ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE CIENCIAS

APLICACIÓN DE ALGORITMOS GENÉTICOS EN LA SELECCIÓN DE VARIABLES PARA LA CONSTRUCCIÓN DE MODELOS DE CREDIT SCORING

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO MATEMÁTICO

PROYECTO DE INVESTIGACIÓN

ELVIS DAVID MANTILLA BOLAÑOS
mantilla668@gmail.com

Director: M.SC. DIEGO PAÚL HUARACA SHAGÑAY
diego.huaracas@epn.edu.ec

Codirector: M.SC. MÉNTHOR OSWALDO URVINA MAYORGA
menthor.urvina@epn.edu.ec

QUITO, JUNIO, 2023

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por ELVIS DAVID MANTILLA BOLAÑOS, bajo nuestra supervisión.

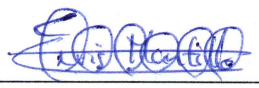
M.Sc. Diego Paúl Huaraca Shagñay
Director del Proyecto

M.Sc. Ménthor Oswaldo Urvina Mayorga
Codirector del Proyecto

DECLARACIÓN

Yo, ELVIS DAVID MANTILLA BOLAÑOS, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.



ELVIS DAVID MANTILLA BOLAÑOS
C.I. 1723818504

AGRADECIMIENTOS

Quiero agradecer en primer lugar a mi madre Silvana Bolaños que sin ella no podría lograr con esta meta de mi vida, ella fue la que me inspiró, la que me inculcó con sus enseñanzas y principios para que hoy en día sea la persona que soy, quiero agradecer por su esfuerzo y dedicación en cada etapa de mi vida.

Quiero agradecer a toda mi familia, a mis tíos: Giovanny y Raúl Bolaños que han sabido mantener a la familia unida, pese a cualquier dificultad familiar que se presente ellos han sabido encontrar solución siempre manteniendo la solidaridad y amor que los caracteriza. Quiero agradecer a mis primos mayores que me han inspirado a cumplir mis metas, ellos han contribuido en mi formación como persona consciente.

Finalmente agradecer a mi director de tesis Diego Huaraca, a mi codirector Menthor Urvina que me han sabido guiar para realizar el presente trabajo, a mis profesores de universidad que han contribuido a mi formación como profesional, a mis compañeros, amigos de la carrera.

DEDICATORIA

A mi madre principalmente que con su dedicación y esfuerzo ha logrado sacarme adelante, ha sabido estar en todo momento de mi vida formándome con valores y principios que en la vida me han servido de mucho, le dedico este trabajo como muestra de agradecimiento.

A mi hermana que ha estado acompañándome como mi mejor amiga, una persona en la que puedo confiar y siempre estará apoyándome en todo lo que me proponga.

Elvis

Índice general

Capítulo 1

Introducción

Las entidades bancarias y financieras que emiten créditos a sus clientes, están en la búsqueda permanente de minimizar los riesgos al emitir dicho crédito, un indicador de este riesgo se refleja en la tasa de morosidad para el sistema bancario que en Ecuador aumentó de 3,00 % en el mes de septiembre de 2019 a 4,07 % en septiembre de 2020[7] de esta manera surgen los modelos Credit Scoring como respuesta a esa búsqueda, generalmente son modelos que clasifican a un individuo en buen pagador o mal pagador además de muchas otras variantes como para saber si un individuo activa o no una tarjeta de crédito, pero el presente trabajo se centrará en el primer caso en donde se predice si un cliente va a ser buen pagador o no.

Para generar un modelo Credit Scoring es necesario tener información sobre los clientes, información antigua que describa el comportamiento del mismo y con base en esa información poder predecir el comportamiento a futuro, por esto es importante entender que la asignación final que se haga a un cliente depende de las diferentes variables que se seleccionan para el modelamiento así como los valores que tenga el individuo en esas variables de la base de datos de la entidad financiera.

En efecto, la selección de variables para elaborar un modelo Credit Scoring es una etapa fundamental y primaria , ya que la continuación o cuerpo del modelo para ser aceptable dependerá de esta selección, es evidente que si se seleccionan variables sin ningún criterio para el modelo de asignación de crédito es muy probable que el modelo sea ineficaz e ineficiente, por esto tener una buena técnica de selección de variables es tan importante, que en la actualidad existen infinidad de métodos: entre ellos métodos tradicionales como forward y backward y en en este trabajo se pondrán a prueba los algoritmos genéticos como métodos para esta etapa.

El nombre hace referencia a la teoría de la evolución de Darwin y como las especies se adaptan a los cambios en la naturaleza y de esta manera siempre sobrevivan las especies más fuertes.

Implementar algoritmos genéticos como una nueva técnica de selección de variables, crea interés por su metodología e inspiración en la Teoría de la Evolución de Darwin(1859). La esencia de esta teoría se centra en los mecanismos que tiene la naturaleza para seleccionar los individuos más aptos de una población, es decir, los que más probabilidad tienen de sobrevivir. Se sabe hoy en día que, para que un individuo pueda sobrevivir debe adaptarse a los cambios que suceden en su entorno, cambios físicos, cambios climáticos, etc., esta adaptación del individuo se ve reflejada en sus genes, para luego estas características (cromosomas) de adaptación deseadas sean transmitidas a sus descendientes cuando este se reproduce[1].

Lo que se buscará es realizar una comparación de resultados entre las técnicas clásicas de selección de variables versus los algoritmos genéticos, en la generación de modelos de Credit Scoring, para de esta manera medir que tan buenos son los algoritmos genéticos en la selección de variables.

1.1. Planteamiento del problema

Con el fin de controlar la tasa de morosidad en la colocación de créditos, las instituciones financieras emplean modelos estadísticos que identifican y clasifican a los clientes en buenos o malos pagadores[13], modelos que son constantemente mejorados por las instituciones bancarias mediante la inclusión de nuevas tecnologías, innovadoras metodologías, mayor cantidad de datos, entre otros factores que aportan significativamente a su mejora[14].

1.1.1. Formulación del problema

Al momento de entrenar un modelo de clasificación (Credit Scoring), la selección de variables explicativas es un paso primordial para que el modelo presente una buena predicción al evaluarlo[5] y por esta misma razón es que la selección de variables se convierte en una etapa clave y sensible, ya que los modelos dependen de una selec-

ción adecuada de variables para varias instituciones financieras que buscan otorgar un crédito, por esto el presente trabajo busca aplicar una nueva metodología para este problema y así mostrar una solución eficiente y basada en métricas.

1.1.2. Descripción del problema

Las entidades bancarias en la actualidad almacenan gran cantidad de datos ya que comprenden los beneficios y utilidades que proporciona el tratamiento de la información, por ejemplo: la identificación de perfiles de crédito, detección de fraudes, alerta de fuga de clientes, crecimiento o no del desembolso en créditos, etc. La utilización de esta gran cantidad de datos surge con el boom del Big Data, la Ciencia de Datos, Machine Learning, entre otros campos que en la actualidad revolucionan la forma de ver, entender y predecir estos datos[9]. Es debido a esto que el número de combinaciones de variables que pueden considerarse al momento de ajustar un modelo es demasiado grande y por tanto encontrar la mejor combinación de variables para la construcción de un modelo Credit Scoring exige de mucho tiempo, altos costos computacionales y operativos, además de que sería ineficiente probar cada combinación posible de variables.

En la actualidad existen varios métodos de selección de variables para la creación de modelos Credit Scoring[5], siendo los más usados por paquetes estadísticos los denominados: selección hacia adelante (forward stepwise), selección hacia atrás (backward stepwise), y selección stepwise que es una combinación de los dos anteriores, también es importante mencionar los métodos de mínimos cuadrados penalizados dentro de estos los métodos "Lasso"[5], la idea clave para entender estos últimos es la penalización ya que evita el sobreajuste del modelo debido al gran número de variables y como es un método de mínimos cuadrados no sólo selecciona variables sino que también estima parámetros del modelo. Además de la existencia de muchos otros métodos derivados que se clasifican según el criterio de introducción o eliminación de la variable en el modelo [3], adicionalmente para generar un modelo de clasificación se realiza un análisis exploratorio donde usualmente se seleccionan las variables con mayor poder predictivo con relación a la variable dependiente.

Haciendo analogía a la Teoría de Darwin, una combinación de variables candidata al modelo representaría un individuo el cual está conformado por cromosomas, que en el caso de la combinación de variables sería una configuración binaria que indica si una

variable va o no en el modelo, se seleccionan los individuos más fuertes (los individuos que generan un mejor modelo según el criterio de fortaleza), luego se llevan a cabo procesos de reproducción (cruce de individuos) y mutación (alteración de uno o más cromosomas con una probabilidad, por lo general baja), para de esta manera obtener nuevos individuos que en su mayoría son más fuertes (generan mejores modelos) que la generación anterior.

1.2. Objetivos

1.2.1. Objetivo general

Comparar la implementación de algoritmos genéticos en la selección de variables explicativas de una base de datos financiera, para la construcción de modelos de colocación de crédito (Credit Scoring), frente a métodos tradicionales de selección.

1.2.2. Objetivos Específicos

- Construir dos modelos Credit Scoring (Regresión Logística y Random Forest) mediante las metodologías tradicionales de selección de variables.
- Elaborar tablas performance que permitan cuantificar diferencias de los modelos obtenidos con algoritmos genéticos versus los modelos obtenidos con metodologías tradicionales, para de esta manera medir su mejoría y la calidad de los modelos resultantes.
- Comprobar si se presenta mejoría o no, tanto en eficiencia como en eficacia de la aplicación de algoritmos genéticos en modelos de colocación de crédito y de esta manera dar a conocer esta metodología de selección de variables.

Capítulo 2

Justificación

2.1. Justificación teórica

Los algoritmos genéticos son métodos robustos de búsqueda, que permiten tratar problemas de optimización donde el objetivo es encontrar un conjunto de parámetros que minimizan o maximizan una función de adaptación [15].

Estos algoritmos operan sobre una población inicial de individuos: $P(t) = \{x_1, x_2, ..x_n\}$ donde t representa la iteración o generación de la población y cada individuo x_i representa una solución a la búsqueda de la mejor combinación de variables para un modelo Credit Scoring[12], el desempeño de cada individuo se evalúa en la función de adaptación (fitness) $f(x_i)$, lo que permite ordenar de mejor a peor los individuos de la población[11].

La población inicial evoluciona sucesivamente hacia mejores poblaciones del espacio de búsqueda mediante procesos probabilísticos [1]: selección de los individuos más adaptados, cruce entre individuos y mutación; hasta alcanzar una solución óptima (mejor modelo) o un criterio de parada[11].

2.2. Justificación práctica

Cuando las entidades bancarias generan un modelo de clasificación para otorgar créditos a sus clientes, buscan reducir al máximo el tiempo de búsqueda de la mejor combinación de variables para el modelo, en este aspecto resultaría totalmente absur-

do tomar decisiones indebidas, como contratar un analista y probar una a una todas las combinaciones, haciendo un desperdicio de esfuerzo, tiempo y dinero.

Por otro lado, uno de los pilares importantes de los cuales parte un algoritmo de un modelo Credit Scoring es el aprovechar la gran variedad de datos que hoy en día almacenan las entidades bancarias y que no han sido explotadas en su totalidad.

Teniendo en cuenta lo mencionado anteriormente, las instituciones financieras al manejar grandes volúmenes de datos, buscarán seleccionar variables que no solo sean significativas para el modelo sino que también sean variables que tengan relación y relevancia con el accionar del negocio, así como variables que sirvan para evaluar a próximos clientes y se logre hacer una predicción que se acerque a la realidad en el otorgamiento o no, de un crédito para al final del día minimizar el riesgo de pérdida al otorgar dicho crédito.

2.3. Justificación metodológica

Existe la necesidad de acortar tiempos en la generación, calibración y aplicación de modelos de Credit Scoring, por esta razón es necesaria la introducción y aplicación de métodos heurísticos como son los algoritmos genéticos [1], métodos que generalmente buscan reducir el tiempo de búsqueda de la solución óptima o en el peor de los casos encontrar una solución tan buena como la óptima y con menores tiempos que métodos deterministas, generalmente se ha visto que los algoritmos genéticos son muy buenos y de bajos costos computacionales cuando de grandes bases de datos se trata[1].

Capítulo 3

Marco teórico

Este capítulo menciona todos los conceptos y definiciones de técnicas y medidas que fueron utilizadas en la metodología del proyecto, así como para la selección, construcción, calibración, y evaluación de los modelos presentados, para esto se menciona primero las medidas de separación y de asociación de las variables tanto cuantitativas como cualitativas, para luego conceptualizar las técnicas estadísticas más usadas en los modelos de Credit Scoring como son: la Regresión Logística y Random Forest, luego se pasará a mencionar las técnicas clásicas de selección de variables, para finalizar con el tema de motivación de este proyecto que son los algoritmos genéticos en la selección de variables.

3.1. Medidas de separación

3.1.1. Prueba de Kolmogórov-Smirnov (KS)

La prueba de Kolmogórov-Smirnov es una prueba no paramétrica, por lo que no es necesario realizar suposiciones a priori sobre la distribución de los datos, la cual mide la bondad de ajuste de dos distribuciones de probabilidad entre sí. Para realizar esta prueba de bondad de ajuste es necesario calcular el estadístico KS correspondiente, para esto primeramente se debe calcular la función de distribución acumulada empírica (FDAE).

Definición 3.1. ***Función de distribución acumulada empírica (FDAE).***
Sea $x_1, x_2, x_3, ..., x_n$ una muestra de tamaño n de una variable aleatoria X, se define la función de distribución acumulada empírica de x como:

$$FDAE_x(x) = \frac{1}{n}\sum_{i=1}^{n}\begin{cases}1 & si\ x_i \leq x\\ 0 & caso\ contrario\end{cases} \tag{3.1.1}$$

A continuación, se va a enunciar la hipótesis que contrasta la prueba KS, para ello se debe considerar dos muestras del mismo tamaño: x_1, x_2, x_3, ..., x_n, y z_1, z_2, z_3, ..., z_n, cada una perteneciente a una variable aleatoria X y Z con funciones de distribución Fx y Fz respectivamente, entonces la prueba KS contrasta:

$$\begin{cases}H_0: & F_X(x) = F_Z(x),\ \ \forall x\\ H_1: & F_X(x) \neq F_Z(x).\end{cases} \tag{3.1.2}$$

Finalmente, el estadístico para contrastar la hipótesis nula H_0 se define a continuación:

$$KS = \max_x |FDAE_X(x) - FDAE_Z(x)| \tag{3.1.3}$$

Es preciso notar que el estadístico KS toma valores entre 0 y 1 incluidos, ya que las funciones de distribuciones acumuladas empíricas de X y Z igualmente toman valores entre 0 y 1 incluidos, además si el estadístico KS toma valores cercanos a 1 indicaría que las distribuciones de X y Z son diferentes y por el contrario si el estadístico KS toma valores cercanos a 0 indicaría que las distribuciones de X y Z son iguales. Mediante el estadístico KS se rechazará la hipótesis nula a un nivel de confianza $(1-\alpha)*100\,\%$ si el valor KS calculado es mayor que el valor crítico KS_α, este valor KS_α se encuentra tabulado para los diferentes valores de α y tamaño de muestra n. La prueba KS generalmente es muy usada cuando se requiere comprobar o contrastar si una muestra observada sigue una distribución determinada o teórica.

Para el caso de los modelos Credit Scoring se requiere que, al escoger las variables para el modelo, el estadístico KS sea el mayor posible es decir que se presente una mayor divergencia entre los casos de pago del crédito y no pago, para esto lo que se hace es dividir la variable a estudiar en X_p, y X_{np}, donde X_p son los valores de la variable X correspondientes a pago del crédito y X_{np} los valores de la variable X correspondientes a no pago del crédito, y el objetivo es comparar las distribuciones de X_p, y X_{np} generando un valor de KS asociado a la variable X, esto se aplica para todas

las variables que se tenga para el modelo, de esta manera se selecciona las variables con el mayor poder predictivo que corresponden a las variables con el KS más alto.

3.2. Medidas de asociación

Las medidas de asociación son utilizadas para cuantificar el poder predictivo de las variables categóricas y su cálculo generalmente se basa en la diferencia de porcentajes entre pago e impago dentro de las categorías de la variable y comparadas con los porcentajes entre pago e impago de la población.

3.2.1. Valor de información (VI)

El valor de información es considerado como una medida del valor predictivo de una variable categórica, sobretodo es muy útil cuando se trata de una clasificación binaria (pago, no pago).

Conceptualmente el valor de información (VI) mide el nivel de asociación entre la probabilidad de que la variable entre en la categoría i, condicionado a que el individuo es buen pagador (pago) y la probabilidad que la variable tome la categoría i, condicionado a que el individuo es mal pagador (impago, no pago).

De esta manera se define el valor de información como sigue.

Definición 3.2.1. ***Valor de información (VI).***
Sea X una variable categórica con m categorías y Z una variable binaria: pago/impago, se define:

$$VI = \sum_{i=1}^{m} \left(\frac{np_i}{NP} - \frac{p_i}{P} \right) ln \left(\frac{\frac{np_i}{NP}}{\frac{p_i}{P}} \right) \tag{3.2.4}$$

Donde:

- p_i, np_i: representan las frecuencias observadas en la categoría i de individuos pagos e impagos respectivamente.
- P,NP: representan el número total de individuos pago y no pago respectivamente.

Es importante destacar que cuanto mayor sea el valor VI, también será mayor la diferencia entre las probabilidades condicionales y por tanto será mayor el poder predictivo de la variable analizada, es por esto que para un modelo de Credit Scoring se usan las variables con mayor valor de información (VI).

3.3. Pruebas de Multicolinealidad

La multicolinealidad describe la dependencia lineal entre las variables explicativas, es un problema que hace difícil cuantificar con precisión el efecto que cada variable explicativa ejerce sobre la variable dependiente.

En la regresión logística la multicolinealidad de las variables explicativas conlleva a tener consecuencias graves en las estimaciones de nuestros parámetros de regresión y por tanto en predicciones que se quiera realizar. La presencia de multicolinealidad se la puede medir, cuando existe una multicolinealidad perfecta significa que una de las variables explicativas se la puede escribir perfectamente como la combinación lineal de otras variables del modelo e imperfecta cuando es una aproximación, lo que buscamos es reducir al mínimo posible la multicolinealidad de las variables explicativas. Si existe multicolinealidad perfecta muy difícil en la realidad simplemente el problema de regresión está mal condicionado ya que la matriz (X'X) sería singular, no tendría inversa y por tanto no se podrían estimar los coeficientes de la regresión. Luego una alta multicolinealidad provoca alta varianza en los coeficientes de regresión esto implica que si tenemos pequeñas variaciones en los datos(variables explicativas) provocarían grandes variaciones en los resultados. [18]

En cuanto a la técnica Random Forest por ser un método no paramétrico y también de clasificación, la multicolinealidad no afecta en gran medida al resultado, principalmente lo que puede provocar son árboles correlacionados pero esto se reduce con la aleatoriedad de elección de las variables para generar los árboles de decisión.[19]

Correlaciones

Para evitar una multicolinealidad alta la correlación entre las variables explicativas de los modelos es menor a $0,65$ en valor absoluto, como se podrá ver en secciones posteriores.

Factor de Inflación de Varianza VIF

El VIF es un indicador de multicolinealidad específica de cada variable predictora del modelo $VIF_j = \frac{1}{1-R_j^2}$ para $j = 1, 2, 3, ..., p$; donde R_j^2 es el coeficiente de determinación de la regresión de X_j respecto a las demás variables predictoras. Como regla general si $VIF_j >= 10$ existe multicolinealidad alta.[18]. El VIF calculado para cada modelo de regresión no supera el criterio establecido y se lo puede revisar en anexo 3.

3.4. Técnicas estadísticas en Credit Scoring

Para la construcción de un modelo Credit Scoring existen muchas técnicas ya sean modelos sencillos como modelos de regresión lineal o modelos de regresión múltiple o ya sea modelos más complejos en los que se necesita de más procesamiento de datos y por tanto de tecnología mucho más avanzada para este proceso, como por ejemplo redes neuronales, máquinas de vectores de soporte (SVM), y en algunos casos modelos de Random Forest, en esta sección se estudiarán dos técnicas en particular las cuales son: Regresión Logística y Random Forest.

3.4.1. Regresión Logística

La técnica de Regresión Logística fue empleada por Wiginton (1.980), en la cual se busca una relación con respecto a las variables independientes, sean numéricas o categóricas, con la que se pueda clasificar a los individuos en pago o impago, esta clasificación se puede hacer con el valor resultante que arroja la relación antes mencionada, este valor es una probabilidad relacionada a que el sujeto pertenezca a una de las categorías pago o impago.

Sea Y una variable dicotómica definida de la siguiente manera:

$$Y = \begin{cases} 1 & impago \\ 0 & pago \end{cases} \tag{3.4.5}$$

Esta variable depende funcionalmente de ciertas variables X_2, X_3, ..., X_k y se tiene una muestra de tamaño n de estas variables.

Notando como $p_i = Pr(y_i = 1)$ la probabilidad de que el individuo i sea impago (mal pagador) y $1 - p_i = Pr(y_i = 0)$ la probabilidad de que el individuo i sea pago (buen pagador), se requiere encontrar una función que arroje p_i con ayuda de las variables independientes X_2, X_3, ..., X_k, para ello se emplea la función logística:

$$p_i = \frac{1}{1 + e^{-(\beta_1 + \beta_2 x_{i2} + \beta_2 x_{i3} + ... + \beta_k x_{ik})}},\ i = 1, 2, 3, ..., n \tag{3.4.6}$$

Si se define:

$$t_i = \beta_1 + \beta_2 x_{i2} + \beta_2 x_{i3} + ... + \beta_k x_{ik} \tag{3.4.7}$$

Se puede ver que t_i es el valor de una regresión lineal múltiple.

La función p_i tiene la siguiente gráfica respecto de t_i:

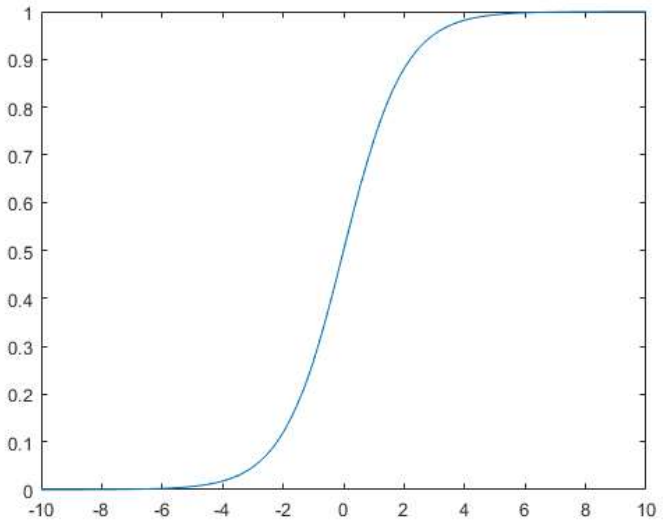


Figura 3.1: Gráfica de la función logit
Elaboración propia

Se puede notar en la gráfica que $0 < p_i < 1$, en efecto ya que p_i es una probabilidad, además no toma los valores extremos es decir 0 o 1, esto quiere decir que no van a existir individuos cuya probabilidad de ser buen pagador sea 1 o 0 exactamente.

Por otro lado si se despeja t_i de la ecuación 3.4.6, definido en la ecuación 3.4.7 se tiene:

$$t_i = ln\ \frac{p_i}{1 - p_i} \tag{3.4.8}$$

A la razón $\frac{p_i}{1-p_i}$ se la denomina razón de probabilidades u odds.

3.4.2. Random Forest

Modelos Bagging

Para hablar de los modelos Random Forest es importante describir a los modelos "Bagging" que como objetivo tienen la reducción de la varianza sobre diferentes muestras aleatorias del conjunto de entrenamiento se obtienen modelos predictivos simples diferentes. Por último, se ensamblan todos ellos en un único modelo, que es el promedio de los anteriores (en caso de regresión) o el más votado (en caso de clasificación)[16].

Los modelos Bagging son recomendables especialmente en modelos con alta varianza (en caso contrario estos no mejoran significativamente). Además, reduce el sobreaprendizaje ya que cada modelo simple es independiente del anterior, con muestras aleatorias diferentes[22].

La técnica de Bagging sigue estos pasos[22]:

1. Divide el conjunto de entrenamiento en distintos subconjunto de datos, obteniendo como resultado diferentes muestras aleatorias con las siguientes características:
    - Muestra uniforme (misma cantidad de individuos en cada conjunto)
    - Muestras con reemplazo (los individuos pueden repetirse en el mismo conjunto de datos).
    - El tamaño de la muestra es igual al tamaño del conjunto de entrenamiento, pero no contiene a todos los individuos ya que algunos se repiten.
    - Si se usan muestras sin reemplazo, suele elegirse el 50 % de los datos como tamaño de muestra.
2. Luego se crea un modelo predictivo con cada conjunto, obteniendo modelos diferentes.
3. Luego se construye o ensambla un único modelo predictivo, que es el promedio de todos los modelos.

Random Forest

La técnica de Random Forest fue propuesta por Leo Breiman[25] a finales del siglo 20 y es una técnica "de ensamble", lo que hace referencia a la utilización de un grupo de

modelos predictivos, para realizar una combinación entre estos y de esta manera alcanzar una mejor precisión y estabilidad del modelo, específicamente los modelos Random Forest utilizan árboles de decisión como el grupo de modelos predictivos y así estos proveen una mejora significativa a los modelos de árboles de decisión.

Un árbol de decisión como todos los modelos sufre de problemas de sesgo y varianza, el sesgo es la diferencia que hay entre los valores predecidos y los valores reales mientras que la varianza es cuan diferente son las predicciones partiendo de un mismo punto o variables de entrada.

Los modelos Random Forest surgen como una solución a estos dos problemas (sesgo, varianza), generalmente un árbol de decisión presenta alto sesgo y baja varianza, es así como la complejidad de los modelos Random Forest generan una reducción en el error de predicción debido a un sesgo más bajo en el modelo.

