Análisis comparativo de optimización de hiperparámetros por búsqueda en grilla y algoritmos genéticos para *forecasting* de XGBoost en ventas *online*

Comparative analysis of hyperparameter optimization by grid search and genetic algorithms for XGBoost forecasting in online sales

https://doi.org/10.15332/24224529.8401

Recibido: 14 de marzo de 2023

Aceptado: 30 de junio de 2023

Leizy Melissa Pinzón Villanueva
Universidad Central
lpinzonv1@ucentral.edu.co
ORCID: 0000-0001-6997-8988

Citar como:

Pinzón Villanueva, L. M. (2023). Análisis comparativo de optimización de hiperparámetros por búsqueda en grilla y algoritmos genéticos para *forecasting* de XGBoost en ventas *online*. *CITAS*, *9*(2).

https://doi.org/10.15332/24224529.8401

Resumen

En el presente estudio, se plantea un análisis comparativo de la optimización de los hiperparámetros utilizados en el modelo XGBoost para el pronóstico de unidades de venta con una serie de tiempo de ventas de comercio electrónico, utilizando la búsqueda en grilla y algoritmos genéticos. La búsqueda en grilla se realiza con el fin de comparar el desempeño de ambos métodos de optimización en su rendimiento con la métrica R^2 y los tiempos de ejecución, tomando el conjunto de datos de ventas, tratándolo, e implementando el modelo XGBoost con sus hiperparámetros por defecto y luego aplicando las optimizaciones en seis hiperparámetros definidos en dos grupos con diferentes valores para proceder con la comparación. Los resultados muestran que la búsqueda en grilla funciona mejor para la optimización en el conjunto de datos utilizado, dando un resultado del 69.24 %, mientras que el algoritmo genético no logra llegar a una predicción efectiva, lo que abre la puerta a explorar otras librerías de algoritmos evolutivos que puedan aportar a la investigación.

Palabras clave: optimización, hiperparámetros, búsqueda en grilla, algoritmos genéticos, pronóstico.

Abstract

In the present study, a comparative analysis of the optimization of the hyperparameters used in the XGBoost model for the forecast of sales units is proposed with a time series of e-commerce sales, using grid search and genetic algorithms. Grid search is carried out in order to compare the performance of both optimization methods in their performance with the R^2 metric and the execution times, taking the sales data set, treating it, and implementing the XGBoost model with its hyperparameters by default and then applying the optimizations in 6 hyperparameters defined in two groups with different values, to proceed with the comparison. The results show that the grid search works better for optimization in the data set used, giving a result of 69.24%, while the genetic algorithm fails to reach an effective prediction, which opens the door to explore other libraries of evolutionary algorithms that can contribute to the investigation.

Keywords: optimization, search grid, genetic algorithm, forecasting, hyperparameters.

Introducción

El pronóstico en el campo de las ventas directas en tiendas de cadena permite que el surtido de vitrinas sea asertivo y oportuno, facilitando su planificación y orientación, además, porque desde allí se prevé la relación entre el inventario en bodega y los productos en el mostrador y, por ende, se disponen los procesos y recursos para las compras y el aprovisionamiento de productos. De esta forma, se evita el exceso de productos sin consumir en el inventario en bodega, y se asegura la exhibición y cantidad adecuada para el público comprador, así como los momentos adecuados para volver a surtir las vitrinas. Actualmente, en la compañía objeto

de estudio no hay ningún proceso de pronóstico implementado, por lo cual las decisiones de compras y de inventarios se toman con apoyo del área de *planning*, cuyo propósito es realizar un control sobre los procesos de compras, enviar inventarios a tiendas, determinar los márgenes para eventos comerciales, gestionar el *long tail*, analizar el surtido y generar el presupuesto de venta y metas de inventarios.

Los anteriores procesos podrían beneficiarse y complementarse al implementar modelos de *forecasting* mejorando la toma de decisiones y la asertividad de las compras. Estos pueden ser construidos con diferentes metodologías como el XGBoost, uno de los algoritmos de *machine learning* de tipo supervisado más usados actualmente. Este algoritmo obtiene buenos resultados de predicción, generando múltiples modelos que permiten encontrar el más fuerte, con mayor predictibilidad y estabilidad, siendo viable para su uso en pronósticos de ventas de canales de comercio electrónico (Ji et ál., 2019).

La comparación y aplicación de los métodos más adecuados para realizar pronósticos frente a la realidad al servicio de las personas, las organizaciones y las empresas es de vital importancia para que se optimicen sus recursos y sus capacidades, reduzcan las pérdidas y los excedentes, mejoren el aprovisionamiento y tengan la posibilidad de organizar mejor el trabajo de las personas. Tal es el caso de la proyección de las unidades de venta en la empresa objeto de estudio, explorando el valor agregado que la optimización de hiperparámetros puede brindar en la construcción de un modelo de pronóstico de ventas.

