Predicción Garantías a Siniestrar

Stiven Sanmartin Jaramillo

Monografía presentada para optar al título de Especialista en Analítica y Ciencia de Datos

Asesora
Daniela Serna Buitrago, Ingeniera de Sistemas

Universidad de Antioquia
Facultad de Ingeniería
Especialización en Analítica y Ciencia de Datos
Medellín, Antioquia, Colombia
2021

Cita	Sanmartin Jaramillo [1]
Referencia Estilo IEEE (2020)	[1] S. Sanmartin Jaramillo, “Predicción Garantías a Siniestrar”, Trabajo de grado especialización, Especialización en Analítica y Ciencia de Datos, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2021.

Especialización en Analítica y Ciencia de Datos, Cohorte II.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

FGA Fondo de Garantías es una sociedad de economía mixta, que actúa como fiador institucional para respaldar la financiación de servicios, créditos de consumo, educativos, créditos digitales, entre otros; a través de garantías de consumo. Las garantías son la figura financiera que respalda el cumplimiento de los créditos que dan entidades como bancos, cooperativas, cajas de compensación, empresas del sector retail, entre otros, y que permite procesos más ágiles y confiables tanto para las personas que los solicitan, en adelante deudores, como para la entidad que los otorga, en adelante intermediario financiero, facilitando de este modo el acceso al crédito.

En este proyecto final, con base en su información demográfica y de comportamiento de cartera de los deudores, se buscó obtener modelos predictivos que permitan prever si una garantía se va a siniestrar.

La metodología empleada inicio con la limpieza y normalización de los datos, seguidamente, el análisis exploratorio de los datos, continuo con la ingeniera de variables, y finalmente, el proceso iterativo de entrenamiento y evaluación de los modelos.

Se concluyó que el modelo que mejor desempeño obtiene con los datos generados es un GradientBoostingClassifier implementado mediante la librería Sklearn, que emplea todas las variables categóricas de las cuales se dispone.

Palabras clave — **Garantías, algoritmos de clasificación, aprendizaje de maquinas, aprendizaje supervisado**

ABSTRACT

FGA Fondo de Garantías is a mixed economy company, which acts as an institutional guarantor to support the financing of services, consumer loans, educational loans, digital loans, among others; through consumer guarantees. Guarantees are the financial figure that supports the fulfillment of the credits given by entities such as banks, cooperatives, compensation funds, companies in the retail sector, among others, and that allows more agile and reliable processes both for the people who request them, as well as for the entity that grants them, thus facilitating access to credit.

In this final project, based on its demographic information and the payments behavior of the debtors, it was sought to obtain predictive models that allow predicting whether a guarantee is going to be sinister.

The methodology used began with the cleaning and normalization of the data, followed by the exploratory data analysis, continued with the feature engineering, and finally, the iterative process of training and evaluation of the models.

It was concluded that the model with the best performance obtained with the data generated is a GradientBoostingClassifier implemented through the Sklearn library, which uses all the categorical variables that are available.

Keywords — **Guarantees, data science, classification algorithms, machine learning, supervised learning**

I. INTRODUCCIÓN

FGA Fondo de Garantías es una sociedad de economía mixta, creada en 1997 por el Fondo Nacional de Garantías (FNG), el Municipio de Medellín y la Cámara de Comercio de Medellín para Antioquia que actúa como fiador institucional para respaldar la financiación de servicios, créditos de consumo, educativos, créditos digitales, entre otros; a través de la línea de negocios garantías de consumo.

Adicionalmente, FGA Fondo de Garantías es el agente comercial del Fondo Nacional de Garantías (FNG) en Antioquia y Chocó para facilitar el acceso a la financiación de la MiPyme colombiana, mediante el otorgamiento de garantías.

A. Garantías

Las garantías son la figura financiera que respalda el cumplimiento de los créditos que dan entidades como bancos, cooperativas, cajas de compensación, empresas del sector retail, entre otros, y que permite procesos más ágiles y confiables tanto para las personas que los solicitan como para la entidad que los otorga, facilitando de este modo el acceso al crédito.

La relación que existe entre FGA y los intermediarios financieros, se basa en la suscripción de un convenio de garantía, mediante el cual FGA, en su calidad de fiador subsidiario, garantiza los créditos que los intermediarios financieros confieren a los usuarios de sus servicios crediticios, en razón del incumplimiento de estos o sus codeudores. Es decir, cuando hay incumplimiento en el pago del crédito por parte de los deudores, FGA le paga a los intermediarios financieros como fiador de ese crédito y luego le puede recobrar al deudor inicial.

B. Proceso para acceder a una garantía

1. El deudor acude a la entidad, intermediario financiero o establecimiento de comercio en busca de una financiación.
2. El intermediario financiero o establecimiento de comercio realiza un estudio de la viabilidad para otorgar el crédito.

3. En este momento ingresa FGA como fiador ante el intermediario financiero, permitiendo así que el deudor pueda acceder al crédito.
4. El intermediario o establecimiento de comercio aprueba el crédito con garantía de FGA.
5. El deudor comienza a realizar sus pagos al intermediario financiero por el valor del crédito pactado y por el valor del servicio de fianza a FGA Fondo de Garantías.
6. El deudor realiza el pago de las cuotas al intermediario, pero si incumple en algún momento con su obligación y el intermediario financiero solicita a FGA hacer efectiva la garantía, FGA realiza el pago de la deuda, e inmediatamente FGA es el nuevo acreedor, por lo que puede proceder con los cobros o disponer de la deuda.

II. DESCRIPCIÓN DEL PROBLEMA

Para FGA Fondo de Garantías es de vital importancia disponer oportunamente de los recursos necesarios para el pago de garantías una vez el intermediario realiza los reclamos de garantías, por ende, mensualmente se hace una la provisión de dinero con base en el comportamiento de los reclamos por intermediario. El disponer de un modelo que ayude a conocer a priori la cantidad y el valor de garantías próximas a siniestrar habilitaría un aprovisionamiento más preciso de dinero, optimizando la utilización de los recursos.

Por otra parte, el costo de capital tributario de provisionar el dinero para el pago de garantías incrementa con el tiempo, es decir, al realizarse el cobro del valor de servicio de fianza, se realizan los pagos de impuestos respectivos y estos no son recuperados hasta que el intermediario financiero hace la reclamación de las garantías, por ende, entre más tiempo pase entre el cobro del valor del servicio de fianza y la reclamación de las garantías, más devaluación presenta el dinero recuperado producto de impuestos. En consecuencia, entre más oportuna sea la reclamación por parte del intermediario, este costo es reducido.

Una vez el intermediario financiero reclama a FGA Fondo de Garantías una garantía, la obligación del cliente del cliente final es subrogada a FGA e inicia el proceso de reclamación, con base en la experiencia adquirida y respaldado por los datos, entre menos tiempo pase entre la otorgación de los créditos y la reclamación de las garantías, es más probable que estos recursos sean recuperados.

Finalmente, en FGA Fondo de Garantías se busca constantemente generar valor a los intermediarios financieros, por tanto, brindarles información para una oportuna toma de decisiones es muy importante.

A. Problema de negocio

Los datos recolectados por FGA Fondo de Garantías, son reportados por los intermediarios financieros. Cuando una garantía es otorgada, dependiendo del convenio realizado entre el intermediario y FGA, se definen los campos asociados a la garantía y los campos demográficos del deudor a ser reportados, dado que cada convenio difiere, no se disponen de los mismos campos para todos los deudores. Por otra parte, comercialmente es complejo que los intermediarios accedan

a reportar toda la información demográfica de la cual disponen. Esto ha dificultado que en FGA se pueda hacer análisis descriptivos y predictivos que generen mayor valor al negocio.

Sin embargo, si se dispone de los comportamientos de cartera de los clientes y de este se obtendrán las principales variables que alimentan el modelo construido.

B. Aproximación desde la analitica de datos

Actualmente se dispone de la historia de cartera de todos los intermediarios, y por decisión del negocio el proyecto se realizará con los datos de uno del sector de retail. Se extrajeron los datos del periodo de tiempo comprendido entre enero de 2019 y junio de 2019, donde se encuentran carteras maduras las cuales ya tienen un estado final de siniestrado o pagado totalmente.

Posteriormente, se realizó un análisis exploratorio de las variables, buscando entender el dataset, limpiarlo e identificar posibles variables estrechamente relacionadas con la variable objetivo. Luego, con la ayuda del negocio se realizó un proceso de ingeniería de variables donde se crearon variables las cuales con base en la experiencia del negocio son de utilidad para identificar las garantías a siniestrar.

Finalmente, se probaron diferentes algoritmos de clasificación y se seleccionó aquel que entregó un mejor desempeño en la métrica definida. Durante los meses posteriores a su puesta en producción se hará monitoreo del mismo, buscando generar métricas de desempeño y una referencia para ejercicios posteriores, dado que es la primera vez que se realiza un ejercicio de este tipo en FGA.

C. Origen de los datos

Los datos fueron extraídos del Datawarehouse, utilizando el modelo relacional de la compañía y la relación entre sus entidades, las cuales se verán en detalle en la próxima sesión. Para esta extracción se decidió en conjunto con el negocio tomar aquellas garantías desembolsadas entre el 1 de enero de 2019 y el 30 de junio de 2019, con el fin de mitigar el efecto de la pandemia asociada al COVID-19 en el comportamiento de pago de los clientes finales y de disponer de garantías maduras, en estados finales como canceladas o siniestradas.

Los datos son propiedad de FGA Fondo de Garantías y solo está permitido el uso de ellos por los autores de esta monografía y con fines académicos.

III. DATOS

A. Datos originales

Como base de datos para el desarrollo del proyecto se dispondrá, de lo que al interior de la compañía se conoce como las tablas de cartera nit, de sp_garantias, y de otras varías maestras que permiten traducir algunos estados.

