Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Iztapalapa

División de Ciencias Básicas e Ingeniería
Departamento de Matemáticas

Credit Scoring:
Un comparativo de un método paramétrico y un no paramétrico

Para obtener el grado de Maestro en Ciencias Matemáticas
(Aplicadas e Industriales)

PRESENTA
Mat. David Hernández Gómez
Matrícula: 2193803504

DIRECTOR
Dra. Blanca Rosa Pérez Salvador

JURADO
Presidente: Dra. Hortensia Josefina Reyes Cervantes

Secretario: Dr. Alberto Castillo Morales

Vocal: Dra. Blanca Rosa Pérez Salvador

Iztapalapa, Ciudad de México a 28 de Noviembre 2022

Dedicatoria

A Michelle, mi mayor tesoro y motor de vida.

Agradecimientos

Gracias a mi madre por ser de los principales promotores de mis sueños, por cada día confiar y creer en mí, apoyarme en mis decisiones de vida y en mis expectativas.

Quiero ser grato con esa persona que se preocupó por mí en cada momento y que siempre quiso lo mejor para mi porvenir, a ella porque en todo momento fue un apoyo incondicional en esta etapa de mi vida.

A mi asesora, la Dra. Pérez por apoyarme y guiarme en este proyecto, dedicada para superar toda duda que me surgiera, agradecerle por la exactitud y claridad con la que enseñó.

A mis amigos de maestría, por hacer el tiempo leve y divertido, y poder llevarme esa amistad para toda la vida.

A la UAM, por haber sido una segunda casa y poder tener el privilegio de haber realizado mis estudios de licenciatura y maestría en una de las mejores universidades de este país.

Índice general

Introducción

Para las empresas del mercado financiero es de suma importancia contar con herramientas que le ayuden a poder predecir o medir con la mayor confianza posible cuando un cliente o empresa cumplirá con lo suscrito en un crédito, para así obtener un mayor beneficio económico al menor riesgo posible. Por ello, surge la necesidad de implementar modelos estadísticos y no estadísticos que ayuden a estimar la probabilidad que se cumpla con las obligaciones de un crédito en tiempo y forma, o mejor aún, la probabilidad que no se cumpla. Se requiere que estos modelos tengan suficiente credibilidad o confianza en su desempeño para la toma de decisiones en negocios de inversión con la información disponible, que ayuden a un mayor y mejor posicionamiento en el sector financiero, pudiendo siempre ser flexibles a ajustes por cambios o ciclos económicos de entorno.

En el capítulo 1, de este trabajo se exponen las ideas y conceptos básicos que definen o componen un modelo *Credit Scoring*. Se aborda desde el surgimiento de la idea de un crédito hasta la evolución a un modelo *Credit Scoring* a través de su historia, así como su diversificación posterior y los sectores donde se aplica de forma cotidiana. Se mencionan los modelos clásicos basados en métodos estadísticos y no estadísticos, también se muestra como interactúa uno de estos modelos con factores de entorno social, cambios económicos o reajustes en las políticas o estrategias de mercado.

En el capítulo 2, se abordan los fundamentos teóricos estadísticos para un modelo paramétrico *Credit Scoring* utilizando la regresión logística, el cual se aplicará posteriormente como parte del proceso de construcción a una base de datos real financiera y se evaluará su desempeño.

Para el capítulo 3, se exponen los fundamentos teóricos de nuestro segundo tipo de modelo perteneciente de una rama de la Inteligencia Artificial: Machine Learning (ML), en el cual se presentan algunos de los algoritmos más usados en la tarea de Clasificación Supervisada. De forma más detallada se revisará un subtema de mayor interés en este tipo de algoritmos: Redes Neuronales Artificiales (ANN[1]) bajo este mismo enforque de clasificación supervisada con el método de optimización backpropagation. Se menciona la estructura y proceso de aprendizaje de este tipo de redes para ser aplicados como modelo no paramétrico para *Credit Scoring* para su también posterior evaluación de resultados.

[1] Por sus siglas en inglés *Artificial Neural Networks*

Para el capítulo 4 se describen las métricas de desempeño más utilizadas en los algoritmos de clasificación supervisada, en particular con los modelos *Credit Scoring*, tales como: *Matriz de Confusión* y sus métricas derivadas de ésta; Precisión, Recall, *Curva ROC* y el *Estadístico Kolmorov-Sminov* ,etc., así como sus interpretaciones. Dentro de este mismo capítulo se revisa un tema importante: técnicas para tratar datos desbalanceados tales como técnicas de remuestreo o costos sensitivos, un tema importante ya que trata el sesgo de clases mayoritarias en la base de información.

En el capítulo 5, se pone en práctica los revisado en los capítulos anteriores y también se explican los procesos de modelación: el proceso de entrenamiento, evaluación, prueba, e implementación o rechazo de un modelo *Credit Scoring*. Contando con la teoría y herramientas necesarias, se puede aplicar dicho proceso tanto para un modelo paramétrico como en uno no paramétrico. Se entrenan varios modelos ANN y de Regresión Logística, tratando según corresponda temas como subajuste o sobreajuste, comparando su desempeño final con las métricas correspondientes. En esta aplicación se realizará su correspondiente análisis en la capacidad de predicción y flexibilidad de ajuste en cada uno de los algoritmos, así como la facilidad de implementación e interpretación. Finalmente, con los resultados anteriores se compararán y determinará cual puede ser el mejor método para crear un Credit Score según los datos históricos utilizados en ese caso en específico, así como políticas y objetivos.