Por otro lado, el abuso de la complejidad del modelo puede generar un sobre ajuste de este lo que significa que el modelo predice muy bien la base de entrenamiento, pero cuando se busca predecir la base de prueba o nuevos individuos el modelo no predice de buena manera, esto se puede observar en la figura 3.2.

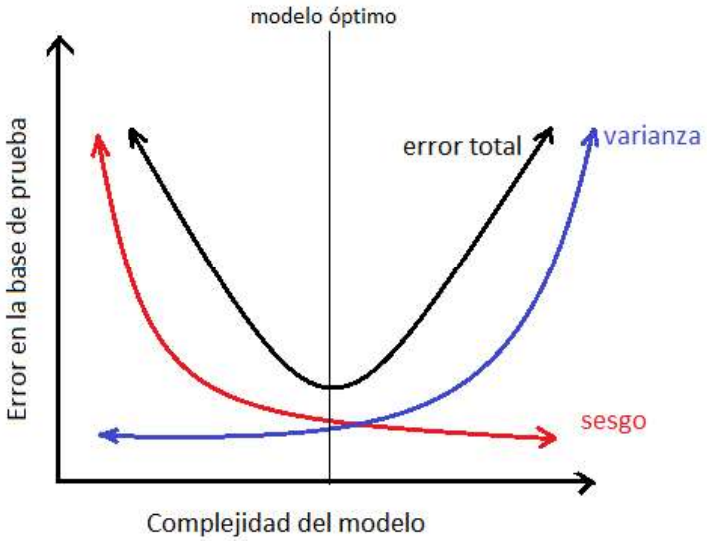


Figura 3.2: Gráfica del modelo óptimo[16]

Los modelos Random Forest como modelos "Bagging", usan modelos predictivos en paralelo, y el principal objetivo de hacer esto es aprovechar la independencia que hay entre estos modelos simples, en este caso aprovechar la independencia de los árboles de decisión, de esta manera reducir el error o sesgo resulta fácil ya que cuando se tiene árboles de regresión se promedia las salidas de los árboles realizados en paralelo y para árboles de clasificación se usa la votación, es como si se diera como respuesta lo que eligiese la mayoría de árboles.

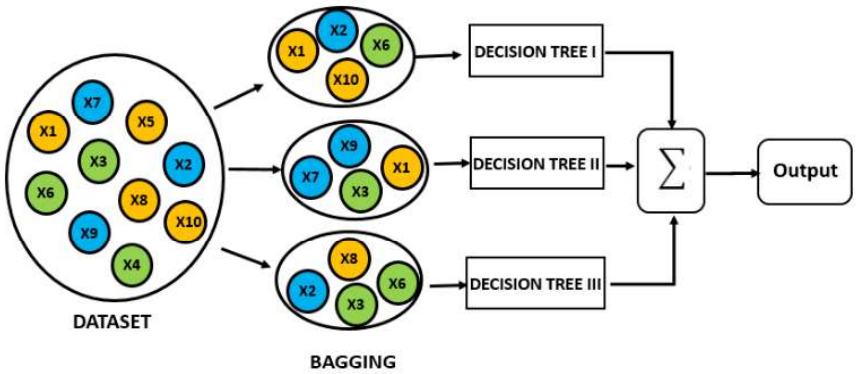


Figura 3.3: Random forest como modelo Bagging[16]

3.4.2.1. Hiperparámetros

A continuación los valores de los hiperparámetros que se exploran en los modelos de Random Forest, estos valores potenciales se consideran en el algoritmo de aprendizaje automático AutoML[26], con estos valores se busca reducir al máximo la varianza del modelo para tener un mejor poder de predicción.

HIPERPARÁMETRO	VALORES
ntrees	$\#predictores$ x $\{8, 9, 10\}$
mtry	$\#predictores$ x $\{0,2; 0,4; 0,6; 0,8\}$

Tabla 3.1: Definición de parámetros para el Random Forest.
Elaboración propia

Donde:

- ntrees: Número de árboles establecidos para el modelo.
- mtry: Número de predictores seleccionados para cada nivel.

3.5. Técnicas clásicas de selección de variables

3.5.1. Métodos Forward

O "Métodos de selección hacia adelante" en estos métodos se parte de un modelo sin ninguna variable explicativa y mediante un criterio de comparación, en pasos sucesivos se van ingresando una a una las variables hasta satisfacer una cierta regla de parada, donde se puede decir que el modelo es el óptimo y no necesita de más variables.

Generalmente las variables a introducir en el modelo son variables altamente correlacionadas con la variable dependiente.

3.5.2. Métodos Backward

O "Métodos de selección hacia atrás" por el contrario con los métodos forward, en estos métodos se parte con un modelo donde todas las variables explicativas están en el mismo, y en cada paso se va eliminando una a una las variables menos significativas o que cumplan una condición de eliminación, hasta que no se pueda eliminar más variables del modelo, es decir hasta satisfacer la regla de parada.

3.5.3. Métodos Stepwise

Este es un método que combina las dos técnicas anteriores lo que hace es que parte inicialmente como un método forward es decir un modelo vacío sin variables explicativas y sigue comportándose como un forward, la diferencia está en que, en cada paso no solo agrega variables al modelo, sino que también puede eliminar una variable según el criterio que se esté usando.

3.5.4. Criterios para los métodos forward, backward y stepwise

Para la introducción o eliminación de variables de un modelo haciendo uso de las técnicas mencionadas antes generalmente se usan dos conjuntos de criterios:

3.5.4.1. Criterios individuales

Los criterios individuales tienen que ver con valores únicos para cada variable, aquí se puede hacer uso de las correlaciones, KS, VI, nivel de significancia (generalmente

usado para modelos de regresión lineal), etc.

Por ejemplo, si se toma como criterio el nivel de significancia para el método forward en cada paso se añadirá la variable más significativa para el modelo, por el contrario, para el método Backward se eliminará la variable menos significativa para el modelo, de la misma manera para los demás criterios.

3.5.4.2. Criterios de ajuste global

En vez de concentrarse en un valor específico de la variable se ve una medida de ajuste del modelo con esa variable, medidas como: GINI, ROC, R^2, Criterio de Información Akaike (AIC), Criterio de Información de Bayes (BIC), etc.

En este caso si se escoge como ejemplo el criterio AIC, para el método forward antes de ingresar la variable se compara el AIC sin la variable y el AIC con la variable y lo que se busca es el menor AIC de esta forma se procederá a introducir o no la variable en el modelo.

Estadístico ROC:

O también conocido como AUROC (Area Under Receiver Operating Characteristic) numéricamente es el área bajo la curva ROC, esta curva se obtiene al graficar la sensibilidad vs (1- especificidad), donde:

- **Sensibilidad**: Es la probabilidad de que a un individuo mal pagador el modelo lo clasifique como mal pagador.
- **Especificidad**: Es la probabilidad de que a un individuo buen pagador el modelo lo clasifique como buen pagador.

Este estadístico mide la separación a lo largo de toda la distribución, si el valor es de 0,5 el modelo no segmenta, si el valor es 1 el modelo segmenta perfectamente, para modelos de riesgo el valor típico se encuentra entre 0,65 y 0,80.

Estadístico GINI:

El estadístico GINI al igual que el estadístico ROC mide que tan bien el modelo clasifica a los individuos buenos o malos pagadores.

Este coeficiente GINI se define como el doble del área entre la curva ROC y la recta $y = x$, es decir:

$$GINI = 2AUROC - 1 \tag{3.5.9}$$

Oscila entre 0 y 1 y cuanto más se acerque a 1 indica una mejor discriminación del modelo.

La desventaja que se tiene al utilizar los métodos Forward, Backward, Stepwise es que al utilizarlos en bases de datos donde existen muchas variables, se vuelven ineficientes ya que en cada paso se elimina o se introduce una variable, es decir se realiza un paso por cada variable y peor aún en los modelos Stepwise donde en cada paso se puede eliminar y agregar una variable, como subpasos dentro del paso, esto provoca un costo computacional alto.

3.5.5. Métodos Lasso

Es un método particular de los métodos de mínimos cuadrados penalizados, lo que se busca con estos métodos es aplicar una penalización que evita el sobre ajuste causado por el excesivo número de variables en el modelo.

Particularmente los métodos Lasso (least alsolute shrinkage and selection operator) introducidos por Tibshirani(1.996) [5] imponen una penalización sobre los coeficientes de regresión, además como es una técnica de mínimos cuadrados no solo selecciona las variables que van en el modelo sino que también estiman los coeficientes de regresión, entonces el método Lasso se reduce a resolver el siguiente problema de mínimos cuadrados:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - x_i'\beta)^2 \tag{3.5.10}$$

sujeto a:

$$\sum_{j=1}^{p} |\beta_j| \leq t \tag{3.5.11}$$

O de forma equivalente minimizando la expresión:

$$\sum_{i=1}^{n} (y_i - x_i'\beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{3.5.12}$$

Donde t y λ son los parámetros de penalización.

Lasso es consistente para la selección de variables, siempre que p no sea demasiado grande y el parámetro de penalización λ crezca más rápido que $\sqrt{nlog(p)}$, n: tamaño de la población y p: número de variables predictoras.[5]

Notar que para valores grandes de λ y valores pequeños de t, los coeficientes de regresión β_j van hacia 0, lo que quiere decir que no se seleccionaría la variable que acompaña a ese β_j, por eso los Métodos Lasso son una técnica de selección además de estimación. En cuanto a la elección del parámetro λ de penalización existen varios métodos que se dedican a eso.

3.6. Algoritmos genéticos (AG) en la selección de variables

Los algoritmos genéticos son técnicas de búsqueda de optimización, en los cuales se simula procesos biológicos de evolución natural, la esencia de estos algoritmos está en que recibe como entrada una población o primera generación de posibles candidatos a soluciones del problema a optimizar, realiza con esta población operaciones de "reproducción" y "mutación" para devolver una generación de mejores soluciones que se acercan más a la solución óptima del problema.

En esta sección se detallará cómo funciona un algoritmo genético, su estructura, su procedimiento metodológico ya que para encontrar la solución al problema de selección de variables se siguen un conjunto de pasos sistematizados, primero se hará referencia al paso tal cual, para un algoritmo genético simple, después se mencionará como este paso se lleva a cabo para la selección de variables.

3.6.1. Codificación de la población inicial

La codificación de la población inicial es muy importante para aplicar un algoritmo genético ya que de esta manera se pueden realizar las fases de reproducción y mutación, el resultado de la codificación se conoce como "cromosoma", existen muchas formas de codificar un individuo, generalmente variando el número de caracteres con los que se quiere representar a los individuos. En este aspecto es muy usada y conocida la codificación binaria en la cual se hace uso de dos caracteres: 0 y 1.

Para hacer uso de la codificación binaria en este caso de estudio referente a la selección

de variables para un modelo óptimo de Credit Scoring, lo que se hace es asignar 1 a la variable que entra al modelo y 0 a la variable que no, además un individuo equivale a una combinación de variables que entran o no al modelo, análogamente las poblaciones o generaciones vienen a ser un conjunto de combinaciones de variables que entran o no al modelo, se hace uso de un ejemplo muy descriptivo:

Suponiendo que se tiene 10 variables independientes: $V_1, V_2, V_3, ..., V_{10}$ y la población inicial de 6 individuos:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
1	0	1	1	0	1	0	0	0	1	1
2	1	0	1	0	1	1	1	0	1	1
3	1	0	1	1	1	1	0	0	0	0
4	0	0	0	0	1	1	1	1	1	1
5	1	0	1	0	1	1	1	0	1	0
6	0	1	0	1	0	0	1	0	1	1

Figura 3.4: Población inicial
Elaboración propia

Esta población conformada por 6 individuos ya está codificada, entonces si se analiza el primer individuo o cromosoma se tiene que la primera variable V_1 tiene la asignación 0 lo que indica que no entra en el modelo, la variable V_2 tiene por codificación 1 lo que indica que, si entra en el modelo, y así para las demás asignaciones del cromosoma, y para los demás individuos.

3.6.2. Cálculo del Fitnees

Una vez realizado la codificación de la población inicial se procede a calcular el "Fitness" o fortaleza del individuo que no es más que un valor medible y comparable, que cuantifica la calidad del individuo como solución del problema de optimización, análogamente hace referencia a cuan apto está un genotipo a la sobrevivencia en la siguiente generación.

Para problemas de optimización los cuales son minimizar o maximizar una función objetivo, el Fitness viene a ser esta función objetivo y si se evalúa los individuos se

podrá comparar cuál de ellos es el mejor candidato a óptimo para el problema.

Específicamente para este caso de estudio el Fitness viene a ser un valor que mida que tan bueno es el modelo con la combinación de variables tomada, para de esta manera comparar los modelos que se generan con las diferentes combinaciones de variables, entonces para el Fitness se toman valores de ajuste del modelo como: R^2, Accuracy, Criterio de Información de Akaike (AIC), KS, etc.

Suponiendo en el ejemplo anterior el Fitness que se va a tomar es el coeficiente de determinación R^2, entonces simplemente se obtiene el R^2 de cada modelo generado por cada uno de los individuos obteniendo:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	R^2
1	0	1	1	0	1	0	0	0	1	1	0,51
2	1	0	1	0	1	1	1	0	1	1	0,74
3	1	0	1	1	1	1	0	0	0	0	0,87
4	0	0	0	0	1	1	1	1	1	1	0,67
5	1	0	1	0	1	1	1	0	1	0	0,66
6	0	1	0	1	0	0	1	0	1	1	0,94

Figura 3.5: Fitness de la población
Elaboración propia

Entonces mediante el Fitness se puede realizar comparaciones de fortaleza entre individuos, en la figura 3.5 se evidencia que el individuo 6 es el más apto o candidato a óptimo, en otras palabras, se diría que la sexta combinación de variables es la que genera el mejor modelo Credit Scoring, por el contrario la primera combinación genera el peor modelo dentro de la población.

3.6.3. Creación de la nueva población

Los algoritmos genéticos buscan siempre nuevas generaciones que se acerquen más a la solución del problema de optimización, la creación de una nueva población se realiza a partir de la población predecesora y para ello se realizan los siguientes pasos:

3.6.3.1. Selección de los individuos

Una subpoblación es seleccionada a partir de la población inicial o predecesora para reproducir una nueva generación, está subpoblación seleccionada es la base para aumentar la fortaleza en las siguientes generaciones, por lo cual en esta selección se deben priorizar los individuos que tengan un fitness alto, entonces lo que se hace es asignar una probabilidad de selección a cada uno de los individuos. Esta probabilidad debe estar correlacionada obviamente con el fitness del individuo.

Para el ejemplo que se estudia, esta selección es conocida como el método de la ruleta, y consiste en asignar un peso entre 0 y 1, o porcentaje respecto a la suma total del fitness, de esta manera:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	R^2	Prob. en la ruleta
1	0	1	1	0	1	0	0	0	1	1	0,51	0,116
2	1	0	1	0	1	1	1	0	1	1	0,74	0,169
3	1	0	1	1	1	1	0	0	0	0	0,87	0,198
4	0	0	0	0	1	1	1	1	1	1	0,67	0,153
5	1	0	1	0	1	1	1	0	1	0	0,66	0,150
6	0	1	0	1	0	0	1	0	1	1	0,94	0,214
									SUMA		4,39	1

Figura 3.6: Probabilidad de selección para proceso de cruce
Elaboración propia

En la tabla anterior se puede ver como el individuo 6 es el individuo más probable en salir en un giro de la ruleta en contraste el individuo 1 es el menos probable en salir en la ruleta, estos simulan al individuo más y menos fuerte en una población y en efecto los individuos más fuertes tienen más probabilidad de sobrevivir a la selección natural.

Ahora si se realiza una simulación del giro de esta ruleta con las diferentes probabilidades de que salga cada uno de los individuos en 6 giros se obtienen los siguientes resultados:

N.- de giro	resultado(individuo)
1er	5
2do	4
3er	6
4to	6
5to	6
6to	5

Figura 3.7: Simulación de selección de los individuos
Elaboración propia

3.6.3.2. Cruce de individuos "Crossover"

Esta fase también es conocida como recombinación, una vez seleccionados los individuos se realizan operaciones de cruce entre ellos (lo que simularía la reproducción en las especies), las operaciones de cruce buscan aprovechar las características que hacen fuertes a los individuos para mostrarlas en las siguientes generaciones, para realizar estas operaciones es necesario que los individuos se encuentren previamente codificados, es principalmente para esta etapa y la de mutación para la cual se hace la codificación de los individuos.

Existen muchas técnicas de cruce de individuos, la más común es la de generar una partición en los cromosomas de los individuos e intercambiar esa partición, en el ejemplo que se esta llevando a cabo, suponiendo que se quiere cruzar los individuos 5 y 6, entonces se tiene:

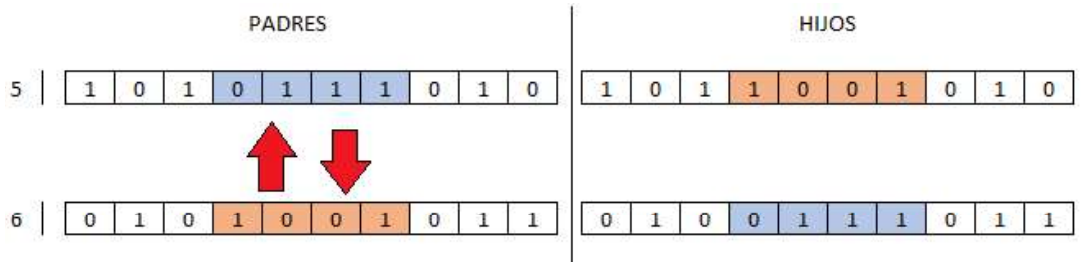


Figura 3.8: Técnica de cruce por partición
Elaboración propia

Para un algoritmo genético simple los puntos donde se realiza la partición para el cruce se toman aleatoriamente, además de que también existe una probabilidad P_c que es la probabilidad de que exista el cruce entre la pareja, en este caso P_c es 1, existen algoritmos genéticos donde los puntos para realizar la partición son fijados, por ejemplo,

puede que se requiera solo recombinar un cromosoma ya que se tiene un buen individuo cercano al óptimo entonces no se necesita cambiarlo tanto.

Es aconsejable por experiencia que la probabilidad de cruce P_c esté entre 0,40 y 0,85 [15].

3.6.3.3. Mutación

El proceso de mutación se realiza en un individuo, no depende de los cromosomas predecesores, para ello se establece una probabilidad de mutación del cromosoma generalmente baja, y consiste en alterar uno o varios caracteres de los genes del hijo, es decir cambiar de 1 a 0 o viceversa los genes del individuo.

Figura 3.9: Mutación de un individuo de la población
Elaboración propia

La probabilidad de realizar el proceso de mutación es recomendable establecerla entre 0,03 y 0,1 [15].

3.6.3.4. Reemplazo de la población antigua

Una vez creados los nuevos individuos hijos provenientes de la técnica de cruce y después de la técnica de mutación, se procede a calcular su fitness o fortaleza para de esta manera observar si se tienen mejores individuos candidatos a óptimo, es importante notar que la nueva generación no necesariamente es mejor que la predecesora, por ello los individuos con mayor fortaleza de la anterior generación no son desechados sino que también pueden formar parte de la próxima generación, la idea aquí es siempre mantener un número constante de individuos en cada generación e ir desechando los individuos con menor fitness.

3.6.3.5. Criterio de parada

Es importante definir un criterio de parada para el algoritmo ya que se puede estar generando individuos infinitamente inclusive individuos repetidos, lo cual solo retrasa y aleja la búsqueda del individuo óptimo, este criterio se encarga de detener el algoritmo y la creación de nuevas generaciones para elegir al individuo más apto como solución del problema, en base a esto algunos de los criterios más comúnmente usados son:

- Establecer un número determinado de generaciones a realizarse.
- Establecer un tiempo de búsqueda determinado del individuo más apto.
- Establecer un límite del número consecutivo de generaciones en donde los individuos no han mejorado los fitness de sus predecesores.
- Se establece una tolerancia que es la diferencia entre el mejor individuo y los demás, si durante n generaciones no se puede acercar o superar la tolerancia, el algoritmo para y deja de crear más generaciones.

Capítulo 4

Marco Metodológico

En este capítulo se realizará una aplicación práctica de los algoritmos genéticos (AG) para la selección de variables, además de su respectiva comparación con las técnicas clásicas de selección de variables para dos modelos de Credit Scoring: un modelo Logit, y un modelo Random Forest, además de que se describirán las técnicas, métodos y pasos llevados a cabo para la generación de estos modelos.

Todo esto se realizará gracias al lenguaje estadístico R y la base de datos utilizada hace referencia a datos reales correspondientes a una institución bancaria ecuatoriana que por privacidad no se mencionarán nombres ni etiquetas, en anexos se puede encontrar un diccionario de las variables utilizadas.

4.1. Credit Scoring para colocación de créditos

Un de modelo Credit Scoring es una metodología estadística y analítica utilizada en el sistema financiero que permite tener una visión y decisión al otorgar un crédito de manera objetiva, minimizando el riesgo asociado con la colocación del mismo.

Un modelo de Credit Scoring se basa en historiales de pago de los clientes, toma comportamientos pasados para pronosticar comportamientos futuros de los créditos.

Como lo explica Schreiner (2.002), un modelo de Credit Scoring utiliza la misma lógica que el analista de crédito, pues se basa en experiencias y seguimientos de créditos otorgados en el pasado y mediante una asociación con características de los nuevos solicitantes puede asignar o no el crédito. La diferencia principal radica en que la

decisión del analista de crédito es subjetiva, dependerá en gran medida de lo que el analista considere "bueno" o "malo", además de muchas otras consideraciones como la capacidad de manejar muchas variables e información que un analista de crédito no puede hacer.

4.2. Generación de la información

La base de datos utilizada representa datos de una institución financiera cuya conformación se extrajo de la siguiente manera.

Se consolidó la información de 5 puntos de observación en el transcurso de un año, desde ago-2017 hasta jul-2018, seleccionando los puntos de observación siguientes:

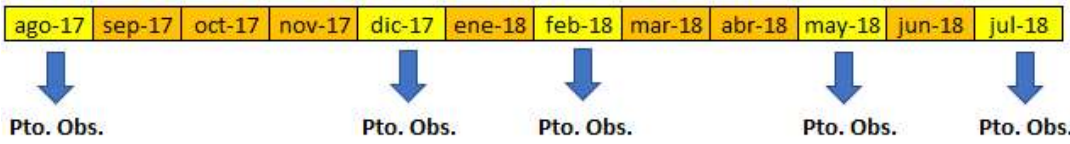


Figura 4.1: Puntos de observación para la conformación de la base de modelamiento. Elaboración propia

Como se puede observar se toma la información de: ago-17, dic-17, feb-18, may-18, jul-18, para la conformación de la base de modelamiento, donde para cada uno de estos puntos de observación se tiene una ventana histórica y una ventana de desempeño:

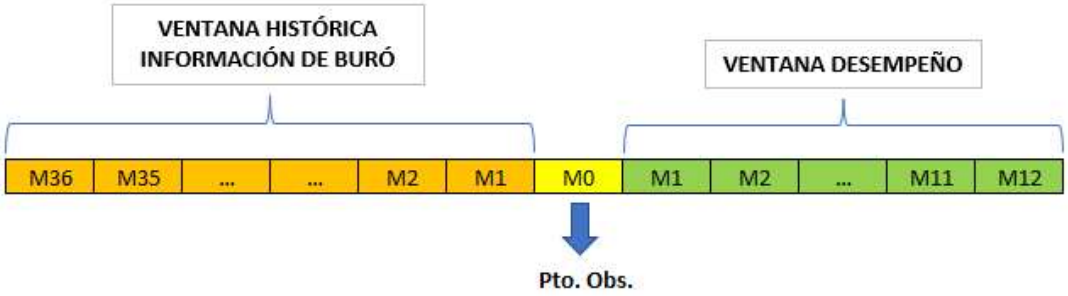


Figura 4.2: Generación de la información.
Elaboración propia

En la figura 4.2 se puede ver que para la conformación de la ventana histórica se toman los meses anteriores al punto de observación y en Ecuador por disposición de la SBS no puede ser superior a los 36 meses (3 años), en esta ventana se generan las variables

asociadas al historial crediticio del individuo entre estas se encuentran: deudas totales, deudas vencidas en tarjetas, número de operaciones de créditos, monto en créditos, etc.

Por otro lado, los meses posteriores al punto de observación constituyen la ventana de desempeño, que generalmente la conforman los 12 meses después del punto de observación y para este caso la conforman los 12 meses. La finalidad de la ventana de desempeño es la de definir a los individuos en buenos o malos pagadores es decir mediante la ventana de desempeño se obtiene la variable dependiente.

La consolidación de los puntos de observación y generación de variables en la ventana histórica y de desempeño establecen una base de datos de exactamente 534.001 registros para este tema de estudio.

4.2.1. Tratamiento de valores atípicos

Uno de los paso más importantes como parte del procesamiento de datos es detectar y tratar valores atípicos, para detectar valores atípicos se ha realizado gráficas de dispersión de las variables explicativas, donde se han identificado principalmente los percentiles.

Y la técnica utilizada ha sido la de acotar a un percentil por encima del percentil 95 donde se notaba un cambio considerable de los valores, con esto se busca tratar de mantener la naturalidad de los datos [23].

4.3. Definición de la variable dependiente

4.3.1. Análisis de Roll-Rate

El análisis de Roll-Rate permite identificar la tasa de deterioro crediticio. Se realiza comparando, de un mes al siguiente si existe un deterioro en los días de vencimiento que tiene un cliente y así para todos los meses que conforman la ventana de desempeño.

Dentro de la ventana de desempeño se obtienen 11 tablas comparando los meses anteriormente mencionados: m1 vs m2, m2 vs m3, m3 vs m4, m4 vs m5, m5 vs m6, m6 vs m7, m7 vs m8, m8 vs m9, m9 vs m10, m10 vs m11, m11 vs m12, que se los puede apreciar a detalle en la sección de anexos.

A continuación se muestra la tabla general que es el promedio de todas las tablas roll

rate, durante la ventana de desempeño que indica la probabilidad de deterioro o no (avanza , no avanza) del estado actual de vencido.