La optimización de los modelos de aprendizaje supervisado incide directamente en la organización y en el mejoramiento de los procesos y del trabajo de las personas y las empresas, siendo aplicable al pronóstico de unidades de venta e implicando la selección del mejor método de optimización, por lo cual durante esta investigación se pretende realizar un análisis comparativo de optimización de hiperparámetros de un modelo XGBoost para *forecasting*, mediante el método de búsqueda en grilla y algoritmo genético, hacia la predicción de una serie de tiempo de ventas de comercio electrónico.

Antecedentes conceptuales

El trabajo magistral “Modelos para la predicción de deserción universitaria de estudiantes de Psicología de la Universidad el Bosque”, de Nicolás Torres Acero (2022), empleó los modelos Random Forest y XGBoost, con base en información académica, sociodemográfica y de personalidad de los estudiantes para realizar el pronóstico planteado, partiendo de la construcción de los modelos, su validación y comparación, a través de la información de los estudiantes de Psicología de la Universidad del Bosque, entre los años 2013 y 2017 (Torres, 2022). En este estudio, se evidenció la superioridad en la exactitud del pronóstico del modelo XGBoost sobre el modelo Random Forest, según lo confirman las diferentes métricas. Se

obtuvieron los hiperparámetros para el funcionamiento de los modelos con base en las primeras iteraciones planteadas y sus ajustes para el funcionamiento de cada uno, en donde fue muy aceptable su asertividad y se ajustó el método XGBoost para identificar los factores de riesgo hacia la deserción (Torres, 2022). Esta investigación arrojó resultados acertados con base en la utilización de matrices de confusión del método Random Forest y de la optimización a través de hiperparámetros con el método XGBoost, en la etapa de entrenamiento, con una exactitud mayor al 75 %. Se encontró, al final, que el modelo Random Forest tuvo resultados mejores de lo esperado, aunque fue superado por la potencia y precisión del modelo XGBoost (Torres, 2022).

Manuela Granda desarrolla la tesis magistral en economía "Determinantes del riesgo de incumplimiento en créditos educativos: un análisis para Colombia" (2020). Allí desarrolló la utilización del método XGBoost para predecir los incumplimientos en los créditos educativos ofrecidos por el Instituto Colombiano de Crédito Educativo (Icetex), entre los años 2015 y 2018 (Granda, 2020). Este trabajo empleó el modelo XGBoost para realizar las mencionadas estimaciones, y se basó en un modelo de equilibrio competitivo, orientado al mercado de los créditos educativos, teniendo en cuenta asimetrías de información (Bardhan y Udry, 1999, en Granda, 2020). De esta forma, en este estudio se planteó el uso de un algoritmo XGBoots, donde los árboles se ajustaron por métodos de *ensambling*, como el *boosting*, reduciendo los errores en las previsiones.

Jesús Espinosa (2020), en su trabajo "Aplicación de algoritmos Random Forest y XGBoost en una base de solicitudes de tarjetas de crédito", planteó el uso de algoritmos de aprendizaje supervisado para el caso de la emisión de tarjetas de crédito, realizando una comparación entre el algoritmo de búsqueda aleatoria Random Forest y el algoritmo de impulso de búsqueda de valores de gradiente extremo XGBoost como métodos de optimización. Para lograrlo, empleó el aprendizaje supervisado, referido hacia una "variable objetivo" de la cual se desea estimar su valor, es decir, apto o no apto para adquirir el servicio de tarjeta de crédito, a través de la asociación de otras variables. El estudio muestra que es mayor la eficiencia del modelo XGBoost, contando con las curvas de ganancia acumulada y de enfoque acumulada para darle una perspectiva práctica a la comparación, haciendo posible que puedan compararse estos resultados con otros métodos, como los algoritmos genéticos y las redes neuronales para continuar el estudio, y complementando la validación de estos, desde las curvas ROC y AUC, empleando coeficientes Kappa o Alfa.

Metodología

Fase 1. Procesamiento de datos y análisis descriptivo

La recolección de datos se hizo directamente de la plataforma de comercio electrónico. Se descargaron las ventas desde julio del 2016, fecha en la cual comenzó a operar esta plataforma hasta diciembre del 2019. Estos archivos fueron almacenados de manera local, organizados por año y cargados posteriormente al programa, en donde se concatenaron para crear el *dataframe* inicial en Python. Para el procesamiento de datos y el desarrollo del ejercicio se utilizó el Python V. 3.9.7 como lenguaje de programación Jupiter Lab para el análisis descriptivo, implementación de XGBoost y búsqueda en grilla y Spider IDE para la optimización con algoritmos genéticos. Estos datos contenían toda la información relacionada con la orden y debieron ser limpiados antes del cargue para anonimizar la base. Adicionalmente, se utilizó un segundo archivo que contenía las categorías a las que correspondía cada producto para poder identificarlos. Las variables resultantes de la anonimización de datos que se cargaron en el *notebook* se muestran en la tabla 1.