En el último capítulo se presentan las conclusiones finales en el proceso del capítulo 5, tales como la implementación de cada uno de los algoritmos, ventajas y desventajas en cada uno de estos, así como posibles mejoras, recomendaciones y observaciones.

Antecedentes

Investigaciones realizadas anteriormente sobre predicción del riesgo han utilizados diversas técnicas y herramientas matemáticas tanto clásicas como modernas para estimar la probabilidad de impago de un crédito con base en diferentes bases de datos relacionadas al sector financiero, comparando su desempeño de predicción, interpretabilidad y estabilidad a través del tiempo; métodos clásicos contra modernos, modelos estadísticos paramétricos y no paramétricos de Machine Learning. Regresión lineal múltiple contra árboles de decisión «The use of profit scoring as an alternative to credit scoring systems in peer-to-peer (P2P) lending.»(Carlos Serrano-Cinca y Begoña Gutiérrez-Nieto, 2016), donde permite un mayor riesgo a costa de un también mayor beneficio monetario. Redes de creencia profunda y Vectores máquina de soporte contra regresión logística «A deep learning approach for credit scoring using credit default swaps.»(Cuicui Luo, 2016), Regresión logística contra AdaBoost, Random Forest, Naive Bayes y Vectores máquina de soporte «Aplicación de Metodologías Machine Learning a Gestión de Riesgo de Crédito.»(David Trujillo Fernández, 2017). Sus resultados cambian y son diferentes dependiendo del enfoque y los datos utilizados en sus comparativas, pero comúnmente apuntando a una ligera y en algunos casos un superior desempeño de los modelos modernos Machine Learning sobre los modelos clásicos paramétricos.

Credit Scoring: *Es el conjunto de modelos de decisión y sus técnicas subyacentes que ayudan a financieros en el otorgamiento de créditos de consumo. Estas técnicas deciden quienes obtendrá el crédito, el monto y qué estrategias operacionales mejorarán la rentabilidad de los deudores a los prestamistas. (Lyn C. [31])*

Modelo Paramétrico: Parte de una función de distribución de probabilidad conocida cuyo problema principal es la estimación del o los parámetros que mejor se ajusten a los datos. $Y \sim F$ donde F es una función de distribución que está contenida en una familia.

Modelo No paramétrico: Estos parten de una distribución libre debido a que no están sujetos a una forma funcional que permiten formular una aproximación de los parámetros de una función y no de una función conocida. $Y \sim F$ donde F es una función de distribución, ésta sigue un modelo no paramétrico si su distribución F solo aplican restricciones de regularidad y que no se puede indexar por un parámetro θ.

Inteligencia Artificial: *"... un campo de la ciencia computacional e ingeniería ocupado con el entendimiento computacional de lo que comúnmente llamamos comportamiento inteligente, y con la creación de artefactos que exhiben dicho comportamiento" (Shapiro, S. C. [29]).*

Red Neuronal Artificial: *Modelos que reproducen neuronas biológicas para explicar o entender su comportamiento o resolver problemas utilizando modelos computacionales inspirados en neuronas (Brandon Reagen [27]).*

Predicción: Salida de un modelo Credit Scoring cuya interpretación es la probabilidad de incumplimiento para pagar un crédito adquirido ($\hat{P}\left(Y = 1|\boldsymbol{x}\right) = \hat{\theta}$). Este dato es utilizado para la toma de decisión en otorgar o denegar dicho crédito con un riesgo límite fijo θ_0:

$$I_{\theta_0}\left(\hat{\theta}\right) = \hat{Y} = \underset{(\theta_0,1)}{\mathbb{1}}\left(\hat{\theta}\right) = \begin{cases} \text{Se otorga} & \text{Si } \theta_0 < \hat{\theta} \\ \text{Se rechaza} & \text{Otro caso .} \end{cases}$$

Métrica de desempeño: Función que se aplica a los resultados de predicción de los modelos cuyas clasificaciones reales de los datos se conocen de antemano a fin de medir la diferencia entre estos. Generalmente miden las proporciones de las clasificaciones predichas contra las reales, así como estadísticos de diferencia de distribución probabilística entre los de mayor y menor riesgo.

Hipótesis

Se plantea que bajo igualdad de condiciones de inicio en los datos históricos del comportamiento de cumplimiento de pago, el modelo no paramétrico de Redes Neuronales Artificiales correctamente estructurada y con los óptimos hiperparámetros tiene un mejor desempeño de predicción de probabilidad de impago que el modelo clásico paramétrico de Regresión logística con la mejor calibración de variables, lo que ayuda a una mejor clasificación para otorgar o denegar un crédito y así, de forma subyacente una mayor redituabilidad financiera, figura 1.