RANGO VENCIDO	VARIABLE	
	NO AVANZA	AVANZA
Sin vencido	93, 20 %	6, 80 %
De 1 a 30 días	87, 28 %	12, 72 %
De 31 a 60 días	60, 91 %	39, 09 %
De 61 a 90 días	46, 04 %	53, 96 %
De 91 a 120 días	32, 73 %	67, 27 %
De 121 a 150 días	26, 71 %	73, 29 %
Más de 150 días	4, 68 %	95, 32 %

Tabla 4.1: Tabla Roll-Rate.
Elaboración propia

Entonces lo que indica la tabla 4.1 es, por ejemplo: si un cliente se encuentra en el rango vencido De 61 a 90 días al punto de observación la probabilidad de que su rango vencido avance o en otras palabras que no pague la deuda en el siguiente mes es 0,5396, así para todas las categorías de rango vencido.

Es interesante notar los extremos de esta tabla y que indican, en el rango: SIN VENCIDO el porcentaje de avanza es del 6,8 % quiere decir que de 100 personas que estén en el rango sin vencido, 6 personas deteriorarán su rango. Por otro lado, en el rango: VENCIDO MÁS DE 150 DÍAS la tasa de avanza es 95,32 % entonces se dice que de 100 personas, 95 van a deteriorar su rango el mes siguiente y es lógico pensar que una persona que lleva más de 150 días de mora va a continuar debiendo y aumentando sus días de mora hasta que se tome acciones como: demanda judicial, expropiación de garantías, etc.

La manera de seccionar cada zona de la tabla 4.1 depende del riesgo que se quiera asumir en la institución financiera:

- zona mal pagador; la zona roja con un avanza mayor o igual al 50 %.
- zona de indeterminados; la zona amarilla con una tasa de avance entre el 10 % y 50 %.
- zona como buen pagador; la zona verde con un avanza menor o igual al 10 %.

Este análisis permite definir la tolerancia del deterioro que se quiere para un sujeto clasificado como buen o mal pagador y de esta manera definir la variable dependiente.

La variable dependiente Y indica si un cliente de la institución bancaria se considera buen pagador o mal pagador (pago o impago).

$$Y = \begin{cases} 1 & mal\ pagador, impago \\ 0 & buen\ pagador,\ pago \end{cases} \tag{4.3.1}$$

4.3.2. Variable Dependiente

La definición de la variable dependiente para este caso práctico se realiza por cada uno de los individuos de la base de datos es por esto por lo que se consolida a nivel cliente la deuda como: saldo total, saldo por vencer, saldo no devenga intereses, saldo vencido, etc.

En función de las tasas obtenidas en la tabla 4.1 se definen las siguientes categorías:

- NO BANCARIZADO: Son los clientes que no tienen historial crediticio de los últimos 12 meses anteriores al punto de observación, es decir si se suma la deuda total en bancos, cooperativas, casas comerciales y otros durante los últimos 12 meses antes del punto de observación y se obtiene una deuda de 0.
- SIN DESEMPEÑO: Son los clientes cuya ventana de desempeño es menor a 6 meses.
- MALO OBSERVADO: Son los clientes que presentan saldo en demanda judicial o cartera castigada al punto de observación.
- MALO: Son los clientes que alcanzaron un rango de vencimiento mayor a 60 días, ver tabla 4.1.
- BUENO: Son los clientes que presentan 0 días de vencido al punto de observación.
- INDETERMINADO: Son todos los clientes que no ingresan en las categorías anteriores.

A continuación la distribución de la variable dependiente:

VarDep	Conteo	%
BUENO	275.402	51,57 %
MALO	65.251	12,22 %
INDETERMINADO	62.994	11,80 %
MALO OBSERVADO	53.210	9,96 %
SIN DESEMPEÑO	69.612	13,04 %
NO BANCARIZADO	7.532	1,41 %
TOTAL	**534.001**	**100,00 %**

Tabla 4.2: Distribución de la VarDep.
Elaboración propia

4.4. Segmentación Clean/Dirty

Esta segmentación Clean/Dirty de la base de datos es muy usada para generar modelos de Credit Scoring se realiza ya que por experiencia se conoce que si una persona tiene retrasos en sus pagos en los últimos meses anteriores al punto de observación va a continuar con ese hábito en el futuro o personas que por motivos de fuerza mayor retrasan sus pagos y lo que se espera es que continúen con esos retrasos al menos por un año.

Esta segmentación surge por la necesidad de buscar dos subgrupos homogéneos y busca diferenciar sujetos que han presentado retrasos en sus pagos (Dirty) en el último año antes del punto de observación de los que no (Clean) y así generar un modelo para cada subgrupo, que en la práctica resulta mucho más eficiente que generar un solo modelo para toda la población.

Para realizar la segmentación se excluyen los registros que se etiquetaron con la categoría "NO BANCARIZADO" ya que como se define esta variable estos registros no tienen un historial crediticio antes del punto de observación y por tanto no se puede realizar la segmentación para estos.

A continuación las definiciones rigurosas para segmentar la población:

Al realizar esta segmentación se espera que los grupos sean representativos ya que suponiendo que uno de los subgrupos representa el 5 % de la población no sería adecuado realizar un modelo para un porcentaje bajo de la población lo ideal sería un 50 %/50 % pero en la realidad es muy difícil obtener este porcentaje, entonces lo que se espera es una partición 80 %/20 % o mejor.

SEGMENTACIÓN	DESCRIPCIÓN
CLEAN	Si el promedio del máximo días de vencido en operaciones vigentes en todo el SCE es menor o igual a 30 días en los últimos 12 meses.
DIRTY	Si el promedio del máximo días de vencido en operaciones vigentes en todo el SCE es mayor a 30 días en los últimos 12 meses.

Tabla 4.3: Definición de la segmentación.
Elaboración propia

SEGMENTACIÓN	CONTEO	%
CLEAN	397.274	75,46 %
DIRTY	129.195	24,54 %
TOTAL	**526.469**	**100,00 %**

Tabla 4.4: Distribución de la segmentación.
Elaboración propia

En este caso práctico se ha obtenido una partición de $75,46\,\%/24,54\,\%$ que está dentro de lo que se busca.

A continuación la distribución de la segmentación en cada una de las categorías de la variable dependiente.

SEGMENTACIÓN	VarDep					TOTAL
	0	1	2	3	4	
CLEAN	263.693	30.474	55.332	5	47.770	397.274
DIRTY	11.709	34.777	7.662	53.205	21.842	129.195
TOTAL	275.402	65.251	62.994	53.210	69.612	526.469

Tabla 4.5: Distribución de la segmentación vs VarDep.
Elaboración propia

En la tabla 4.5 se definen:

0: Bueno

1: Malo

2: Indeterminado

3: Malo observado

4: Sin desempeño

Es importante tomar en cuenta la distribución de la segmentación en las categorías Bueno y Malo de la variable dependiente ya que con estas se construyen el modelo de asignación de crédito y se presumiría que el segmento "Clean" va a tener una menor tasa de malos que el segmento "Dirty" por cómo se definen los segmentos.

SEGMENTO	**Bueno**	**%Fila**	**%Columna**	**Malo**	**%Fila**	**%Columna**	**Total**
CLEAN	263.693	89,64 %	95,75 %	30.474	10,36 %	46,70 %	294.167
DIRTY	11.709	25,19 %	4,25 %	34.777	74,81 %	53,30 %	46.486
TOTAL	**275.402**	**80,85 %**	**100,00 %**	**65.251**	**19,15 %**	**100,00 %**	**340.653**

Tabla 4.6: Distribución de la segmentación vs Bueno/Malo.
Elaboración propia

En efecto se puede apreciar en la tabla 4.6 que en el segmento "Clean" la tasa de malo es de 10,36 % y en el segmento "Dirty" es de 74,81 %, para el segmento "Clean" la tasa de malo tiende a ser baja y para entrenar un modelo en este segmento se requiere que la tasa suba al menos al 20 %[2] para lograr una muestra homogénea por lo que para la base de modelamiento de este segmento se ha aplicado remuestreo.

4.5. Muestra de modelamiento y validación

Considerando que la población de modelamiento presenta una cantidad suficientemente grande de registros se ha decidido generar dos muestras a partir de esta población de igual tamaño, las cuales se generarán tras aplicar muestreo aleatorio simple sin reemplazo.

La primera muestra se denominará de modelamiento y será empleada para entrenar el modelo de Credit Scoring, mientras que la segunda muestra será denominada de validación y servirá para evaluar el desempeño del modelo en una muestra distinta a la de entrenamiento.

Lo anterior, nos permitirá evaluar posibles sobreajustes de los modelos.Para la base de modelamiento se toma en cuenta únicamente las categorías Bueno y Malo de la variable dependiente, así se entrenarán los modelos y para la base de validación si se tomarán en cuenta las demás categorías.

A continuación las muestras de modelamiento y validación para este caso práctico:

Segmento Dirty:

Dirty VarDep	**Modelamiento**	**%**
0	5.872	25,39 %
1	17.256	74,61 %
TOTAL	**23.128**	**100,00 %**

Tabla 4.7: Distribución de la segmentación vs Bueno/Malo.
Elaboración propia

Dirty VarDep	**Validación**
0	5.837
1	17.521
2	3.836
3	26.542
4	10.978
TOTAL	**64.714**

Tabla 4.8: Bases de modelamiento y validación segmento Dirty.
Elaboración propia

Segmento Clean:

Clean VarDep	**Modelamiento**	**%**
0	131.850	89,58 %
1	15.330	10,42 %
TOTAL	**147.180**	**100,00 %**

Tabla 4.9: Distribución de la segmentación vs Bueno/Malo.
Elaboración propia

Clean	Validación
VarDep	
0	131.843
1	15.144
2	27.569
3	2
4	23.963
TOTAL	**198.521**

Tabla 4.10: Bases de modelamiento y validación segmento Clean.
Elaboración propia

4.6. Balanceo de categorías

En la tabla 4.7 se puede apreciar que la tasa de malos de la base de modelamiento para el segmento Dirty es de 74,61 % tasa significativa para realizar los modelos.

En cambio en la tabla 4.9 se puede notar que para el segmento Clean la tasa de malos de la base de modelamiento es del 10,42 % tasa muy baja para entrenar un modelo, lo que indica un desbalanceo de la variable dependiente, para obtener una mejor tasa de malos se puede aplicar diferentes técnicas de remuestreo o balanceo de bases de datos como son: Random Over Sampling (ROS), Random Under Sampling (RUS), Synthetic Minority Over Sampling Thechnique (SMOTE), entre otras[17].

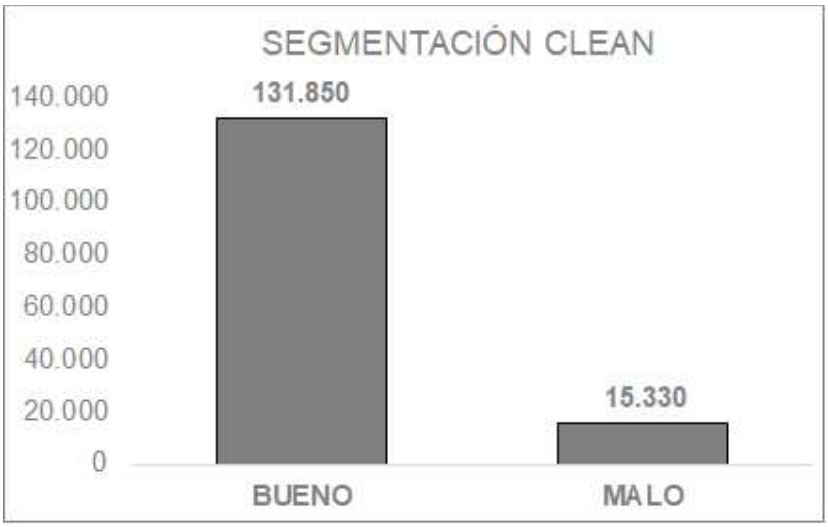


Figura 4.3: Desbalance segmentación clean.
Elaboración propia

Para el presente trabajo se utilizó las técnicas ROS y RUS.

4.6.1. Random Over Sampling (ROS)

En todo conjunto de datos desbalanceado se va a tener una clase que tiene el menor número de registros (minoritario) y una clase que tiene el mayor número de registros (mayoritario), este método replica aleatoriamente registros de la clase minoritaria para lograr un balanceo con la clase mayoritaria.

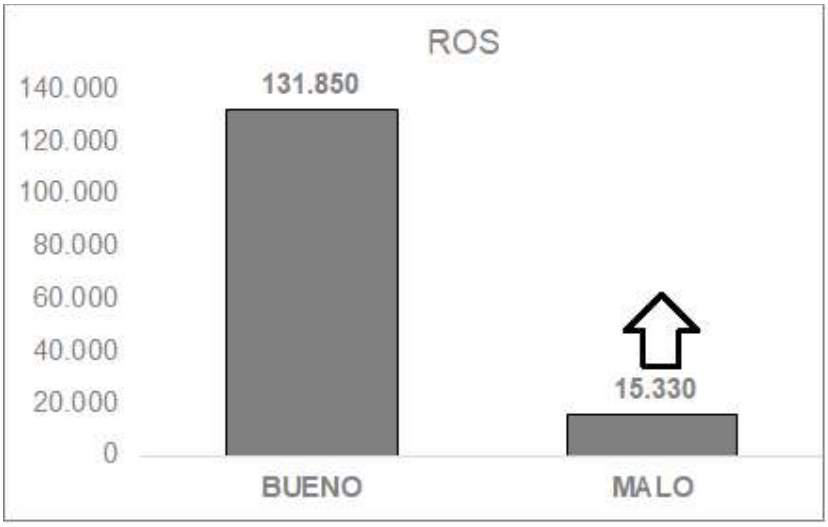


Figura 4.4: Técnica ROS
Elaboración propia

4.6.2. Random Under Sampling (RUS)

Este método al contrario que el anterior elimina aleatoriamente registros de la clase mayoritaria para lograr un balanceo con la clase minoritaria.

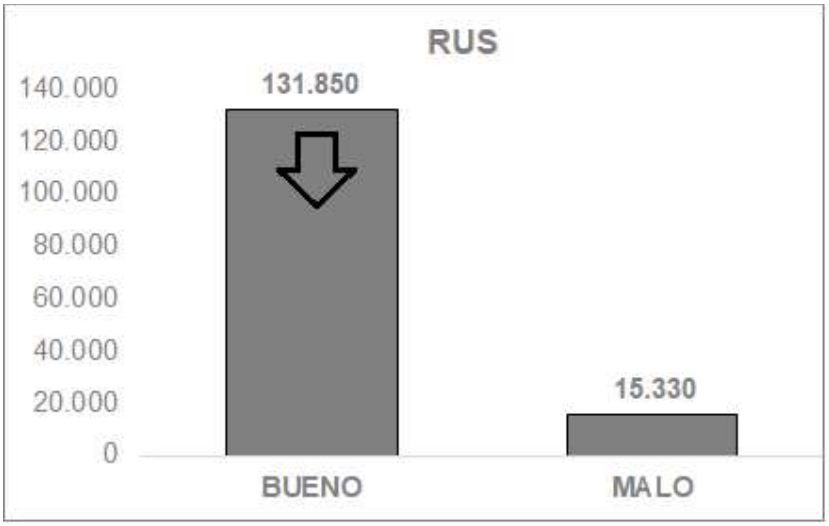


Figura 4.5: Técnica RUS.
Elaboración propia

También se puede hacer una combinación de ambas técnicas:

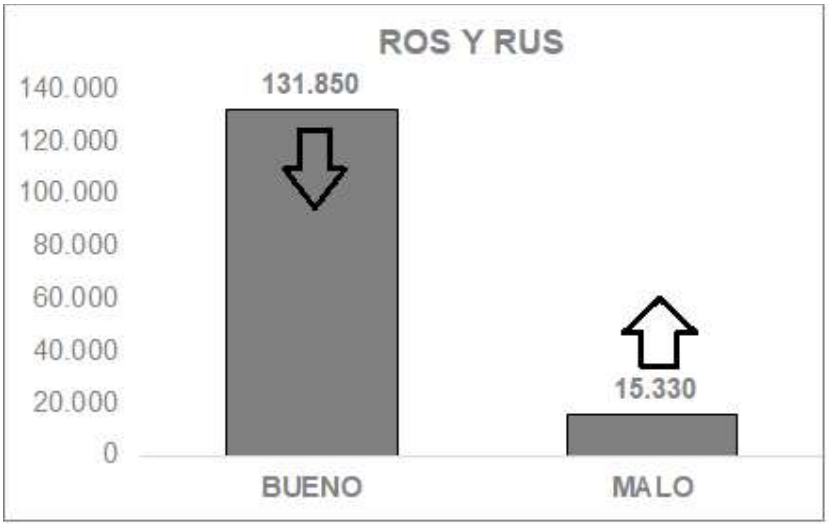


Figura 4.6: Combinación ROS y RUS.
Elaboración propia

Mediante los métodos descritos se realiza un remuestreo para que la tasa de malos suba al menos al 20 % (el código utilizado en lenguaje R se lo puede revisar en anexos) obteniendo lo siguiente:

Clean VarDep	Modelamiento	%
0	141.369	80,0 %
1	35.247	20,0 %
TOTAL	**176.616**	**100,00 %**

Tabla 4.11: Remuestreo base modelamiento segmento Clean.
Elaboración propia

Los modelos serán entrenados con las bases de modelamiento antes planteadas y se realizarán dos modelos uno por cada segmento de la base, luego se procederá a evaluar cada modelo en las respectivas bases de validación.

4.7. Calibración de hiperparámetros para los Random Forest

Para optimizar los hiperparámetros que generan el random forest inicialmente se fijó el mtry en 3, para luego establecer el número de árboles que minimicen el error o a partir de cuantos árboles la disminución del error no sea significativa, y luego establecer de la misma forma el mtry, como se muestra en las gráficas.

Random Forest sin AG

Segmento Dirty:

Se tomó ntree = 100

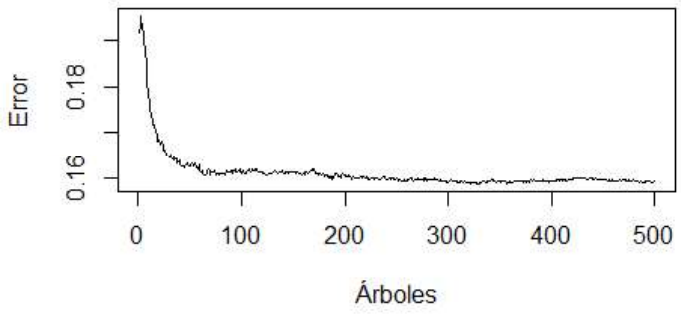


Figura 4.7: Grafica error vs #árboles.
Elaboración propia

Para después tomar mtry=2

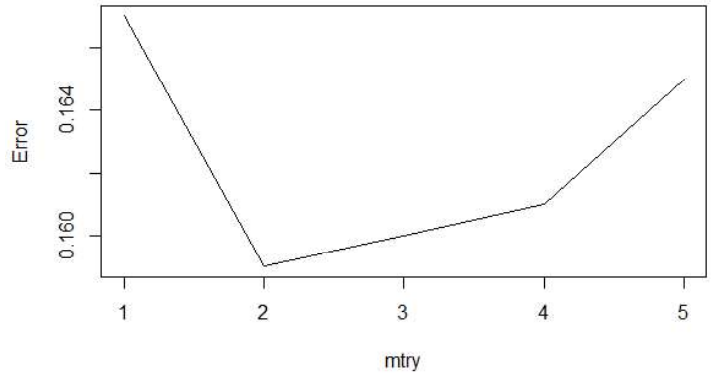


Figura 4.8: Grafica error vs mtry.
Elaboración propia

Segmento Clean:

Se tomó ntree = 100

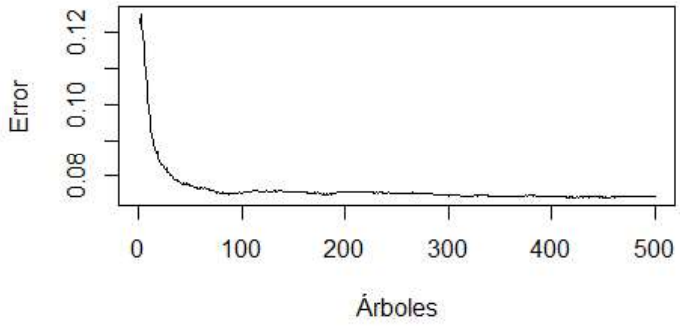


Figura 4.9: Grafica error vs #árboles.
Elaboración propia

Para después tomar mtry=4

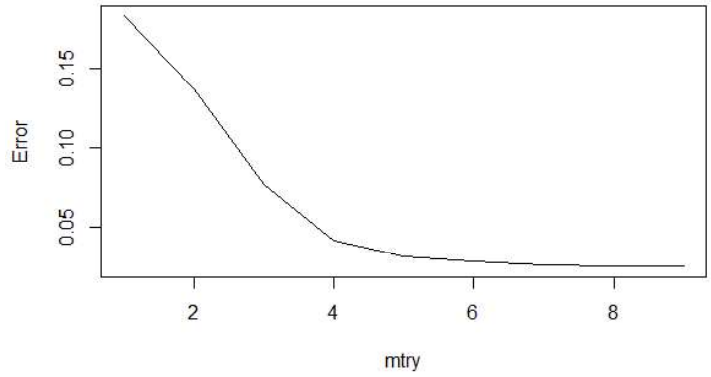


Figura 4.10: Grafica error vs mtry.
Elaboración propia

Random Forest con AG

Segmento Dirty:

Se tomó ntree = 100

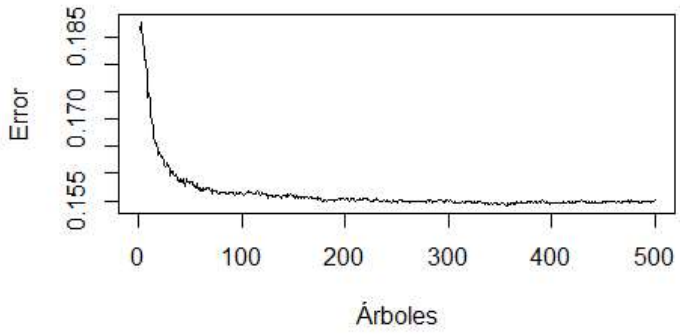


Figura 4.11: Grafica error vs #árboles.
Elaboración propia

Para después tomar mtry=2

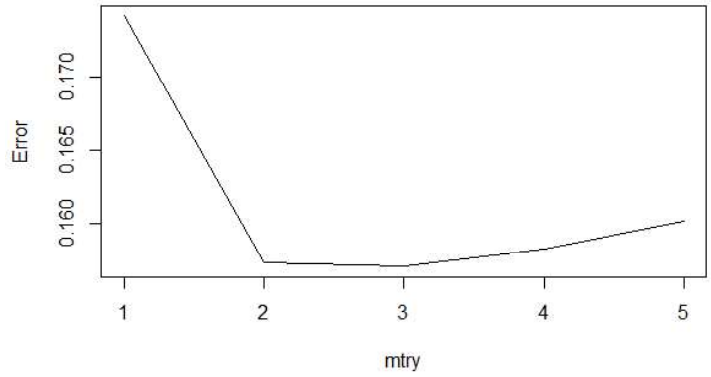


Figura 4.12: Grafica error vs mtry.
Elaboración propia

Segmento Clean:

Se tomó ntree = 100

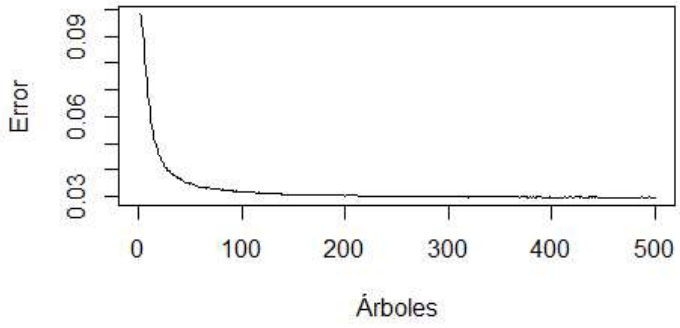


Figura 4.13: Grafica error vs #árboles.
Elaboración propia

Para después tomar mtry=3

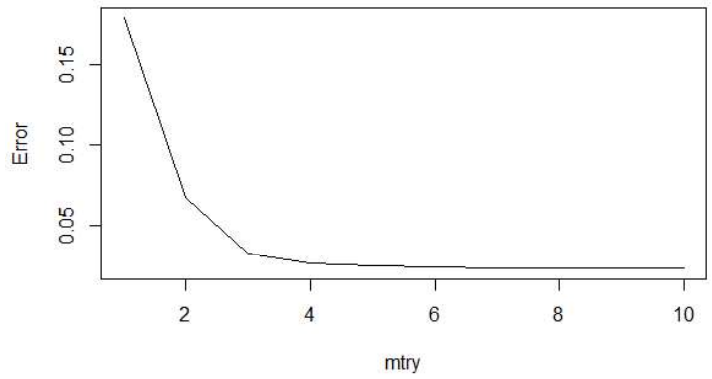


Figura 4.14: Grafica error vs mtry.
Elaboración propia

4.8. Credit Scoring sin AG/con AG

En esta sección se presentarán todos los resultados de los modelos entrenados con la muestra de entrenamiento descrita en la sección previa, se presentan matrices de correlaciones de los modelos así como tablas de significancia de los coeficientes de los modelos de Regresión Logística.

Los modelos fueron entrenados teniendo en cuenta algunos aspectos generales:

- Previo al entrenamiento de los modelos se ha calculado el coeficiente KS para todas las variables de las dos segmentaciones para de esta manera tener una mejor decisión al seleccionar las variables.
- Así mismo se ha calculado el coeficiente VI para las variables categóricas de la muestra de entrenamiento.
- Se ha establecido un rango para el número de variables de los modelos, entre 8 a 16 variables explicativas.

4.8.1. Credit Scoring sin AG

4.8.1.1. Regresión Logística

Los coeficientes estimados para las variables de los modelos de Regresión Logística son significantes a un nivel de confianza del 95 %.