TABLA I
TABLAS, COLUMNAS, Y DESCRIPCIÓN DE LOS DATOS ORIGINALES

Tabla	**Columna**	**Descripción**
sp_garantias	id_garantia	Identificador único de las garantías
sp_garantias	perfil	Perfil del deudor
sp_garantias	ocupacion	Ocupación del deudor
sp_garantias	ciudad	Ciudad de origen del deudor
sp_garantias	almacen	Almacén donde fue desembolsado el crédito
sp_garantias	negocio	Negocio del intermediario donde fue desembolsado el crédito
sp_garantias	estrato	Estrato del deudor
sp_garantias	nombre_producto	Clasificación del intermediario en la cual fue otorgada el producto
sp_garantias	estado_civil	Estado civil del deudor
sp_garantias	regional	Regional a la que pertenece el deudor según el intermediario
sp_garantias	genero	Generó del deudor
sp_garantias	segmentacion	Segmentación según el intermediario
sp_garantias	nivel_academico	Nivel académico del deudor
sp_garantias	nivel_ingresos	Nivel de ingresos del deudor en la SMLMV
sp_garantias	cod_estado_garantia	Código del estado de la garantía
sp_garantias	cal_riesgo_obligacion	Calificación del riesgo de la obligación según el intermediario

sp_garantias	vlr_monto_des	Valor del monto desembolsado
sp_garantias	convenio	Convenio entre el intermediario y FGA bajo el cual se entregó la garantía
sp_garantias	fecha_desembolso	Fecha en la cual se desembolsó el crédito
sp_garantias	fecha_nacimiento	Fecha de nacimiento del deudor
sp_cartera_nit	id_cartera	Id que identifica cada deporte de cartera en un mes de corte y asociado a una garantía
sp_cartera_nit	nit_intermediario	Nit del intermediario
sp_cartera_nit	id_cliente	Id único de los deudores
sp_cartera_nit	pagare	Identificador único del crédito para el intermediario
sp_cartera_nit	saldo_capital	Saldo capital a la fecha de corte de la fila
sp_cartera_nit	saldo_total	Saldo_total a la fecha de corte de la fila
sp_cartera_nit	fecha_corte	Fecha de corte del reporte de cartera
sp_cartera_nit	num_cuotas_mora	Número de cuotas en mora
sp_cartera_nit	fec_inicio_mora	Fecha de inicio de la mora
sp_cartera_nit	fecha_cancelacion	Fecha de cancelación total de la garantía
sp_cartera_nit	estado_operacion	Estado de la operación
sp_cartera_nit	mes_cartera	Mes de la fecha de corte
sp_cartera_nit	fecha_pago	Fecha de pago de la garantía por parte de FGA en caso de siniestrarse
sp_cartera_nit	total_pagado	Total pagado por parte de FGA
sp_cartera_nit	archivo	Nombre del archivo con el que se cargó la cartera
sp_cartera_nit	anio_corte	Año de la fecha de corte
sp_cartera_nit	mes_corte	Mes de la fecha de corte

productos	producto	Producto del convenio entre FGA y el intermediario bajo el cual se otorgó la garantía
convenios	mora_minima	Mora mínima en días que debe tener la garantía para ser reclamada
convenios	plazo_max_reclamacion	Mora máxima en días que debe tener la garantía para ser reclamada
sp_estado_garantia	estado_garantia	Estado de la garantía

B. Datasets

A partir de las tablas previamente nombradas, se construyó una ETL en Tableau Prep, donde se consolidaron todos los campos en un extracto de Tableau cuya extensión es .hyper, formato estandarizado en la compañía para las fuentes de datos.

En primer lugar la ETL, mediante un query de SQL une todas las entidades mencionadas en el apartado “DATOS ORIGINALES”, en el paso siguiente se corrigieron tipos de datos erróneamente clasificados por el software, se anonimiza la identidad del intermediario, y finalmente se guarda en una ubicación previamente determinada el extracto .hyper de Tableau.

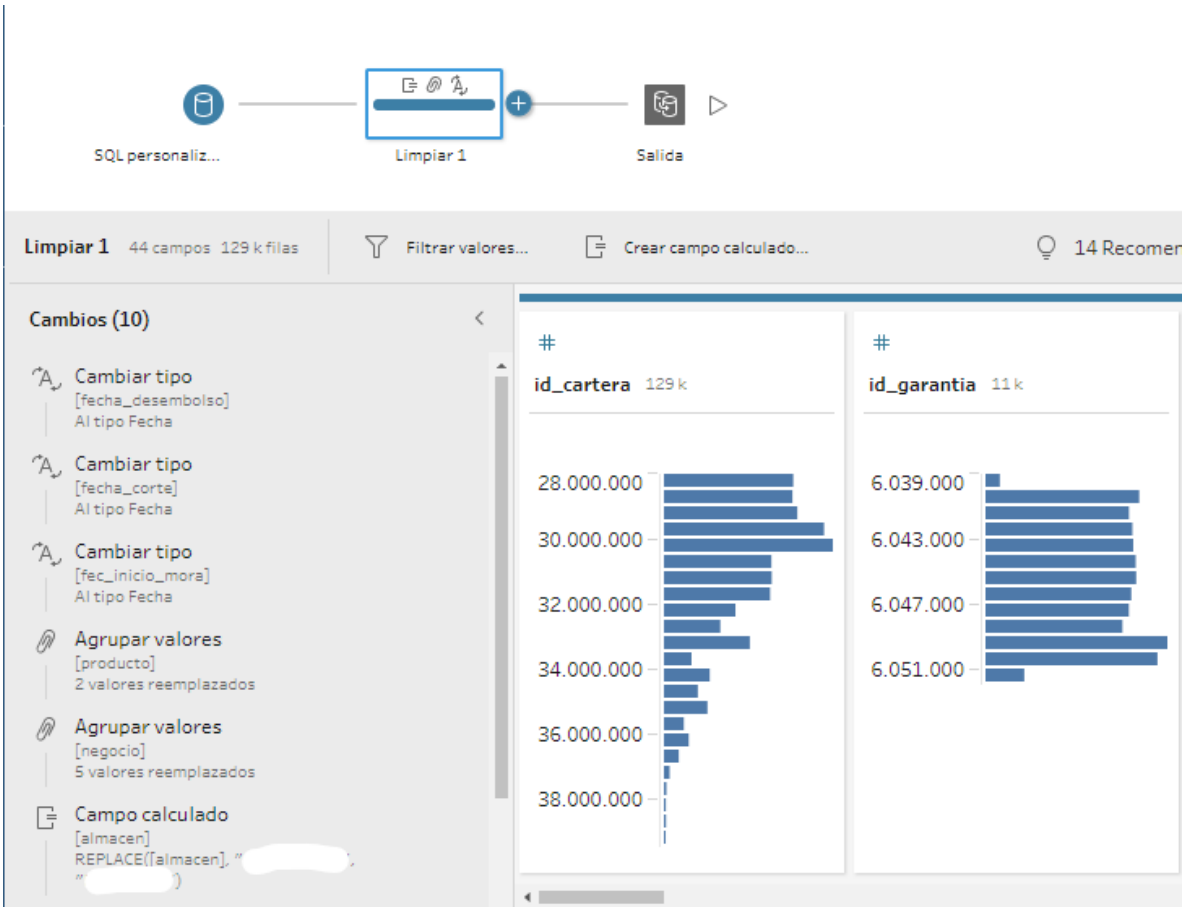


Fig. 1. ETL para la extracción inicial de los datos

C. Descriptiva

El análisis exploratorio de los datos, evidenciado en el notebook "*Proyecto Garantías a Siniestrar.ipyn*" con un mayor detalle, toma como base el archivo "*historico_cartera_clientexgarantia.hyper*" el cual consta de 2.008.004 registros, los cuales describen el comportamiento de cartera de 168.382 garantías.

El análisis está segmentado en dos partes, la inicial donde se verificaron las columnas, se eliminó aquellas que por su naturaleza no agregan valor al modelo, se verificó que no hubiese duplicados, se reemplazan valores vacíos por nulos, se eliminaron las columnas cuya proporción de nulos sea superior al 50% y finalmente, se verificaron los posibles valores en las variables categóricas con el fin de normalizarlos cuando aplique.

La segunda parte, se realiza después de tener la ingeniería de variables realizada, y cuando el nivel de detalle es una fila por cada garantía. Se inició con las variables categóricas. En primer lugar, se exploran las ciudades.

1) Ciudad:

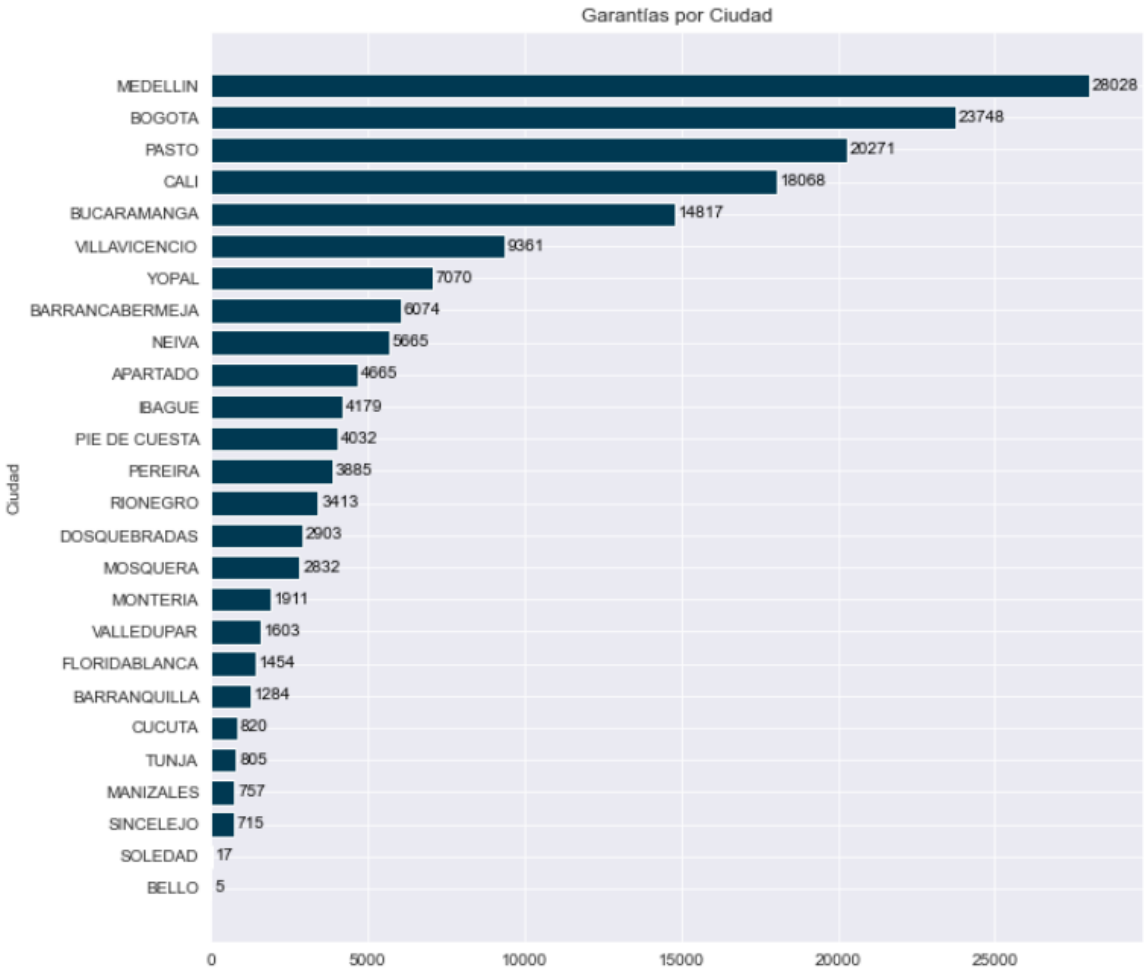


Fig. 2. Garantías por ciudad

Dónde se evidenció que las ciudades donde más se desembolsaron créditos es en Medellín, seguido por Bogotá y demás ciudades principales. Ahora en la siguiente Figura se evidencia el porcentaje de participación de cada ciudad respecto a las garantías siniestradas.