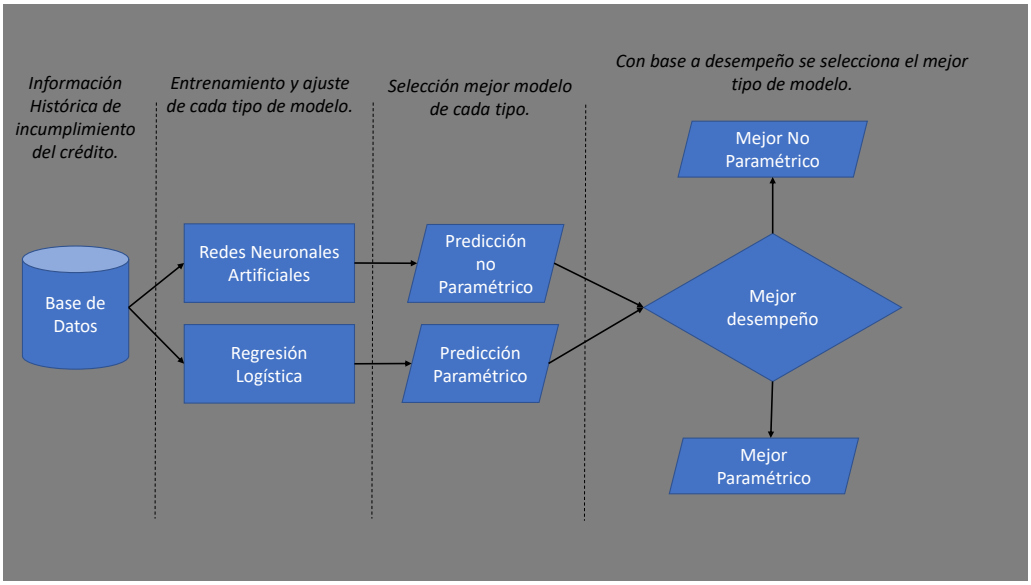


Figura 1: Flujo general para la comparativa de desempeño y selección de Modelo.

Variables

Variable dependiente : Probabilidad de incumplimiento para el pago del un crédito.

Variables independientes : Datos demográficos del solicitante y perfil como cliente ante la entidad financiera.

Variable	**Tipo de variable**	**Espacio muestra**l
Balance de cuenta	Categórica	Mayor o igual a 0.
Duración del Crédito meses	Cuantitativa	1 mes-10años.
Historial de Crédito	Categórica	Sin deuda-Cuentas crítica.
Propósito del Préstamo	Categórica	Coche(nuevo o usado), negocio, etc.
Monto del Crédito	Cuantitativa	100-10,000,000.
Monto en la cuenta de ahorro	Categórica	Mayor o igual a 0.
Tiempo empleo actual	Categórica	Desempleado, mayor igual 1 año.
% ingreso para pagar	Cuantitativa	Mayor a 0.
Sexo y Estado Civil	Categórica	Masculino, Femenino, Casado, etc.
Garantes/Deudores	Categórica	Ninguno, cliente, etc.
Años residencia actual	Cuantitativa	mayor igual 1 año.
Posesiones	Categórica	Inmuebles, automóvil, etc.
Edad	Categórica	Años cumplidos.
Otros créditos	Categórica	Banco, tiendas, ninguno.
Alojamiento	Categórica	Alquiler, propio, gratis.
No. Créditos en el banco	Cuantitativa	Mayor o igual a 0.
Trabajo	Categórica	Empleado, desempleado, etc.
No. Beneficiarios	Cuantitativa	Mayor o igual a 0.
Teléfono	Categórica	Ninguno, sí a nombre cliente.
Trabajador extrajeron	Categórica(binaria)	Sí, No.
Probabilidad de impago	Cuantitativa	(0,1).

Cuadro 1: Variables para el entrenamiento o aprendizaje del modelo.

Capítulo 1

Credit Scoring

Hoy en día la forma en que se realizan las operaciones comerciales, bancarias o de otra índole se ha diversificado al grado de que se pueden realizar de forma personal y también de forma digital a través de diversos dispositivos tales como celulares y computadoras, entre otros; esto obliga a las instituciones del sector productivo y de servicios a actualizarse para incrementar la eficacia de sus operaciones. El sector productivo puede producir más a menor tiempo y costo, y el sector de servicios, puede atender a más clientes en menor tiempo y con una mayor precisión usando las nuevas tecnologías. Particularmente, en el ramo bancario del sector financiero, es de suma importancia contar con instrumentos que aumenten la eficiencia de las operaciones, ya que esto implica menor costo o mayor ganancia en el mercado del crédito. Se entiende por eficiencia en el mercado crediticio que para cada trámite y evaluación de solicitudes se brinde una respuesta ya sea favorable o no, en el menor tiempo y con la mayor confianza posible. Por otro lado, teniendo en cuenta que en este mercado las pérdidas se tienen si los clientes no pagan según el contrato correspondiente, es de suma importancia el realizar un buen análisis o valuación al cliente para así evitar o mitigar dichos eventos.

A lo largo del tiempo las instancias financieras han utilizado diversos métodos de discriminación o clasificación de clientes mediante la estimación de la probabilidad de incumplimiento con lo suscrito; desde análisis en las incidencias en los datos hasta algoritmos de inteligencia artificial, pasando por modelos estadísticos. Estos métodos han mostrado gran utilidad para evitar pérdidas, incrementar ganancias o conseguir un mayor posicionamiento en mercado, sin embargo, todavía no hay resultados claros sobre cuál es el mejor dada la gran diversidad de métodos que se emplean.

En este trabajo revisaremos algunas de las herramienta o métodos para clasificar a los clientes potenciales basados en los modelos estadísticos más usuales utilizando datos históricos reales, con el objetivo de identificar a los que incumplirán a los suscrito, además de aplicar algunas técnicas para apoyar a muchas entidades financieras en la selección del mejor modelo. El Credit Scoring que es un método con el que se evalúa el riesgo crediticio de los solicitantes a un crédito. Para los pequeños y medianos negocios del sector financiero, el Credit Scoring constituye una herramienta muy útil y ampliamente utilizada por las ventajas que brinda para clasificar a los posibles clientes, ya que genera pronta respuesta a una solicitud de crédito con la mayor confianza y al mínimo riesgo, es decir, a una menor probabilidad de pérdida, lo que permite una diversificación del riesgo o una mayor expansión en el mercado[19].

