1-score	support
0	0.83	0.95	0.89	21
1	0.83	0.56	0.67	9
accuracy			0.83	30
macro avg	0.83	0.75	0.78	30
weighted avg	0.83	0.83	0.82	30



En este caso se obtienen métricas que evidencian una habilidad del modelo para predecir adecuadamente ambas clases. Se tiene un f1-score de 0.82, siendo mejor para la clase de videos no exitosos que exitosos. Esto puede verse en la matriz de confusión.

4.5.4 Optimización de hiperparámetros con Cross-Validation y Regularización

A los hiperparámetros a optimizar se agregan los siguientes:

- Gamma: es la penalización aplicada para la creación de hojas en un árbol. Controla la cantidad mínima de reducción requerida en la pérdida de la función objetivo para que

una partición adicional en el nodo sea considerada. Un valor alto en este hiperparámetro lleva a una mayor regularización y ayuda a evitar el sobreajuste.

- Reg_alpha: corresponde con la regularización L1 (Lasso) aplicado a los pesos de las características de los árboles de decisión. Un valor más alto incrementa la regularización y contribuye a reducir la complejidad del modelo al penalizar los pesos más grandes. Esto puede resultar en modelos más simples y generalizables.

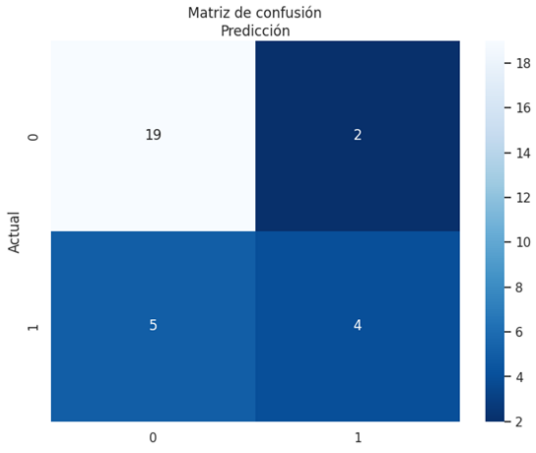
En este caso se obtiene:

- Subsample: 0.9
- Reg_alpha: 1
- Max_depth: 4
- Learning_rate: 0.08
- Gamma: 0.4
- Colsample_bytree: 0.5

Al entrenar el modelo con estos valores se logran las métricas que se muestran en la gráfica abajo. Como se observa, tanto el set de entrenamiento como el de testeo va mejorando sus métricas de evaluación con cada época (nuevo árbol creado). Para el set de entrenamiento se logra un logloss de 0.24133, un error de 0.02857 y un AUC de 0.99428. Para el set de testeo se llega a un logloss de 0.40153, error de 0.2333 y AUC de 0.91005.

Adicionalmente, el modelo entrega un accuracy de 0.77, con un f1-score general de 0.75. Como es posible ver, el modelo es mejor para la predicción de videos no exitosos que de videos exitosos. A

	precision	recall	f1-score	support
0	0.79	0.90	0.84	21
1	0.67	0.44	0.53	9
accuracy			0.77	30
macro avg	0.73	0.67	0.69	30
weighted avg	0.75	0.77	0.75	30