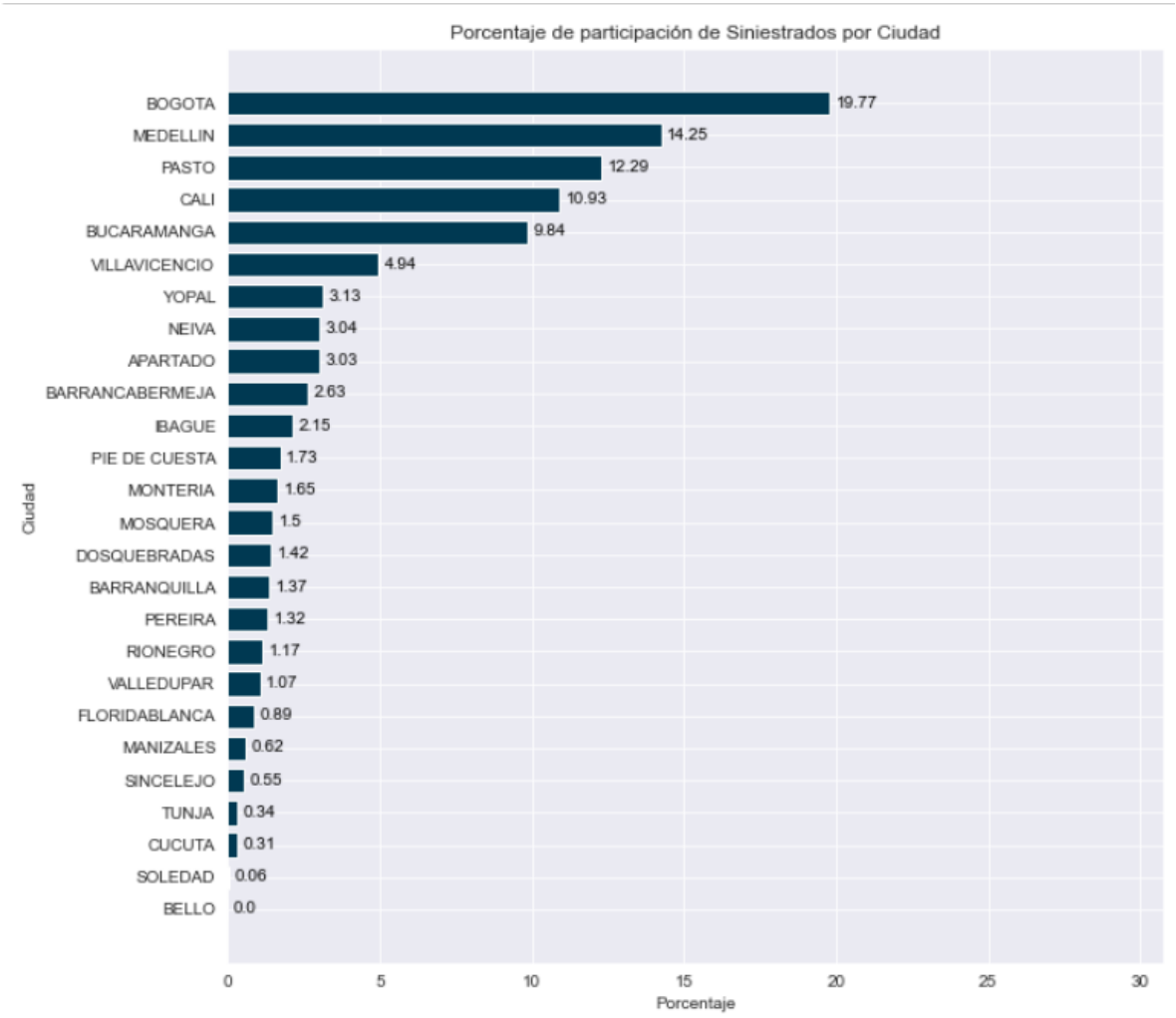


Fig. 3. Porcentaje de participación de siniestrados por Ciudad.

Aunque es proporcional a la cantidad de créditos desembolsados, se ve que el primer lugar es ocupado por Bogotá, así en Medellín sea donde más se desembolsen créditos.

2) Negocio:

El intermediario de interés segmenta los créditos en diferentes negocios.

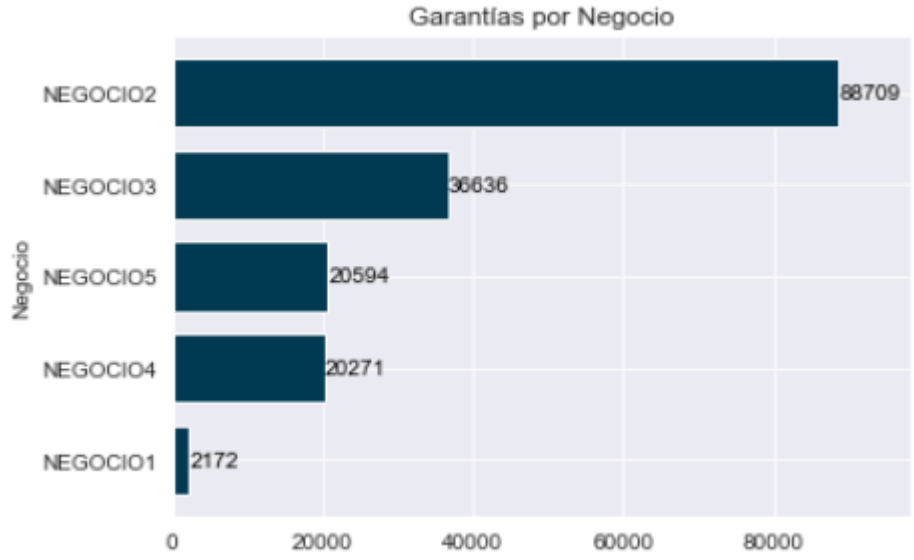


Fig. 4. Garantías por negocio

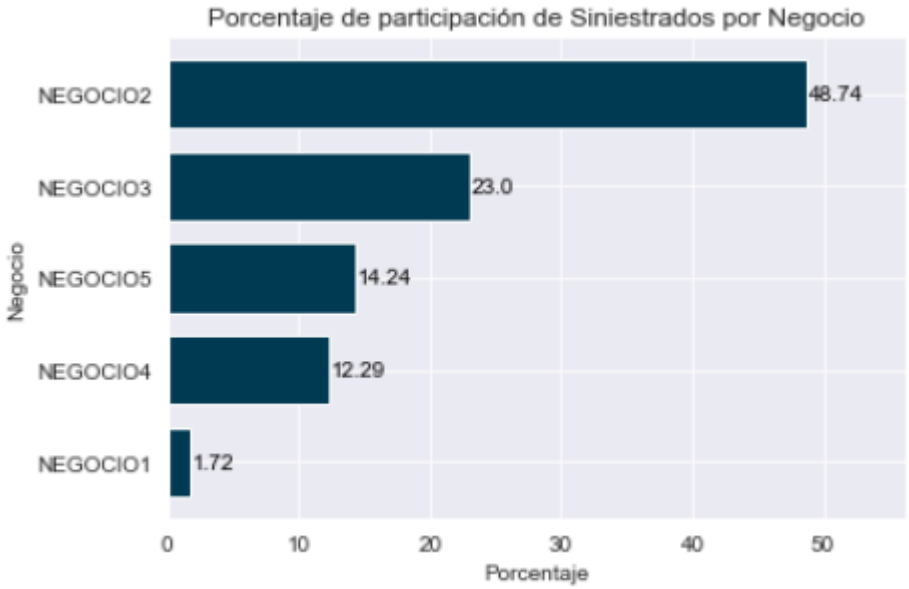


Fig. 5. Porcentaje de partición de siniestrados por negocios

Aquí se evidencia que las proporciones de siniestralidad se conservan con respecto a la asignación de créditos.

3) Almacén:

El almacén está altamente correlacionado con el negocio, dado que en un almacén generalmente se desembolsan créditos de solo un negocio, máximo dos. Del mismo modo, los almacenes dependen de la ciudad, como podemos ver en la Figura 6.

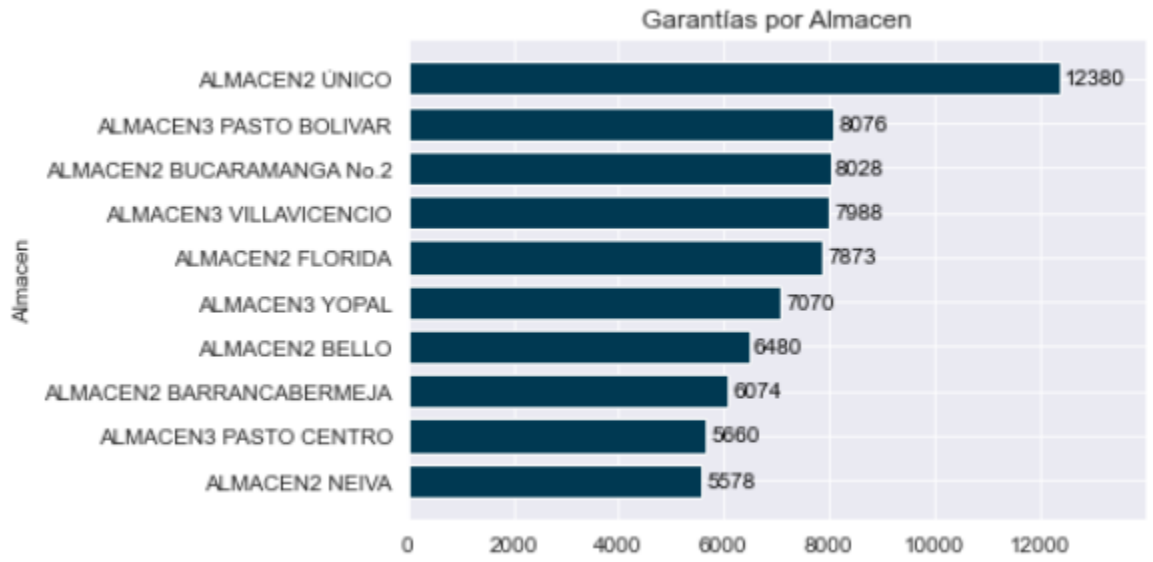


Fig. 6. Top 10 almacenes por número de garantías

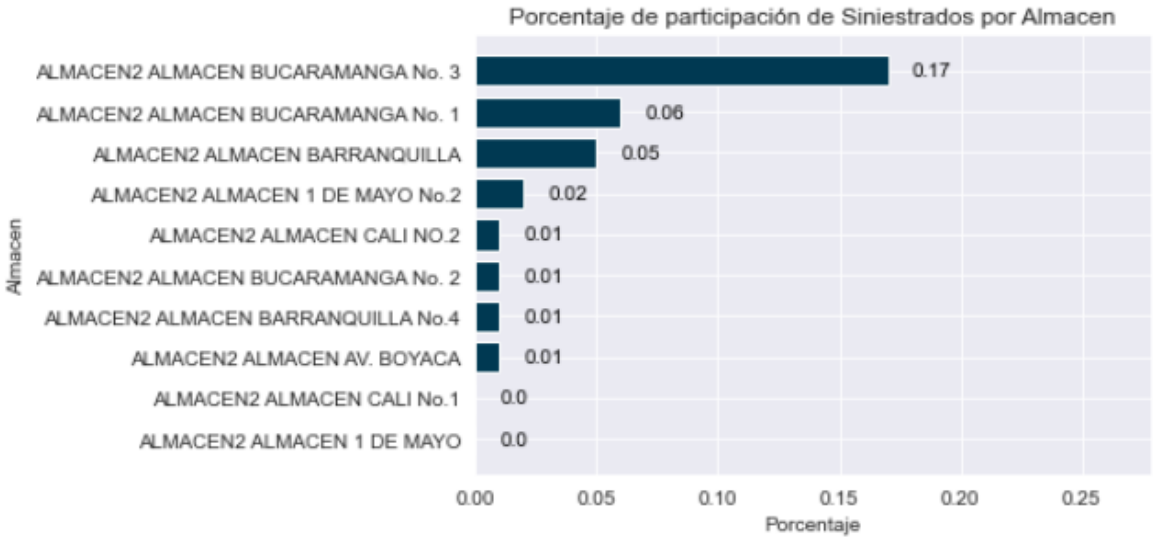


Fig. 7. Top 10 almacenes con mayor participación en siniestrados

Se evidencia que los almacenes donde más se desembolsa, no son aquellos donde se tiene una mayor siniestralidad.