Segmento Dirty:

Variable	Estimate	Std. Error	Z Value	Sign.
Intercepto	2,17	0,11	18,24	0,0000
ln_NMES_ULT_VENC_SBS_SC_12M	-0,66	0,02	-32,31	0,0000
ln_PROM_VEN_SBS_6M	0,19	0,01	17,78	0,0000
NENT_VEN_SCE_3M	0,57	0,05	10,81	0,0000
ln_NTC_VENC_TC_3M	1,00	0,34	2,96	0,0031
prbb_genero_ecivil	-3,07	0,41	-7,44	0,0000
r_NOPE_XVEN_OP_3s6M	-0,44	0,04	-9,78	0,0000
r_NOPE_VENC_31AMAS_OP_3s6M	0,39	0,06	6,33	0,0000
r_DEUDA_TOTAL_SICOM_OP_12s24M	0,51	0,09	5,65	0,0000

Tabla 4.12: Resumen del modelo - logit - dirty.
Elaboración propia

Los resultados de este modelo evidencian 3 variables que premian (con signo negativo) es decir disminuyen la probabilidad de mal pagador y 5 variables que castigan (con signo positivo) es decir que aumentan la probabilidad de mal pagador, adicionalmente todas las variables son significativas, el modelo contiene variables en diferentes períodos (3, 6 , 12, 24 meses) las más recientes de 3 meses son las que más aportan en la clasificación del modelo.

Correlaciones	1	2	3	4	5	6	7	8
ln_NMES_ULT_VENC_SBS_SC_12M	1,00							
ln_PROM_VEN_SBS_6M	-0,52	1,00						
NENT_VEN_SCE_3M	-0,49	0,45	1,00					
ln_NTC_VENC_TC_3M	-0,16	0,28	0,46	1,00				
prbb_genero_ecivil	0,12	-0,10	-0,04	-0,05	1,00			
r_NOPE_XVEN_OP_3s6M	0,24	-0,25	-0,12	-0,05	0,08	1,00		
r_NOPE_VENC_31AMAS_OP_3s6M	-0,17	0,01	0,47	-0,08	0,02	-0,05	1,00	
r_DEUDA_TOTAL_SICOM_OP_12s24M	0,49	-0,18	0,11	-0,03	0,10	0,16	0,15	1,00

Tabla 4.13: Matriz de correlaciones - logit - dirty.
Elaboración propia

Segmento Clean:

Variable	Estimate	Std. Error	Z Value	Sign.
Intercepto	5,82	0,24	23,82	0,0000
NENT_VEN_SCE_36M	0,79	0,01	66,86	0,0000
r_NTC_APERT_SCE_12a36M	0,38	0,02	22,50	0,0000
NTC_XVEN_TC_3M	0,18	0,01	38,79	0,0000
r_DEUDA_TOTAL_SCE_6a12M	0,44	0,03	12,67	0,0000
prbb_genero_ecivil	-5,81	0,19	-29,93	0,0000
r_NOPE_APERT_SBS_OP_12s24M	0,46	0,01	30,81	0,0000
ln_NMES_ULT_REFIN_TC_12M	-0,47	0,03	-14,11	0,0000
ln_NMES_ULT_REFIN_OP_12M	-0,29	0,02	-12,01	0,0000
r_PROM_VENSBSsPROM_DEUDASBS_TC36M	-1,48	0,22	-6,78	0,0000

Tabla 4.14: Resumen del modelo - logit - clean.
Elaboración propia

Los resultados de este modelo evidencian 4 variables que premian (con signo negativo) es decir disminuyen la probabilidad de mal pagador y 5 variables que castigan (con signo positivo) es decir que aumentan la probabilidad de mal pagador, adicionalmente todas las variables son significativas, la variable que más aporta en la clasificación del modelo es la de prbb_genero_ecivil.

Correlaciones	1	2	3	4	5	6	7	8	9
NENT_VEN_SCE_36M	1,00								
r_NTC_APERT_SCE_12a36M	0,00	1,00							
NTC_XVEN_TC_3M	0,10	0,35	1,00						
r_DEUDA_TOTAL_SCE_6a12M	-0,04	0,14	-0,13	1,00					
prbb_genero_ecivil	0,01	-0,09	-0,02	-0,10	1,00				
r_NOPE_APERT_SBS_OP_12s24M	-0,01	-0,07	-0,17	0,31	-0,01	1,00			
ln_NMES_ULT_REFIN_TC_12M	-0,03	-0,06	-0,06	-0,01	0,00	0,00	1,00		
ln_NMES_ULT_REFIN_OP_12M	-0,01	0,03	0,02	0,01	0,00	-0,06	0,05	1,00	
rPROMVENSBSsPROM_DEUDASBS_TC36M	0,20	-0,01	-0,02	0,00	-0,01	0,00	-0,01	-0,01	1,00

Tabla 4.15: Matriz de correlaciones - logit - clean.
Elaboración propia

4.8.2. Random Forest

Se han entrenado los modelos tomando en cuenta las variables con mayor KS, a continuación las variables seleccionadas con las correlaciones entre las mismas:

Segmento Dirty:

Correlaciones	1	2	3	4	5	6	7	8	9	10	11
ln_NMES_ULT_VENC_SBS_SC_12M	1,00										
ln_PROM_VEN_SBS_6M	-0,52	1,00									
NENT_VEN_SCE_3M	-0,49	0,45	1.00								
ln_MVALVEN_SBS_TC_6M	-0,35	0,57	0.32	1,00							
r_MVALEN_SBSsDEUDA_TOTAL	-0,13	0,21	0.05	0,41	1,00						
rPROM_VENSBSsPROM_DEUDATOTAL	-0,19	0,36	0.12	0,33	0,42	1,00					
rPROM_VENSBS_TCsPROMDEUDA	-0,20	0,31	0.09	0,57	0,37	0,41	1.00				
r_NOPE_XVEN_OP_3s6M	0,24	-0,25	-0,12	-0,15	-0,07	-0,20	-0,41	1,00			
NOPE_XVEN_OP_3M	0,22	-0,23	-0,08	-0,17	-0,10	-0,17	-0,33	0,58	1,00		
NTC_VENC_31AMAS_TC_24M	-0,25	0,47	0.26	0,58	0,31	0,52	0.52	-0,18	-0,19	1,00	
prbb_genero_ecivil	0,12	-0,10	-0,04	-0,11	-0,08	-0,04	-0,09	0,08	0,13	-0,07	1,00

Tabla 4.16: Matriz de correlaciones - rf - dirty.
Elaboración propia

Segmento Clean:

Correlaciones	1	2	3	4	5	6	7	8	9	10	11
r_DEUDA_TOTAL_SBS_TC_3s6M	1,00										
NENT_VEN_SCE_36M	0,07	1,00									
NTC_APERT_SCE_24M	0,54	-0,01	1,00								
NOPE_XVEN_OP_3M	-0,26	0,03	-0,14	1,00							
ln_DEUDA_TOTAL_SCE_3M	-0,04	0,05	0,03	0,41	1,00						
r_DEUDA_TOTAL_SCE_3a6M	0,10	-0,03	0,00	0,00	-0,02	1,00					
ln_PROM_VEN_SCE_24M	0,01	0,63	-0,04	0,02	0,03	-0,01	1,00				
r_DEUDA_TOTAL_SCE_6a12M	-0,01	-0,04	-0,01	0,00	-0,03	0,62	-0,03	1,00			
ln_MVALVEN_SBS_TC_36M	0,17	0,59	0,04	-0,07	0,07	-0,04	0,33	-0,07	1.00		
r_NOPE_APERT_SBS_OP_12s24M	-0,21	-0,01	-0,09	0,33	0,12	0,17	-0,01	0,31	-0,06	1,00	
prbb_genero_ecivil	-0,07	0,01	-0,08	0,09	0,13	-0,07	0,01	-0,10	0.00	-0,01	1,00

Tabla 4.17: Matriz de correlaciones - rf - clean.
Elaboración propia

4.9. Credit Scoring con AG

Para estos modelos se ha tomado las 60 variables con mayor KS y que no estén correlacionadas (correlación menor a 0,65 en valor absoluto), de esta manera el algoritmo genético devuelve la combinación de variables que considere mejor.

4.9.1. Regresión Logística

Segmento Dirty:

Variable	Estimate	Std. Error	Z Value	Sign.
Intercepto	0,81	0,08	10,65	0,0000
ln_NMES_ULT_VENC_SBS_SC_12M	-0,61	0,02	-29,74	0,0000
NENT_VEN_SCE_3M	0,41	0,05	8,12	0,0000
ln_MVALVEN_SBS_TC_6M	0,17	0,01	19,28	0,0000
rPROM_VENSBS_TCsPROMDEUDA_TOTALSBS_OP12M	0,18	0,06	3,14	0,0017
NOPE_XVEN_OP_3M	-0,03	0,01	-2,21	0,0270
NOPE_VENC_31AMAS_OP_3M	0,33	0,03	10,01	0,0000
r_DEUDA_TOTAL_SICOM_OP_12s24M	0,47	0,09	5,05	0,0000
NENT_VEN_SCE_12M	0,25	0,05	5,24	0,0000
ln_MVALVEN_SBS_OP_3M	0,20	0,01	14,32	0,0000
r_NOPE_APERT_SCE_12a24M	-0,34	0,05	-6,65	0,0000

Tabla 4.18: Resumen del modelo - logit - dirty - con AG.
Elaboración propia

Los resultados de este modelo evidencian 3 variables que premian (con signo negativo) es decir disminuyen la probabilidad de mal pagador y 7 variables que castigan (con signo positivo) es decir que aumentan la probabilidad de mal pagador adicionalmente todas las variables son significativas.

Correlaciones	1	2	3	4	5	6	7	8	9	10
ln_NMES_ULT_VENC_SBS_SC_12M	1,00									
NENT_VEN_SCE_3M	-0,49	1,00								
ln_MVALVEN_SBS_TC_6M	-0,35	0,32	1,00							
rPROM_VENSBS_TCsPROMDEUDA_TOTALSBS	-0,20	0,09	0,57	1,00						
NOPE_XVEN_OP_3M	0,22	-0,08	-0,17	-0,33	1,00					
NOPE_VENC_31AMAS_OP_3M	-0,23	0,47	-0,28	-0,28	-0,06	1,00				
r_DEUDA_TOTAL_SICOM_OP_12s24M	0,49	0,11	-0,11	-0,11	0.23	0,08	1,00			
NENT_VEN_SCE_12M	-0,22	0,60	0,29	0,10	0,03	0,28	0,11	1,00		
ln_MVALVEN_SBS_OP_3M	-0,30	0,29	-0,20	-0,26	-0.04	0,38	-0,10	0,14	1,00	
r_NOPE_APERT_SCE_12a24M	0,19	-0,08	-0,11	-0,25	0.40	-0,07	0,18	-0,03	-0,03	1,00

Tabla 4.19: Matriz de correlaciones - logit - dirty - con AG.
Elaboración propia

Segmento Clean:

Variable	Estimate	Std. Error	Z Value	Sign.
Intercepto	-3,31	0,05	-68,51	0,0000
NENT_VEN_SCE_36M	0,69	0,01	47,83	0,0000
PROM_DEUDA_TOTAL_SBS_TC_3M	0,001	0,00	23,54	0,0000
NTC_APERT_SCE_24M	0,20	0,01	23,57	0,0000
NTC_XVEN_TC_3M	0,05	0,01	6,11	0,0000
LN_DEUDA_TOTAL_SCE_3M	0,06	0,00	12,53	0,0000
NOPE_APERT_SCE_24M	0,05	0,00	26,38	0,0000
MVALVEN_SBS_TC_36M	0,001	0,00	8,40	0,0000
NENT_VEN_SCE_24M	0,27	0,03	9,45	0,0000
r_NOPE_APERT_SBS_OP_12s24M	0,23	0,02	14,53	0,0000
r_DEUDA_TOTAL_SCE_12a24M	1,14	0,03	35,62	0,0000

Tabla 4.20: Resumen del modelo - logit - clean - con AG.
Elaboración propia

Los resultados de este modelo evidencian 1 intercepto con signo negativo que hace disminuir la probabilidad de mal pagador y 10 variables que castigan (con signo positivo) es decir que aumentan la probabilidad de mal pagador adicionalmente todas las variables son significativas.

Correlaciones	1	2	3	4	5	6	7	8	9	10
NENT_VEN_SCE_36M	1,00									
PROM_DEUDA_TOTAL_SBS_TC_3M	0,10	1,00								
NTC_APERT_SCE_24M	-0,01	0,21	1,00							
NTC_XVEN_TC_3M	0,10	0,61	0,59	1,00						
LN_DEUDA_TOTAL_SCE_3M	0,05	0,31	0,03	0,18	1,00					
NOPE_APERT_SCE_24M	0,04	-0,03	-0,10	-0,16	0,25	1,00				
MVALVEN_SBS_TC_36M	0,31	0,27	0,01	0,16	0,09	-0,01	1,00			
NENT_VEN_SCE_24M	0,57	0,19	0,05	0,23	0,06	-0,05	0,45	1,00		
r_NOPE_APERT_SBS_OP_12s24M	-0,01	-0,06	-0,09	-0,17	0,12	0,27	-0,02	-0,07	1,00	
r_DEUDA_TOTAL_SCE_12a24M	-0,12	-0,11	0,03	-0,18	-0,02	0,05	-0,07	-0,11	0,30	1,00

Tabla 4.21: Matriz de correlaciones - logit - clean - con AG.
Elaboración propia

4.9.2. Random Forest

A continuación las variables seleccionadas con sus correlaciones:

Segmento Dirty:

Correlaciones	1	2	3	4	5	6	7	8	9	10	11
NMES_ULT_VENC_SBS_SC_12M	1,00										
PROM_VEN_SBS_24M	-0,04	1,00									
MAVALEN_SBS_TC_6M	-0,13	0,03	1,00								
PROM_VEN_SBS_TC_6M	-0,08	0,08	0,51	1,00							
NTC_VENC_TC_3M	-0,25	0.00	0,46	0,36	1,00						
r_MVALVEN_SBSsDEUDA_TOTAL_SBS_TC	-0,16	-0,01	0,17	0,11	0,32	1,00					
rPROM_VENSBS_TCsPROMDEUDA_TOTAL	-0,18	-0,01	0,21	0,21	0,45	0,33	1,00				
NOPE_XVEN_OP_3M	0,17	-0,05	-0,07	-0,08	-0,16	-0,10	-0,31	1,00			
NOPE_NDI_OP_3M	-0,21	-0,02	-0,10	-0,06	-0,19	-0,15	-0,24	0,13	1,00		
rMVALVEN_SBSTCsMVALVEN_SBSOP_12M	-0,29	-0,03	0,37	0,22	0,64	0,51	0,62	-0,20	-0,35	1,00	
DEUDA_TOTAL_SICOM_OP_12M	0,36	-0,02	-0,03	-0,03	-0,04	-0,05	-0,05	0,17	-0,07	-0,07	1,00

Tabla 4.22: Matriz de correlaciones - rf - dirty - con AG.
Elaboración propia

Segmento Clean:

Correlaciones	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NENT_VEN_SCE_36M	1,00														
PROM_DEUDA_TOTAL_SBS_TC_3M	0,10	1,00													
NTC_XVEN_TC_3M	0,10	0,61	1,00												
NOPE_XVEN_OP_3M	0,03	-0,05	-0,20	1,00											
DEUDA_TOTAL_SCE_3M	0,01	0,16	0,08	0,14	1,00										
LN_DEUDA_TOTAL_SCE_3M	0,05	0,31	0,18	0,41	0,23	1,00									
r_DEUDA_TOTAL_SCE_3a6M	-0,03	-0,05	-0,09	0,00	-0,01	-0,02	1,00								
r_DEUDA_TOTAL_SCE_6a12M	-0,04	-0,08	-0,13	0,00	-0,01	-0,03	0,62	1.00							
r_NOPE_APERT_SBSsSCE_OP_24M	-0,03	-0,02	-0,15	0,32	0,05	0,15	0,05	0.09	1,00						
NMES_ULT_REFIN_TC_12M	-0,02	-0,03	-0,05	0,02	0,00	-0,01	-0,02	-0,01	-0,00	1,00					
NTC_APERT_SCE_6M	0,00	0,10	0,37	-0,09	0,00	-0,03	0,12	0.12	-0,06	-0,06	1,00				
NMES_ULT_REFIN_OP_12M	-0,01	0,01	0,02	-0,01	0,00	-0,01	0,00	0.01	-0,05	0,33	0,02	1,00			
r_DEUDA_TOTAL_SCE_12a24M	-0,12	-0,11	-0,18	0,04	-0,01	-0,02	0,28	0.58	0,16	0,01	0,08	0,01	1,00		
r_DEUDA_TOTAL_SBS_OP_3s12M	-0,02	-0,07	-0,18	0,29	0,03	0,15	0,40	0.40	0,59	-0,01	-0,06	-0,03	0,25	1.00	
r_NOPE_APERT_SCE_12a24M	-0,03	-0,11	-0,23	0,32	0,02	0,15	0,24	0.41	0,39	-0,01	-0,05	-0,03	0,39	0.47	1,00

Tabla 4.23: Matriz de correlaciones - rf - clean - con AG.
Elaboración propia

4.10. Alineación de Scores

En este capítulo se aplica la técnica de Alineación de Scores para los segmentos Clean y Dirty para así obtener un solo modelo para la población a partir de los resultados obtenidos en los scores de las muestras.

4.10.1. Importancia de la Alineación de Scores

La importancia de alinear los scores de los segmentos Clean y Dirty radica en obtener un solo modelo unificado que pueda evaluarse en el cliente ya que los modelos por separados tienes diferentes valores extremos en los rangos de los 10 intervalos y también de esta manera proporcionar la misma evaluación e interpretación del riesgo.

4.10.2. Procedimiento para Alineación de Scores

1. Seleccionar un puntaje base, para este caso el puntaje del segmento Clean.
2. Ajustar una regresión exponencial entre el puntaje medio del rango y la probabilidad de incumplimiento (PD).
3. Mediante la transformación anterior obtenemos los scores de la segmentación Dirty.
4. Obtenemos los nuevos rangos de score alineados, los rangos del segmento Clean se mantienen invariantes ya que se tomaron como puntaje base.

Como resultados obtenemos las tablas performance mencionadas en el capítulo 6.

Capítulo 5

Comparación de resultados

En este capítulo se busca comparar y dejar en evidencia si existe o no mejoría al aplicar algoritmos genéticos en la selección de variables para modelos de credit scoring mediante medidas de calidad de ajuste de los modelos entrenados.

Como se pudo ver en el capítulo anterior se ha entrenado en total 8 modelos credit scoring, 4 modelos sin Algoritmos Genéticos (sin AG) es decir seleccionando las variables explicativas mediante métodos tradicionales mencionados en el anterior capítulo y 4 con Algoritmos Genéticos (con AG), usando dos técnicas estadísticas: Regresión Logistica y Random Forest para dos segmentaciones de la base total: segmentación Dirty y segmentación Clean.

5.1. Tablas Performance

Las tablas performance de los modelos realizados son herramientas que permiten visualizar detalladamente la calidad de discriminación del modelo generalmente dividida por deciles de la muestra, donde cada decil tiene rangos de probabilidad estimada de impago además del número de sujetos pago e impago.

Las tablas tal cual se pueden ver en el cápitulo 6, se tomará la primera de ellas como ejemplo para describirlas.

KS	ROC	GINI
61,4	86,5	73,1

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
657	999	2.313	10 %	10 %	646	4 %	4 %	27,93 %	27,9 %
570	657	2.313	10 %	20 %	946	5 %	9 %	40,90 %	34,4 %
386	570	2.312	10 %	30 %	1.084	6 %	16 %	46,89 %	38,6 %
182	385	2.313	10 %	40 %	1.535	9 %	24 %	66,36 %	45,5 %
118	182	2.313	10 %	50 %	1.940	11 %	36 %	83,87 %	53,2 %
85	118	2.313	10 %	60 %	2.112	12 %	48 %	91,31 %	59,5 %
60	85	2.313	10 %	70 %	2.189	13 %	61 %	94,64 %	64,6 %
40	60	2.312	10 %	80 %	2.243	13 %	74 %	97,02 %	68,6 %
23	40	2.313	10 %	90 %	2.266	13 %	87 %	97,97 %	71,9 %
1	23	2.313	10 %	100 %	2.295	13 %	100 %	99,22 %	74,6 %
Total		23.128			17.256				

Tabla 5.1: Tabla performance modelo logit - segmentación dirty - base de modelamiento.
Elaboración propia

La tabla 5.1 es la tabla performance describe la calidad de discriminación del modelo logit (Regresión Logistica) para la segmentación dirty sin algoritmos genéticos en la selección de variables.

Encabezando a la tabla podemos encontrar tres medidas de calidad de discriminación del modelo: KS, ROC, GINI, las cuales servirán para poder comparar los modelos sin AG con AG, en este modelo los 3 coeficientes son altos, el máximo es 100 pero en la práctica es irreal obtener este valor.

En las columnas tenemos:

- Score (Min): El puntaje mínimo de score del rango.
- Score (Max): El puntaje máximo de score del rango.
- Total (Int): El número total de sujetos en el rango.
- Total (Int %): El porcentaje que representa el número total de sujetos en el rango respecto al total de la población.
- Total (Cum %):El porcentaje de sujetos acumulados hasta ese rango.
- Malo (Int): El número de sujetos malos en el rango.

- Malo (Int %): El porcentaje que representa el número de sujetos malos en el rango respecto al total de malos de la población.
- Malo (Cum %): El porcentaje de sujetos malos acumulado hasta ese rango.
- Malo Rate (Int): El porcentaje de malos en el rango respecto al total de la población del rango.
- Malo Rate (Cum): El porcentaje de malos acumulado hasta ese rango.

La columna Malo Rate (Cum) ayuda a la institución a definir el riesgo máximo que quiera asumir y de esta manera definir un punto de corte.

A continuación se presenta un cuadro resumen comparativo con las medidas de calidad de discriminación de todos los modelos realizados en donde se presentan las medidas KS, ROC, GINI y se busca comparar principalmente el uso de algoritmos genéticos frente a su omisión.

MODELO			SIN AG			CON AG		
			KS	ROC	GINI	KS	ROC	GINI
Regresión Logística	CLEAN	MOD	24,9	66,2	32,3	25,6	67,5	34,9
		VAL	24,1	66,0	32,0	26,3	67,6	35,3
	DIRTY	MOD	61,4	86,5	73,1	62,4	86,9	73,7
		VAL	70,1	91,1	82,2	63,4	88,2	76,5
Random Forest	CLEAN	MOD	32,2	72,1	44,2	34,7	73,6	47,2
		VAL	29,6	70,1	40,2	32,4	71,5	43,0
	DIRTY	MOD	65,1	89,0	78,0	65,4	88,9	77,9
		VAL	64,5	89,6	79,2	62,6	89,2	78,5

Tabla 5.2: Tabla comparativa modelos sin AG vs modelos con AG.
Elaboración propia

5.2. Tablas Performance con Alineación de Scores

A continuación se muestra un cuadro comparativo de los modelos resultantes al aplicar la técnica Alineación de Scores:

MODELO		SIN AG			CON AG		
		KS	ROC	GINI	KS	ROC	GINI
Regresión Logística	MOD	38,2	75,2	50,3	39,5	76,0	52,0
	VAL	64,2	87,9	75,7	64,9	88,4	76,9
Random Forest	MOD	43,4	79,9	59,8	45,3	81,0	62,0
	VAL	62,4	88,0	76,0	63,7	89,0	78,1

Tabla 5.3: Tabla comparativa modelos sin AG vs modelos con AG.
Elaboración propia

5.3. Resultados

- En general se obtuvo mejores modelos con algoritmos genéticos del total de 8 evaluaciones con las medidas KS, GINI, ROC, 5 mejoraron en todas las medidas y 1 modelo mejoró en la medida KS al utilizar AG, véase Tabla 5.2.
- En cuanto a las 2 evaluaciones que no mejoraron con algoritmos genéticos se tratan de modelos de la segmentación dirty, donde los modelos tradicionales que se obtuvieron fueron buenos en cuanto a las medidas de calidad.
- Para la segmentación clean al aplicar algoritmos genéticos hubo una mejora en todos los modelos como se puede evidenciar en la Tabla 5.2, los modelos tradicionales en esta segmentación no fueron tan buenos como los de la segmentación dirty.
- Los modelos Random Forest tienen una mejor calidad de discriminación que los modelos de Regresión Logística como se puede ver en la Tabla 5.2.
- Cuando se aplica la técnica de Alineación de Scores para fusionar las tablas performance de los modelos de los segmentos dirty y clean en uno solo se obtuvo mejores medidas KS, GINI, ROC.

Capítulo 6

Tablas Performance

6.1. Modelos sin AG

6.1.1. Regresión Logística

Segmento Dirty:

BASE DE MODELAMIENTO

KS	**ROC**	**GINI**
61,4	86,5	73,1

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
657	999	2.313	10 %	10 %	646	4 %	4 %	27,93 %	27,9 %
570	657	2.313	10 %	20 %	946	5 %	9 %	40,90 %	34,4 %
386	570	2.312	10 %	30 %	1.084	6 %	16 %	46,89 %	38,6 %
182	385	2.313	10 %	40 %	1.535	9 %	24 %	66,36 %	45,5 %
118	182	2.313	10 %	50 %	1.940	11 %	36 %	83,87 %	53,2 %
85	118	2.313	10 %	60 %	2.112	12 %	48 %	91,31 %	59,5 %
60	85	2.313	10 %	70 %	2.189	13 %	61 %	94,64 %	64,6 %
40	60	2.312	10 %	80 %	2.243	13 %	74 %	97,02 %	68,6 %
23	40	2.313	10 %	90 %	2.266	13 %	87 %	97,97 %	71,9 %
1	23	2.313	10 %	100 %	2.295	13 %	100 %	99,22 %	74,6 %
Total		23.128			17.256				

Tabla 6.1: Tabla performance modelo logit - segmentación dirty - base de modelamiento.
Elaboración propia

La tabla 6.1 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 27,93 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 99,22 % de malos pagadores.