Tabla 1. Variables iniciales

Variable	Definición
Order	Número de orden
Sequence	Número de transacción
Creation Date	Fecha de creación de la orden
Creation Hour	Hora de creación de la orden
UF	Departamento de Colombia
City	Ciudad
Postal Code	Código postal
SLA Type	Tipo de transportadora que enviara la orden
Courrier	Nombre de la transportadora que llevará la orden
Status	Estado de la orden
Payment System Name	Nombre del método de pago
Installments	Número de cuotas (si la orden fue pagada con tarjeta de crédito)
Payment Value	Valor pagado
Quantity_SKU	Cantidad de productos comprados
ID_SKU	Identificador del producto plataforma de *ecommerce*
Category Ids Sku	Categoría del producto
Reference Code	Identificador del producto en la cadena
SKU Name	Nombre del producto
SKU Value	Precio base del producto
SKU Selling Price	Precio de venta del producto
SKU Total Price	Valor total pagado por producto
Shipping List Price	Valor pagado por el envío
Shipping Value	Valor total de la orden pagada por envío
Total Value	Valor total de la orden de productos más envío
Discounts Totals	Descuentos aplicados a la orden

Fuente: elaboración propia.

Para el caso del departamento se necesitó un procesamiento adicional, ya que el que se muestra en la variable "Category Ids Sku" estuvo conformado por la concatenación de los ID de las categorías que existían dentro de la plataforma de *ecommerce* en el siguiente formato, siendo el primer valor el departamento y los siguientes la categoría, subcategoria1 y subcategoria2 al que corresponde el producto: *Category Ids Sku: /5/35/134/232/.* Es por esto que fue necesario identificar textualmente el departamento (*Department_name*) y la categoría (*Category_name*) al que correspondían los ID del archivo para que a su vez fuera posible identificar el departamento y la categoría a la que correspondía cada producto comprado. Para este propósito, se descargó de la plataforma de *ecommerce* un listado de los departamentos y categorías con ID y Nombre para identificar su origen, se cargaron los archivos de órdenes y de categorías al *notebook*, en donde se concatenaron para crear un único *dataframe*, y se unió con el archivo de Departamentos, donde se crearon dos columnas nuevas. Se hicieron transformaciones a la columna "Creation Date" convirtiéndola a fecha y extrayendo el año en el cual se realizó la orden, creando una columna adicional llamada "Year". También se excluyeron del *dataframe* las órdenes que no eran efectivas (la orden fue cancelada por el cliente o se procesó de manera incorrecta) y finalmente se redujo el *dataframe* a los datos necesarios para el análisis, manteniendo las siguientes variables: *Order*, *Year*, *Total_Value*, *Quantity_SKU*, *department_name*, *category_name*, *City*, *Reference Code*, *SKU Name* e *ID_SKU.*

El análisis descriptivo mostró información de la cantidad de datos por utilizar y el número de variables, el total de ventas por año, las unidades vendidas por año y por categoría. También se analizó la venta por categoría para identificar si estos productos fueron vendidos durante el periodo definido de los datos y se determinaron las categorías y su participación total en la serie de tiempo. Se analizó la venta por año y por categoría al igual que el número de unidades vendidas para definir las categorías más representativas del estudio. De acuerdo con el análisis realizado, se definió el departamento con el que se trabajaron las siguientes fases del proyecto, seleccionando el departamento más representativo en ventas. Para los modelos que se evaluaron en este estudio, se hizo la sumatoria por mes de los productos vendidos en la cadena, omitiendo el campo de SKU y teniendo en cuenta también la eliminación de datos nulos en el caso de que existieran.

Fase 2. Desarrollo e implementación del modelo

Para este modelo se utilizó la técnica de aprendizaje automático XGBoost (Extreme Gradient Boosting). El paso inicial fue visualizar los datos para analizar el comportamiento y la presencia de estacionalidad y tendencia, y, en caso de que no lo fueran, se obtuvo la diferencia en ventas contra el mes anterior buscando que los datos fuesen estacionarios. A partir de este resultado, se construyó el modelo, partiendo de que la variable de unidades vendidas por mes correspondiera a la categoría con mayor representación en la cadena. Se definieron también

los datos que se utilizarían para entrenamiento y para prueba en un porcentaje de 70-30, teniendo en cuenta que el porcentaje mayor correspondería a los datos de entrenamiento de la serie y el porcentaje menor a los datos de prueba. Luego de la separación de los datos en entrenamiento y prueba, se ejecutó el modelo XGBoost con los hiperparámetros generados por defecto, para lo cual se utilizó la función fit() y se procedió con la predicción. La métrica empleada para la validación de los modelos fue R^2.