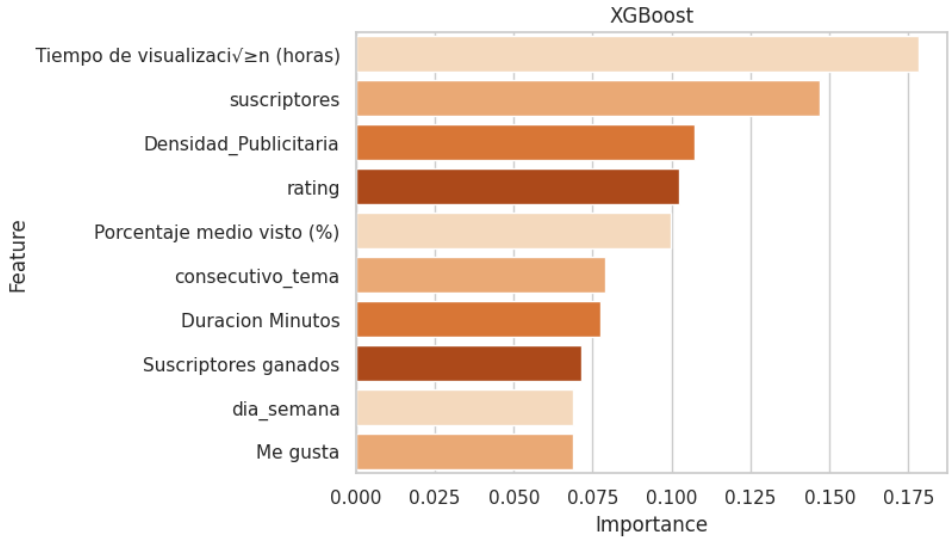


pesar de que el modelo cuenta con una precisión de 0.67 para la clase de videos no exitosos, su recall es de apenas 0.44 en esta clase. Esto indica que el modelo clasificó correctamente el 44% de los casos positivos en comparación con el total de casos positivos reales en el conjunto de datos.

En el problema que se trata, se podría dar mayor importancia a la precisión que al recall, dado que se busca minimizar la cantidad de casos negativos (videos no exitosos), que son clasificados incorrectamente como positivos (exitosos).

Es importante resaltar que este modelo fue entrenado con 5 fold cross-validation.

Finalmente, al revisar la importancia de las variables en el modelo, se identifican 10 que están contribuyendo al f1-score.



Aquí se identifica el tiempo de visualización como la variable más importante, seguida por el número de suscriptores y la densidad publicitaria. Se dejaron por fuera del modelo 5 variables: Clics en teaser de tarjeta, Tarjetas mostradas, Clics en elementos de pantalla final, Comentarios a√±adidos y No me gusta.

5. Análisis y conclusiones

- El modelo seleccionado fue la regresión logística. A pesar de ser el modelo más sencillo desde el punto de vista matemático y computacional comparado con los otros estimados fue el que mejores resultados tuvo en términos de métricas como el accuracy, f1-score, recall, etc, cada conjunto de datos sigue una estructura funcional para este caso un modelo lineal logístico fue suficiente.
- Una regresión logística podría ser preferible a los otros modelos implementados por las siguientes razones, interpretabilidad, menor complejidad, menor riesgo de sobreajuste y Eficiencia computacional.
- A pesar de que la regresión logística solo alcanza un recall de 0.56 para la clase de videos exitosos, logra una precisión de 0.83. Dado que se busca minimizar la cantidad de casos

negativos que son incorrectamente clasificados como positivos, se da más relevancia a la precisión que al recall.

- La regresión logística alcanzó un accuracy de 0.83, comparado con 0.77 del árbol de decisión y XGBoost, y 0.8 del Random Rorest.
- Cada problema en ciencia de datos tiene ciertas particularidades las cuales son importantes comprender antes de empezar a modelar datos, por se debe ir revisando los resultados a medida que se obtienen y combinar indicadores de calidad y presión de los modelos.
- A pesar de que el algoritmo de YouTube es desconocido, y que según la documentación consultada son muchas las variables que influyen, mediante las variables propias de un video publicado en un canal, es posible estimar si el contenido será exitoso o no.
- Al integrar el análisis estadístico y los modelos de machine learning, junto con una sólida preparación de datos y selección de características, se puede obtener una comprensión más completa y precisa del problema de clasificación, maximizando tanto la interpretabilidad como el rendimiento del modelo.
- Los modelos por su naturaleza tienen unos parámetros por defecto, los cuales puede ser ajustados para mejorar el rendimiento y la clasificación de los modelos, como se evidenció este proceso mejora significativamente los resultados, también es relevante aplicar varios tipos de modelos y comparar para seleccionar el mejor en función de métricas definidas previamente y el criterio experto juega un papel muy importante, para interpretar los resultados en un contexto específico.
- Se hicieron intervenciones no paramétricas para evitar las suposiciones sobre los datos y reducción del sesgo en la selección de características.