BASE DE VALIDACIÓN

KS	**ROC**	**GINI**
70,1	91,1	82,2

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
628	999	6.471	10 %	10 %	1.434	3 %	3 %	22,16 %	22,2 %
549	628	6.472	10 %	20 %	2.107	4 %	7 %	32,56 %	27,4 %
226	549	6.471	10 %	30 %	2.724	6 %	13 %	42,10 %	32,3 %
103	226	6.472	10 %	40 %	4.596	10 %	23 %	71,01 %	42,0 %
62	103	6.471	10 %	50 %	5.660	12 %	34 %	87,47 %	51,1 %
41	62	6.471	10 %	60 %	6.077	13 %	47 %	93,91 %	58,2 %
30	41	6.472	10 %	70 %	6.222	13 %	60 %	96,14 %	63,6 %
17	30	6.471	10 %	80 %	6.256	13 %	73 %	96,68 %	67,8 %
8	17	6.472	10 %	90 %	6.373	13 %	87 %	98,47 %	71,2 %
1	8	6.471	10 %	100 %	6.442	13 %	100 %	99,55 %	74,0 %
Total		64.714			47.891				

Tabla 6.2: Tabla performance modelo logit - segmentación dirty - base de validación. Elaboración propia

La tabla 6.2 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 22,16 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 99,55 % de malos pagadores.

Segmento Clean:

BASE DE MODELAMIENTO

KS	**ROC**	**GINI**
24,9	66,2	32,3

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
894	999	17.662	10 %	10 %	1.757	5 %	5 %	9,9 %	9,9 %
878	894	17.661	10 %	20 %	1.892	5 %	10 %	10,7 %	10,3 %
862	878	17.662	10 %	30 %	2.154	6 %	16 %	12,2 %	11,0 %
847	862	17.661	10 %	40 %	2.548	7 %	24 %	14,4 %	11,8 %
829	847	17.662	10 %	50 %	2.843	8 %	32 %	16,1 %	12,7 %
807	829	17.662	10 %	60 %	2.937	8 %	40 %	16,6 %	13,3 %
780	807	17.661	10 %	70 %	3.838	11 %	51 %	21,7 %	14,5 %
743	780	17.662	10 %	80 %	4.479	13 %	64 %	25,3 %	15,9 %
675	743	17.661	10 %	90 %	5.259	15 %	79 %	29,7 %	17,4 %
1	675	17.662	10 %	100 %	7.540	21 %	100 %	42,6 %	20,0 %
Total		176.616			35.247				

Tabla 6.3: Tabla performance modelo logit - segmentación clean - base de modelamiento.
Elaboración propia

La tabla 6.3 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 9,9 % de malos pagadores más de 10 puntos porcentuales de diferencia con el segmento dirty, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 42,6 % de malos pagadores.

BASE DE VALIDACIÓN

KS	**ROC**	**GINI**
24,1	66,0	32,0

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
896	999	19.852	10 %	10 %	1.013	6 %	6 %	5,10 %	5,1 %
880	896	19.852	10 %	20 %	935	6 %	12 %	4,71 %	4,9 %
864	880	19.852	10 %	30 %	1.041	6 %	18 %	5,24 %	5,0 %
849	864	19.852	10 %	40 %	1.235	7 %	25 %	6,22 %	5,3 %
831	849	19.852	10 %	50 %	1.336	8 %	33 %	6,73 %	5,6 %
810	831	19.853	10 %	60 %	1.546	9 %	42 %	7,79 %	6,0 %
783	810	19.852	10 %	70 %	1.727	10 %	53 %	8,70 %	6,4 %
746	783	19.852	10 %	80 %	2.077	12 %	65 %	10,46 %	6,9 %
677	746	19.852	10 %	90 %	2.377	14 %	79 %	11,97 %	7,4 %
1	677	19.852	10 %	100 %	3.497	21 %	100 %	17,62 %	8,5 %
Total		198.521			16.784				

Tabla 6.4: Tabla performance modelo logit - segmentación clean - base de validación. Elaboración propia

La tabla 6.4 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 5,1 % de malos pagadores más de 17 puntos porcentuales de diferencia con el segmento dirty, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 17,62 % de malos pagadores más de 80 % puntos porcentuales de diferencia con el segmento dirty, las empresas otorgadoras de crédito buscarían centrarse en esta segmentación clean ya que genera mucho menos pérdida que la segmentación dirty.

6.1.2. Random Forest

Segmento Dirty:

BASE DE MODELAMIENTO

KS	ROC	GINI
65,1	89,0	78,0

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
693	999	2.313	10 %	10 %	551	3,1 %	3,1 %	23,8 %	23,8 %
586	693	2.313	10 %	20 %	761	4,4 %	7,6 %	32,9 %	28,3 %
407	586	2.312	10 %	30 %	1.123	6,5 %	14,1 %	48,5 %	35,1 %
216	406	2.313	10 %	40 %	1.613	9,3 %	23,4 %	69,7 %	43,7 %
123	216	2.313	10 %	50 %	1.977	11,4 %	34,9 %	85,4 %	52,1 %
60	123	2.313	10 %	60 %	2.138	12,3 %	47,3 %	92,4 %	58,8 %
30	60	2.313	10 %	70 %	2.241	12,9 %	60,2 %	96,8 %	64,2 %
19	30	2.312	10 %	80 %	2.263	13,1 %	73,4 %	97,8 %	68,4 %
9	19	2.313	10 %	90 %	2.287	13,2 %	86,6 %	98,8 %	71,8 %
1	9	2.313	10 %	100 %	2.302	13,3 %	100,0 %	99,5 %	74,6 %
Total		23.128			17.256				

Tabla 6.5: Tabla performance modelo rf - segmentación dirty - base de modelamiento. Elaboración propia

La tabla 6.5 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 23,8 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 99,5 % de malos pagadores, además el KS es casi 4 puntos más alto que el modelo de regresión logística.

BASE DE VALIDACIÓN

KS	**ROC**	**GINI**
64,5	89,6	79,2

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
683	999	6.471	10 %	10 %	1.116	2,57 %	2,5 %	17,2 %	17,2 %
573	683	6.472	10 %	20 %	1.800	4,14 %	6,7 %	27,8 %	22,5 %
299	573	6.471	10 %	30 %	2.693	6,19 %	12,9 %	41,6 %	28,8 %
208	299	6.472	10 %	40 %	3.524	8,10 %	21,0 %	54,4 %	35,2 %
156	208	6.471	10 %	50 %	4.556	10,48 %	31,4 %	70,4 %	42,3 %
99	156	6.471	10 %	60 %	5.122	11,78 %	43,2 %	79,1 %	48,4 %
32	99	6.472	10 %	70 %	5.804	13,35 %	56,6 %	89,6 %	54,3 %
19	32	6.471	10 %	80 %	6.189	14,23 %	70,8 %	95,6 %	59,5 %
10	19	6.472	10 %	90 %	6.292	14,47 %	85,3 %	97,2 %	63,6 %
1	10	6.471	10 %	100 %	6.393	14,70 %	100,0 %	98,7 %	67,2 %
Total		64.714			43.489				

Tabla 6.6: Tabla performance modelo rf - segmentación dirty - base de validación. Elaboración propia

La tabla 6.6 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 17,2 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 98,7 % de malos pagadores.

Segmento Clean:

BASE DE MODELAMIENTO

KS	ROC	GINI
32,2	72,1	44,2

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
919	999	17.662	10 %	10 %	579	2 %	2 %	3,28 %	3,3 %
886	919	17.661	10 %	20 %	1.311	4 %	5 %	7,42 %	5,4 %
868	886	17.662	10 %	30 %	1.681	5 %	10 %	9,52 %	6,7 %
850	868	17.661	10 %	40 %	2.338	7 %	17 %	13,24 %	8,4 %
822	850	17.662	10 %	50 %	2.822	8 %	25 %	15,98 %	9,9 %
790	822	17.662	10 %	60 %	3.337	9 %	34 %	18,89 %	11,4 %
758	790	17.661	10 %	70 %	3.986	11 %	46 %	22,57 %	13,0 %
718	758	17.662	10 %	80 %	4.708	13 %	59 %	26,66 %	14,7 %
663	718	17.661	10 %	90 %	5.940	17 %	76 %	33,63 %	16,8 %
1	663	17.662	10 %	100 %	8.545	24 %	100 %	48,38 %	20,0 %
Total		176.616			35.247				

Tabla 6.7: Tabla performance modelo rf - segmentación clean - base de modelamiento. Elaboración propia

La tabla 6.7 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 3,28 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 48,38 % de malos pagadores, y un KS con de 7 puntos más que la regresión logística.

BASE DE VALIDACIÓN

KS	**ROC**	**GINI**
29,6	70,1	40,2

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
918	999	19.852	10 %	10 %	461	3 %	3 %	2,32 %	2,3 %
886	918	19.852	10 %	20 %	708	4 %	7 %	3,57 %	2,9 %
868	886	19.852	10 %	30 %	1.024	6 %	13 %	5,16 %	3,7 %
851	868	19.852	10 %	40 %	1.097	6 %	19 %	5,53 %	4,1 %
824	851	19.852	10 %	50 %	1.369	8 %	27 %	6,90 %	4,7 %
793	824	19.853	10 %	60 %	1.629	10 %	37 %	8,21 %	5,3 %
761	793	19.852	10 %	70 %	1.961	12 %	49 %	9,88 %	5,9 %
723	761	19.852	10 %	80 %	2.281	13 %	62 %	11,49 %	6,6 %
666	723	19.852	10 %	90 %	2.628	15 %	78 %	13,24 %	7,4 %
1	666	19.852	10 %	100 %	3.802	22 %	100 %	19,15 %	8,5 %
Total		198.521			16.960				

Tabla 6.8: Tabla performance modelo rf - segmentación clean - base de validación. Elaboración propia

La tabla 6.8 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 2,32 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 19,15 % de malos pagadores, además de un 8 % de malos pagadores en toda la población, y un KS con 5 puntos más que la regresión logística.

6.2. Modelos con AG

6.2.1. Regresión Logística

Segmento Dirty: BASE DE MODELAMIENTO

KS	**ROC**	**GINI**
62,4	86,9	73,7

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
669	999	2.313	10 %	10 %	652	4 %	4 %	28,1 %	28,2 %
570	669	2.313	10 %	20 %	870	5 %	9 %	37,6 %	32,9 %
377	570	2.312	10 %	30 %	1.111	6 %	15 %	48,1 %	38,0 %
185	377	2.313	10 %	40 %	1.537	9 %	24 %	66,5 %	45,1 %
119	185	2.313	10 %	50 %	1.969	11 %	36 %	85,1 %	53,1 %
79	119	2.313	10 %	60 %	2.106	12 %	48 %	91,1 %	59,4 %
58	79	2.313	10 %	70 %	2.210	13 %	61 %	95,6 %	64,6 %
40	58	2.312	10 %	80 %	2.258	13 %	74 %	97,7 %	68,7 %
21	40	2.313	10 %	90 %	2.258	13 %	87 %	97,6 %	71,9 %
1	21	2.313	10 %	100 %	2.285	13 %	100 %	98,8 %	74,6 %
Total		23.128			17.256				

Tabla 6.9: Tabla performance modelo logit - dirty - base de modelamiento - con AG. Elaboración propia

La tabla 6.9 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 28,10 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 98,80 % de malos pagadores, y un KS con 1 punto más que la regresión logística sin AG.

BASE DE VALIDACIÓN

KS	**ROC**	**GINI**
63,4	88,2	76,5

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
673	999	6.471	10 %	10 %	1.349	3 %	3 %	20,8 %	20,8 %
574	673	6.472	10 %	20 %	1.831	4 %	7 %	28,2 %	24,6 %
228	574	6.471	10 %	30 %	2.585	6 %	13 %	39,9 %	29,7 %
162	228	6.472	10 %	40 %	3.974	9 %	22 %	61,4 %	37,6 %
116	162	6.471	10 %	50 %	5.028	11 %	33 %	77,7 %	45,6 %
89	116	6.471	10 %	60 %	5.112	12 %	45 %	79,0 %	51,2 %
59	89	6.472	10 %	70 %	5.717	13 %	58 %	88,3 %	56,5 %
36	59	6.471	10 %	80 %	6.020	14 %	72 %	93,0 %	61,1 %
17	36	6.472	10 %	90 %	6.200	14 %	86 %	95,8 %	64,9 %
1	17	6.471	10 %	100 %	6.341	14 %	100 %	98,0 %	68,2 %
Total		64.714			44.157				

Tabla 6.10: Tabla performance modelo logit - segmentación dirty - base de validación
Elaboración propia

La tabla 6.10 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 20,8 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 98 % de malos pagadores.

Segmento Clean:

BASE DE MODELAMIENTO

KS	**ROC**	**GINI**
25,6	67,5	34,9

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
897	999	17.662	10 %	10 %	1.476	4 %	4 %	8,36 %	8,4 %
880	657	17.661	10 %	20 %	1.698	5 %	9 %	9,61 %	9,0 %
863	880	17.662	10 %	30 %	2.142	6 %	15 %	12,13 %	10,0 %
846	863	17.661	10 %	40 %	2.498	7 %	22 %	14,14 %	11,1 %
828	846	17.662	10 %	50 %	2.760	8 %	30 %	15,63 %	12,0 %
811	828	17.662	10 %	60 %	3.339	9 %	39 %	18,90 %	13,1 %
790	811	17.661	10 %	70 %	3.802	11 %	50 %	21,53 %	14,3 %
749	790	17.662	10 %	80 %	4.329	12 %	63 %	24,51 %	15,6 %
669	749	17.661	10 %	90 %	5.254	15 %	77 %	29,75 %	17,2 %
1	669	17.662	10 %	100 %	7.949	23 %	100 %	45,01 %	20,0 %
Total		176.616			35.247				

Tabla 6.11: Tabla performance modelo logit - segmentación clean - base de modelamiento - con AG.
Elaboración propia

La tabla 6.11 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 8,36 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 45,01 % de malos pagadores, y un KS con casi 1 punto mayor que el modelo sin AG.

BASE DE VALIDACIÓN

KS	**ROC**	**GINI**
26,3	67,6	35,3

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
901	999	19.852	10 %	10 %	784	5 %	5 %	3,95 %	3,9 %
883	901	19.852	10 %	20 %	882	5 %	10 %	4,44 %	4,2 %
866	883	19.852	10 %	30 %	986	6 %	16 %	4,97 %	4,5 %
849	866	19.852	10 %	40 %	1.074	6 %	22 %	5,41 %	4,7 %
831	849	19.852	10 %	50 %	1.374	8 %	30 %	6,92 %	5,1 %
813	831	19.852	10 %	60 %	1.594	9 %	40 %	8,03 %	5,6 %
791	813	19.852	10 %	70 %	1.826	11 %	51 %	9,20 %	6,1 %
750	791	19.852	10 %	80 %	2.091	12 %	63 %	10,53 %	6,7 %
665	750	19.852	10 %	90 %	2.534	15 %	78 %	12,76 %	7,4 %
1	665	19.852	10 %	100 %	3.687	22 %	100 %	18,57 %	8,5 %
Total		198.521			16.832				

Tabla 6.12: Tabla performance modelo logit - segmentación clean - base de validación - con AG. - Elaboración propia.

La tabla 6.12 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 3,95 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 18,57 % de malos pagadores, y un KS con 2 puntos mayor que el modelo sin AG.

6.2.2. Random Forest

Segmento Dirty:

BASE DE MODELAMIENTO

KS	ROC	GINI
65,4	88,9	77,9

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
689	999	2.313	10 %	10 %	541	3 %	3 %	23,39 %	23,4 %
594	689	2.313	10 %	20 %	785	5 %	8 %	33,94 %	28,7 %
398	594	2.312	10 %	30 %	1.088	6 %	14 %	47,06 %	34,8 %
217	398	2.313	10 %	40 %	1.625	9 %	23 %	70,26 %	43,7 %
121	217	2.313	10 %	50 %	2.005	12 %	35 %	86,68 %	52,3 %
62	121	2.313	10 %	60 %	2.132	12 %	47 %	92,17 %	58,9 %
35	62	2.313	10 %	70 %	2.240	13 %	60 %	96,84 %	64,3 %
21	35	2.312	10 %	80 %	2.264	13 %	73 %	97,92 %	68,5 %
11	21	2.313	10 %	90 %	2.275	13 %	87 %	98,36 %	71,8 %
1	11	2.313	10 %	100 %	2.301	13 %	100 %	99,48 %	74,6 %
Total		23.128			17.256				

Tabla 6.13: Tabla performance modelo rf - segmentación dirty - base de modelamiento - con AG.- Elaboración propia.

La tabla 6.13 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 23,39 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 99,48 % de malos pagadores, y un KS 0,5 puntos mayor que el modelo sin AG.

BASE DE VALIDACIÓN

KS	**ROC**	**GINI**
62,6	89,2	78,5

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
704	999	6.471	10 %	10 %	1.151	3 %	3 %	17,79 %	17,8 %
610	704	6.472	10 %	20 %	1.558	4 %	6 %	24,07 %	20,9 %
305	610	6.471	10 %	30 %	2.539	6 %	13 %	39,24 %	27,0 %
275	305	6.472	10 %	40 %	3.464	8 %	21 %	53,52 %	33,7 %
226	275	6.471	10 %	50 %	3.771	9 %	30 %	58,28 %	38,6 %
110	226	6.471	10 %	60 %	4.711	11 %	41 %	72,80 %	44,3 %
36	110	6.472	10 %	70 %	5.787	14 %	55 %	89,42 %	50,7 %
23	36	6.471	10 %	80 %	6.185	15 %	70 %	95,58 %	56,3 %
9	23	6.472	10 %	90 %	6.262	15 %	85 %	96,76 %	60,8 %
1	9	6.471	10 %	100 %	6.404	15 %	100 %	98,96 %	64,6 %
Total		64.714			41.832				

Tabla 6.14: Tabla performance modelo rf - segmentación dirty - base de validación - con AG.
Elaboración propia

La tabla 6.14 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 17,79 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 98,96 % de malos pagadores.

Segmento Clean:

BASE DE MODELAMIENTO

KS	ROC	GINI
34,7	73,6	47,2

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
914	999	17.662	10 %	10 %	562	2 %	2 %	3,18 %	3,2 %
891	914	17.661	10 %	20 %	1.098	3 %	5 %	6,22 %	4,7 %
867	891	17.662	10 %	30 %	1.612	5 %	9 %	9,13 %	6,2 %
844	867	17.661	10 %	40 %	2.208	6 %	16 %	12,50 %	7,8 %
817	844	17.662	10 %	50 %	2.700	8 %	23 %	15,29 %	9,3 %
792	817	17.662	10 %	60 %	3.176	9 %	32 %	17,98 %	10,7 %
760	792	17.661	10 %	70 %	3.969	11 %	43 %	22,47 %	12,4 %
719	760	17.662	10 %	80 %	4.767	14 %	57 %	26,99 %	14,2 %
662	719	17.661	10 %	90 %	6.076	17 %	74 %	34,40 %	16,5 %
1	662	17.662	10 %	100 %	9.079	26 %	100 %	51,40 %	20,0 %
Total		176.616			35.247				

Tabla 6.15: Tabla performance modelo rf - segmentación clean - base de modelamiento - con AG.
Elaboración propia

La tabla 6.15 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 3,18 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 51,40 % de malos pagadores, y un KS 2 puntos mayor que el modelo sin AG.

BASE DE VALIDACIÓN

KS	**ROC**	**GINI**
32,4	71,5	43,0

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
914	999	19.852	10 %	10 %	427	3 %	3 %	2,15 %	2,2 %
892	914	19.852	10 %	20 %	711	4 %	7 %	3,58 %	2,9 %
869	892	19.852	10 %	30 %	876	5 %	12 %	4,41 %	3,4 %
847	869	19.852	10 %	40 %	1.059	6 %	18 %	5,33 %	3,9 %
820	847	19.852	10 %	50 %	1.222	7 %	26 %	6,16 %	4,3 %
795	820	19.853	10 %	60 %	1.557	9 %	35 %	7,84 %	4,9 %
764	795	19.852	10 %	70 %	1.916	11 %	46 %	9,65 %	5,6 %
723	764	19.852	10 %	80 %	2.375	14 %	60 %	11,96 %	6,4 %
667	723	19.852	10 %	90 %	2.841	17 %	77 %	14,31 %	7,3 %
1	667	19.852	10 %	100 %	3.849	23 %	100 %	19,39 %	8,5 %
Total		198.521			16.833				

Tabla 6.16: Tabla performance modelo rf - segmentación clean - base de validación - con AG.
Elaboración propia

La tabla 6.16 pone en manifiesto que de el 10 % de la población con el score más alto se podría alcanzar un 2,15 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 19,39 % de malos pagadores, y un KS 2 puntos mayor que el modelo sin AG.

6.3. Alineación de Scores sin AG

6.3.1. Regresión Logística:

BASE DE MODELAMIENTO

KS	**ROC**	**GINI**
38,2	75,2	50,3

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
892	999	19.974	10 %	10 %	2.049	4 %	4 %	10,3 %	10,3 %
874	892	19.975	10 %	20 %	2.054	4 %	8 %	10,3 %	10,3 %
856	874	19.974	10 %	30 %	2.838	5 %	13 %	14,2 %	11,6 %
837	856	19.975	10 %	40 %	2.811	5 %	19 %	14,3 %	12,2 %
816	837	19.974	10 %	50 %	3.395	6 %	25 %	17 %	13,2 %
787	816	19.974	10 %	60 %	4.033	8 %	33 %	20,2 %	14,3 %
747	787	19.975	10 %	70 %	4.795	9 %	42 %	24 %	15,7 %
675	747	19.974	10 %	80 %	5.964	11 %	53 %	29,9 %	17,5 %
497	675	19.975	10 %	90 %	7.820	15 %	68 %	39,1 %	19,9 %
1	497	19.974	10 %	100 %	16.744	32 %	100 %	83,8 %	26,3 %
Total		199.744			52.503				

Tabla 6.17: modelo logit sin AG - base de modelamiento. - Elaboración propia.

La tabla 6.17 alinea los scores de los segmentos dirty y clean para obtener una sola tabla performance en la que se evidencia que de el 10 % de la población con el score más alto se podría alcanzar un 10,3 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 83,8 % de malos pagadores.

BASE DE VALIDACIÓN

KS	**ROC**	**GINI**
64,2	87,9	75,7

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
329	999	26.324	10 %	10 %	457	1 %	1 %	1,7 %	1,7 %
254	329	26.323	10 %	20 %	1.529	3 %	4 %	5,8 %	3,8 %
218	254	26.323	10 %	30 %	1.429	3 %	7 %	5,4 %	4,3 %
187	218	26.324	10 %	40 %	1.636	3 %	10 %	6,2 %	4,8 %
160	187	26.324	10 %	50 %	1.882	4 %	13 %	7,1 %	5,3 %
132	160	26.323	10 %	60 %	2.096	4 %	18 %	8 %	5,7 %
102	132	26.323	10 %	70 %	2.787	5 %	23 %	10,6 %	6,4 %
59	102	26.324	10 %	80 %	3.446	7 %	30 %	13,1 %	7,2 %
23	59	26.324	10 %	90 %	11.458	22 %	52 %	43,5 %	11,3 %
1	23	26.323	10 %	100 %	24.767	48 %	100 %	94,1 %	19,6 %
Total		263.235			51.487				

Tabla 6.18: modelo logit sin AG - base de validación. - Elaboración propia

La tabla 6.18 alinea los scores de los segmentos dirty y clean para obtener una sola tabla performance en la que se evidencia que de el 10 % de la población con el score más alto se podría alcanzar un 1,7 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 94,1 % de malos pagadores.

6.3.2. Random Forest:

BASE DE MODELAMIENTO

KS	ROC	GINI
43,4	79,9	59,8

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
916	999	19.974	10 %	10 %	607	1 %	1 %	3 %	3 %
882	916	19.975	10 %	20 %	1.422	3 %	4 %	7,1 %	5,1 %
863	882	19.974	10 %	30 %	1.963	4 %	8 %	9,8 %	6,7 %
838	863	19.975	10 %	40 %	3.103	6 %	14 %	15,5 %	8,9 %
802	838	19.974	10 %	50 %	3.445	7 %	20 %	17,2 %	10,6 %
766	802	19.974	10 %	60 %	4.278	8 %	28 %	21,4 %	12,4 %
728	766	19.975	10 %	70 %	5.135	10 %	38 %	25,7 %	14,3 %
677	728	19.974	10 %	80 %	6.370	12 %	50 %	31,9 %	16,5 %
534	677	19.975	10 %	90 %	8.777	17 %	67 %	43,9 %	19,5 %
1	534	19.974	10 %	100 %	17.403	33 %	100 %	87,1 %	26,3 %
Total		199.744			52.503				

Tabla 6.19: Tabla performance modelo rf sin AG - base de modelamiento. Elaboración propia

La tabla 6.19 alinea los scores de los segmentos dirty y clean para obtener una sola tabla performance en la que se evidencia que de el 10 % de la población con el score más alto se podría alcanzar un 3 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 87,1 % de malos pagadores.

BASE DE VALIDACIÓN

KS	**ROC**	**GINI**
62,4	88,0	76,0

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
863	999	26.324	10 %	10 %	751	1 %	1 %	2,9 %	2,9 %
823	863	26.323	10 %	20 %	1.279	2 %	4 %	4,9 %	3,9 %
799	823	26.323	10 %	30 %	1.553	3 %	6 %	5,9 %	4,5 %
776	799	26.324	10 %	40 %	1.700	3 %	9 %	6,5 %	5 %
748	776	26.324	10 %	50 %	2.320	4 %	13 %	8,8 %	5,8 %
714	748	26.323	10 %	60 %	2.949	5 %	19 %	11,2 %	6,7 %
674	714	26.323	10 %	70 %	3.738	7 %	25 %	14,2 %	7,8 %
554	674	26.324	10 %	80 %	5.069	9 %	34 %	19,3 %	9,2 %
23	554	26.324	10 %	90 %	12.243	22 %	56 %	46,5 %	13,3 %
1	23	26.323	10 %	100 %	24.882	44 %	100 %	94,5 %	21,5 %
Total		263.235			56.484				

Tabla 6.20: Tabla performance modelo rf sin AG - base de validación.
Elaboración propia

La tabla 6.20 alinea los scores de los segmentos dirty y clean para obtener una sola tabla performance en la que se evidencia que de el 10 % de la población con el score más alto se podría alcanzar un 2,9 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 94,5 % de malos pagadores.