Fase 3. Desarrollo e implementación de la optimización de hiperparámetros

En la fase de optimización de hiperparámetros se determinaron los hiperparámetros que se iban a optimizar, teniendo en cuenta la naturaleza de cada uno de ellos logrando determinar los valores válidos que aceptaría cada uno. También se estableció una condición que permitió al algoritmo detenerse para mejorar los tiempos de la optimización y finalmente ejecutar los procesos correspondientes en donde se establecieron los hiperparámetros para evaluar, la cantidad de valores que se van a pasar por cada iteración, ejecutar los procesos de entrenamiento los cuales deben ser comparables en las mismas condiciones para poder evaluar los resultados calculando puntuaciones del modelo.

Los hiperparámetros optimizados en esta investigación fueron los siguientes:

-*Gamma*: reducción mínima de pérdidas necesaria para realizar una nueva partición en un nodo hoja del árbol. Cuanto mayor sea gamma, más conservador será el algoritmo. Rango [0, ∞]

-*Learning rate*: la tasa de aprendizaje. Valor por defecto 0.3 en un rango de [0, 1]

-*Max_depth*: profundidad máxima del árbol. Rango [0, ∞]

-*Min_child_weight*: número de instancias necesarias en cada nodo. Rango [0, ∞]

-*N_estimators*: número máximo de iteraciones y de árboles que se realizarán antes de detener el proceso de ajuste.

-*Subsample*: proporción del muestreo de cada árbol. Rango de [0, 1]

Para la identificación de los valores aproximados de los hiperparámetros se tomaron valores aleatorios dentro del rango permitido para cada uno de ellos, teniendo en cuenta como valor central los hiperparámetros que arrojó el modelo por defecto y buscando los mejores resultados en un rango mayor y menor de cada uno de estos. Para lograr una mayor profundidad en el ejercicio, se validaron conjuntos de hiperparámetros aleatoriamente usando una función en uno de ellos que tuviera mayor profundidad en el análisis, buscando un mejor desempeño. También se tomaron dos grupos de referencia en cada una de las optimizaciones, utilizando valores diferentes y en rangos mas amplios para validar el proceso.

a) Desarrollo e implementación de la optimización por medio de búsqueda en grilla

Para la optimización por medio de la búsqueda en grilla, se usó la función GridSearchCV() de la librería scikit-learn, en donde se establecieron los estimadores y un conjunto de valores finitos y razonables para cada hiperparámetro teniendo como referente la validación cruzada (valor cv = 5, que es el valor por defecto de la función) a través de un subconjunto de hiperparámetros especificados manualmente, que fueron los mismos para ambas optimizaciones, de los cuales el algoritmo devolvió el modelo con la mejor puntuación en el proceso de evaluación a través de los atributos mencionados anteriormente. Estos hiperparámetros se utilizaron para ejecutar el modelo inicial y realizar la correspondiente comparación. También se realizó el análisis en dos grupos, modificando los valores de los hiperparámetros.

Tabla 2. Grupo 1 y 2 de hiperparámetros de búsqueda en grilla

Hiperparámetro	Valor grupo 1	Valor grupo 2
gamma	[1, 3, 5]	[1, 5, 7, 9]
learning_rate	[0.1, 0.2, 0.3]	[0.1, 0.3, 0.5, 0.7, 1]
max_depth	[1, 3, 5]	[1, 5, 7, 9]
min_child_weight	[1, 3, 5]	[1, 5, 7, 9]
n_estimators	[50, 100, 150, 200]	[50, 100, 200, 300, 400]
Subsample	[0.1, 0.3, 0.5]	[0.1, 0.3, 0.5, 0.7, 1]

Fuente: elaboración propia.

Como se observa en la tabla 2, los valores del segundo grupo tuvieron un rango más amplio, lo que permitió ver resultados diferentes, y en donde se esperó evaluar también los tiempos de ejecución. Estos valores fueron incluidos dentro de una variable tipo diccionario llamada *tuned_parameters,* en donde se alojaron los valores de cada grupo, y por medio de la función GridSearchCV(), con una validación cruzada de cinco bloques. Tan pronto el modelo fue entrenado, se accedió a la mejor puntuación y los mejores parámetros por medio de los atributos *best_score_* y *best_params_* respectivamente, lo que arrojó los mejores valores de cada hiperparámetro y, por último, al R^2 resultado del ejercicio. También se incluyó dentro del código el comando mágico *%%time*, que es parte de IPython, el cual imprime el tiempo de ejecución de la función.

b) Desarrollo e implementación de la optimización por medio de algoritmos genéticos

Para la implementación del algoritmo genético como método de optimización se utilizó un algoritmo genético personalizado para XGBoost creado por Mohit Jain (2018) que contiene funciones que siguieron los siguientes cuatro pasos:

-*Inicialización*: en esta fase, los hiperparámetros se inicializaron aleatoriamente para crear la población, formando la primera generación. En este proceso se generó un vector que contenía los seis hiperparámetros que se optimizaron y se les asignó un rango que estuvo dentro de los

límites que se establecieron para la búsqueda en grilla, por lo cual estuvieron entre el valor menor y el valor mayor asignado en cada uno de los grupos.