1.1. Historia del Crédito

Como refieren Raymond A. [2] y Lyn C. [31] existen registros de operaciones crediticias que datan del 2000 a.c. en la antigua Babilonia, donde se encontró una piedra tallada en la que se pudo leer una especie de lo que ahora llamaríamos pagaré, en la que se indica que se recibe un bien en temporada de siembra y con fecha de pago al tiempo de cosecha con un interés correspondiente. También es sabido del uso de un crédito relativamente avanzado en el imperio Romano, no obstante, para los siguientes siglos fueron oscuros ya que para el crédito hubo un muy pequeño desarrollo y no fue sino hasta el siglo XII donde vuelve a tener un mejor desarrollo en Europa a través de las casas de empeño. En este mismo periodo surgió la primera legislación medieval para la Bancarrota, la cual se centraba en proteger a los acreedores en el mercado de financiamiento, dicha protección contenía desde encarcelamiento, tortura, hasta la pena de muerte, esto con la finalidad de reducir incumplimiento de pago y maximizar el valor de los activos [2]. Se tiene también registros de crédito hipotecario (Inglaterra 1187-1189) y el primer anuncio publicitario en periódico para créditos de consumo (1730, Londres).

Ya en el siglo XX, el verdadero desarrollo comenzó en el año 1920, cuando la gente en Estados Unidos comenzaba a demandar *autos de motor*, como también creció la demanda de bienes en masa, además por medio de publicidad por catálogo se invitaba a adquirir bienes y pagarlo a un plazo extendido. Así, para mediados del siglo XX explotó la demanda de crédito, observándose altos índices de crecimiento en el sector comercial, así, en 1960 nacen las tarjetas de crédito, las cuales son esenciales hoy en día para muchas operaciones. En este contexto y en orden de crecimiento ascendente se tuvo créditos personales, compras a plazos y créditos hipotecarios. En el siguiente cuadro 1.1 resume el desarrollo del crédito [2]:

Fecha.	Evento
2000 a.C.	Primer uso del crédito en Babilonia y Egipto.
1100 d.C.	Las primeras casas de empeño en Europa iniciando como casa de caridad.
1350	Continúan las casas, pero con lucro.
1536	Cargo de interés considerado aceptable por la Iglesia Protestante.
1780	Primer uso de cheque en Inglaterra.
1841	*Mercantile Agency* es la primera agencia de reporte de crédito.
1851	El primer uso de ranking crediticio para acreedores comerciales.
1856	*Singer Sewing Machine* ofrece los créditos para consumo.
1869	*Retailers Commercial Agency* es el primer buró de consumo.
1886	*Sears* lanza su catálogo.
1927	El primer buró de crédito de Alemania.
1934	Primer registro de crédito público en Alemania.
1936	*Fisher* hace uso de técnicas estadísticas para discriminar entre especies de Iris.
1941	*David D.* escribe un artículo donde sugiere la estadística para decisiones crediticias.
199?	*Henry Wells* usa Credit Scoring en la empresa *Spiegel Inc*.
1958	Se hace uso por primera vez del score aplicado por *American Investments*.
1960s	Se adopta ampliamente el Credit Scoring por compañías de tarjetas de crédito.
1966	*Credit Data Corp.* se convierte en el primer buró de crédito automático.
1978	Se implementan las primeras Scorecards para el financiamiento automotriz en África.
1984	FI desarrolla la primera scores de buró usado para preselección.
2000	Moody's KMV introduce RiskCalc para radios Scoring financieros (FRS).

Cuadro 1.1: Resumen historia del Crédito.

En el caso especial de la tarjeta de crédito, para después de la Segunda Guerra Mundial hubo un gran crecimiento en urbanización y el movimiento entre ciudades de Europa y Estados Unidos [15], lo que trajo consigo una creciente demanda del crédito para consumo, esto llevó a casas financieras y bancos a realizar grandes inversiones en este tipo de créditos. Por ello, en 1951, *Diners Club* emitió su primera tarjeta de crédito a dos centenas de sus clientes, aunque solo era posible su uso en restaurantes de Nueva York.

Estas tarjetas eran digitales y de deslizamiento para su correcto procesamiento y no contaron con la banda magnética característica sino hasta 1970. Por otro lado, en 1960 *Bank of America* lanza su Bank Americard y cuyo nombre fue cambiado en 1977 por *Visa*. Aunque en un inicio la marca solo era usada de forma local, posteriormente licitó a otros bancos alrededor del mundo, ya en 1979 nace el nombre *MasterCard*. Las tarjetas de crédito en un inicio tuvieron una tasa fija en la mayoría de los bancos emisores, no obstante, notaron la necesitad de la diversificación en los costos o precios para dicho servicio, entraron en la llamada tendencia pagos de usuario, tales como cobro por pago tardío, comisiones por sobregiro, tasa de interés no fija, etc., estos precios diversificados continuaron con American Express. Posteriormente se introduce la valuación basados en el riesgo, lo que llevó a una contracción de las tasas de interés en las tarjetas de crédito y ello tuvo como consecuencia la caída de emisores de tarjetas de un 70 a un 44 por ciento.

En 1936 el estadístico inglés *Ronald Aylmer Fisher* publicó su artículo "linear discriminant analysis" en el cual hace una clasificación entre dos tipos de Iris mediante sus magnitudes físicas. Esta idea por retomada por *David Durand*[1] quien implementó esta técnica para discriminar a los buenos y malos negocios [15]. Con el inicio de la segunda guerra mundial, todas las casas financieras comenzaron a presentar dificultades con los manejos de los créditos, ya que muchas de las personas con esta experiencia fueron reclutados por el ejército [31], por lo que algunas de las medidas fueron escribir y aplicar reglas de dedo para la gente nueva sin experiencia, no obstante esto no fue eficiente.