6.4. Alineación de Scores con AG

6.4.1. Regresión Logística:

BASE DE MODELAMIENTO

KS	**ROC**	**GINI**
39,5	76,0	52,0

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
896	999	19974	10 %	10 %	1.745	3 %	3 %	8,7 %	8,7 %
876	896	19975	10 %	20 %	2.036	4 %	7 %	10,2 %	9,5 %
857	876	19974	10 %	30 %	2.491	5 %	12 %	12,5 %	10,5 %
837	857	19975	10 %	40 %	2.938	6 %	18 %	14,7 %	11,5 %
817	837	19974	10 %	50 %	3.408	6 %	24 %	17,1 %	12,6 %
795	817	19974	10 %	60 %	4.233	8 %	32 %	21,2 %	14,1 %
753	795	19975	10 %	70 %	4.623	9 %	41 %	23,1 %	15,4 %
660	753	19974	10 %	80 %	6.090	12 %	52 %	30,5 %	17,2 %
47	660	19975	10 %	90 %	8.637	16 %	69 %	43,2 %	20,1 %
1	47	19974	10 %	100 %	1.6302	31 %	100 %	81,6 %	26,3 %
Total		199.744			52.503				

Tabla 6.21: Tabla performance modelo logit con AG - base de modelamiento. Elaboración propia

La tabla 6.21 alinea los scores de los segmentos dirty y clean para obtener una sola tabla performance en la que se evidencia que de el 10 % de la población con el score más alto se podría alcanzar un 8,7 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 81,6 % de malos pagadores, y un KS con un punto más que el modelo sin AG.

BASE DE VALIDACIÓN

KS	**ROC**	**GINI**
64,9	88,4	76,9

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
896	999	26.324	10 %	10 %	1.077	2 %	2 %	4,1 %	4,1 %
873	896	26.323	10 %	20 %	1.216	2 %	4 %	4,6 %	4,4 %
850	873	26.323	10 %	30 %	1.417	2 %	6 %	5,4 %	4,7 %
826	850	26.324	10 %	40 %	1.838	3 %	9 %	7 %	5,3 %
800	826	26.324	10 %	50 %	2.307	4 %	13 %	8,8 %	6 %
751	800	26.323	10 %	60 %	2.617	4 %	17 %	9,9 %	6,6 %
606	751	26.323	10 %	70 %	3.497	6 %	23 %	13,3 %	7,6 %
67	606	26.324	10 %	80 %	5.730	10 %	33 %	21,8 %	9,4 %
23	67	26.324	10 %	90 %	16.703	28 %	61 %	63,5 %	15,4 %
1	23	26.323	10 %	100 %	23.606	39 %	100 %	89,7 %	22,8 %
Total		263.235			60.008				

Tabla 6.22: Tabla performance modelo logit con AG - base de validación.
Elaboración propia

La tabla 6.22 alinea los scores de los segmentos dirty y clean para obtener una sola tabla performance en la que se evidencia que de el 10 % de la población con el score más alto se podría alcanzar un 4,1 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 89,7 % de malos pagadores, y un KS con 0,7 puntos más que el modelo sin AG.

6.4.2. Random Forest:

BASE DE MODELAMIENTO

KS	ROC	GINI
45,3	81,0	62,0

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
912	999	19.974	10 %	10 %	535	1 %	1 %	2,7 %	2,7 %
886	912	19.975	10 %	20 %	1.274	2 %	3 %	6,4 %	4,5 %
860	886	19.974	10 %	30 %	1.891	4 %	7 %	9,5 %	6,2 %
831	860	19.975	10 %	40 %	2.796	5 %	12 %	14 %	8,1 %
802	831	19.974	10 %	50 %	3.262	6 %	19 %	16,3 %	9,8 %
769	802	19.974	10 %	60 %	4.273	8 %	27 %	21,4 %	11,7 %
728	769	19.975	10 %	70 %	5.173	10 %	37 %	25,9 %	13,7 %
678	728	19.974	10 %	80 %	6.528	12 %	49 %	32,7 %	16,1 %
555	678	19.975	10 %	90 %	9.265	18 %	67 %	46,4 %	19,5 %
1	555	19.974	10 %	100 %	17.506	33 %	100 %	87,6 %	26,3 %
Total		199.744			52.503				

Tabla 6.23: Tabla performance modelo rf con AG - base de modelamiento. Elaboración propia

La tabla 6.23 alinea los scores de los segmentos dirty y clean para obtener una sola tabla performance en la que se evidencia que de el 10 % de la población con el score más alto se podría alcanzar un 2,7 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 87,6 % de malos pagadores, y un KS con 2 puntos más que el modelo sin AG.

BASE DE VALIDACIÓN

KS	**ROC**	**GINI**
63,7	89,0	78,1

Score		Total			Malo			Malo Rate	
Min	Max	Int	Int %	Cum %	Int	Int %	Cum %	Int	Cum
908	999	26.324	10 %	10 %	636	1 %	1 %	2,4 %	2,4 %
878	908	26.323	10 %	20 %	1.037	2 %	3 %	3,9 %	3,2 %
848	878	26.323	10 %	30 %	1.372	2 %	5 %	5,2 %	3,9 %
813	848	26.324	10 %	40 %	1.688	3 %	8 %	6,4 %	4,5 %
778	813	26.324	10 %	50 %	2.266	4 %	12 %	8,6 %	5,3 %
726	778	26.323	10 %	60 %	3.033	5 %	18 %	11,5 %	6,4 %
645	726	26.323	10 %	70 %	3.844	7 %	24 %	14,6 %	7,5 %
476	645	26.324	10 %	80 %	5.272	9 %	34 %	20 %	9,1 %
23	476	26.324	10 %	90 %	13.285	23 %	57 %	50,5 %	13,7 %
1	23	26.323	10 %	100 %	24.719	43 %	100 %	93,9 %	21,7 %
Total		263.235			57.152				

Tabla 6.24: Tabla performance modelo rf con AG - base de validación.
Elaboración propia

La tabla 6.24 alinea los scores de los segmentos dirty y clean para obtener una sola tabla performance en la que se evidencia que de el 10 % de la población con el score más alto se podría alcanzar un 2,4 % de malos pagadores, por el contrario en el 10 % de la población con el score más bajo se podría alcanzar un 93,9 % de malos pagadores, y un KS con 1 punto más que el modelo sin AG.

Capítulo 7

Conclusiones y Recomendaciones

7.1. Conclusiones

1. Los modelos de Credit Scoring en los que se aplicó Algoritmos Genéticos en la selección de variables explicativas para generarlos presentaron mejores medidas de calidad de discriminación que los modelos en los que no se aplicaron Algoritmos Genéticos, esto se evidencia en la tabla 5.3.

2. La implementación de algoritmos genéticos además de llegar a un buen modelo Credit Scoring reduce mucho el tiempo de búsqueda de una combinación de variables para un modelo, y esto se debe sobre todo a que el algoritmo está automatizado en lenguaje R, lo que evita un manejo manual al seleccionar las variables para el modelo.

3. Se constató que los algoritmos genéticos son métodos heurísticos que se fundamentan en procedimientos aleatorios que buscan encontrar una solución óptima o al menos cercana a ella, en este estudio mejorando los dos modelos (Regresión Logística y Random Forest).

4. En cuanto a los modelos entrenados para las segmentaciones dirty y clean, como se puede ver en la tabla comparativa 5.2 los modelos con algoritmos genéticos presentaron una mejor calidad de discriminación que los modelos sin algoritmos genéticos a excepción de 3 modelos (pintados de amarillo) donde se presenta una menor calidad de discriminación.

5. Entrenar un modelo para la segmentación clean es más complicado que para la segmentación dirty, y una de las causas es que para la segmentación clean el porcentaje de representatividad de malos es bajo, lo que obstaculiza en encontrar un buen modelo y se requiere realizar técnicas de remuestreo para esta segmentación.

6. Los modelos Random Forest generalmente tienen un mayor poder predictivo que los modelos de Regresión Logística, como se puede ver en la tabla 5.2.

7. Las tablas performance de los modelos son herramientas de gran ayuda para medir la calidad de discriminación de un modelo de credit scoring.

7.2. Recomendaciones

1. Se recomienda tener una extensa información de los clientes, tanto variables cuantitativas como cualitativas y tomar correctamente los valores correspondientes, de esta manera se obtendrá un modelo con un mayor poder predictivo.

2. Usar el software libre R, ya que es muy útil para la construcción de modelos Credit Scoring de una manera más automatizada a diferencia de otros softwares, además de que está en constante actualización con la inclusión de diferentes paquetes creados por sus usuarios.

3. Para entrenar el modelo Credit Scoring es recomendable generar variables de ratios que comparen el desempeño del cliente durante la ventana de comportamiento estas variables ayudan a tener una mejor calidad predictiva en el modelo, por ejemplo el ratio saldo mora 3 últimos meses vs los últimos 6 meses; si este ratio es alto indicaría un endeudamiento del cliente en los últimos 3 meses con relación a los últimos 6 meses, diferencia de tener solo la variable de saldo mora en los últimos 3 meses que no explicaría la evolución de la mora del cliente.

4. Si se realiza una segmentación como la realizada en el presente trabajo se recomienda realizar un proceso de remuestreo para la segmentación clean para que de esta manera suba el porcentaje de malos en la muestra.

5. Se recomienda realizar la Alineación de Scores para la obtención de un solo modelo que pueda discriminar a la población y no tener que clasificar a un individuo en clean o dirty para luego aplicar el modelo.

Bibliografía

[1] John McCall, 2005, *Genetic algorithms for modelling and optimisation*, Journal of Computational and Applied Mathematics, Volume 184, Issue 1, pág 205-210

[2] Aggarwal, Nikita, 2020, *THE NORMS OF ALGORITHMIC CREDIT SCORING*, University of Oxford, pág 5-10.

[3] IBM SPSS, 2011, *IBM SPSS Statics 20 Algorithms*, SPSS Inc. Chicago 531, pág 205-212

[4] Joaquín Amat, 2018, *Algoritmo genético para selección de predictores*, RPubs, pág 1.

[5] Ana Gonzáles, 2015, *Selección de variables:una revisión de métodos existentes*, Universidade da Coruña, Facultad Informática, pág 14-19.

[6] Narváez García Andrés Mauricio, 2019, *Variables determinantes de la probabilidad de incumplimiento de los créditos comerciales en una institución financiera del Ecuador, aproximación bajo el modelo de Regresión Logística Binaria.*, Universidad Técnica de Ambato, Facultad de contabilidad y auditoría, pág 1-5.

[7] Reditum.ec, 2020, *Análisis Mensual de la Morosidad de los Bancos en el Ecuador / Junio 2020*, Reditum Ecuador, pág 1,2.

[8] Asobanca, 2020, *Bancos - Indicadores, en porcentajes.*, DataLab, pág 1.

[9] Erik Brynjolfsson and Andrew McAfee, 2011, *The Big Data Boom Is the Innovation Story of Our Time.*, The Atlantic, pág 1.

[10] Juan Sebastián Herrera, Miguel Jara Manríquez, 2005, *Los Bancos y las Nuevas Tecnologías.*, Universidad de Chile, Facultad de Ciencias Económicas y Administrativas, pág 3-5.

[11] Pablo Estévez Valencia, 1997, *Optimización mediante algoritmos genéticos.*, Anales del Instituto de Ingenieros de Chile,pág 83-92.

[12] Steven Finlay, 2008, *Are we modelling the right thing? The impact of incorrect problem specification in credit scoring*, Departament of Management Science, Lancaster University Management School, United Kindom,pág 2.

[13] Beltrán Pascual, Muñoz Alamillos, Muñoz Martiez, 2012, *Un nuevo clasificador de préstamos bancarios a través de la minería de datos*, SEIO, pág 1.

[14] Micha David, 2016, *Credit Scoring Using Genetic Programming*, IMS, Lisboa, pág 2,3.

[15] Xunbo Shuai, Xiangguang Zhou, 2011, *A Genetic Algorithm Based on Combination Operators*, Research Institute of Petroleum Exploration and Development-Langfang, Langfang, China, pág 346.

[16] Johanna Orellana Alvear, 2018, *Arboles de decision y Random Forest*, RPubs, pág 1.

[17] Bart Baesens, 2018, *Dealing with imbalanced datasets* , KU Leuven, pág 1-64.

[18] Vega José, 2010, *regresión pls y pca como solución al ´ problema de multicolinealidad en regresión múltiple*, CIMPA – UCR pág 10.

[19] Carolin Strobl, Anne-Laure Boulesteix,Thomas Kneib, 2008, *Conditional variable importance for random forests*, BMC Bioinformatics, pág 1.

[20] Capelo Julia, 2012, *Modelo de aprobación de tarjetas de crédito en la población ecuatoriana bancarizada a través de una metodología analítica*, Escuela Politécnica Nacional, pág 1-98.

[21] Pérez Alex, 2014, *Modelo de activación de tarjetas de crédito en el mercado crediticio ecuatoriano a través de una metodología analítica y automatizada en R*, Escuela Politécnica Nacional, pág 1-95.

[22] Leo Breiman, 1996, *Bagging predictors*, University of California, pág 123-140.

[23] Vargas Gissela, 2021, *Modelo de clasificación binaria para una población bancarizada empleando metodología de ensamble*, Escuela Politécnica Nacional, pág 1-133.

[24] Leo Breiman, 2001, *RANDOM FORESTS*, University of California, Berkeley pág 1.

[25] H2O.AI, 2016, *Automatic Machine Learning*,RPubs, pág 1.

Anexos

Anexo 1:

Tablas Roll Rate:

RANGO VENCIDO	VARIABLE	
	NO AVANZA	AVANZA
Sin vencido	92,58 %	7,42 %
De 1 a 30 días	75,41 %	24,59 %
De 31 a 60 días	53,90 %	46,10 %
De 61 a 90 días	42,64 %	57,36 %
De 91 a 120 días	32,23 %	67,77 %
De 121 a 150 días	25,45 %	74,55 %
Más de 150 días	17,51 %	82,49 %

Tabla 7.1: Tabla Roll-Rate m1 vs m2.
Elaboración propia

RANGO VENCIDO	VARIABLE	
	NO AVANZA	AVANZA
Sin vencido	92,82 %	7,18 %
De 1 a 30 días	75,80 %	24,20 %
De 31 a 60 días	55,88 %	44,12 %
De 61 a 90 días	42,00 %	58,00 %
De 91 a 120 días	32,95 %	67,05 %
De 121 a 150 días	27,77 %	72,23 %
Más de 150 días	16,57 %	83,43 %

Tabla 7.2: Tabla Roll-Rate m2 vs m3.
Elaboración propia

RANGO VENCIDO	VARIABLE	
	NO AVANZA	AVANZA
Sin vencido	93,27 %	6,73 %
De 1 a 30 días	75,89 %	24,11 %
De 31 a 60 días	56,43 %	43,57 %
De 61 a 90 días	44,52 %	55,48 %
De 91 a 120 días	32,47 %	67,53 %
De 121 a 150 días	28,41 %	71,59 %
Más de 150 días	16,18 %	83,82 %

Tabla 7.3: Tabla Roll-Rate m3 vs m4.
Elaboración propia

RANGO VENCIDO	VARIABLE	
	NO AVANZA	AVANZA
Sin vencido	93,36 %	6,64 %
De 1 a 30 días	76,09 %	23,91 %
De 31 a 60 días	56,72 %	43,28 %
De 61 a 90 días	44,17 %	55,83 %
De 91 a 120 días	33,72 %	66,28 %
De 121 a 150 días	27,82 %	72,18 %
Más de 150 días	15,69 %	84,31 %

Tabla 7.4: Tabla Roll-Rate m4 vs m5.
Elaboración propia

RANGO VENCIDO	VARIABLE	
	NO AVANZA	AVANZA
Sin vencido	93,73 %	6,27 %
De 1 a 30 días	77,58 %	22,42 %
De 31 a 60 días	59,37 %	40,63 %
De 61 a 90 días	44,47 %	55,53 %
De 91 a 120 días	32,98 %	67,02 %
De 121 a 150 días	29,91 %	70,09 %
Más de 150 días	15,11 %	84,89 %

Tabla 7.5: Tabla Roll-Rate m5 vs m6.
Elaboración propia

RANGO VENCIDO	VARIABLE	
	NO AVANZA	AVANZA
Sin vencido	94,13 %	5,87 %
De 1 a 30 días	76,65 %	23,35 %
De 31 a 60 días	59,11 %	40,89 %
De 61 a 90 días	47,11 %	52,89 %
De 91 a 120 días	34,01 %	65,99 %
De 121 a 150 días	29,18 %	70,82 %
Más de 150 días	14,60 %	85,40 %

Tabla 7.6: Tabla Roll-Rate m6 vs m7.
Elaboración propia

RANGO VENCIDO	VARIABLE	
	NO AVANZA	AVANZA
Sin vencido	94,40 %	5,60 %
De 1 a 30 días	76,89 %	23,11 %
De 31 a 60 días	58,91 %	41,09 %
De 61 a 90 días	46,95 %	53,05 %
De 91 a 120 días	34,53 %	65,47 %
De 121 a 150 días	27,67 %	72,33 %
Más de 150 días	13,29 %	86,71 %

Tabla 7.7: Tabla Roll-Rate m7 vs m8.
Elaboración propia

RANGO VENCIDO	VARIABLE	
	NO AVANZA	AVANZA
Sin vencido	94,70 %	5,30 %
De 1 a 30 días	76,97 %	23,03 %
De 31 a 60 días	60,38 %	39,62 %
De 61 a 90 días	47,26 %	52,74 %
De 91 a 120 días	35,31 %	64,69 %
De 121 a 150 días	28,94 %	71,06 %
Más de 150 días	11,35 %	88,65 %

Tabla 7.8: Tabla Roll-Rate m8 vs m9.
Elaboración propia

RANGO VENCIDO	VARIABLE	
	NO AVANZA	AVANZA
Sin vencido	95,05 %	4,95 %
De 1 a 30 días	77,28 %	22,72 %
De 31 a 60 días	60,23 %	39,77 %
De 61 a 90 días	48,51 %	51,49 %
De 91 a 120 días	35,53 %	64,47 %
De 121 a 150 días	28,79 %	71,21 %
Más de 150 días	9,80 %	90,20 %

Tabla 7.9: Tabla Roll-Rate m9 vs m10.
Elaboración propia

RANGO VENCIDO	VARIABLE	
	NO AVANZA	AVANZA
Sin vencido	95,42 %	4,58 %
De 1 a 30 días	75,75 %	24,25 %
De 31 a 60 días	60,50 %	39,50 %
De 61 a 90 días	48,81 %	51,19 %
De 91 a 120 días	35,97 %	64,03 %
De 121 a 150 días	31,74 %	68,26 %
Más de 150 días	7,99 %	92,01 %

Tabla 7.10: Tabla Roll-Rate m10 vs m11.
Elaboración propia

RANGO VENCIDO	VARIABLE	
	NO AVANZA	AVANZA
Sin vencido	94,43 %	5,57 %
De 1 a 30 días	86,04 %	13,96 %
De 31 a 60 días	61,88 %	38,12 %
De 61 a 90 días	48,53 %	51,47 %
De 91 a 120 días	35,25 %	64,75 %
De 121 a 150 días	33,63 %	66,37 %
Más de 150 días	4,27 %	95,73 %

Tabla 7.11: Tabla Roll-Rate m11 vs m12.
Elaboración propia

Anexo 2:

Código de remuestreo en R:

```
library(ROSE)

remuestreo <- function(data){
    # Porcentaje de malos en el rebalanceo: p=20%
    # Incremento de la data: N=115%
    set.seed(12345)
    res <- ovun.sample(VarDepF ~ ., data=data, N=1.2*nrow(data),
                       p=0.2, seed=1, method="both")$data
    return(res)
}

rmod <- remuestreo(mod)
```

Anexo 3:

Código de implementación de algoritmos genéticos en R:

```
##########              Algoritmos Geneticos              ##########

crear_poblacion <- function(n_poblacion, n_variables, n_max = NULL, n_min
    = NULL, verbose = TRUE){
    # ARGUMENTOS
    # ================================
    # n_poblacion: "integer"
    #   nÃºmero total de individuos de la poblaciÃ³n.
    # n_variables: "integer"
    #   longitud de los individuos.
    # n_max: "integer"
    #   nÃºmero mÃ¡ximo de TRUEs que puede contener un individuo.
    # n_min: "integer"
    #   nÃºmero mÃnimo de TRUEs que puede contener un individuo.
    # verbose: "logical"
    #   mostrar informaciÃ³n del proceso por pantalla.
```

```
# RETORNO
# ========================================
# Matriz de tamaÃ±o n_poblacion x n_variables que representa una
    poblaciÃ³n.

# Comprobaciones.
if (isTRUE(n_max > n_variables)) {
    stop("n_max no puede ser mayor que n_variables.")
}
# Si no se especifica n_max, el nÃºmero mÃ¡ximo de predictores (TRUEs
    ) que puede
# contener un individuo es igual al nÃºmero total de variables
    disponibles.
if (is.null(n_max)) {
    n_max <- n_variables
}
# Si no se especifica n_min, el nÃºmero mÃnimo de predictores (TRUEs
    ) que puede
# contener un individuo es 1.
if (is.null(n_min)) {
    n_min <- 1
}
# Matriz donde almacenar los individuos generados.
poblacion <- matrix(data = NA, nrow = n_poblacion, ncol = n_variables
    )
# Bucle para crear cada individuo.
for (i in 1:n_poblacion) {
    # Se selecciona (con igual probabilidad) el nÃºmero de valores
       TRUE que puede
    # tener el individuo, dentro del rango acotado por n_min y n_max.
    n_true <- sample(x = n_min:n_max, size = 1)

    # Se crea un vector con todo FALSE que representa el individuo.
    individuo <- rep(FALSE, times = n_variables)

    # Se sustituyen n_true posiciones aleatorias por valores TRUE.
    individuo[sample(x = 1:n_variables, size = n_true)] <- TRUE

    # Se aÃ±ade el nuevo individuo a la poblaciÃ³n.
    poblacion[i, ] <- individuo
}
if (verbose) {
    print("PoblaciÃ³n inicial creada")
```

```
        print("-------------------------")
        print(paste("NÃºmero de individuos =", n_poblacion))
        print(paste("NÃºmero de predictores mÃnimo por individuo =", n_
            min))
        print(paste("NÃºmero de predictores mÃ¡ximo por individuo =", n_
            max))
        cat("\n")
        print(poblacion)
        cat("\n")
    }
    return(poblacion)
}

calcular_fitness_individuo_glm <- function(x, y, cv, metrica = NULL,
    nivel_referencia = NULL, seed = 123, verbose = TRUE, ...){
    # ARGUMENTOS
    # ============================================
    # x: "matrix" o "data.frame"
    #   matriz de predictores.
    #
    # y: "vector"
    #   variable respuesta.
    #
    # cv: "integer"
    #   nÃºmero de particiones de validaciÃ³n cruzada.
    #
    # seed: "integer"
    #   semilla para garantizar reproducibilidad en el proceso de CV.
    #
    # metrica: ("f1", "kappa", "accuracy")
    #   mÃ©trica utilizada para calcular el fitness. Por defecto es el -
        MSE para
    #   regresiÃ³n y Accuracy para clasificaciÃ³n. En el caso de
        clasificaciÃ³n
    #   binaria, se acepta tambiÃ©n f1 y kappa.
    #
    # nivel_referencia: "character"
    #   valor de la variable respuesta considerado como referencia.
        Necesario
    #   cuando la mÃ©trica es f1 o kappa.
    #
    # verbose: "logical"
    #   mostrar informaciÃ³n del proceso por pantalla.
```

```
# RETORNO
# ============================================
# fitness ("numeric") del individuo obtenido por validaciÃ³n cruzada
    empleando
# regresiÃ³n logÃstica como modelo.

library(MLmetrics)
library(caret)

# Comprobaciones
if (!is.character(y) & !is.factor(y)) {
    stop(
        paste(
            "El modelo glm solo puede emplearse para problemas de
                clasificaciÃ³n ,",
            "la variable respuesta debe ser character o factor."
        )
    )
}

if (is.character(y)) {
    y <- as.factor(y)
}

if (length(levels(y)) != 2) {
    stop(
        paste(
            "El modelo glm solo puede emplearse para problemas de
                clasificaciÃ³n binaria ,",
            "la variable respuesta debe tener dos niveles."
        )
    )
}

if(!metrica %in% c("accuracy", "f1", "kappa")) {
    stop(
        "Las Ãºnicas mÃ©tricas permitidas con el modelo glm son
            accuracy, f1 y kappa"
    )
}

# ReparticiÃ³n de las observaciones para la validaciÃ³n cruzada (CV).
set.seed(seed)
indices_cv <- caret::createFolds(y = y, k = cv, list = FALSE)
```

```
# Vector para almacenar el fitness de cada iteraciÃ³n CV.
fitness_cv <- rep(NA, times = cv)

for (i in 1:cv) {
    datos_modelo <- as.data.frame(cbind(x[indices_cv != i, , drop =
        FALSE],
                                        y = y[indices_cv != i]))
    modelo <- glm(
        formula = y ~ .,
        family = "binomial",
        data = datos_modelo
    )

    predicciones <- predict(
        modelo,
        newdata = as.data.frame(
            x[indices_cv == i, , drop = FALSE]
        ),
        type = "response"
    )
    predicciones <- unname(predicciones)
    predicciones <- ifelse(predicciones < 0.5, levels(y)[1], levels(y
        )[2])
    predicciones <- factor(x = predicciones, levels = levels(y))

    if (is.null(metrica) || metrica == "accuracy") {
        # Si y es factor (clasificaciÃ³n), el fitness por defecto es
            el accuracy.
        metrica <- "accuracy"
        accuracy <- MLmetrics::Accuracy(
            y_pred = predicciones,
            y_true = y[indices_cv == i]
        )
        fitness_cv[i] <- accuracy
    } else if (metrica == "kappa") {
        if (is.null(nivel_referencia)) {
            stop("Para la mÃ©trica kappa es necesario indicar el
                nivel de referencia.")
        } else {
            mat_confusion <- caret::confusionMatrix(
                table(predicciones,
                      y[indices_cv == i]),
                positive = nivel_referencia
```

```
                )
                kappa <- mat_confusion$overall["Kappa"]
                fitness_cv[i] <- kappa
            }
        } else if (metrica == "f1") {
            if (is.null(nivel_referencia)) {
                stop("Para la mÃ©trica f1 es necesario indicar el nivel
                    de referencia.")
            } else if (sum(predicciones == nivel_referencia) < 1) {
                print(paste(
                    "NingÃºn valor predicho ha sido el nivel de
                        referencia,",
                    "por lo tanto, el valor de precisiÃ³n no puede ser
                        calculado",
                    "y tampoco la mÃ©trica F1."
                ))
            } else {
                f1 <- MLmetrics::F1_Score(
                    y_true = y[indices_cv == i],
                    y_pred = predicciones,
                    positive = nivel_referencia
                )
                fitness_cv[i] <- f1
            }
        }

        if (verbose) {
            if (is.na(fitness_cv[i])) {
                print(paste("En la particion", i, "de CV, fitness del
                    individuo no",
                            "ha podido ser calculado."
                ))
                print("Valores reales:")
                print(y[indices_cv == i])
                print("Valores predichos:")
                print(predicciones)
            }
        }
    }

    if (any(is.na(fitness_cv))) {
        warning(paste(
            "En una o mÃ¡s particiones de CV, el fitness del individuo no
                ",
```

```
            "ha podido ser calculado."
        ))
    }

    if (is.na(mean(fitness_cv, na.rm = TRUE))) {
        stop("El fitness del individuo no ha podido ser calculado.")
    }

    if (verbose) {
        print(paste(
            "El fitness calculado por CV =", cv, "empleando la mÃ©trica",
            metrica, "es de:", mean(fitness_cv, na.rm = TRUE)
        ))
        cat("\n")
    }
    return(mean(fitness_cv, na.rm = TRUE))
}
calcular_fitness_individuo_rf <- function(x, y, cv, metrica = NULL, nivel
   _referencia = NULL, n_tree = 50, seed = 123, verbose = TRUE,
                                          ...) {
    # ARGUMENTOS
    # ==========================
    # x: "matrix" o "data.frame"
    #   matriz de predictores.
    #
    # y: "vector"
    #   variable respuesta.
    #
    # cv: "integer"
    #   nÃºmero de particiones de validaciÃ³n cruzada.
    #
    # seed: "integer"
    #   semilla para garantizar reproducibilidad en el proceso de CV.
    #
    # metrica: ("-mse", "f1", "kappa", "accuracy")
    #   mÃ©trica utilizada para calcular el fitness. Por defecto es el -
       MSE para
    #   regresiÃ³n y Accuracy para clasificaciÃ³n. En el caso de
       clasificaciÃ³n
    #   binaria, se acepta tambiÃ©n f1 y kappa.
    #
    # nivel_referencia: "character"
    #   valor de la variable respuesta considerado como referencia.
       Necesario
```

```
#   cuando la mÃ©trica es f1 o kappa.
#
# ntree: "integer"
#   nÃºmero de Ã¡rboles del modelo randomforest.
#
# verbose: "logical"
#   mostrar informaciÃ³n del proceso por pantalla.

# RETORNO
# ==========================================
# fitness ("numeric") del individuo obtenido por validaciÃ³n cruzada
    empleando
# random forest como modelo.

library(MLmetrics)
library(caret)

if(!is.null(metrica) && metrica != "-mse" && is.numeric(y)) {
    warning(
        paste(
            "La mÃ©trica", metrica, "no es valida para problemas de
                regresiÃ³n.",
            "Se emplea el -mse en su lugar."
        )
    )
}

# ReparticiÃ³n de las observaciones para la validaciÃ³n cruzada (CV).
set.seed(seed)
indices_cv <- caret::createFolds(y = y, k = cv, list = FALSE)

# Vector para almacenar el fitness de cada iteraciÃ³n CV.
fitness_cv <- rep(NA, times = cv)

for (i in 1:cv) {
    set.seed(seed)
    modelo <- randomForest::randomForest(
        x = x[indices_cv != i, , drop = FALSE],
        y = y[indices_cv != i],
        ntree = n_tree
    )
    predicciones <- predict(modelo, newdata = x[indices_cv == i, ,
        drop = FALSE])
```

```
        if (is.numeric(y)) {
            # Si y es numÃ©rico (regresiÃ³n), el fitness es igual al -MSE
                .
            metrica  <- "-mse"
            residuos <- predicciones - y[indices_cv == i]
            fitness_cv[i] <- -(mean(residuos^2))
        } else {
            if (is.null(metrica) || metrica == "accuracy") {
                # Si y es factor (clasificaciÃ³n), el fitness por defecto
                    es el accuracy.
                metrica <- "accuracy"
                accuracy <- MLmetrics::Accuracy(
                    y_pred = predicciones,
                    y_true = y[indices_cv == i]
                )
                fitness_cv[i] <- accuracy
            } else if (metrica == "kappa") {
                if (is.null(nivel_referencia)) {
                    stop("Para la mÃ©trica kappa es necesario indicar el
                        nivel de referencia.")
                } else {
                    mat_confusion <- caret::confusionMatrix(
                        table(predicciones,
                              y[indices_cv == i]),
                        positive = nivel_referencia
                    )
                    kappa <- mat_confusion$overall["Kappa"]
                    fitness_cv[i] <- kappa
                }
            } else if (metrica == "f1") {
                if (is.null(nivel_referencia)) {
                    stop("Para la mÃ©trica f1 es necesario indicar el
                        nivel de referencia.")
                } else if (sum(predicciones == nivel_referencia) < 1) {
                    print(paste(
                        "NingÃºn valor predicho ha sido el nivel de
                            referencia,",
                        "por lo tanto, el valor de precisiÃ³n no puede
                            ser calculado",
                        "y tampoco la mÃ©trica F1."
                    ))
                } else {
                    f1 <- MLmetrics::F1_Score(
                        y_true = y[indices_cv == i],
```

```
                        y_pred = predicciones,
                        positive = nivel_referencia
                    )
                    fitness_cv[i] <- f1
                }
            }
        }

        if (verbose) {
            if (is.na(fitness_cv[i])) {
                print(paste("En la particion", i, "de CV, fitness del
                    individuo no",
                            "ha podido ser calculado."
                ))
                print("Valores reales:")
                print(y[indices_cv == i])
                print("Valores predichos:")
                print(predicciones)
            }
        }
    }

    if (any(is.na(fitness_cv))) {
        warning(paste(
            "En una o mÃ¡s particiones de CV, el fitness del individuo no
                ",
            "ha podido ser calculado."
        ))
    }

    if (is.na(mean(fitness_cv, na.rm = TRUE))) {
        stop("El fitness del individuo no ha podido ser calculado.")
    }

    if (verbose) {
        print(paste(
            "El fitness calculado por CV =", cv, "empleando la mÃ©trica",
            metrica, "es de:", mean(fitness_cv, na.rm = TRUE)
        ))
        cat("\n")
    }
    return(mean(fitness_cv, na.rm = TRUE))
}
calcular_fitness_poblacion <- function(poblacion, x, y, cv, seed = 123,
```

```
modelo = "glm", metrica = NULL, nivel_referencia = NULL,
                                        verbose = TRUE){
# ARGUMENTOS
# ==================================
# poblacion: "matrix"
#   matriz boleana que representa la poblaciÃ³n de individuos.
#
# x: "matrix" o "data.frame"
#   matriz de predictores.
#
# y: "vector"
#   variable respuesta.
#
# cv: "integer"
#   nÂºmero de particiones de validaciÃ³n cruzada.
#
# seed: "integer"
#   semilla para garantizar reproducibilidad en el proceso de CV.
#
# modelo: ("lm", "glm", "rf").
#   tipo de modelo empleado para calcular el fitness.
#
# metrica: ("-mse", "f1", "kappa", "accuracy").
#   mÃ©trica utilizada para calcular el fitness. Por defecto es el -
    MSE para
#   regresiÃ³n y Accuracy para clasificaciÃ³n. En el caso de
    clasificaciÃ³n
#   binaria, se acepta tambiÃ©n f1 y kappa.
#
# nivel_referencia: "character"
#   valor de la variable respuesta considerado como referencia.
    Necesario
#   cuando la mÃ©trica es f1 o kappa.
#
# verbose: "logical"
#   mostrar informaciÃ³n del proceso por pantalla.


# RETORNO
# ==================================
# Vector ("numeric") con el fitness de todos los individuos de la
    poblaciÃ³n,
# obtenido por validaciÃ³n cruzada. El orden de los valores se
    corresponde con
```

```
    # el orden de las filas de la matriz poblaciÃ³n.

    # Vector donde almacenar el fitness de cada individuo.
    fitness_poblacion <- rep(NA, times = nrow(poblacion))

    # Tipo de modelo utilizado para calcular el fitness.
    if(modelo == "lm"){
        calcular_fitness_individuo <- calcular_fitness_individuo_lm
    }else if(modelo == "rf"){
        calcular_fitness_individuo <- calcular_fitness_individuo_rf
    }else if(modelo == "glm"){
        calcular_fitness_individuo <- calcular_fitness_individuo_glm
    }else{
        stop(paste(
            "El modelo empleado para calcular el fitness debe ser",
            "lm (linear model)",
            "glm (logistic regresion)",
            "o rf (randomforest)."
        ))
    }

    if(is.null(metrica)){
        if(is.numeric(y)){
            metrica <- "-mse"
        }else{
            metrica <- "accuracy"
        }
    }

    for (i in 1:nrow(poblacion)) {
        individuo <- poblacion[i, ]

        if (verbose) {
            print(paste("Individuo", i, ":", paste(individuo, collapse =
                " ")))
        }

        fitness_individuo <- calcular_fitness_individuo(
            x  = x[, individuo, drop = FALSE],
            y  = y,
            cv = cv,
            metrica = metrica,
            nivel_referencia = nivel_referencia,
            seed = seed,
```

```
            verbose = verbose
        )
        fitness_poblacion[i] <- fitness_individuo
    }

    if(verbose){
        print("Fitness de la poblaciÃ³n calculado")
        print("--------------------------------")
        print(paste("Modelo empleado para el cÃ¡lculo del fitness:",
            modelo))
        print(paste("MÃ©trica empleado para el cÃ¡lculo del fitness:",
            metrica))
        cat("\n")
    }

    return(fitness_poblacion)
}
calcular_fitness_poblacion_paral <- function(poblacion, x, y, cv, seed =
    123, modelo = "rf", metrica = NULL, nivel_referencia = NULL,
                                                verbose = TRUE) {
    # ARGUMENTOS
    # ==========================
    # poblacion: "matrix"
    #   matriz boleana que representa la poblaciÃ³n de individuos.
    #
    # x: "matrix" o "data.frame"
    #   matriz de predictores.
    #
    # y: "vector"
    #   variable respuesta.
    #
    # cv: "integer"
    #   nÃºmero de particiones de validaciÃ³n cruzada.
    #
    # seed: "integer"
    #   semilla para garantizar reproducibilidad en el proceso de CV.
    #
    # modelo: ("lm", "glm", "rf").
    #   tipo de modelo empleado para calcular el fitness.
    #
    # metrica: ("-mse", "f1", "kappa", "accuracy").
    #   mÃ©trica utilizada para calcular el fitness. Por defecto es el -
        MSE para
    #   regresiÃ³n y Accuracy para clasificaciÃ³n. En el caso de
```

```
    clasificaciÃ³n
#   binaria, se acepta tambiÃ©n f1 y kappa.
#
# nivel_referencia: "character"
#   valor de la variable respuesta considerado como referencia.
    Necesario
#   cuando la mÃ©trica es f1 o kappa.
#
# verbose: "logical"
#   mostrar informaciÃ³n del proceso por pantalla.

# RETORNO
# ================================
# Vector ("numeric") con el fitness de todos los individuos de la
    poblaciÃ³n,
# obtenido por validaciÃ³n cruzada. El orden de los valores se
    corresponde con
# el orden de las filas de la matriz poblaciÃ³n.

# Paquetes necesarios para paralelizar.
library(foreach)
library(doParallel)

# Tipo de modelo utilizado para calcular el fitness.
if (modelo == "lm") {
    calcular_fitness_individuo <- calcular_fitness_individuo_lm
} else if (modelo == "rf") {
    calcular_fitness_individuo <- calcular_fitness_individuo_rf
} else if (modelo == "glm") {
    calcular_fitness_individuo <- calcular_fitness_individuo_glm
} else {
    stop(paste(
        "El modelo empleado para calcular el fitness debe ser",
        "lm (linear model)",
        "glm (logistic regresion)",
        "o rf (randomforest)."
    ))
}

if(is.null(metrica)) {
    if(is.numeric(y)) {
        metrica <- "-mse"
    }else {
        metrica <- "accuracy"
```

```
        }
    }
    # Se emplean todos los cores del ordenador menos 1.
    registerDoParallel(parallel::detectCores() - 1)
    # Se emplea la funciÃ³n foreach para paralelizar.
    fitness_poblacion <- foreach::foreach(i = 1:nrow(poblacion)) %dopar%
        {
        individuo <- poblacion[i, ]
        if (verbose) {
            print(paste("Individuo", i, ":", paste(individuo, collapse =
                " ")))
        }
        fitness_individuo <- calcular_fitness_individuo(x = x[, individuo
            , drop = FALSE],
            y = y,
            cv = cv,
            seed = seed,
            metrica = metrica,
            nivel_referencia = nivel_referencia,
            verbose = verbose
        )
        fitness_individuo
    }
    if (verbose) {
        print("Fitness de la poblaciÃ³n calculado")
        print("--------------------------------")
        print(paste("Modelo empleado para el cÃ¡lculo del fitness:",
            modelo))
        print(paste("MÃ©trica empleado para el cÃ¡lculo del fitness:",
            metrica))
        cat("\n")
    }
    # Se vuelve a un Ãºnico core.
    options(cores = 1)
    return(unlist(fitness_poblacion))
}
cruzar_individuos <- function(parental_1, parental_2, metodo_cruce = "
    uniforme", verbose = TRUE) {
    # Esta funciÃ³n devuelve un individuo resultado de cruzar dos
        individuos
    # parentales con el mÃ©todo de cruzamiento uniforme.
    #
    # ARGUMENTOS
    # ==========
```

```
# parental_1:
#   vector que representa a un individuo.
# parental_2:
#   vector que representa a un individuo.
# metodo_cruce: (uniforme", "punto_simple")
#   estrategia de cruzamiento.

# RETORNO
# ==========================================
# Un vector que representa a un nuevo individuo.

# Para crear el nuevo individuo, se emplea el mÃ©todo de cruzamiento
    uniforme,
# con la misma probabilidad de que el valor proceda de cada parental.

if (length(parental_1) != length(parental_2)) {
    stop(paste0(
        "La longitud de los dos vectores que representan a los ",
        "individuos debe ser la misma."
    ))
}
if (!(metodo_cruce %in% c("uniforme", "punto_simple"))) {
    stop("El mÃ©todo de cruzamiento debe ser: uniforme o punto_simple
        .")
}

# Se crea el vector que representa el nuevo individuo
descendencia <- rep(NA, times = length(parental_1))

if (metodo_cruce == "uniforme") {
    # Se seleccionan aleatoriamente las posiciones que se heredan del
        parental_1.
    herencia_parent_1 <- sample(
        x = c(TRUE, FALSE),
        size = length(parental_1),
        replace = TRUE
    )
    # El resto de posiciones se heredan del parental_2.
    herencia_parent_2 <- !herencia_parent_1

    descendencia[herencia_parent_1] <- parental_1[herencia_parent_1]
    descendencia[herencia_parent_2] <- parental_2[herencia_parent_2]
} else {
    punto_cruce <- sample(
```

```
            x    = 2:length(parental_1),
            size = 1
        )
        descendencia <- c(
            parental_1[1:(punto_cruce-1)],
            parental_2[punto_cruce: length(parental_1)]
        )
    }

    if(verbose){
        print("Cruce realizado")
        print("----------------")
        print(paste("MÃ©todo: ", metodo_cruce))
        cat("\n")
        print("Parental 1")
        print(parental_1)
        print("Parental 2")
        print(parental_2)
        cat("\n")
        print("Descendencia")
        print(descendencia)
        cat("\n")
    }
    return(descendencia)
}
seleccionar_individuo <- function(vector_fitness, metrica, metodo_
    seleccion = "tournament", verbose = TRUE) {
    # ARGUMENTOS
    # ==================================
    # vector_fitness: "numeric"
    #   un vector con el fitness de cada individuo.
    #
    # metrica: ("-mse", "f1", "kappa", "accuracy")
    #   mÃ©trica empleada para calcular el fitness.
    #
    # metodo_seleccion: ("ruleta", "rank", o "tournament").
    #   mÃ©todo para establecer la probabilidad de seleccÃ³n.

    # RETORNO
    # ==================================
    # El Ã ndice ("integer") que ocupa el individuo seleccionado.


    # SeleccÃ³n del individuo
```

```
    if (metodo_seleccion == "ruleta") {
        if (metrica == "-mse") {
            probabilidad_seleccion <- (1/vector_fitness) / sum(1/vector_
                fitness)
        } else if (metrica %in% c("accuracy", "f1", "kappa")) {
            probabilidad_seleccion <- (vector_fitness) / sum(vector_
                fitness)
        }

        ind_seleccionado <- sample(
            x = 1:length(vector_fitness),
            size = 1,
            prob = probabilidad_seleccion
        )
    } else if (metodo_seleccion == "rank") {
        if (metrica == "-mse") {
            vector_fitness <- (-1/vector_fitness)
        }
        probabilidad_seleccion <- 1 / rank(-1*vector_fitness)
        probabilidad_seleccion <- probabilidad_seleccion / sum(
            probabilidad_seleccion)

        ind_seleccionado <- sample(
            x = 1:length(vector_fitness),
            size = 1,
            prob = probabilidad_seleccion
        )
    } else if (metodo_seleccion == "tournament") {
        # Se seleccionan aleatoriamente dos parejas de individuos.
        ind_candidatos_a <- sample(x = 1:length(vector_fitness), size = 2
            )
        ind_candidatos_b <- sample(x = 1:length(vector_fitness), size = 2
            )

        # De cada pareja se selecciona el de mayor fitness.
        ind_ganador_a <- ifelse(
            vector_fitness[ind_candidatos_a[1]] > vector_fitness[ind_
                candidatos_a[2]],
            ind_candidatos_a[1],
            ind_candidatos_a[2]
        )
        ind_ganador_b <- ifelse(
            vector_fitness[ind_candidatos_b[1]] > vector_fitness[ind_
                candidatos_b[2]],
```

```
            ind_candidatos_b[1],
            ind_candidatos_b[2]
        )

        # Se comparan los dos ganadores de cada pareja.
        ind_seleccionado <- ifelse(
            vector_fitness[ind_ganador_a] > vector_fitness[ind_ganador_b
                ],
            ind_ganador_a,
            ind_ganador_b
        )
    } else {
        stop("El argumento metodo_seleccion debe ser: ruleta, rank o
            tournament")
    }

    if (verbose) {
        print("Individuo seleccionado")
        print("----------------------")
        print(paste("MÃ©trica de selecciÃ³n empleada: ", metrica))
        print(paste("MÃ©todo de selecciÃ³n empleado: ", metodo_seleccion)
            )
        print(paste("Indice seleccionado:", ind_seleccionado))
        cat("\n")
    }
    return(ind_seleccionado)
}
mutar_individuo <- function(individuo, prob_mut = 0.01) {
    # Este mÃ©todo somete al individuo a un proceso de mutaciÃ³n en el
        que, cada
    # una de sus posiciones, puede verse modificada con una probabilidad
        `prob_mut`.
    #
    # ARGUMENTOS
    # ==========
    # individuo:
    #   vector que representa a un individuo.
    # prob_mut:
    #   probabilidad que tiene cada posiciÃ³n del vector de mutar.

    # RETORNO
    # ==========
    # Un vector que representa al individuo tras someterse a las
        mutaciones.
```

```
    # SeleccÃ³n de posiciones a mutar.
    posiciones_mutadas <- runif(n = length(individuo), min = 0, max = 1)
        < prob_mut

    # Se modifica el valor de aquellas posiciones que hayan sido
        seleccionadas para
    # mutar. Si el valor de prob_mut es muy bajo, las mutaciones serÃ¡n
        muy poco
    # frecuentes y el individuo devuelto serÃ¡ casi siempre igual al
        original.
    individuo[posiciones_mutadas] <- !(individuo[posiciones_mutadas])
    return(individuo)
}

seleccionar_predictores <- function(x, y, n_poblacion = 20, n_
    generaciones = 10, n_max_predictores = 10, n_min_predictores = 6,
                                    modelo = "glm", cv = 3, metrica = NULL
                                        , nivel_referencia = NULL, elitismo
                                         = 0.1, prob_mut = 0.01,
                                    metodo_seleccion = "tournament",
                                        metodo_cruce = "uniforme", parada_
                                        temprana = FALSE,
                                    rondas_parada = NULL, tolerancia_
                                        parada = NULL, paralelizado = FALSE
                                        , verbose = TRUE){
    # ARGUMENTOS
    # ==========================================
    # x: "matrix" o "data.frame"  matriz de predictores.
    # y: "vector" variable respuesta.
    # n_poblacion: "integer" nÃºmero total de individuos de la poblaciÃ³n
        .
    # n_variables: "integer" longitud de los individuos.
    # n_max_predictores: "integer" nÃºmero mÃ¡ximo de predictores que
        puede contener un individuo.
    # n_min_predictores: "integer" nÃºmero mÃnimo de predictores que
        puede contener un individuo.
    # modelo: ("lm", "glm", "rf").  tipo de modelo empleado para calcular
         el fitness.
    # cv: "integer" nÃºmero de particiones de validaciÃ³n cruzada.
    # seed: "integer" semilla para garantizar reproducibilidad en el
        proceso de CV.
    # metrica: ("-mse", "f1", "kappa", "accuracy") mÃ©trica utilizada
        para calcular el fitness. Por defecto es el -MSE para
```

```
#   regresiÃ³n y Accuracy para clasificaciÃ³n. En el caso de
    clasificaciÃ³n binaria, se acepta tambiÃ©n f1 y kappa.
# nivel_referencia: "character" valor de la variable respuesta
    considerado como referencia. Necesario
#   cuando la mÃ©trica es f1 o kappa.
# elitismo: "numeric" porcentaje de mejores individuos de la
    poblaciÃ³n actual que pasan directamente a la siguiente
# poblaciÃ³n. De esta forma, se asegura que, la siguiente generaciÃ³n
    , no sea nunca peor.
# prob_mut: "numeric" probabilidad que tiene cada posiciÃ³n del
    individuo de mutar.
# metodo_seleccion: ("ruleta", "rank", "tournament") mÃ©todo de
    selecciÃ³n de los individuos para los cruces.
# metodo_cruce: (uniforme", "punto_simple") estrategia de cruzamiento
    .
# parada_temprana: "logical" si durante las Ãºltimas `rondas_parada`
    generaciones la diferencia absoluta entre mejores
# individuos no es superior al valor de `tolerancia_parada`, se
    detiene el algoritmo y no se crean nuevas generaciones.
# rondas_parada: "int" nÃºmero de generaciones consecutivas sin
    mejora mÃnima para que se active la parada temprana.
# tolerancia_parada: "numeric" valor mÃnimo que debe tener la
    diferencia de generaciones consecutivas
#   para considerar que hay cambio.
# ntree: "integer" nÃºmero de Ã¡rboles del modelo randomforest.
# paralelizado: "logical" si se paraleliza o no el algoritmo de
    busqueda.
# verbose: "logical" mostrar informaciÃ³n del proceso por pantalla.
# RETORNO
# ==========================
# Objeto "lista" con los resultados del algoritmo genÃ©tico:
# fitness: fitness del mejor individuo de cada generaciÃ³n.
# mejor_individuo: informaciÃ³n del mejor individuo de cada
    generaciÃ³n.
# porcentaje_mejora: mejora respecto a la generaciÃ³n anterior.
# df_resultados: dataframe con los resultados del proceso en cada
    generaciÃ³n.
# mejor_individuo_global: mejor individuo encontrado en todo el
    proceso.
library(foreach)
library(doParallel)
library(randomForest)
library(caret)
library(MLmetrics)
```

```
# COMPROBACIONES INICIALES
if(!is.numeric(y) & modelo == "lm"){
    stop("El modelo lm solo puede aplicarse a problemas de regresiÃ³n
        , (variable respuesta numÃ©rica).")
}
if(is.numeric(y) & modelo == "glm"){
    stop("El modelo glm solo puede emplearse para problemas de
        clasificaciÃ³n binaria.")
}
if((length(levels(y)) != 2) & modelo == "glm"){
    stop("El modelo glm solo puede emplearse para problemas de
        clasificaciÃ³n binaria.")
}
# El nÃºmero mÃ¡ximo de predictores no puede superar el nÃºmero de
    columnas de x.
if(n_max_predictores > ncol(x)){
    stop("El nÃºmero mÃ¡ximo de predictores no puede superar al
        nÃºmero de variables disponibles en x.")
}
# Si se activa la parada temprana, hay que especificar los argumentos
    : rondas_parada y tolerancia_parada.
if(isTRUE(parada_temprana) & (is.null(rondas_parada) | is.null(
    tolerancia_parada))){
    stop("Para activar la parada temprana es necesario indicar un
        valor de rondas_parada y de tolerancia_parada.")
}
# ALMACENAMIENTO DE RESULTADOS
# ==============================
# Por cada generaciÃ³n se almacena el mejor individuo, su fitness, y
    el porcentaje de mejora respecto a la Ãºltima generaciÃ³n.
resultados_fitness    <- vector(mode = "list", length = n_generaciones
    )
resultados_individuo <- vector(mode = "list", length = n_generaciones
    )
porcentaje_mejora     <- vector(mode = "list", length = n_generaciones
    )
# CREACIÃN DE LA POBLACIÃN INICIAL
poblacion <- crear_poblacion(n_poblacion = n_poblacion, n_variables =
    ncol(x), n_max = n_max_predictores,
    n_min = n_min_predictores, verbose = verbose)
# ITERACIÃN DE POBLACIONES
for(i in 1:n_generaciones){
    if(verbose){
        print("----------------------")
```

```
    print(paste("GeneraciÃ³n: ", i))
    print("----------")
}
# CALCULAR FITNESS DE LOS INDIVIDUOS DE LA POBLACIÃ N
# ==========
if(!paralelizado){
    fitness_ind_poblacion <- calcular_fitness_poblacion(poblacion
        = poblacion, x = x, y = y, modelo = modelo,
        cv = cv, metrica = metrica, nivel_referencia = nivel_
            referencia, verbose = verbose)
}
if(paralelizado){
    fitness_ind_poblacion <- calcular_fitness_poblacion_paral(
        poblacion = poblacion, x = x, y = y, modelo = modelo,
        cv = cv, metrica = metrica, nivel_referencia = nivel_
            referencia, verbose = verbose)
}
# SE ALMACENA EL MEJOR INDIVIDUO DE LA POBLACIÃ N ACTUAL
# ==========
fitness_mejor_individuo    <- max(fitness_ind_poblacion)
mejor_individuo            <- poblacion[which.max(fitness_ind_
    poblacion), ]
resultados_fitness[[i]]    <- fitness_mejor_individuo
resultados_individuo[[i]] <- colnames(x)[mejor_individuo]
# SE CALCULA LA MEJORA RESPECTO A LA GENERACIÃ N ANTERIOR
# ==========
# El porcentaje de mejora solo puede calcularse a partir de la
    segunda generaciÃ³n.
if(i > 1){
    porcentaje_mejora[[i]] <- 1 - (resultados_fitness[[i]]/
        resultados_fitness[[i - 1]])
}
# NUEVA POBLACIÃ N
# ==========
nueva_poblacion <- matrix(data = NA, nrow = nrow(poblacion), ncol
    = ncol(poblacion))
# ELITISMO
# ==========
# El elitismo indica el porcentaje de mejores individuos de la
    poblaciÃ³n actual que pasan directamente a la siguiente
# poblaciÃ³n. De esta forma, se asegura que, la siguiente
    generaciÃ³n, no sea nunca inferior.
if(elitismo > 0){
    n_elitismo          <- ceiling(nrow(poblacion)*elitismo)
```

```
      posicion_n_mejores <- order(fitness_ind_poblacion, decreasing
          = TRUE)
      posicion_n_mejores <- posicion_n_mejores[1:n_elitismo]
      nueva_poblacion[1:n_elitismo, ] <- poblacion[posicion_n_
          mejores, ]
  }else{
      n_elitismo <- 0
  }
  # CREACIÃ N DE NUEVOS INDIVIDUOS POR CRUCES
  # ==============================================
  for(j in (n_elitismo + 1):nrow(nueva_poblacion)){
      # Seleccionar parentales
      metrica <- ifelse(test = is.numeric(y), "-mse", "accuracy")
      indice_parental_1 <- seleccionar_individuo(vector_fitness =
          fitness_ind_poblacion, metrica = metrica,
          metodo_seleccion = metodo_seleccion, verbose = verbose)
      indice_parental_2 <- seleccionar_individuo(vector_fitness =
          fitness_ind_poblacion, metrica = metrica,
          metodo_seleccion = metodo_seleccion, verbose = verbose)
      parental_1 <- poblacion[indice_parental_1, ]
      parental_2 <- poblacion[indice_parental_2, ]
      # Cruzar parentales para obtener la descendencia
      descendencia <- cruzar_individuos(parental_1 = parental_1,
          parental_2 = parental_2, metodo_cruce = metodo_cruce,
          verbose = verbose)
      # Mutar la descendencia
      descendencia <- mutar_individuo(individuo = descendencia,
          prob_mut = prob_mut)
      # Almacenar el nuevo descendiente en la nueva poblaciÃ³n:
          puede ocurrir que el individuo resultante del cruce y
      # posterior mutaciÃ³n no contenga ningÃºn predictor (todas
          sus posiciones son FALSE). Si esto ocurre, y para evitar
      # que se produzca un error al ajustar el modelo, se introduce
           aleatoriamente un valor TRUE.
      if(all(descendencia == FALSE)){
          descendencia[sample(x = 1:length(descendencia), size = 1)
              ] <- TRUE
      }
      nueva_poblacion[j, ] <- descendencia
  }
  poblacion <- nueva_poblacion
  # CRITERIO DE PARADA
  # ==============================================
  # Si durante las Ãºltimas n generaciones el mejor individuo no ha
```

```
            mejorado por una cantidad superior a
        # tolerancia_parada, se detiene el algoritmo y no se crean nuevas
            generaciones.
        if(parada_temprana && (i > rondas_parada)){
            ultimos_n <- tail(unlist(porcentaje_mejora), n = rondas_
                parada)
            if(all(ultimos_n < tolerancia_parada)){
                print(paste("Algoritmo detenido en la generacion", i, "
                    por falta de mejora mÃnima de", tolerancia_parada,
                    "durante", rondas_parada, "generaciones consecutivas.
                        " ))
                break()
            }
        }
        #
    }
    # IDENTIFICACIÃ N DEL MEJOR INDIVIDUO DE TODO EL PROCESO
    mejor_individuo_global <- resultados_individuo[[which.max(unlist(
        resultados_fitness))]]
    # RESULTADOS
    # La funciÃ³n devuelve una lista con 4 elementos:
    # fitness: una lista con el fitness del mejor individuo de cada
        generaciÃ³n.
    # mejor_individuo: una lista con la combinaciÃ³n de predictores  del
        mejor individuo de cada generaciÃ³n.
    # porcentaje_mejora: una lista con el porcentaje de mejora entre el
        mejor individuo de cada generaciÃ³n.
    # df_resultados: un dataframe con todos los resultados anteriores.
    # mejor_individuo_global: la combinaciÃ³n de predictores  del mejor
        individuo encontrado en todo el proceso.

    # Para crear el dataframe se convierten las listas a vectores del
        mismo tamaÃ±o.
    fitness     <- unlist(resultados_fitness)
    predictores <- resultados_individuo[!sapply(resultados_individuo, is.
        null)]
    predictores <- sapply(X = predictores, FUN = function(x){paste(x,
        collapse = ", ")})
    mejora <- c(NA, unlist(porcentaje_mejora))

    df_resultados <- data.frame(generacion = seq_along(fitness), fitness
        = fitness, predictores = predictores, mejora = mejora)
    return(list(fitness = resultados_fitness, mejor_individuo =
        resultados_individuo, porcentaje_mejora = porcentaje_mejora,
```

```
            df_resultados = df_resultados, mejor_individuo_global = mejor
                _individuo_global))
}
```