-*Selección de padres*: en este punto se entrenó el modelo usando la población inicial y calculando el valor de aptitud, que para el ejercicio será el R^2. Se definieron ocho padres de los cuales se seleccionaron los cuatro más aptos, creando cuatro generaciones monitoreando el R^2. La mitad de los padres de la siguiente generación fueron los padres más aptos de la generación anterior, manteniendo un mejor puntaje y se creó una matriz de acuerdo al valor de aptitud.

-*Cruce*: se utilizó un cruce uniforme en donde cada parámetro para el hijo se seleccionó de manera independiente de los padres, y un cruce para listas ordenadas.

-*Mutación*: con la mutación se buscó introducir diversidad en los hijos seleccionando al azar uno de los parámetros y alterando su cantidad de manera aleatoria teniendo en cuenta los límites que se establecieron al inicio.

Por último, se generaron los resultados por cada una de las generaciones, mostrando el mejor valor para cada uno de los hiperparámetros y el R^2 resultado del ejercicio.

Fase 4. Comparación de métricas y resultados

Una vez terminada la fase de desarrollo e implementación de la optimización de hiperparámetros con búsqueda en grilla y algoritmos genéticos, se procede a compararlos. Como se mencionó con anterioridad, la variable para pronosticar es el número de unidades vendidas por mes, teniendo como base de comparación el resultado del R^2, lo que explica en qué porcentaje la variabilidad observada es capturada por el modelo. El resultado del proceso anterior es una métrica confiable en cada uno de los modelos y grupos. Así se generaron dos resultados de búsqueda en grilla y dos resultados para los algoritmos genéticos, que fueron comparados por valores utilizados y por técnica usada.

Resultados

Procesamiento de datos y análisis descriptivo

A continuación, se evidencian los datos que se utilizaron para el análisis descriptivo. Durante los cuatro años, se realizaron un total de 1 584 524 órdenes, además, se generaron $129 192 960 000 millones de pesos en ventas *online*, vendiendo 622 913 unidades.

Tabla 3. Transacciones, ventas totales y unidades vendidas por año

Año	Order	Ventas totales	Cantidad
2016	105 800	$ 25 387 780 000	113 672
2017	103 955	$ 24 874 060 000	112 839

2018	123 041	$ 26 618 640 000	133 827
2019	232 898	$ 52 312 480 000	262 575

Fuente: elaboración propia.

Al validar la presencia de ventas por categoría, se generó una tabla mostrando las unidades vendidas por categoría y por año.

Tabla 4. Ventas por departamento por año

Año	Alimentos	Bebés	Belleza	Deportes	Electro	Escolar	Hogar	Juguetería	Libros	Maletas	Mascotas	Moda	Tecnologia
2016	0	21 598	1170	1270	4622	1318	7400	57620	226	172	1334	15900	72
2017	0	23 163	836	2315	12527	1290	5862	44087	252	90	0	21137	0
2018	0	23 837	2823	5347	13927	1411	5637	58146	239	111	2582	18630	51
2019	33	59 004	6609	7716	20465	3586	25 675	90764	364	386	6785	39814	237

Fuente: elaboración propia.

En la tabla 4 se observa que solo los departamentos Bebés, Belleza, Deportes, Electro, Hogar, Juguetería, Libros, Maletas, Moda, Videojuegos y Zapatos, tuvieron ventas durante toda la serie de tiempo, por lo que se procedió a eliminar las otras categorías que no presentasen ventas constantes. Luego de la extracción, quedó como resultado un *dataframe* con 555 949 órdenes.

Se observó que el departamento Juguetería era el que generaba más ventas, seguido de Bebés y de Moda, siendo las categorías más fuertes del *dataframe*. Las demás categorías mostraron un comportamiento constante. Así, pues, se mantuvieron dentro del *dataframe* las tres categorías más representativas tanto en ventas como en unidades (Bebés, Moda y Juguetería). Después de esta extracción, quedó un *dataframe* con 438 573 órdenes.

Para la categoría de Bebés se observó que la cantidad máxima vendida fue de 59 004 unidades por año, la mediana se encontró en 23 500 unidades y la cantidad mínima fue de 21 598 unidades; el rango intercuartílico mostró que las unidades vendidas fluctuaron entre 22 380 y 41 420 unidades. Para Juguetería se observó que la cantidad máxima vendida fue de 90 674 unidades por año, la mediana se encontró en 57 883 unidades y la cantidad mínima fue de 44 087 unidades; el rango intercuartílico evidenció que las unidades vendidas fluctuaron entre 50 853 y 74 410 unidades. Para Moda se observó que la cantidad máxima vendida fue de 39 814 unidades por año, la mediana se encontró en 19 883 unidades y la cantidad mínima fue de 15 900 unidades; el rango intercuartílico mostró que las unidades vendidas fluctuaron entre 17 265 y 30 475 unidades.