4) Perfil:

El perfil es la profesión reportada por el deudor al momento de solicitar el crédito

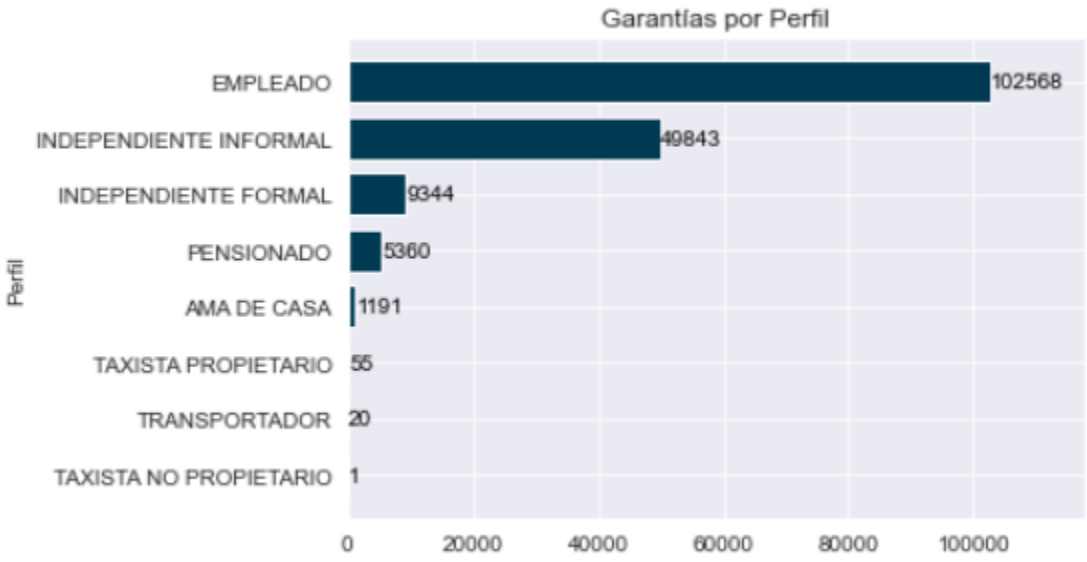


Fig. 8. Garantías por perfil

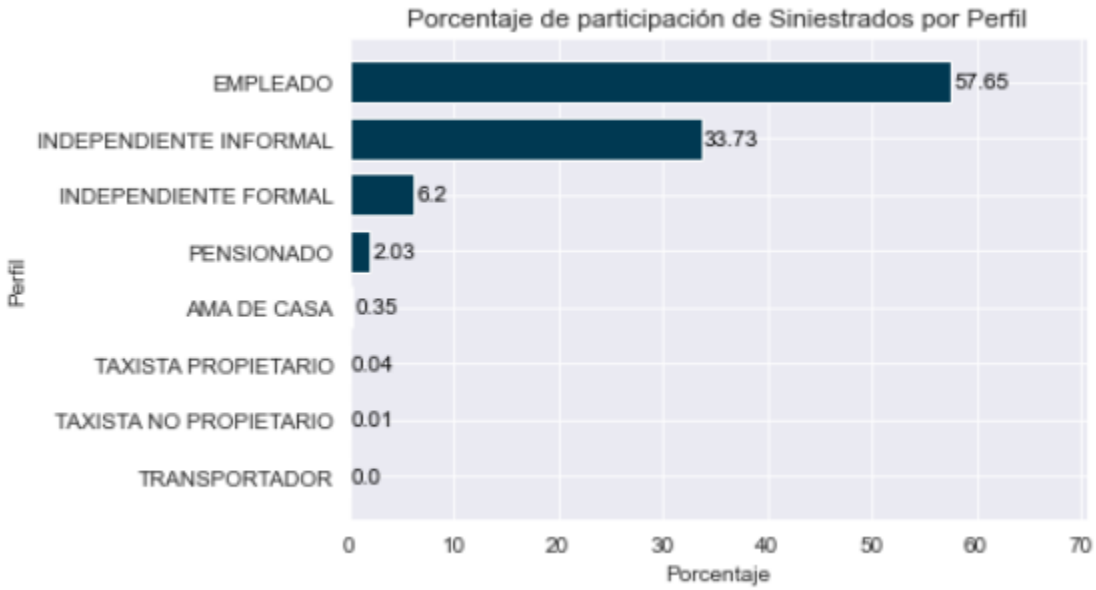


Fig. 9. Porcentaje de participación de siniestrados por perfil

Se evidencia que las proporciones se mantienen, en cuanto a créditos desembolsados y porcentaje de participación en los siniestrados, es notable que la barra de pensionados es menor en participación de siniestrados que en garantías desembolsadas

5) Producto:

El producto es una definición entre el intermediario financiero y FGA Fondo de Garantías, entre productos se cobran diferentes valores de servicio de fianza, esto depende de la cantidad de siniestrados esperados por la naturaleza específica del crédito, o producto.

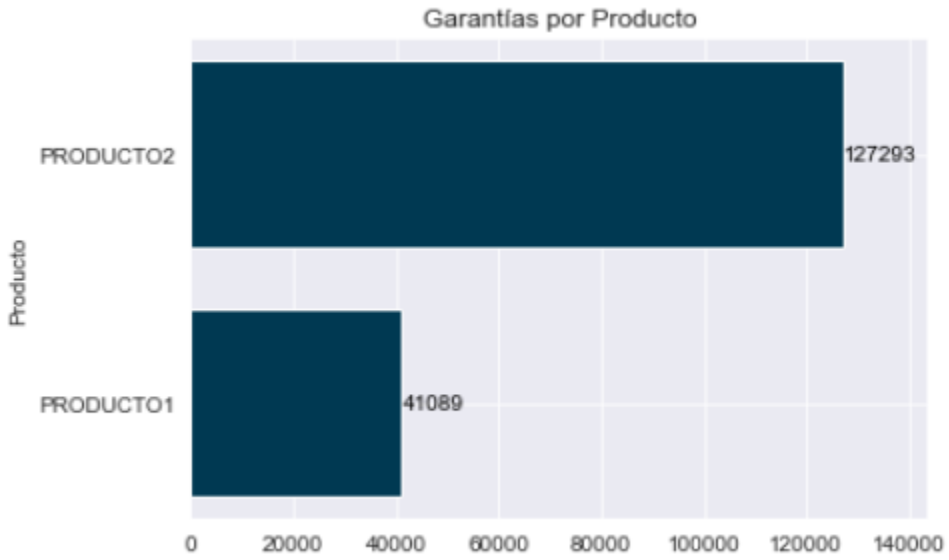


Fig. 10. Garantías por producto

Fig. 11. Porcentaje de participación de siniestrados por productos

IV. PROCESO DE ANALÍTICA

A. Preprocesamiento

En la etapa de procesamiento se tomó la salida de la etl, y se realizó el análisis exploratorio respectivo, construyendo las variables que por experiencia del negocio fueron sugeridas y consideradas relevantes para la predicción del modelo.

El nivel de detalle del dataset al inicio de la exploración es una fila por cada fecha de corte y garantía, la cual describe el comportamiento de cartera, saldo total, saldo capital, y si entró o no en mora para esa fecha de corte. Lo que se buscó es generar variables que resuman los comportamientos de cartera y se asignarlas como un atributo más de las garantías, por lo tanto, al final de este preprocesamiento, el nivel de detalle es de una fila por cada garantía.

1) Identificación de variables:

- Veces en mora: Veces en mora describe cuántas veces una garantía entró en mora, a continuación en la Figura 12 se evidencia su distribución cuando la garantía se siniestró y cuando no.

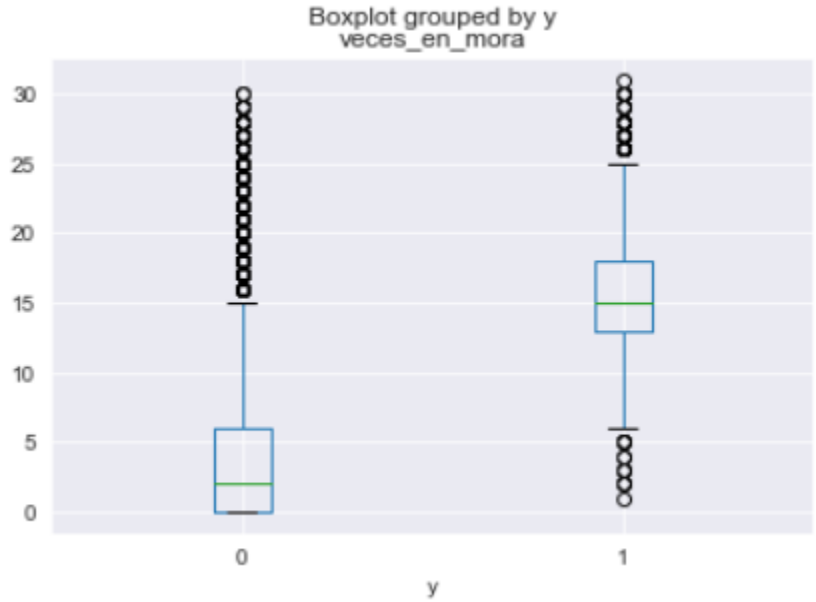


Fig. 12. Distribución de la variable veces en mora para siniestrados y no siniestrados

Se puede ver que cuando la garantía se siniestra, generalmente entra un mayor número de veces en mora, 15 en promedio.

- Veces saldo total mayor a desembolso: Que en una fecha de corte el saldo total sea mayor al desembolso se da cuando los intereses son los suficientes para hacer que se deba más de lo que se prestó, se suele dar en dos circunstancias, cuando se entra en mora desde la primera fecha de corte, o cuando después de haber cumplido los primeros meses se entra en mora durante los suficientes meses consecutivos para que los intereses vuelvan a superar el valor desembolsado. A continuación, en la Figura 13 se puede ver sus distribuciones en garantías siniestradas y no siniestradas

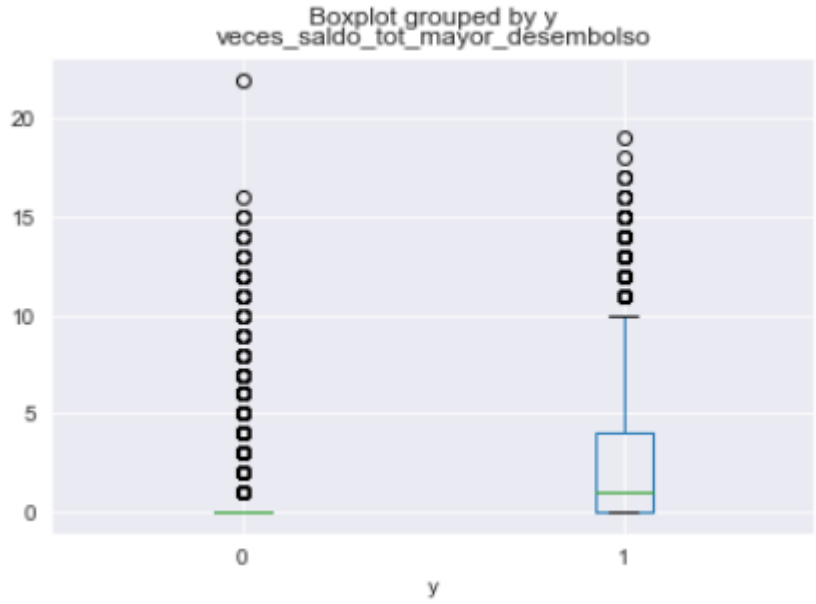


Fig. 13. Distribución de la variable saldo total mayor a desembolso para siniestrados y no siniestrados

- Veces cero pagos: Veces cero pagos es calculado como la cantidad de fechas de corte en las cuales el saldo capital es igual al valor desembolsado, es decir, son aquellas garantías las cuales desde la primera fecha de corte entran en mora.