Anexo 4:

VIF de los modelos de regresión:

Variable	**VIF**
ln_NMES_ULT_VEC_SBS_SC_12M	2,58
ln_PROM_VEN_SBS_6M	1,37
NENT_VEN_SCE_3M	2,57
ln_NTC_VENC_TC_3M	1,09
prbb_genero_ecivil	1,01
r_NOPE_XVEN_OP_3s6M	1,03
r_NOPE_VENC_31AMAS_OP_3s6M	1,95
r_DEUDA_TOTAL_SICOM_OP_12s24M	2,29

Tabla 7.12: VIF - logit - dirty.
Elaboración propia

Variable	**VIF**
NENT_VEN_SCE_36M	1,06
r_NTC_APERT_SCE_12a36M	1,18
NTC_XVEN_TC_3M	1,20
r_DEUDA_TOTAL_SCE_6a12M	1,19
prbb_genero_ecivil	1,02
r_NOPE_APERT_SBS_OP_12s24M	1,14
ln_NMES_ULT_REFIN_TC_12M	1,01
ln_NMES_ULT_REFIN_OP_12M	1,01
r_PROM_VEN_SBSsPROM_DEUDA_TOTAL_SBS_TC_36M	1,05

Tabla 7.13: VIF - logit - clean.
Elaboración propia

Variable	VIF
ln_NMES_ULT_VENC_SBS_SC_12M	1,34
NENT_VEN_SCE_3M	1,74
ln_MVALVEN_SBS_TC_6M	1,13
r_PROM_VEN_SBS_TCsPROM_DEUDA_TOTAL_SBS_OP_12M	1,36
NOPE_XVEN_OP_3M	1,19
NOPE_VENC_31AMAS_OP_3M	1,27
r_DEUDA_TOTAL_SICOM_OP_12s24M	1,66
NENT_VEN_SCE_12M	1,83
ln_MVALVEN_SBS_OP_3M	1,28
r_NOPE_APERT_SCE_12a24M	1,96

Tabla 7.14: VIF - logit - dirty - con AG.
Elaboración propia

Variable	VIF
NENT_VEN_SCE_36M	8,16
PROM_DEUDA_TOTAL_SBS_TC_3M	2,02
NTC_APERT_SCE_24M	1,78
NTC_XVEN_TC_3M	3,06
LN_DEUDA_TOTAL_SCE_3M	1,28
NOPE_APERT_SCE_24M	1,19
MVALVEN_SBS_TC_36M	1,45
NENT_VEN_SCE_24M	5,84
r_NOPE_APERT_SBS_OP_12s24M	1,21
r_DEUDA_TOTAL_SCE_12s24M	1,23

Tabla 7.15: VIF - logit - clean - con AG.
Elaboración propia

Anexo 5:

Descripción de las variables independientes:

Variable	Descripción
ln_NMES_ULT_VENC_SBS_SC_12M	Logaritmo natural del número de meses últimos vencidos en los sistemas SBS y SC en los 12 meses anteriores al punto de observación.
ln_PROM_VEN_SBS_6M	Logaritmo natural del promedio de saldos vencidos en el SBS en los 6 meses anteriores al punto de observación.
NENT_VEN_SCE_3M	Número de entidades en el SCE en las que tiene saldo vencido en los 3 meses anteriores al punto de observación.
ln_NTC_VENC_TC_3M	Logaritmo natural del número de tarjetas de crédito con saldo vencido en los 3 meses anteriores al punto de observación.
prbb_genero_ecivil	Probabilidad de que un cliente sea bueno sabiendo su género y estado civil.
r_NOPE_XVEN_OP_3s6M	Ratio del número de operaciones por vencer de los últimos 3 meses respecto a los últimos 6 meses.
r_NOPE_VENC_31AMAS_OP_3s6M	Ratio del número de operaciones vencidas 31 días a más de los últimos 3 meses respecto a los últimos 6 meses.
r_DEUDA_TOTAL_SICOM_OP_12s24M	Ratio de la deuda total en el sistema comercial en operaciones de los últimos 12 meses respecto a los últimos 24 meses.
NENT_VEN_SCE_36M	Número de entidades en el SCE en las que tiene saldo vencido en los 36 meses anteriores al punto de observación.
r_NTC_APERT_SCE_12s36M	Ratio del número de tarjetas de crédito aperturadas en el SCE en los últimos 12 meses respecto a los últimos 36 meses.
NTC_XVEN_TC_3M	Número de tarjetas de crédito con saldo x vencer en los últimos 3 meses.
r_DEUDA_TOTAL_SCE_6a12M	Ratio de la deuda total en el SCE en los últimos 6 meses respecto a los últimos 12 meses.

Tabla 7.16: Elaboración propia

Variable	Descripción
r_NOPE_APERT_SBS_OP_12s24M	Ratio del número de operaciones aperturadas en el SBS en los últimos 12 meses respecto a los últimos 24 meses.
ln_NMES_ULT_REFIN_TC_12M	Logaritmo natural del número de meses últimos refinanciados en el sistema TC en los 12 meses anteriores al punto de observación.
ln_NMES_ULT_REFIN_OP_12M	Logaritmo natural del número de meses últimos refinanciados de operaciones en los 12 meses anteriores al punto de observación.
r_PROM_VEN_SBSsPROM_DEUDA_TOTAL_SBS_TC_36M	Razón del promedio de saldos vencidos sobre el promedio de deuda total en el SBS y TC de los últimos 36 meses.
ln_MVALVEN_SBS_TC_6M	Logaritmo natural del máximo valor vencido en el SBS de tarjetas de crédito en los últimos 6 meses anteriores al punto de observación.
r_MVALVEN_SBSsDEUDA_TOTAL_SBS_TC_12M	Razón del máximo valor vencido en el SBS sobre la deuda total en el SBS de tarjetas de crédito en los últimos 12 meses anteriores al punto de observación.
r_PROM_VEN_SBSsPROM_DEUDA_TOTAL_SBS_TC_12M	Razón del promedio de saldos vencidos sobre el promedio de deuda total en el SBS y TC de los últimos 12 meses.
r_PROM_VEN_SBS_TCs PROM_DEUDA_TOTAL_SBS_OP_12M	Razón del promedio de saldos vencidos en el SBS en tarjetas de crédito sobre el promedio de deuda total en el SBS de operaciones de los últimos 12 meses.
NOPE_XVEN_OP_3M	Número de operaciones por vencer de los últimos 3 meses anteriores al punto de observación.
NTC_VENC_31AMAS_TC_24M	Número de tarjetas de crédito con saldo vencido 31 a más días en los últimos 24 meses anteriores al punto de observación.
r_DEUDA_TOTAL_SBS_TC_3s6M	Ratio de la deuda total en el SBS en los últimos 3 meses respecto a los últimos 6 meses.
NTC_APERT_SCE_24M	Número de tarjetas de crédito aperturadas en el SCE los últimos 24 meses antes del punto de observación.
ln_DEUDA_TOTAL_SCE_3M	Logaritmo natural de la deuda total en el SCE los últimos 3 meses antes del punto de observación.
r_DEUDA_TOTAL_SCE_3s6M	Ratio de la deuda total en el SCE de los últimos 3 meses respecto a los últimos 6 meses.
ln_PROM_VENC_SCE_24M	Logaritmo natural del promedio de saldos vencidos en el SCE los últimos 24 meses antes del punto de observación.

Tabla 7.17: Elaboración propia

Variable	Descripción
ln_MVALVEN_SBS_TC_36M	Logaritmo natural del máximo valor vencido en el SBS de tarjetas de crédito los últimos 36 meses antes del punto de observación.
NOPE_VENC_31AMAS_OP_3M	Número de operaciones vencidas 31 a más días los últimos 3 meses antes del punto de observación.
NENT_VEN_SCE_12M	Número de entidades en el SCE en las que tiene saldo vencido en los 12 últimos meses anteriores al punto de observación.
ln_MVALVEN_SBS_OP_3M	Logaritmo natural del máximo valor vencido en el SBS de operaciones los últimos 3 meses antes del punto de observación.
r_NOPE_APERT_SCE_12s24M	Ratio del número de operaciones aperturadas en el SCE en los últimos 12 meses respecto a los últimos 24 meses.
PROM_DEUDA_TOTAL_SBS_TC_3M	Promedio de las deudas totales en el SBS en tarjetas de crédito en los últimos 3 meses anteriores al punto de observación.
NOPE_APERT_SCE_24M	Número de operaciones aperturadas en el SCE los últimos 24 meses antes del punto de observación.
MVALVEN_SBS_TC_36M	Máximo valor vencido en el SBS en tarjetas de crédito los últimos 36 meses.
NENT_VEN_SCE_24M	Número de entidades en el SCE en las que tiene saldo vencido en los 24 meses anteriores al punto de observación.
r_DEUDA_TOTAL_SCE_12s24M	Ratio de la deuda total en el SCE en los últimos 12 meses respecto a los últimos 24 meses.
NMES_ULT_VENC_SBS_SC_12M	Número de meses últimos vencidos en los sistemas SBS y SC en los 12 meses anteriores al punto de observación.
PROM_VEN_SBS_24M	Promedio de saldos vencidos en el SBS los últimos 24 meses anteriores al punto de observación.
MVALVEN_SBS_TC_6M	Máximo valor vencido en el SBS en tarjetas de crédito los últimos 6 meses.
PROM_VEN_SBS_TC_6M	Promedio de saldos vencidos en el SBS en tarjetas de crédito los últimos 6 meses.
NTC_VENC_TC_3M	Número de tarjetas de crédito con saldo vencido en los últimos 3 meses.
r_PROM_VEN_SBS_TCs PROM_DEUDA_TOTAL_SBS_OP_24M	Razón del promedio de saldos vencidos en el SBS en tarjetas de crédito sobre el promedio de deuda total en el SBS de operaciones de los últimos 24 meses.

Tabla 7.18: Elaboración propia

Variable	Descripción
r_MVALVEN_SBS_TCs MVALVEN_SBS_OP_12M	Razón del máximo valor vencido en el SBS de tarjetas de crédito sobre el máximo valor vencido en el SBS de operaciones en los últimos 12 meses anteriores al punto de observación.
DEUDA_TOTAL_SICOM_ OP_12M	Deuda total en el sistema comercial de operaciones los últimos 12 meses anteriores al punto de observación.
DEUDA_TOTAL_SCE_3M	Deuda total en el SCE los últimos 3 meses anteriores al punto de observación.
r_NOPE_APERT_SBSsSCE_ OP_24M	Razón del número de operaciones aperturadas en el SBS sobre el SCE de operaciones en los últimos 24 meses.
NMES_ULT_REFIN_TC_ 12M	Número de meses últimos refinanciados de tarjetas de crédito en los últimos 12 meses anteriores al punto de observación.
NTC_APERT_SCE_6M	Número de tarjetas aperturadas en el SCE en los últimos 6 meses anteriores al punto de observación.
NMES_ULT_REFIN_OP_ 12M	Número de meses últimos refinanciados de operaciones en los últimos 12 meses anteriores al punto de observación.
r_DEUDA_TOTAL_SCE_ 12s24M	Ratio de la deuda total en el SCE en los últimos 12 meses respecto a los últimos 24 meses anteriores al punto de observación.
r_DEUDA_TOTAL_SBS_OP_ 3s12M	Ratio de la deuda total en el SBS de operaciones en los últimos 3 meses respecto a los últimos 12 meses anteriores al punto de observación.
r_NOPE_APERT_SCE_ 12a24M	Ratio del número de operaciones aperturadas en el SCE de los últimos 12 meses respecto a los últimos 24 meses anteriores al punto de observación.

Tabla 7.19: Elaboración propia

Anexo 6:

CÁLCULO DEL KS Y VI:

SEGMENTO CLEAN:

Variable	**KS**
PORC_USO_CUPO	0,2302
r_DEUDA_TOTAL_SBS_TC_3s6M	0,1504
NENT_VEN_SCE_36M	0,1497
PROM_DEUDA_TOTAL_SBS_TC_3M	0,1489
NTC_APERT_SCE_24M	0,1217
r_NTC_APERT_SCE_12a36M	0,1173
NTC_XVEN_TC_3M	0,1148
NOPE_XVEN_OP_3M	0,1105
IDENTIFICACION	0,1065
DEUDA_TOTAL_SCE_3M	0,1036
LN_DEUDA_TOTAL_SCE_3M	0,1036
NOPE_APERT_SCE_24M	0,1028
r_DEUDA_TOTAL_SCE_3a6M	0,0989
PROM_VEN_SCE_24M	0,0935
r_DEUDA_TOTAL_SCE_6a12M	0,0935
MVALVEN_SBS_TC_36M	0,0934
NENT_VEN_SCE_24M	0,0934
r_MVALVEN_SBS_TCsMVALVEN_SBS_OP_36M	0,093
PorcCuentaAhorrosCorriente	0,0914
PROM_VEN_SCE_36M	0,0907
r_NOPE_APERT_SBS_OP_12s24M	0,0906
PromVehiculosUsoHogar	0,09
PromVehiculosxVivienda	0,09
PromLocalesCom	0,09
NOPE_VENC_OP_36M	0,0898
r_NOPE_APERT_SBSsSCE_OP_24M	0,0897
PromVehiculosUsoComercial	0,0897
NMES_ULT_REFIN_TC_12M	0,0884
NTC_APERT_SCE_6M	0,0881
PorcCuentaCoop	0,0881
NMES_ULT_REFIN_OP_12M	0,0881
ANTIGUEDAD_TC_SBS	0,088
ANTIGUEDAD_OP_SBS	0,0871
PorcPersComprasCred	0,0869
r_PROM_XVEN_SBS_OP_12s36M	0,0864

Tabla 7.20: Elaboración propia

Variable	KS
PromNumTerrenos	0,0861
PorcVivPropiaProv	0,0859
PorcTelefonoProv	0,0856
NTC_NDI_TC_36M	0,0837
PROM_VEN_SBS_TC_36M	0,0825
PROM_NDI_SBS_TC_36M	0,0824
r_PROM_VEN_SBSsPROM_DEUDA_TOTAL_SBS_TC_36M	0,0823
r_PROM_VEN_SBS_TCsPROM_DEUDA_TOTAL_SBS_OP_36M	0,0823
PorcHogaresTerrenos	0,0822
PROM_MAX_DVEN_SBS_TC_36M	0,082
r_PROM_DEUDA_TOTAL_SBS_OP_3s12M	0,0801
r_DEUDA_TOTAL_SCE_12a24M	0,0784
r_DEUDA_TOTAL_SBS_OP_3s12M	0,0774
PorcPrestamo	0,0769
r_NOPE_APERT_SCE_12a24M	0,076
PorcHipotecas	0,0757
r_NOPE_APERT_SCE_6a12M	0,0748
PromEdadJefeHogar	0,0746
r_NOPE_APERT_SBS_OP_6s12M	0,0725
PorcCuentaMutualista	0,0717
PromPagoLuz	0,0715
PorcCuentaFinanciera	0,0706
PorcCuentaCaja	0,0705
EDAD	0,0693
PorcCuentaPropia	0,0684
r_NOPE_XVEN_OP_6s24M	0,067
r_NOPE_VENCsNOPE_APERT_SBS_36M	0,0665

Tabla 7.21: Elaboración propia

Variable	VI
CodParroquia	0,0769
PROVINCIA	0,0556
PROFESION	0,0415
ESTADO_CIVIL	0,0242
PARROQUIA	0,0219
GENERO	0,0126
CANTON	0,0107

Tabla 7.22: Elaboración propia

SEGMENTO DIRTY:

Variable	**KS**
NMES_ULT_VENC_SBS_SC_12M	0,6043
PROM_VEN_SBS_6M	0,4741
NENT_VEN_SCE_3M	0,4386
PROM_VEN_SBS_24M	0,38
MVALVEN_SBS_TC_6M	0,3224
PROM_VEN_SBS_TC_6M	0,3221
r_MVALVEN_SBSsDEUDA_TOTAL_SBS_TC_6M	0,3142
PROM_MAX_DVEN_N_TC_3M	0,3111
NTC_VENC_TC_3M	0,3057
r_PROM_VEN_SBS_TCsPROM_DEUDA_TOTAL_SBS_OP_3M	0,3035
NMES_ULT_VENC_SBS_TC_12M	0,3017
r_PROM_VEN_SBS_TCsPROM_DEUDA_TOTAL_SBS_OP_6M	0,3001
r_MVALVEN_SBS_TCsMVALVEN_SBS_OP_3M	0,2999
r_PROM_VEN_SBS_TC_6s24M	0,291
r_MVALVEN_SBS_TCsMVALVEN_SBS_OP_6M	0,2829
PROM_NDI_SBS_TC_12M	0,282
PROM_VEN_SBS_TC_36M	0,282
r_MVALVEN_SBSsDEUDA_TOTAL_SBS_TC_12M	0,2781
r_MVALVEN_SBSsDEUDA_TOTAL_SBS_TC_24M	0,2677
r_NTC_VENC_31AMAS_TC_6s24M	0,2657
r_PROM_VEN_SBSsPROM_DEUDA_TOTAL_SBS_TC_12M	0,2638
r_PROM_VEN_SBS_TCsPROM_DEUDA_TOTAL_SBS_OP_12M	0,2486
r_PROM_VEN_SBS_TCsPROM_DEUDA_TOTAL_SBS_OP_24M	0,2454
NTC_NDI_TC_12M	0,2365
r_NOPE_XVEN_OP_3s6M	0,2343
NOPE_XVEN_OP_3M	0,2336
NTC_VENC_31AMAS_TC_24M	0,2307
NTC_VENC_1A30_TC_6M	0,2283
PORC_USO_CUPO	0,2222
NOPE_NDI_OP_3M	0,211

Tabla 7.23: Elaboración propia

Variable	KS
r_NOPE_XVEN_OP_6s24M	0,2103
NOPE_VENC_31AMAS_OP_3M	0,2093
r_NOPE_VENC_31AMAS_OP_3s6M	0,2093
r_MVALVEN_SBS_TCsMVALVEN_SBS_OP_12M	0,206
r_DEUDA_TOTAL_SICOM_OP_12s24M	0,2048
DEUDA_TOTAL_SICOM_OP_12M	0,2047
r_PROM_VEN_SICOM_OP_12s24M	0,2047
r_DEUDA_TOTAL_SICOMsSCE_12M	0,2047
PROM_VEN_SICOM_OP_12M	0,2046
r_MVALVEN_SICOMsDEUDA_TOTAL_SICOM_12M	0,2045
r_MVALVEN_SICOMsDEUDA_TOTAL_SBS_12M	0,2045
r_PROM_VEN_SICOMsPROM_DEUDA_TOTAL_SBS_12M	0,2045
NENT_VEN_SCE_12M	0,2039
r_MVALVEN_SBS_TCsMVALVEN_SBS_OP_24M	0,2022
MVALVEN_SBS_OP_3M	0,2007
r_MVALVEN_SBS_OP_3s12M	0,2007
r_NTC_VENC_1A30_TC_24s36M	0,2003
PROM_VEN_SBS_OP_3M	0,2002
r_DEUDA_TOTAL_SBSsSCE_12M	0,2001

Tabla 7.24: Elaboración propia

Variable	VI
CodParroquia	0,0962
GENERO	0,0469
INSTRUCCION	0,0362
PARROQUIA	0,0339
PROVINCIA	0,0321
CANTON	0,0114
ESTADO_CIVIL	0,0105
PROFESION	0,0100

Tabla 7.25: Elaboración propia