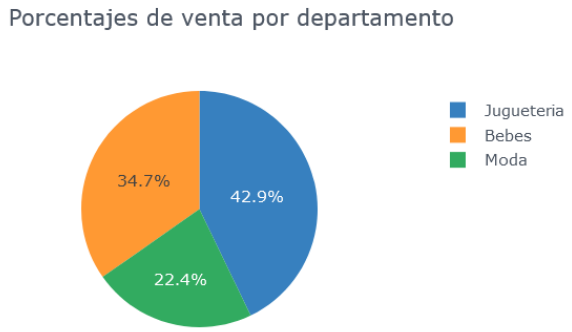

Figura 1. Participación de venta principales departamentos

Fuente: elaboración propia.

Finalmente, se halló el porcentaje de unidades vendidas por departamento en el 2019, donde se observó que el departamento de Juguetería tuvo el 42.9 % de participación, seguido de Bebés con un 34.7 % y finalmente Moda con un 22.4 %. Se definió de acuerdo con los resultados que el estudio se centraría en el departamento de Juguetería, que fue el de mayor participación en ventas.

Desarrollo e implementación del modelo

En la implementación del modelo, se tomaron los datos de la categoría de Juguetería en el periodo de tiempo de Julio del 2016 a diciembre del 2019. Se sumaron las cantidades totales vendidas por mes y se graficaron en el periodo establecido, mostrando que los datos no eran estacionarios (figura 2).

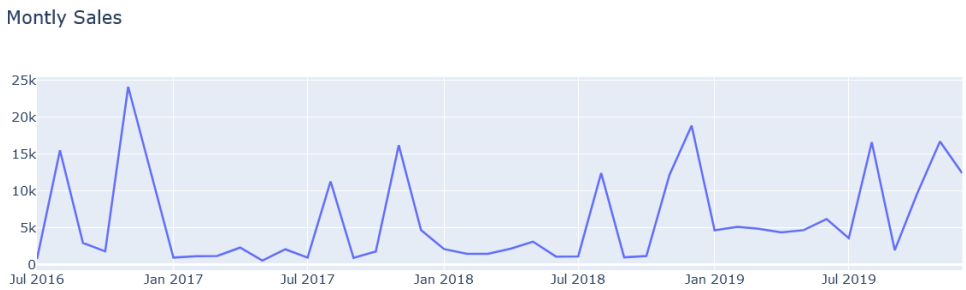

Figura 2. Ventas por mes

Fuente: elaboración propia.

Para convertir los datos en estacionarios, se obtuvo la diferencia en las ventas comparadas contra el mes anterior.

Tabla 5. Diferencia de ventas comparadas contra el mes anterior

	date	Quantity_SKU	prev_sales_1	diff
1	2016-08-01	15490	762.0	14728.0
2	2016-09-01	2954	15490.0	-12536.0
3	2016-10-01	1808	2954.0	-1146.0
4	2016-11-01	24078	1808.0	22270.0
5	2016-12-01	12528	24078.0	-11550.0

Fuente: elaboración propia.

A partir de la diferencia, se usaron los datos de los 12 meses anteriores para pronosticar los siguientes y así completar el conjunto de datos para el pronóstico, validando la utilidad de las características del modelo con R^2, en donde las 12 variables creadas explicaron el 63 % de la variación (tabla 6).

Tabla 6. Resultados de ventanas del modelo

	date	Quantity_SKU	diff	lag_1	lag_2	lag_3	lag_4	lag_5	lag_6	lag_7	lag_8	lag_9	lag_10	lag_11	lag_12
0	2017-08-01	11280	10320.0	-1126.0	1514.0	-1746.0	1125.0	52.0	174.0	-11561.0	-11550.0	22270.0	-1146.0	-12536.0	14728.0
1	2017-09-01	917	-10363.0	10320.0	-1126.0	1514.0	-1746.0	1125.0	52.0	174.0	-11561.0	-11550.0	22270.0	-1146.0	-12536.0
2	2017-10-01	1798	881.0	-10363.0	10320.0	-1126.0	1514.0	-1746.0	1125.0	52.0	174.0	-11561.0	-11550.0	22270.0	-1146.0
3	2017-11-01	16169	14371.0	881.0	-10363.0	10320.0	-1126.0	1514.0	-1746.0	1125.0	52.0	174.0	-11561.0	-11550.0	22270.0
4	2017-12-01	4686	-11483.0	14371.0	881.0	-10363.0	10320.0	-1126.0	1514.0	-1746.0	1125.0	52.0	174.0	-11561.0	-11550.0

Fuente: elaboración propia.