6. Referencias

Altman, E., & Jiménez, T. (2019). Measuring Audience Retention in YouTube. *Valuetools*. https://doi.org/10.1145/3306309

Breiman, L. (2001a). *Random Forests* (Vol. 45).

Breiman, L. (2001b). *Random Forests* (Vol. 45).

Celebi, M. E., Kingravi, H. A., & Vela, P. A. (2013). A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, *40*(1), 200–210. https://doi.org/10.1016/j.eswa.2012.07.021

Chen, T., & Guestrin, C. (2016a). XGBoost: A Scalable Tree Boosting System. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 785–794. https://doi.org/10.1145/2939672.2939785

Chen, T., & Guestrin, C. (2016b). XGBoost: A Scalable Tree Boosting System. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 785–794. https://doi.org/10.1145/2939672.2939785

Cruz, J. (2020, November 14). *Así creció YouTube en Latinoamérica en 2020*. https://folou.co/internet/youtube-audiencias-latinoamerica/

DelcaVideography. (2022). *DelcaVideography - YouTube*. https://www.youtube.com/@DelcaX

Fons, R., & BIGSEO. (n.d.). *CRECETUBE | Curso Especializado para Acelerar Tu Crecimiento en YouTube®*. Retrieved June 5, 2023, from https://crecetube.com/

Garnett, R. (2023). Bayesian Optimization. *Cambridge University Press*, *I*.

Hastie, Trevor., Tibshirani, Robert., & Friedman, J. H. (Jerome H.). (2001a). *The elements of statistical learning : data mining, inference, and prediction : with 200 full-color illustrations*. 533. https://www.worldcat.org/title/46809224

Hastie, Trevor., Tibshirani, Robert., & Friedman, J. H. (Jerome H.). (2001b). *The elements of statistical learning : data mining, inference, and prediction : with 200 full-color illustrations*. 533. https://www.worldcat.org/title/46809224

IBM. (n.d.-a). *¿Qué es un árbol de decisión?* Retrieved December 7, 2022, from https://www.ibm.com/es-es/topics/decision-trees

IBM. (n.d.-b). *¿Qué es un árbol de decisión?* Retrieved December 8, 2022, from https://www.ibm.com/es-es/topics/decision-trees

Kikuchi, Y., Nishimura, I., & Sasaki, T. (2022). Wild birds in YouTube videos: Presence of specific species contributes to increased views. *Ecological Informatics*, *71*, 101767. https://doi.org/10.1016/J.ECOINF.2022.101767

Lior Rokach and Oded Maimon. (2008). *Data mining with decision trees: theory and applications*. World Scientific.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Https://Doi.Org/*, *5.1*, 281–298.

mathworks. (2022). *mathworks*. https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/233459a6-523d-4cf7-91f3-ff539a1b58ce/f6c9980c-ed0d-4564-8289-e95c9274b48e/images/screenshot.png

Melanie, M. (1999). *0−262−63185−7 (PB) 1. Genetics-Computer simulation.2. Genetics-Mathematical models*.

Minitab. (2023). *Estadísticos kappa y coeficientes de Kendall*. Minitab. https://support.minitab.com/es-mx/minitab/20/help-and-how-to/quality-and-process-improvement/measurement-system-analysis/supporting-topics/attribute-agreement-analysis/kappa-statistics-and-kendall-s-coefficients/#fnsrc_1

Molinero, L. M. (2001). *LA REGRESION LOGISTICA*. https://web.archive.org/web/20130629005527/http://www.seh-lelha.org/rlogis1.htm

Mousinho, A. (2021). *SEO: guía completa del posicionamiento en buscadores [2021]*. https://rockcontent.com/es/blog/que-es-seo/