Por otro lado, una compañía creó su propio sistema de calificación crediticia (Credit Scoring) empleando técnicas estadísticas para desarrollar modelos de decisión, siendo esta empresa pionera. No obstante, en un principio dos principales factores derivaron en una baja adopción del Credit Scoring:

i Resistencia organizacional al uso de nuevas tecnologías para la tomar decisiones.

ii Los cálculos estadísticos y la aplicación del score eran tediosos y difícil de explicar.

En las décadas siguientes a la posguerra el implemento y desarrollo de Scorecards bajo el análisis de cuentas canceladas, lo cual brindaba información para el proceso de otorgamiento de crédito, particularmente en esta economía creciente donde la experiencia en créditos era muy escasa. Ya a mediados de los 60's muchas empresas petroleras comenzaron a presentar problemas con sus operaciones de créditos debido a fraudes, robos de tarjetas e incumplimiento de pago, como consecuencia al brindar créditos en grandes volúmenes y con comisiones bajas para un pensado mejor posicionamiento en el mercado de créditos. Ante ello, implementaron un otorgamiento más conservador y una aplicación de un Credit Scoring que llevó a la toma de mejores decisiones, logrando así una caída en las pérdidas en un 50 %, apreciando así la potencia de esta herramienta [15].

En 1962 se publica el primer trabajo académico donde relaciona la probabilidad con el comportamiento del score, lo describe como una *Cadena de Markov*, ya que a través del tiempo se mueve de un estado a otro, por ejemplo, de un estado "bueno" o "al corriente" a un estado en "malo" o "falta de pago". En 1980 se toma un cambio en la forma de el otorgamiento de los créditos en el Reino Unido, pues si bien en Estados Unidos y muchos países había tomado una forma tradicionalista a la evaluación del riesgo, Reino Unido tomó otro enfoque [2]:

I Los bancos comenzaron a otorgar créditos a *no clientes*, acción que se pensaba no era conveniente.

II Un enorme crecimiento del mercado de las tarjetas de crédito.

III Hubo un cambio de enfoque, de los grandes préstamos corporativos, donde el objetivo era evitar pérdidas, a los préstamos al consumidor donde el objetivo fue maximizar ganancias.

[1] Investigador de la *U.S. National Bureau of Economics Research*

Retomando un poco el punto III, Carlos Serrano y Begoña Gutiérrez [3] publican sobre la calificación crediticia en el mercado de préstamos personales (*Peer to Peer Lending*), donde mide la rentabilidad en relación a la tasa interna de retorno[2] o IRR[3] (*Internal Rate of Return*), donde se concluye que incluso cuando muchos clientes con alta probabilidad de no pagar también son altamente rentables con esta tasa de retorno, es decir, se enfocan en maximizar ganancias.

Cabe resaltar que algunos de los métodos estadísticos más utilizados en un principio para un modelo Credit Scoring eran:

Análisis Discriminante: Es una técnica multivariable para separar en grupos de dos o más observaciones.

Modelo lineal: Es un modelo de regresión lineal donde la variable dependiente es binaria y las variables explicativas pueden ser binarias o continuas.

Actualmente debido al gran avance en el poder de cómputo uno de los modelos más utilizados es el modelo de regresión logística ya que la solución requiere aproximaciones numéricas, además de modelos con Redes Neuronales Artificiales, entre otros que también puede requerir capacidad computacional alta. Para los 90's en adelante, el Credit Scoring se fue aplicando en otras áreas tales como: hipotecas, créditos de casa, créditos para ingresos bajos, préstamos de bajo riesgo, créditos para pequeños negocios, etc.

Adicionalmente, el Credit Scoring no estaba asociado para el riesgo en activos de grandes compañías, no obstante, en el año 2000 es lanzado el modelo Credit Scoring *RiskCalc* diseñado para calcular las frecuencias de *pérdidas esperadas* para pequeñas y medianas empresas, cabe señalar que se basaban en información de sus estados financieros. Posteriormente con esta idea, se crea la subsidiaria Moody's KMV, la cual se enfoca en proveer a prestamistas herramientas de análisis de crédito para valuación de negocios.

Una característica subyacente del Credit Scoring es su pragmatismo y empirismo, ya que el objetivo principal de éste es la predicción del riesgo y no el de explicarlo. Sin poner mucha atención en qué ramo se aplique, su principal razón es la predicción y no necesariamente el explicar por qué unos clientes pagan y otros no. El principal soporte del Credit Scoring es que su metodología es sólida y su información con la que es construido deriva de lo práctico, es decir, el sistema Credit Scoring se sustenta en la información del comportamiento de los clientes del pasado reciente similares a aquellos que aplican para un diagnóstico crediticio. Intuitivamente, muchas de las características de los clientes con la que se pretende realizar dicha predicción del riesgo tiene que estar lógicamente relacionados, por ejemplo, si tiene o no cuenta en la entidad financiera (lo que brindaría mayor información sobre su comportamiento), estabilidad financiera como son otros créditos, sus ingresos, tiempo de estar trabajando para la misma empresa o número de dependientes económicos, tipo de residencia o vivienda, estatus migratorio, etc.