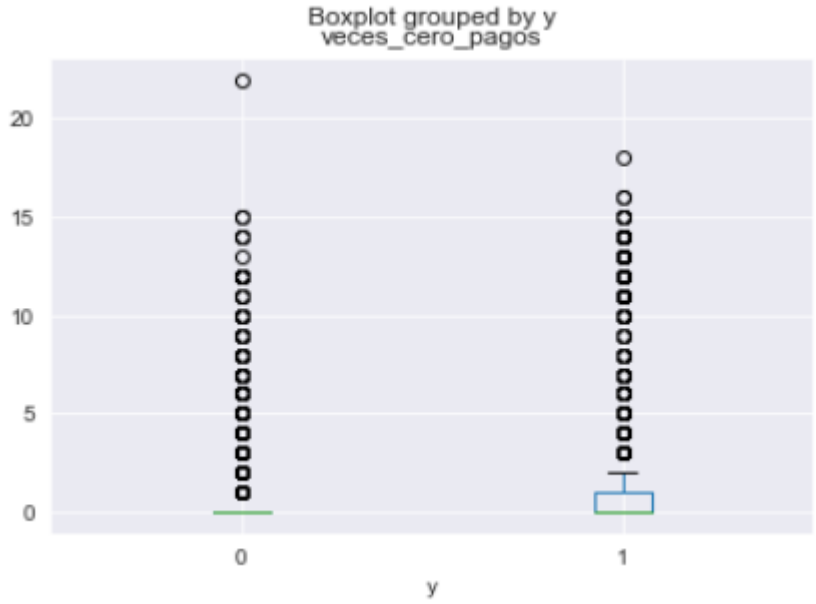


Fig. 14. Distribución de la variable saldo total mayor a desembolso para siniestrados y no siniestrados

- Moras continuas: Desde el negocio, surgieron dos definiciones para la continuidad en las moras, una es cuando dos o más moras se dan contiguas, y la otra es cuando o una o más moras de dan contiguas y ambas fueron generadas, la primera se llama “moras_continuas” y la segunda “moras_continuas_2”, a continuación, se ven sus distribuciones para garantías siniestradas y no siniestradas.

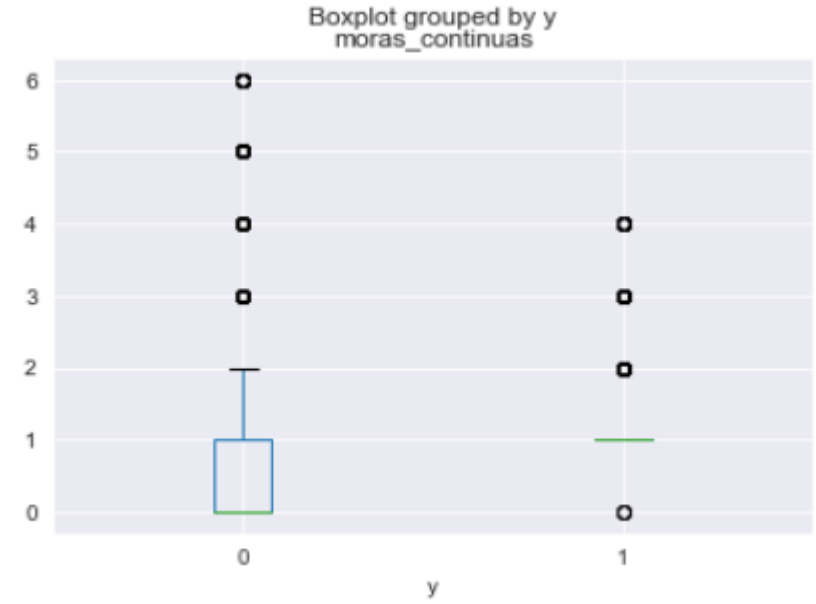


Fig. 15. Distribución de la variable moras continuas para siniestrados y no siniestrados.

Aquí se evidencia que las garantías siniestradas generalmente solo tienen una consecutividad de dos o más moras.

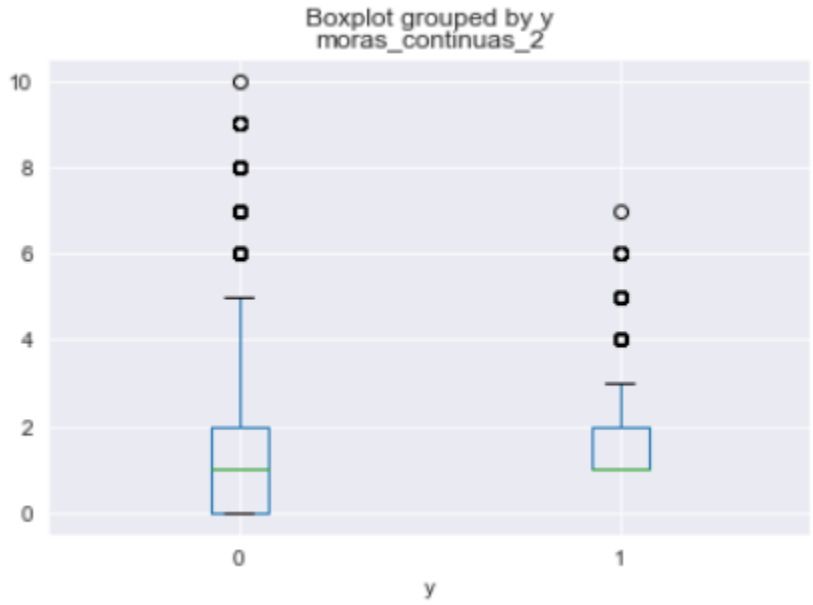


Fig. 16. Distribución de la variable moras continuas 2 para siniestrados y no siniestrados.

B. Modelos

Dado que el problema es de clasificación, por su buen desempeño en este tipo de problemas se eligieron tres algoritmos de la familia de los árboles; RandomForestClassifier, GradientBoostingClassifier y AdaBoostClassifier. Fueron implementados mediante la librería de Sklearn.

```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier
from sklearn.model_selection import train_test_split, StratifiedShuffleSplit, GridSearchCV
from sklearn.metrics import f1_score, accuracy_score, plot_confusion_matrix, precision_score, recall_score
```

Fig. 17. Librerías empleadas para el desarrollo de los modelos.

1) RandomForestClassifier:

El RandomForestClassifier es un meta estimador que entrena un de árbol de decisión en varios subconjuntos del conjunto de datos y usa el promedio para mejorar la precisión de la predicción y controlar el sobre ajustamiento del modelo. A este modelo se le varió la cantidad de

estimadores, es decir la cantidad de árboles empleados, el criterio de bifurcación y criterio para definir el número de variables a considerar cuando se está buscando por la mejor bifurcación del árbol.[1]

2) GradientBoostingClassifier:

El GradientBoostingClassifier construye un modelo aditivo de manera progresiva por etapas; permite la optimización de funciones de pérdida diferenciables arbitrarias. En cada etapa, n_classes_ árboles de regresión se ajustan al gradiente negativo de la función de pérdida de desviación binomial o multinomial.
A este modelo le varió la cantidad de estimadores, el criterio que mide la calidad de una bifurcación, y la función de pérdida a ser optimizada.[1]

3) AdaBoostClassifier:

AdaBoostClassifier es un meta estimador que comienza ajustando un clasificador en el conjunto de datos original y luego ajusta copias adicionales del clasificador en el mismo conjunto de datos, pero donde los pesos de instancias clasificadas incorrectamente se ajustan de modo que los clasificadores posteriores se enfocan más en casos difíciles.[1]

C. Métricas

En algoritmos de clasificación se dispone de diferentes métricas de desempeño tales como la exactitud, exhaustividad, precisión, puntaje f1 y la matriz de confusión. Sklearn habilita la implementación de todas estas medidas de una manera fácil y rápida.

1) Matriz de confusión o error:

De acuerdo con la documentación, por definición, una matriz de confusión C es tal que C_{ij} es igual al número de observaciones que se sabe que están en el grupo i y que se predice que están en el grupo j. Por ende, en clasificación binario, el conteo de verdaderos negativos (TN) es $C_{0,0}$, falsos negativos (FN) $C_{1,0}$, verdaderos positivos (TP) es $C_{1,1}$, y falsos positivos (FP) es $C_{0,1}$.[2]

2) Exactitud:

La exactitud, también muy conocida por su nombre en inglés: accuracy, es la medida más intuitiva y directa de la calidad de un clasificador, y básicamente es el total de elementos clasificados correctamente sobre el total de elementos.[2]

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (1)$$

3) Precisión, Exhaustividad y medidas F:

De acuerdo con Sklearn, intuitivamente, precisión (precision en inglés) es la capacidad del clasificador de no etiquetar como positiva una muestra negativa, exhaustividad (recall) es la capacidad del clasificador de encontrar todas las muestras positivas.[2]

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

Por otra parte, el puntaje F1 (F1 score) puede ser interpretado como la media armónica de la precisión y la exhaustividad.[2]

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

Para la evaluación del modelo, se usó el puntaje F1 dado que nos facilita comparar los rendimientos tanto de la precisión como de la exhaustividad en un solo indicador.