Finalmente, se separaron los datos en conjunto de entrenamiento y de prueba en un porcentaje 70-30. Los hiperparámetros utilizados en este modelo fueron los arrojados por

defecto y como resultado del modelo, el R^2 para datos de entrenamiento es del 99.9 % y para los datos de test del 83.72 %

Desarrollo e implementación de la optimización por medio de búsqueda en grilla

De acuerdo con los hiperparámetros definidos anteriormente, se buscó la optimización de resultados por medio de la búsqueda en grilla. Teniendo en cuenta los resultados arrojados por defecto del XGBoost, se tomaron valores cercanos a este resultado y se generaron dos grupos (tabla 2). En el primer grupo, se tomó un número menor de valores. En el segundo grupo, se tomaron más valores y un rango más amplio buscando un resultado más preciso. Para conseguirlo, se asignaron los valores de cada grupo en una variable tipo diccionario llamada *tuned_parameters*, en donde se alojan los valores de cada grupo, y por medio de la función GridSearchCV() de la librería scikit learn, con una validación cruzada de cinco bloques. Tan pronto el modelo fue entrenad o, se accedió a la mejor puntuación y a los mejores parámetros por medio de los atributos *best_score_* y *best_params_*, respectivamente. Luego de ejecutar el modelo con los datos de entrenamiento y obtener los mejores parámetros, se procedió a ejecutarlo con los datos de test, asignando dichos parámetros con la función *best_estimator_* para lograr el resultado del R^2. Con el primer grupo, los resultados arrojaron un R^2 del 67.77 % en los datos de entrenamiento y el 65.76 % en datos de test, mientras que con el segundo grupo los resultados arrojaron un R^2 del 68.96 % en los datos de entrenamiento y el 69.24 % en datos de test.

Los mejores hiperparámetros resultantes de la validación del grupo 1 y 2 se muestran en la tabla 7.

Tabla 7. Resultados grupo 1 y 2. Búsqueda en grilla

Hiperparámetro	**Grupo 1**			**Grupo 2**		
	Valor	**R^2 E**	**R^2 T**	**Valor**	**R^2 E**	**R^2 T**
Gamma	1	67.77%	65.76%	1	68.96%	69.24%
learning_rate	0.2			0.1		
max_depth	1			1		
min_child_weight	1			1		
n_estimators	200			400		
Subsample	0.3			0.3		
Tiempo	4 minutos, 45 segundos			54 minutos, 47 segundos		

Fuente: elaboración propia.

Desarrollo e implementación de la optimización por medio de algoritmos genéticos

Para el análisis con algoritmos genéticos se tomó el rango mínimo y máximo de los valores asignados a los hiperparámetros usados en la búsqueda en grilla en cada uno de los grupos, teniendo en cuenta que el algoritmo genético tomó valores aleatorios dentro de estos rangos. Se realizó la selección de padres y el valor de aptitud R^2, definiendo ocho padres que darían paso a los cuatro más aptos, creando cuatro generaciones. Luego de esto se realizó un cruce uniforme y se mutó, buscando diversidad en los hijos, teniendo en cuenta los límites de los rangos establecidos al inicio. Finalmente, se generaron los resultados por cada una de las generaciones, mostrando al final el mejor valor para cada uno de los hiperparámetros y el R^2 resultado del ejercicio.

Con el primer grupo los resultados arrojaron un R^2 del -1.24 % en los datos de entrenamiento y el -1.24 % en datos de test. Con el segundo grupo los resultados arrojaron un R^2 del -1.24 % en los datos de entrenamiento y el -1.24 % en datos de test. Además, el comportamiento del R^2, en cada generación fue el mismo en cada uno de los grupos; sin embargo, con respecto a los valores de los hiperparámetros, se observó una variación en la primera generación y en las siguientes generaciones un comportamiento similar que no da mayores resultados (tabla 8).

Tabla 8. Resultados grupo 1 y 2. Algoritmo genético

Hiperparámetro	**Grupo 1**			**Grupo 2**		
	Valor	**R^2 E**	**R^2 T**	**Valor**	**R^2 E**	**R^2 T**
Gamma	2.02			3.04		
learning_rate	0.21			0.59		
max_depth	3	-	-	5	-	-
min_child_weight	1.77	1.24%	1.24%	2.55	1.24%	1.24%
n_estimators	75			125		
Subsample	0.41			0.79		
Tiempo	9 segundos			9 segundos		

Fuente: elaboración propia.