Claramente las características de los clientes nos ayudan a construir un buen Credit Scoring, sin embargo, hay algunas características que están protegidas ante la ley o que no se pueden agregar a pesar de que serían de utilidad, esto debido a que pudieran existir dilemas racistas o considerarse discriminatorias, tales como género, raza o etnia, creencias religiosas, costumbres, etc. Refiere [31] que, según algunos estudios realizados, si se tomará el género como factor o variable para realizar el Credit Scoring, a más clientes femeninos se les otorgarían créditos. Lo anterior ya había llevado a un debate en los 80's acerca de la ética de los Credit Scoring incluso cuando la principal razón era el maximizar las ganancias en esta disyuntiva riegos-beneficio.

[2] En términos simples es la tasa de interés o rentabilidad que ofrece una inversión.

[3] David L. [18] define IRR: Para una inversión es la tasa de interés que es igual a la tasa anual compuesta obtenida en un ahorro con el mismo flujo de efectivo.

Buró de Crédito

Las entidades financieras principalmente tienen tres fuentes de información: La del cliente mismo, la interna en relación a su historial y la externa, esta última es precisamente la fuente de la que se hablará. Dicha fuente se conoce como *Buró de Crédito*, la cual ha ayudado a extenderse en el mundo a las financieras en los dos últimos siglos. El acceso a información compartida sobre créditos entre prestamistas fue en 1803 por The Mutual Communication Society of London en Reino Unido. Nace como un esfuerzo de diversos sastres quienes reunieron información de aquella gente que no pagaba sus cuentas [2].

Los *burós de crédito* o también llamadas *agencias de consulta de crédito* son entidades que almacenan información histórica y de comportamiento sobre personas que tienen o han tenido algún crédito en alguna entidad financiera afiliada[4] y cuya información sirve de referencia para las mismas financieras en la toma de decisiones para otorgar o negar un crédito. Basándose en la información que las mismas entidades financieras le brindan y actualizan de sus clientes que hayan tenido un crédito o que esté activo, generan un modelo Credit Scoring propio y con el cual evalúan el riesgo a cada cliente.

La necesidad de estas agencias de consulta radica en los años de mucha accesibilidad crediticia debido a su expansión y en la aparición de *malos pagadores* y defraudadores, pues solicitaban varios créditos en diversas instituciones en tiempos relativamente cortos, es decir, aparición de potenciales solicitudes fraudulentas. Por tanto, la creación de estas instituciones de consulta ayudó de forma significativa en la toma de decisiones para el mercado financiero, contribuyendo en una mejor predicción del riesgo.

1.2. ¿Qué es el Credit Scoring?

El origen etimológico la palabra *Crédito* proviene del latín *creditum* que significa “cosa confiada”, es decir, está relacionada la palabra confianza, no obstante, en el día a día comúnmente se entiende por crédito a “compre ahora y pague después”. Un Crédito es simplemente una operación financiera donde un acreedor presta cierta cantidad monetaria a otro llamado deudor y que se compromete a retornar esa cantidad en un tiempo estipulado más una cantidad adicional llamada interés.

Aquella *Confianza* referidas anteriormente conlleva de forma subyacente algunos términos: *solvencia*, que es la voluntad y posibilidad de pagar, *riesgo crediticio* que es el impacto financiero ante el cambio de la solvencia. Por otra parte, el *Scoring* se refiere a una herramienta numérica para clasificar en este contexto gente, empresas, etc., tomando en cuenta algunas cualidades o atributos y así asegurar de forma objetiva y consistente las decisiones. Para lo anterior, se sustenta de toda la información disponible integrada en un solo valor que implique alguna cualidad o de idoneidad, en nuestro contexto, la cualidad de que cumplirá el acuerdo o trato financiero. Estas cualidades son variables $X_1, X_2, ..., X_n$ y se pueden clasificar principalmente en cuantitativas o categóricas, donde se expresa una métrica numérica o una cualidad respectivamente tal como se ejemplifica en cuadro 1.2.

No existe una definición única del Credit Scoring, por ejemplo, se tienen algunas definiciones del *Credit Scoring* según algunos autores:

Raymond A. [2]: *Es el uso de modelos estadísticos que transforma información relevante a medidas numéricas que guíen en las decisiones de crédito.*

Loretta J. [19] : *Es un método estadístico usado para predecir la probabilidad que un aplicante a crédito o un crédito ya concedido no realice el pago o se trate de un fraude.*

[4] Actualmente las dos agencias de consulta de crédito más importantes en México son *Buró de Crédito* y *Círculo de Crédito*

Nombre de la Variable	Tipo de variable	Espacio muestral
X_1: Edad	Cuantitativa	De 18 a 99 años de edad.
X_2: Sexo	Categórica (binaria)	Masculino o Femenino.
X_3: Historial de Crédito	Categórica	Sin deuda-Cuentas crítica.
X_4: Monto del Crédito	Cuantitativa	100-1,000,000.
X_5: Duración del Crédito	Cuantitativa	1 mes-10años.
$\vdots$	$\vdots$	$\vdots$
X_{n-1}: Propósito del Cliente	Categórica	Coche (nuevo o usado), negocio, etc.
X_n: Alojamiento	Cuantitativa	Alquiler, propio, gratis.
Y: Clase Cliente	Categórica (binaria)	Bueno o Malo.

Cuadro 1.2: Ejemplo de posibles características del subscriptor a un crédito.