V. METODOLOGÍA

A. Baseline

Como baseline para el modelo se empleó un RandomForestClassifier, para el cual se usaron todas las variables categóricas, sólo se especifica el parámetro class_weight como 'balanced_subsample' para indicarle al algoritmo que la distribución de la variable objetivo es desbalanceada, no se realiza búsqueda de híper-parámetros.

```
df_baseline = df.copy()
#Realizamos el proceso de one_hot encoding
for i in ['producto', 'perfil', 'ciudad', 'almacen', 'negocio']:
    df_baseline = df_baseline.merge(pd.get_dummies(df_baseline[i], prefix=i), left_index=True, right_index=True).copy()
#Eliminamos las variables posterior al one hot encoding
df_baseline.drop(['producto', 'perfil', 'ciudad', 'almacen', 'negocio', 'id_garantia'], axis=1, inplace=True)
#División del dataset el train y test
X_train, X_test, y_train, y_test = train_test_split(df_baseline.loc[:, df_baseline.columns != 'y'], df_baseline['y'], test_size=0
#Definición del modelo
rfc = RandomForestClassifier(class_weight='balanced_subsample')
#Entrenamiento del modelo
rfc.fit(X_train,y_train)
#Predicciones
train_predictions = rfc.predict(X_train)
test_predictions = rfc.predict(X_test)
#Matriz de confusion
plot_confusion_matrix(rfc, X_test, y_test, cmap='Blues', colorbar=False)
plt.show()
#Impresión de los resultados
print(rfc)
print('f1_score: ', round(f1_score(y_test, test_predictions), 3))
print('accuracy_score: ', round(accuracy_score(y_test, test_predictions), 3))
print('precision_score: ', round(accuracy_score(y_test, test_predictions), 3))
print('recall_score: ', round(recall_score(y_test, test_predictions), 3))
```

Fig. 18. Código del baseline

B. Iteraciones y evolución

Para poder realizar las iteraciones de una forma óptima, y evitar la constante reutilización de código se construyó la función “probar_modelos”, la cual se describe en detalle en la sección 5.2.1. Una vez realizada la función, se realizan diversas iteraciones, evaluando el error con base en la importancia de las variables para cada modelo.

1) Función probar_modelos

Parámetros:

df: *dataframe de pandas, default = None*

Dataset completo, a procesar y dividir para entrenar los modelos.

one_hot: ***bool, default = False***

Booleano que indica si se realizará one hot encoding sobre el conjunto de datos.

one_hot_list: ***list, default = []***

Lista con los nombres de las columnas a las cuales se les aplicará el proceso de codificación one-hot.

drop_list: ***list, default = ['producto', 'perfil', 'ciudad', 'almacen', 'negocio', 'id_garantia']***

Listo de columnas a eliminar del dataframe de Pandas.

Ahora, se detalla el funcionamiento de la función; inicialmente, se crea una copia del dataframe recibido, con el fin de no afectar el objeto recibido. Posterior a esto, con base en el parámetro *one_hot* y *one_hot_list* se realiza el proceso de codificación *one-hot.* Para finalizar, se borran las columnas iniciales, dado que no tienen una codificación apropiada y no son entendidas por los algoritmos de aprendizaje.

```
#realizamos una copia del dataframe
df_model = df.copy()
#realizamos el proceso de one_hot encoding
if one_hot:
    for i in one_hot_list:
        df_model = df_model.merge(pd.get_dummies(df_model[i], prefix=i), left_index=True, right_index=True).copy()
df_model.drop(drop_list, axis=1, inplace=True)
```

Fig. 19. Parte 2 de la función que entrena los modelos

En este punto, se realiza la división del conjunto de datos en datos para entrenamiento y datos para prueba, indicando a la función que el conjunto de datos es desproporcionado. Seguidamente, se instancia un validador cruzado que tiene en cuenta las proporciones de la variable objetivo.

```
#divimos el dataset en entrenamiento y pruebas
X_train, X_test, y_train, y_test = train_test_split(df_model.loc[:, df_model.columns != 'y'], \
                                                    df_model['y'], test_size=0.3, stratify=df_model['y'])
st = StratifiedShuffleSplit(n_splits=5, test_size=0.3, random_state=0)
```

Fig. 20. Parte 1 de la función que entrena los modelos

Luego, se instancian los clasificadores, se definen los hiper-parámetros para cada uno, y se realiza el gridsearch para cada modelo.

```
#definimos los modelos y los parámetros para el gridsearch
rfc = RandomForestClassifier(class_weight='balanced_subsample')
gbc = GradientBoostingClassifier()
abc = AdaBoostClassifier()
parameters_rfc = {'n_estimators':[100,120,140], 'criterion':['gini', 'entropy'], 'max_features': ['auto', 'sqrt', 'log2']}
parameters_gbc = {'n_estimators':[100,120,140], 'criterion': ['friedman_mse', 'mse'], 'loss': ['deviance', 'exponential']}
parameters_abc = {'n_estimators':[40,50,60], 'algorithm':['SAMME', 'SAMME.R']}
models = [rfc, gbc, abc]
parameters = [parameters_rfc, parameters_gbc, parameters_abc]

grid_searchs = []
#realizamos la busqueda de los mejores hiperparámentros
for i in range(len(models)):
    gs = GridSearchCV(estimator=models[i], param_grid=parameters[i], cv=st, scoring='balanced_accuracy')
    gs.fit(X_train,y_train)
    grid_searchs.append(gs)
```

Fig. 21. Parte 3 de la función que entrena los modelos

En el siguiente paso, se grafican las matrices de confusión de los 3 algoritmos para una fácil comparación.

```
#gráficamos las matrices de confusión y calculamos los errores en los dataset de entrenamiento y pruebas
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15,10))
for cls, ax in zip(grid_searchs, axes.flatten()):
    train_predictions = cls.best_estimator_.predict(X_train)
    test_predictions = cls.best_estimator_.predict(X_test)
    train_error = f1_score(y_train, train_predictions)
    test_error = f1_score(y_test, test_predictions)
    plot_confusion_matrix(cls.best_estimator_, X_test, y_test, ax=ax, cmap='Blues', colorbar=False)
    ax.title.set_text(type(cls.best_estimator_).__name__)
    txt=f"Train F1 score {train_error}\nTest F1 score: {test_error}"
    ax.text(0.5,-0.3, txt, size=12, ha="center", transform=ax.transAxes)
plt.tight_layout()
plt.show()
```

Fig. 22. Parte 4 de la función que entrena los modelos

Finalmente se imprimen las métricas para cada modelo, se retorna un dataframe con la importancia de cada variable para cada modelo, y una lista con los modelos entrenados.

```
#accuracy_score, precision_score, recall_score
for cls in grid_searchs:
    test_predictions = cls.best_estimator_.predict(X_test)
    print(cls.best_estimator_)
    print('f1_score: ', round(f1_score(y_test, test_predictions), 3))
    print('accuracy_score: ', round(accuracy_score(y_test, test_predictions), 3))
    print('precision_score: ', round(accuracy_score(y_test, test_predictions), 3))
    print('recall_score: ', round(recall_score(y_test, test_predictions), 3))
    print('\n')


importances = pd.DataFrame(list(zip(grid_searchs[0].best_estimator_.feature_importances_, \
                                    grid_searchs[1].best_estimator_.feature_importances_, grid_searchs[2].best_estimator_.feature_importances_)),
           columns =['RandomForest', 'GradientBoosting', 'AdaBoost'], index = X_train.columns)
#retornamos el dataframe con los features importances y los modelos
return importances, grid_searchs
```

Fig. 23. Parte 5 de la función que entrena los modelos

2) Iteraciones:

Además del baseline se realizaron las siguientes iteraciones, en todas se emplean los 3 modelos y se hace búsqueda de hiper-parámetros en cada uno, lo que cambia son las variables empleadas con base en la importancia entregada por cada modelo.

- Iteración 1: Todas las variables categóricas;
- Iteración 2: Sin la variable categórica almacén
- Iteración 3: Sin la variable categórica negocio
- Iteración 4: Sin variables categóricas almacén y negocio
- Iteración 5: Sin variables categóricas

Cabe resaltar que siempre se tuvieron en cuenta las variables numéricas, dado que en la importancia de los modelos quedaban en los primeros puestos.

3) Herramientas:

Para el desarrollo del proyecto se empleó Tableau Prep y Anaconda. Con Tableu Prep realizamos la ETL, que consolida y anonimiza los datos, por otra parte, con Anaconda creamos un ambiente específico para el desarrollo del proyecto, y mediante Jupyter Notebook, instalado dentro de Anaconda y corriendo sobre Python 3.9.4 se realizó el resto del proyecto.

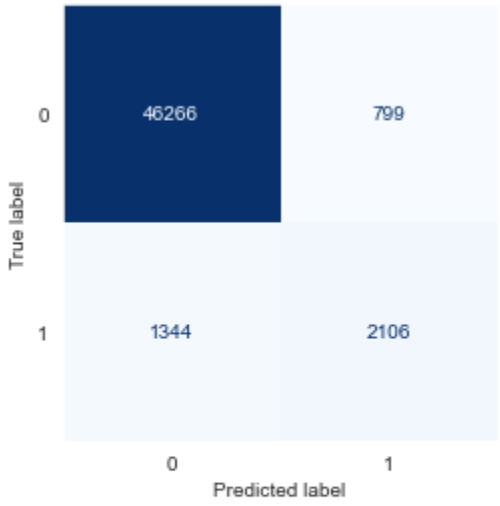
Respecto a las librerías, se empleó Pandas para el procesamiento de los datos, Matplotlib y Seaborn para las visualizaciones, y Sklearn para el modelado.

VI. RESULTADOS

A. Métricas

1) Baseline:

Como se mencionó en la sección 5.1, se empleó como Baseline un RandomForestClassifier, con un único parámetro y es el class_weigth, este modelo retornó un puntaje F1 de 0.659




```
RandomForestClassifier(class_weight='balanced_subsample')
f1_score:  0.663
accuracy_score:  0.958
precision_score:  0.725
recall_score:  0.61
```

Fig. 24. Matriz de confusión para el Baseline

TABLA II
MÉTRICAS OBTENIDAS PARA EL BASELINE

Modelo	Puntaje F1	Exactitud	Precisión	Exhaustividad
Baseline	0.663	0.958	0.725	0.61

2) Iteración 1: Todas las variables categóricas

En esta iteración, el modelo de mejor desempeño el GradientBoostingClassifier, con los siguientes hiper-parámetros:

```
GradientBoostingClassifier(loss='exponential', n_estimators=140)
```