Nananukul, N. (2022). An Inference Model for Online Media Users. *Journal of Data Science*, *11*(1), 143–155. https://doi.org/10.6339/JDS.2013.11(1).1129

Parsa, A. B., Movahedi, A., Taghipour, H., Derrible, S., & Mohammadian, A. (Kouros). (2020). Toward safer highways, application of XGBoost and SHAP for real-time accident detection and feature analysis. *Accident Analysis & Prevention*, *136*, 105405. https://doi.org/https://doi.org/10.1016/j.aap.2019.105405

Pinelis, M., & Ruppert, D. (2020). Machine Learning Portfolio Allocation. *Journal of Finance and Data Science*, *8*, 35–54. https://doi.org/10.1016/j.jfds.2021.12.001

Social Blade. (n.d.). Retrieved December 6, 2022, from https://socialblade.com/youtube/channel/UCDDQf-3UNw3B66FzV53KykQ

Spearman, C. (1904). *The proof and measurement of association between two things*. https://doi.org/10.1093/ije/dyq191

Statista. (2022, April 27). *Número de usuarios de Youtube a nivel mundial entre 2012 y 2021*. https://es.statista.com/previsiones/1289041/usuarios-de-youtube-en-todo-el-mundo

YouTube Creators. (2022, August 9). *Búsqueda y descubrimiento de YouTube: Preguntas frecuentes sobre el algoritmo y el rendimiento - YouTube*. https://www.youtube.com/watch?v=fApg7tzlLjY

Zappin, A., Malik, H., Shakshuki, E. M., & Dampier, D. A. (2022). YouTube Monetization and Censorship by Proxy: A Machine Learning Prospective. *Procedia Computer Science*, *198*, 23–32. https://doi.org/10.1016/J.PROCS.2021.12.207

7. Anexos

Anexo 1

Tabla 1

Variables 1

#	Nombre variable
1	nombre_video

2 fecha_publicacion
3 suscriptores
4 tus_ingresos_estimados_(usd)
5 porcentaje_de_clics_de_las_impresiones_(%)
6 id
7 likes
8 dislikes
9 rating
10 viewCount
11 Tiempo_Recomendación1
12 Tiempo_Recomendación2
13 Tiempo_Recomendación3
14 Tiempo_Publicidad_1
15 Tiempo_Publicidad_2
16 Tiempo_Publicidad_3
17 Tiempo_Publicidad_4
18 Tiempo_Publicidad_5
19 Tiempo_Publicidad_6
20 Tiempo_Publicidad_7
21 Tiempo_Publicidad_8
22 Polarity
23 fecha_publicacion_ff
24 dia_semana_str
25 dia_semana
26 duracion_video_minutos
27 porc_visualizacion
28 marca_exito
29 rangos_duracion_video
30 rangos_Cant_Publicidad
31 consecutivo_tema
32 Duracion Minutos
33 Densidad_Publicitaria
34 porc_min_publi1
35 porc_min_publi2

36 porc_min_publi3
37 porc_min_publi4
38 porc_min_publi5
39 porc_min_publi6
40 porc_min_publi7
41 porc_min_publi8
42 Contenido
43 T√≠tulo del v√≠deo
44 Hora de publicaci√≥n del v√≠deo
45 Clics por elemento de pantalla final mostrado (%)
46 Clics en elementos de pantalla final
47 Clics en teaser por teaser de tarjeta mostrado (%)
48 Clics en teaser de tarjeta
49 Clics por tarjeta mostrada (%)
50 Tarjetas mostradas
51 Me gusta (vs. No me gusta) (%)
52 No me gusta
53 Me gusta
54 Mostrado en el feed
55 Vistos (frente a saltados) (%)
56 Comentarios a√±adidos
57 Suscriptores ganados
58 Porcentaje medio visto (%)
59 Tiempo de visualizaci√≥n (horas)
60 Impresiones
61 Porcentaje de clics de las impresiones (%)