Lyn C. [31]: *Es el conjunto de modelos de decisión y sus técnicas subyacentes que ayudan a financieros en el otorgamiento de créditos de consumo. Estas técnicas deciden quienes obtendrá el crédito, el monto y qué estrategias operacionales mejorarán la rentabilidad de los deudores a los prestamistas.*

De estas definiciones se puede observar las siguientes palabras clave: *Riesgo*, *modelo*, *estadística*, *herramienta*, *información* y *decisión*. Así, podemos tener la idea de modelos basados en datos que provean predicciones para tomar de decisiones financieras, es decir, podemos usar la siguiente definición: *Son modelos estadísticos que permiten evaluar de forma óptima el riesgo o probabilidad de incumplimiento de pago o fraude de un solicitante o un cliente de un crédito para una mejor toma de decisiones bajo los objetivos de una entidad financiera.*

1.2.1. Principales ventajas

El Credit Scoring cambió la relación convencional crédito-criterio humano, ya que se tenía un contacto más personal, por ejemplo, en el casos de los bancos, para que un cliente tuviera acceso a un crédito generalmente debía contar con cierta antigüedad, que el personal conociera de forma más cercana al cliente, lo cual daba la idea de un riesgo pequeño, no obstante, la eficiencia no era para nada aceptable, entonces, el *Credit Scoring* trajo consigo grandes ventajas [2]:

- Mayor objetividad en las decisiones.
- Automatización implicando grandes volúmenes.
- Mayor penetración en mercado.
- Menores costos.
- Crecimiento de las financieras.
- Exploración de nuevos mercados.

1.2.2. Tipos

En el ámbito de aplicación de un Credit Scoring y recordando que la intención principal es dar un valor al riesgo subyacente, como lo señala Nieto S. [23] y Raymond A. [2] algunos tipos importantes son:

Aplication Score: También llamado Score de originación, es usado para la evaluación de un solicitante a un nuevo producto y esta toma en cuenta información del cliente; histórico de transacciones y buró de crédito.

Behavioral Score: Este Score es de comportamiento, va enfocado con una mejor dirección de una cuenta individual, características como autorizaciones, manejo por sobregiro, establecimiento de límites, etc., es decir, predice la probabilidad de incumplimiento de los clientes que ya cuentan con un producto financiero.

Collections Score: Este es usado por área de cobranza, mediante el proceso de recuperación ya que incorpora el comportamiento de los clientes ante ausencia de pagos y buró de crédito.

Customer Score: Este combina el Behavioral Score sobre varias cuentas y es usado para el mejoramiento de este y aplicación de ventas cruzadas también sobre los clientes ya existentes.

Bureau Score: Es un Score que es provisto por un *Buró de Crédito* y normalmente resume la información que se les brinda por entidades financieras para predecir fraudes o bancarrota.

1.3. Flujo de desarrollo

El flujo para el desarrollo de un modelo Credit Scoring ilustra los pasos, conceptos directos y subyacentes que requiere. Mediante el diagrama de la figura 5.28 se sintetiza el flujo de desarrollo *flujo Credit Scoring* que a continuación se expondrá. Este flujo lo propone Raymond A. ([2]), donde toma en cuenta cuatro aspectos principales: *ENTORNO, APLICACIÓN, DESARROLLO y MODELADO*.

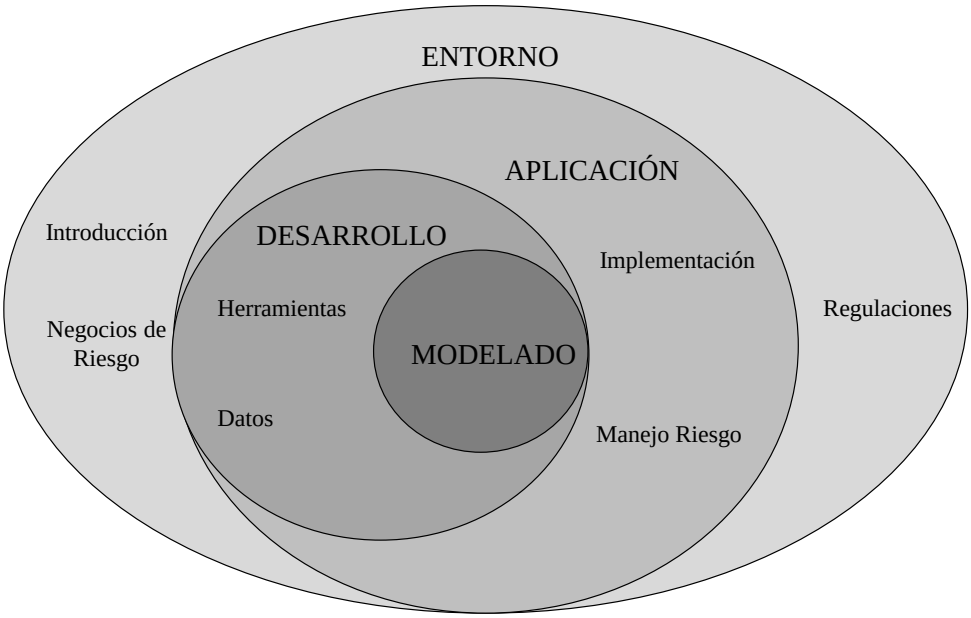


Figura 1.1: Diagrama Credit Scoring.

El *ENTORNO* está compuesto por la *INTRODUCCIÓN*, donde se menciona la historia del Crédito y del Credit Scoring, también por *Negocios de Riesgo* en el cual justifica el por qué, el dónde y cómo será usado en el ámbito de negocios y por último las *Regulaciones*, es decir, el entorno legislativo y su impacto en los clientes tales como protección de datos, no discriminación, etc.

En el rubro de *DESARROLLO*, ya provisto de la información del *ENTORNO* se basa en dos elementos principales: Herramientas y Datos. El primero se compone por resultados teóricos matemáticos, tales como algoritmos, estadística, métricas de predicción y evaluación, etc. Para el segundo, es la parte fundamental

ya que los Datos son indispensables para la estimación de los parámetros del modelo a saber.