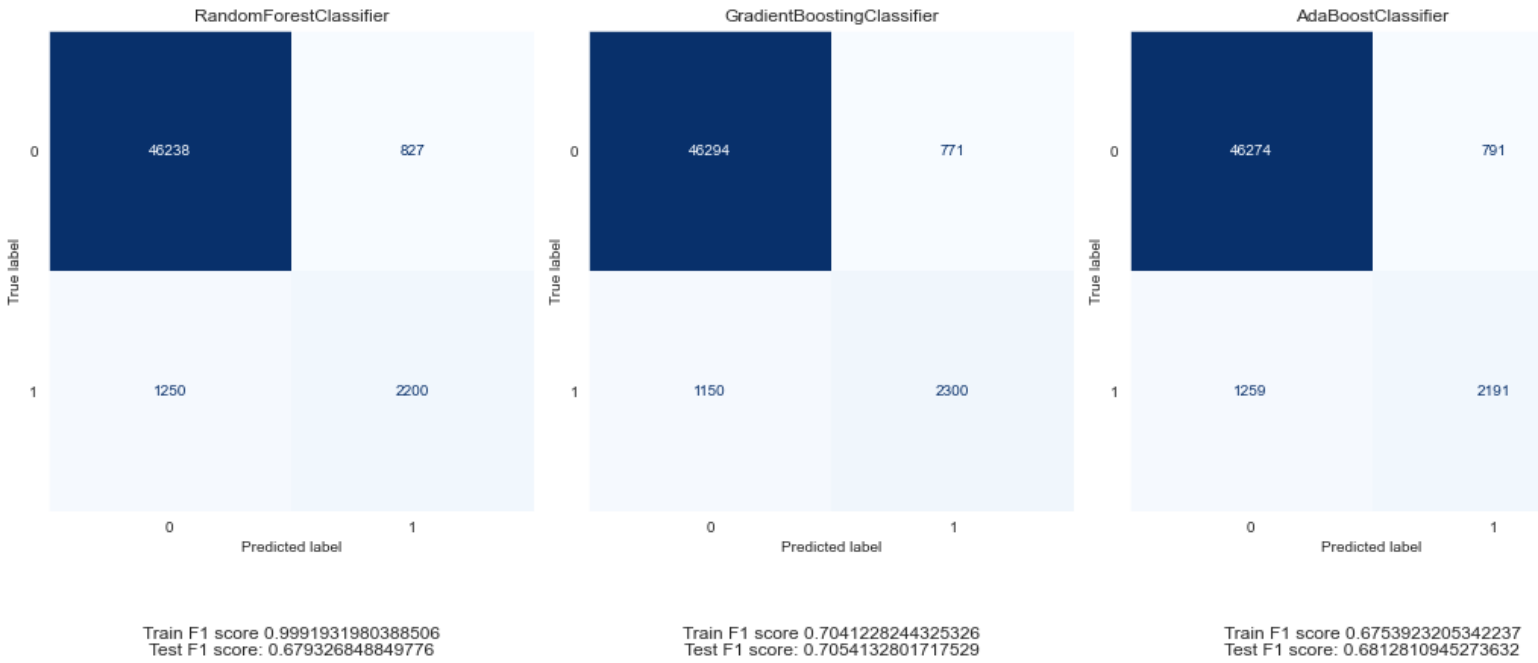


Fig. 25. Matrices de confusión para la primera iteración

TABLA III
MÉTRICAS OBTENIDAS PARA LA PRIMER ITERACIÓN

Modelo	Puntaje F1	Exactitud	Precisión	Exhaustividad
RandomForestClassifier	0.679	0.959	0.727	0.638
GradientBoostingClassifier	0.705	0.962	0.749	0.667
AdaBoostClassifier	0.681	0.959	0.735	0.635

3) Iteración 2: Sin la variable categórica almacén

Para esta iteración el modelo con mejor desempeño el GradientBoostingClassifier con los siguientes hiper-parámetros:

```
GradientBoostingClassifier(loss='exponential', n_estimators=140)
```

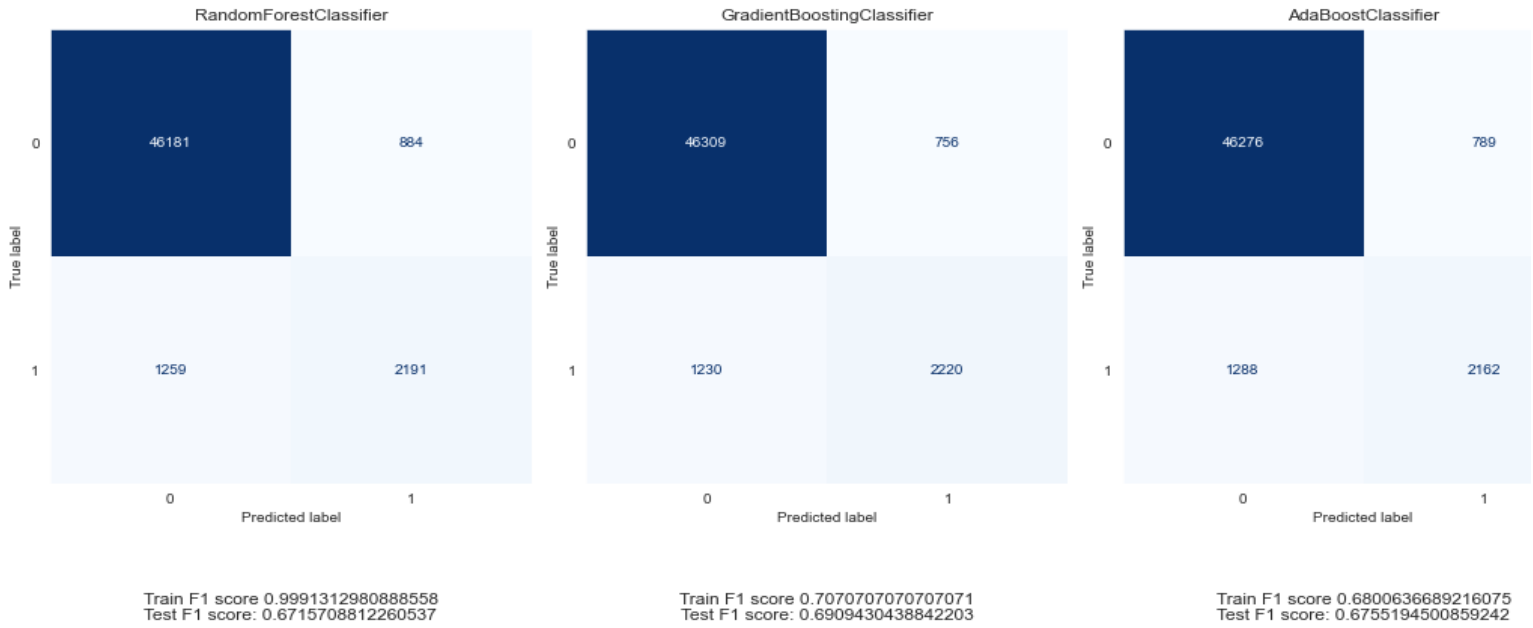


Fig. 26. Matrices de confusión para la segunda iteración

TABLA IV
MÉTRICAS OBTENIDAS PARA LA SEGUNDA ITERACIÓN

Modelo	Puntaje F1	Exactitud	Precisión	Exhaustividad
RandomForestClassifier	0.672	0.958	0.713	0.635
GradientBoostingClassifier	0.691	0.961	0.746	0.643
AdaBoostClassifier	0.676	0.959	0.733	0.627

4) Iteración 3: Sin la variable categórica negocio

Para esta iteración el modelo con mejor desempeño fue el GradientBoostingClassifier con los siguientes hiper-parámetros:

```
GradientBoostingClassifier(loss='exponential', n_estimators=140)
```

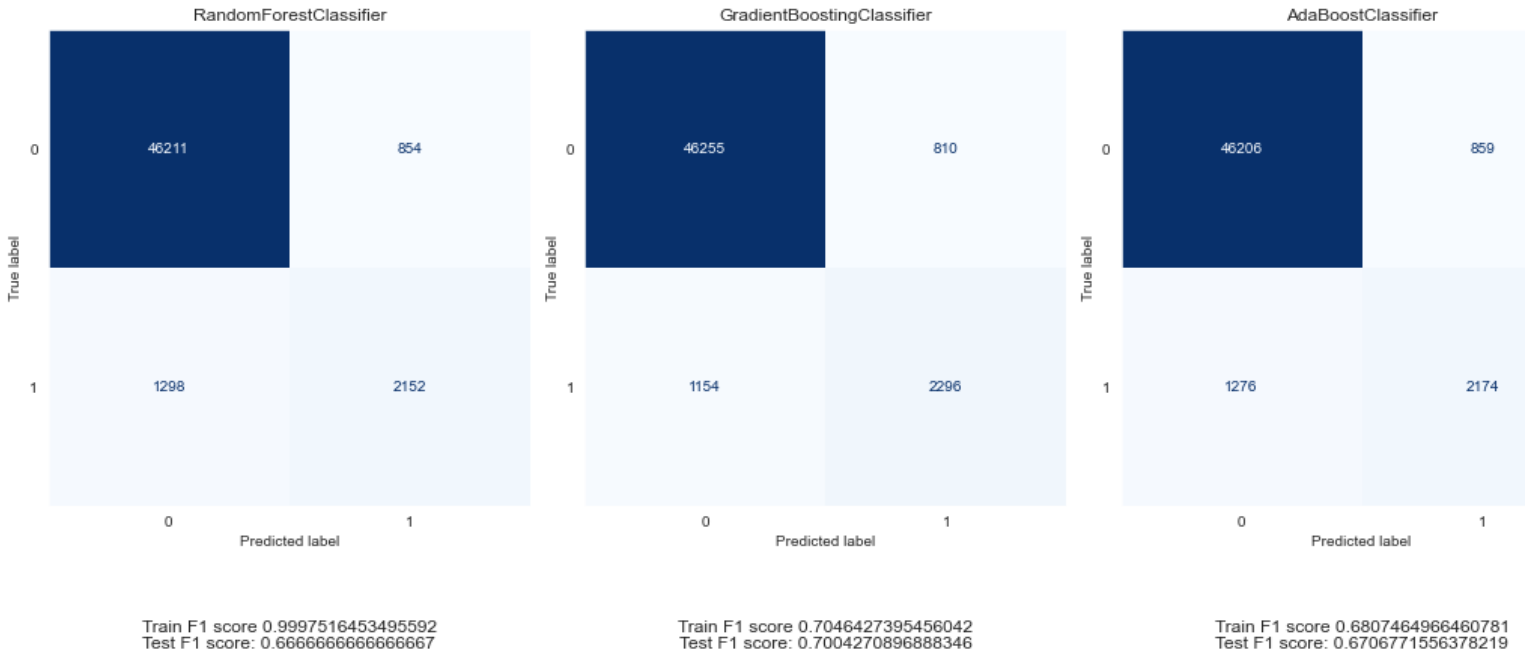


Fig. 27. Matrices de confusión para la tercera iteración

TABLA V
MÉTRICAS OBTENIDAS PARA LA TERCERA ITERACIÓN

Modelo	Puntaje F1	Exactitud	Precisión	Exhaustividad
RandomForestClassifier	0.667	0.957	0.716	0.624
GradientBoostingClassifier	0.7	0.961	0.739	0.666
AdaBoostClassifier	0.671	0.958	0.717	0.63

5) Iteración 4: Sin variables categóricas almacén y negocio

En esta iteración el modelo con mejor desempeño fue GradientBoostingClassifier, con los siguientes hiper-parámetros:

```
GradientBoostingClassifier(loss='exponential', n_estimators=140)
```