Comparación de métricas y resultados

Luego de realizar los procesos de limpieza de datos y preparación del dataset para el análisis, en donde se determinó que los datos que se utilizarían para el ejercicio eran los datos del departamento de Juguetería de la cadena, y se realizara todo el proceso de limpieza y afinamiento de los datos, se llevó a cabo inicialmente el pronóstico utilizando el modelo XGBoost sin ningún tipo de optimización, en donde se obtuvieron resultados de R^2 para

datos de entrenamiento de un 99.9 % y para datos de test del 83.72 %. Estos resultados indicaron que el modelo por sí solo se sobreajustaba, ya que presentaba buen desempeño con los datos de entrenamiento; sin embargo, los resultados con los datos de test fueron más bajos. En cuanto al tiempo de ejecución de este modelo, se ejecutó de manera inmediata.

En el caso de la búsqueda en grilla, se asignaron valores diferentes a dos grupos con los que fueron entrenados los modelos, al igual que en la optimización con algoritmos genéticos. La diferencia es que en la búsqueda en grilla se indicaron los valores de los hiperparámetros puntuales por optimizar, mientras que al algoritmo genético se le indicó un rango, ya que él buscó valores aleatorios. Inicialmente, se evidenció que, con la utilización de la búsqueda en grilla, se logró eliminar el sobreajuste, además, con un número menor de valores, los tiempos de ejecución fueron más cortos; sin embargo, el R^2 obtenido fue más bajo. Para el caso del segundo grupo, en donde se incluyó un numero de hiperparámetros más alto, el tiempo de ejecución aumentó significativamente y también los resultados del R^2 mejoraron, tanto en datos de entrenamiento como en los datos de test. Comparado contra la ejecución del modelo sin ningún tipo de optimización, el resultado evidenció mayor estabilidad y un consumo mayor de tiempo de ejecución.

En cuanto al algoritmo genético, los resultados no fueron positivos debido a que el resultado obtenido en cada uno de los grupos fue de -1.24 %, lo que indicó que el modelamiento no fue efectivo para este conjunto de datos. Se realizaron cambios sobre el número de padres y generaciones y el resultado obtenido era el mismo. En cuanto al tiempo de ejecución, los resultados se obtuvieron en nueve segundos, mostrando así una ejecución bastante rápida, pero de igual manera ineficiente.

Al comparar los resultados obtenidos por la búsqueda en grilla y el algoritmo genético se encuentra que la mejor opción es la búsqueda en grilla, ya que proporciona datos coherentes en el ejercicio.

Conclusiones y recomendaciones

Al ejecutar los dos ejercicios con dos diferentes conjuntos de hiperparámetros se definió que el mejor grupo es el conjunto de hiperparámetros que contenía mayor número de datos y se utilizó con la búsqueda en grilla, arrojando un R^2 del 69.24 %. No fue posible hacer una comparación en cuanto a efectividad entre los dos modelos de optimización, ya que el algoritmo genético no arrojó valores efectivos del pronóstico realizado. Sin embargo, sí se logró observar que el número de valores de hiperparámetros utilizados en la búsqueda en grilla impactaron directamente sobre el resultado del R^2, lo que indicó que un mayor número de valores arrojaría un resultado más cercano a lo deseado, pero a su vez impactando en los tiempos de ejecución, lo que implicaría posiblemente el uso de máquinas que tengan mayor

poder computacional. Adicionalmente, el tiempo de ejecución podría ser una desventaja al trabajar con un dataset de mayor tamaño y con mayor número de variables. Para este ejercicio no representó mayor complejidad, pues el tiempo de ejecución fue aceptable dentro de las validaciones que hizo el algoritmo. Es necesario agregar un mayor número de valores a los hiperparámetros para lograr un resultado mejor en el pronóstico, que pueda ser utilizado con confianza en un ejercicio real; sin embargo, tomará un mayor tiempo de ejecución llegar a este resultado.

Referencias

Espinosa Zúñiga, J. J. (2020). Aplicación de algoritmos Random Forest y XGBoost en una base de solicitudes de tarjetas de crédito. *Revista Ingeniería Investigación y Tecnología*, *XXI*(3), 1-16. https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-77432020000300002

Granda Rodríguez, M. A. (2020). *Determinantes del riesgo de incumplimiento en créditos educativos: un análisis para Colombia*. (Tesis de Maestría en Economía), Universidad Eafit, Medellín.

Jain, M. (2018). *Hyperparameter Tuning in XGBoost Using Genetic Algorithm*. https://towardsdatascience.com/hyperparameter-tuning-in-xgboost-using-genetic-algorithm-17bd2e581b17

Ji, S., Wang, X., Zhao, W. y Guo, D. (2019). An Application of a Three-Stage XGBoost: Based Model to Sales Forecasting of a Cross-Border E-Commerce Enterprise. *Mathematical Problems in Engineering*, (2019), 1-15. https://www.hindawi.com/journals/mpe/2019/8503252/

Torres Acero, N. (2022). *Modelos para la predicción de deserción universitaria de estudiantes de Psicología de la Universidad el Bosque*. (Maestría en Estadística), Universidad del Bosque, Bogotá, D. C.