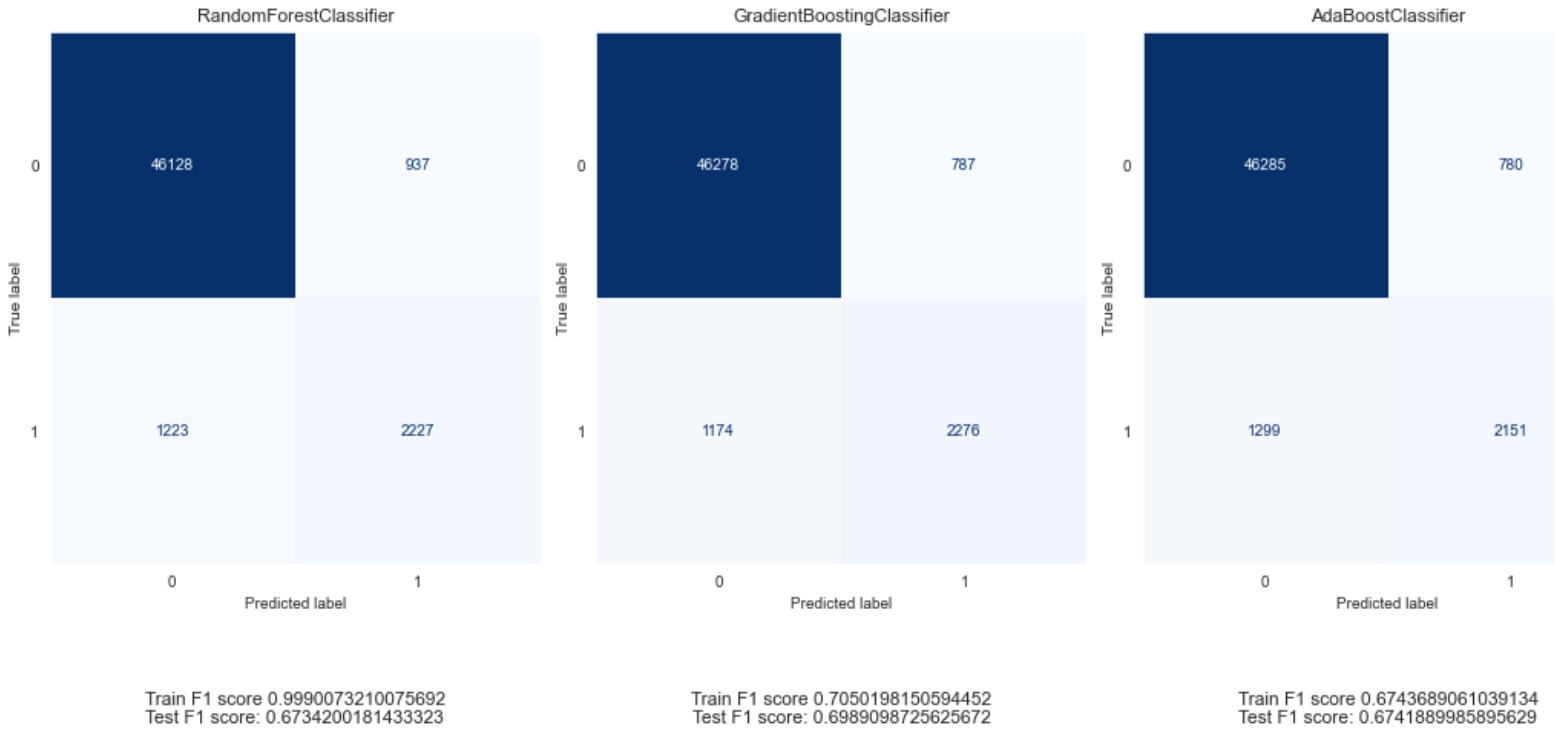


Fig. 28. Matrices de confusión para la cuarta iteración

TABLA VI
MÉTRICAS OBTENIDAS PARA LA CUARTA ITERACIÓN

Modelo	Puntaje F1	Exactitud	Precisión	Exhaustividad
RandomForestClassifier	0.673	0.957	0.704	0.464
GradientBoostingClassifier	0.699	0.961	0.743	0.66
AdaBoostClassifier	0.674	0.959	0.734	0.623

6) Iteración 5: Sin variables categóricas

En esta iteración el modelo con mejor desempeño fue GradientBoostingClassifier, con los siguientes hiper-parámetros:

```
GradientBoostingClassifier(loss='exponential')
```

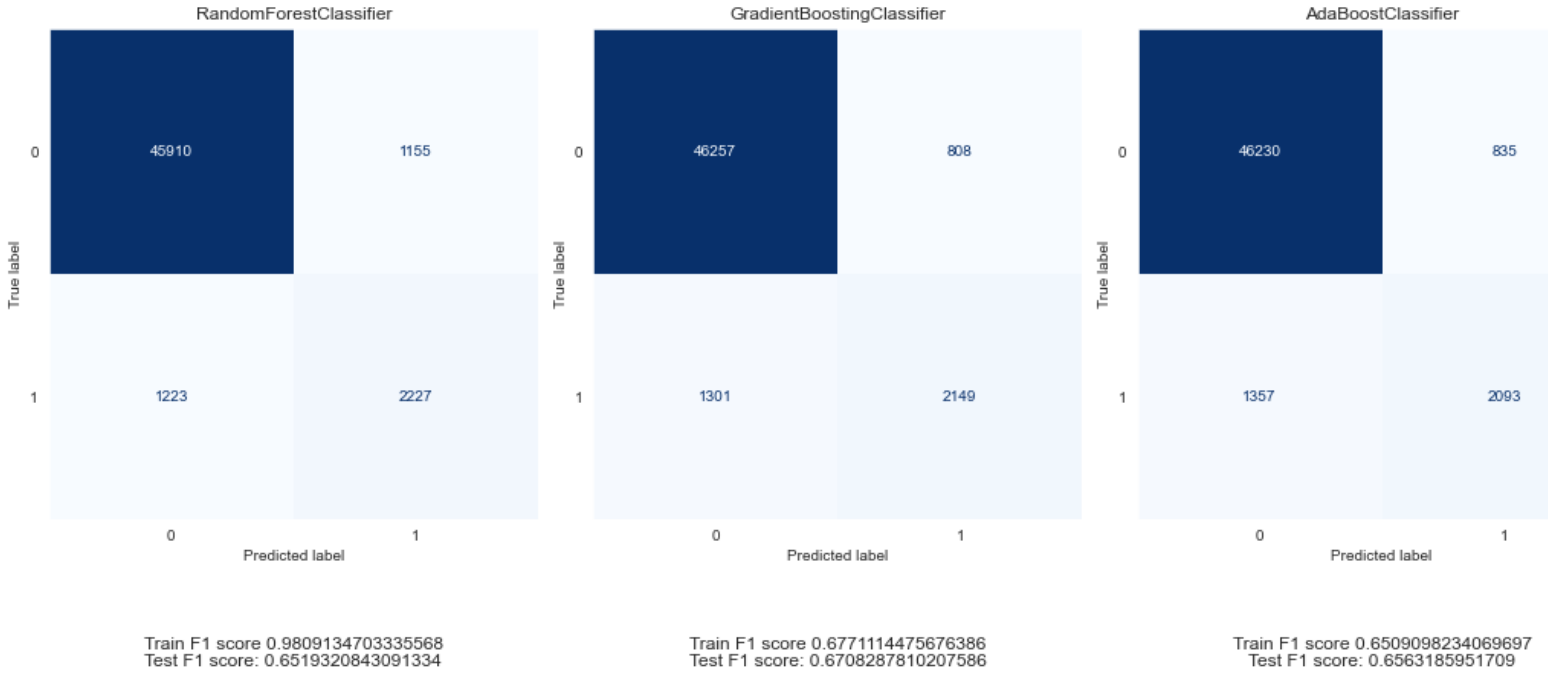


Fig. 29. Matrices de confusión para la quinta iteración

TABLA VII
MÉTRICAS OBTENIDAS PARA LA QUINTA ITERACIÓN

Modelo	Puntaje F1	Exactitud	Precisión	Exhaustividad
RandomForestClassifier	0.668	0.953	0.658	0.646
GradientBoostingClassifier	0.671	0.958	0.727	0.623
AdaBoostClassifier	0.656	0.957	0.715	0.607

B. Evaluación cualitativa

Evidenciamos en las leyendas de las matrices de confusión que el RandomForestClassifier es el algoritmo que tiende a sobre ajustarse pues su desempeño es bueno en los datos de entrenamiento, pero en los datos de test su puntaje F1 ronda 0.6. El GradientBoostingClassifier es quien mejor desempeño tuvo en todas las iteraciones. Por otra parte, los modelos obtenidos en la iteración número 1, 3 y 5 obtuvieron resultados muy similares, aunque su tiempo de búsqueda de hiper-parámetros y entrenamiento difiere sustancialmente. La primera iteración tomó 45 minutos en entrenarse, la tercera 44 minutos, y la quinta 22 minutos. Por ende, se pondrá en producción el modelo GradientBoostingClassifier de la quinta iteración.

C. Consideraciones de producción

Dada la complejidad de los cálculos para la ingeniería de variables y los grandes volúmenes de datos con los que se dispone, se deben tener en consideración los siguientes puntos para el despliegue a producción del modelo:

- Se debe implementar un proceso específico para el preprocesamiento de los datos, el cual debe realizarse sobre un cluster de pyspark, pues las técnicas tradicionales no responderían en los tiempos deseados
- El proyecto se desarrolló sobre los datos de un intermediario del sector de retail, por ende, el modelo final elegido será solo de utilidad para los datos de este intermediario.
- Dependiendo del producto e intermediario, los comportamientos de cartera son diferentes, por ejemplo, los deudores cuyo producto es un crédito educativo tiene una menor tasa de siniestralidad que los deudores cuyo producto es un crédito de retail. Por ende, se debe pensar en entrenar modelos por línea de crédito, homologando los campos demográficos.

VII. CONCLUSIONES

Para FGA Fondo de Garantías, los datos de comportamiento de cartera se convierten en un activo importante para la compañía, en ellos se dispone información de diversos sectores y líneas de crédito del mercado colombiano. Con el desarrollo de este proyecto, se genera un hito en la compañía, pues es la primera vez que se utilizan los datos asociados a comportamiento de cartera para la generación de modelos predictivos. Este servirá como inicio para futuros desarrollos para otros intermediarios y sectores de mercado, dando así, el uso esperado del activo previamente mencionado. Del mismo modo, apalancará el crecimiento de la infraestructura tecnológica pues las demandas de cómputo para el preprocesamiento de los datos y el entrenamiento de los modelos así lo requieren. Por otra parte, demandará la implementación de mejores prácticas como Mlops, para un adecuado ciclo de vida de los proyectos de analítica.

De cara al negocio, se contará con modelos que optimizarán su operación, al provisionar los recursos oportunamente, disminuir el costo de capital e incrementar la probabilidad de recuperación de las operaciones pagadas.

En el entrenamiento de los modelos, se evidenció que, al diversificar las variables empleadas, en las diferentes iteraciones se conserva la proporción de desempeño, pero se evidencia una optimización en los tiempos de ejecución. En una de las iteraciones, el descartar almacén y negocio como variables categóricas, se obtiene un 50% de mejora en los tiempos de entrenamiento del modelo.

En las métricas de desempeño, para todos los modelos, se evidencia que en la exactitud se obtienen medidas de alrededor de 0.9. Teniendo en cuenta que la exactitud es el total de elementos clasificados correctamente sobre el total de elementos, la obtención de estos puntajes se explica en la desproporción que existe en los datos, y en buena capacidad de predicción de los modelos para los no siniestrados, los cuales son los más numerosos, como se evidencia en las matrices de confusión. Esta capacidad de predicción, se estima, mejorará la exactitud en el aprovisionamiento en un 50%.

Finalmente, se evidencia la importancia del proceso iterativo en el desarrollo de proyectos analíticos, dado que con los análisis de correlación y de importancia de las variables de los modelos, se llegaba a pensar en el descarte de algunas variables categóricas, pero en los experimentos se

notaba que la presencia o no de las mismas, no afectaba sustancialmente las métricas de desempeño, aunque si los tiempos de ejecución.

VIII. REFERENCIAS

[1] "Forests of randomized trees," Nov. 19, 2021. https://scikit-learn.org/stable/modules/ensemble.html#forest (accessed Nov. 18, 2021).

[2] "Metrics and scoring: quantifying the quality of predictions," Nov. 19, 2021. https://scikit-learn.org/stable/modules/model_evaluation.html (accessed Nov. 18, 2021).