Con lo anterior, se llega al aspecto del *Modelado*, es decir, la parte del proceso de desarrollo del modelo Credit Scoring, este proceso es en el que nos enfocaremos en el capítulo ***Aplicación Credit Scoring***. El proceso está compuesto por:

- Análisis Exploratorio de los Datos (EDA).
- Selección y transformación de las variables.
- Selección y entrenamiento del modelo.
- Ajuste y validación del modelo.
- Implementación o eliminación del modelo.

Después del *DESARROLLO* se procede a la *APLICACIÓN*, donde la *IMPLEMENTACIÓN* del modelo Credit Scoring conlleva un monitoreo y control, por otra parte, para el Manejo de Riesgos son consideradas políticas de marketing para ofertar los créditos. Por tanto, este flujo mimetiza la forma en que se construye un modelo de Credit Scoring, atendiendo limitaciones del mercado financiero, leyes, y riegos hasta su implementación real en el crédito.

1.4. Modelo Credit Scoring

Como ya se ha mencionado los modelos Credit Scoring son utilizados para evaluar la credibilidad de un evento futuro basado en nuestra experiencia, muchos derivan de su información histórica. El uso de nuevas tecnologías para la automatización en la combinación de scores y estrategias de mercado para la toma de decisiones de negocios, la cual reduce de forma sustancial los costos, esto es, el *Business Intelligence*. Por tanto, a partir del conjunto de datos que contiene la información histórica de las variables de cada uno de los clientes $X_1, X_2, ..., X_p$, se puede construir un modelo Credit Scoring $f\left(X_1, X_2, ..., X_p\right)$ dependiente de las características o atributos estos clientes, de esta manera, se puede clasificar a cada futuro cliente para otorgar o rechazar el crédito dado su riesgo estimado.

Para exponer la dinámica de un crédito de forma resumida supongamos que se evaluará un cliente potencial con vector de características o atributos $\boldsymbol{x} = \left(X_1 = x_1, X_2 = x_2, ..., X_p = x_p\right) \in \mathbb{R}^p$ y sea $f(\boldsymbol{x})$ un modelo *Credit Scoring*, definamos la variable Y como la categoría del cliente, 0: Bueno y 1: Malo, entendiéndose que se aprobará o rechazará el otorgamiento respectivamente. Por otro lado, el cliente con categoría Y tiene asociado un riesgo de incumplimiento $\theta \in (0, 1)$, el cual será estimado mediante el modelo como se muestra en la ecuación (1.1).

$$f\left(X_1, X_2, ..., X_p\right) = \hat{\theta} \in (0, 1)\,. \tag{1.1}$$

Donde esta última estimación tiene la interpretación,

$$\hat{\theta} = \hat{P}\left(Y = 1|\boldsymbol{x}\right). \tag{1.2}$$

Por otra parte, para ciertas condiciones de entorno, estrategias de mercado y políticas de la financiera se fija un riesgo dispuesto $\theta_0 \in (0, 1)$, también llamado *cut-off* o *corte de decisión*, de esta manera, se puede tomar la decisión de otorgamiento mediante:

$$I_{\theta_0}\left(\hat{\theta}\right) = \hat{Y} = \underset{(\theta_0,1)}{\mathbb{1}}\left(\hat{\theta}\right) = \begin{cases} 1 & \text{Si } \theta_0 < \hat{\theta} \\ 0 & \text{Otro caso .} \end{cases} \tag{1.3}$$

Por tanto, mediante (1.3) se ha asignado una clasificación al solicitante, es decir, una estimación de la clasificación real y que a futuro se sabrá si ésta fue correcta o errónea, de ser acertada traerá beneficios y en caso contrario, implicará un costo o pérdida para la financiera.

Claramente un buen modelo brindará o estimará un riesgo muy próximo a su clasificación real, así, el entrenamiento y selección del mejor modelo es de suma importancia. En capítulos posteriores se revisarán las métricas de selección y proceso de entrenamiento para implementar o desechar modelos Credit Scoring en sus dos enfoques para este trabajo: Paramétrico y no paramétrico.

1.5. Modelos paramétricos y no paramétricos

Como se ha mencionado los modelos Credit Scoring derivan de una gran cantidad de técnicas estadísticas dependiendo si se conoce o no de la forma funcional de la distribución de la variable dependiente Y, se clasifican en *paramétricos* y *no paramétricos*. En esta parte se tratará de mencionar de forma resumida qué son los modelos paramétricos y no paramétricos, así como su diferencia, ya que se aplicarán se evaluará su desempeño en el contexto Credit Scoring.

Paramétricos: Estos parten de una función de distribución conocida y el problema principal es la estimación del parámetro que mejor se ajuste a los datos, aunque pueden ser muy sensible ante violación de sus hipótesis. Para una variable aleatoria Y tal que $Y \sim F$ donde F es una función de distribución que está contenida en una familia $\mathscr{F}_{\Theta}$ de distribuciones indexada, entonces Y sigue un modelo paramétrico si

$$F \in \mathscr{F}_{\Theta} = \{F_\theta : \theta \in \Theta \subset \mathbb{R}^{\,k}\}. \tag{1.4}$$

Algunos ejemplos de modelos paramétricos pueden ser el *modelo logístico* y *análisis discriminante*, etc., éste primero se implementará en este trabajo.

No Paramétricos: También llamados de distribución libre debido a que no están sujetos a una forma funcional. Estos permiten formular una aproximación de los parámetros de una función y no de una función conocida. Sea Y una variable aleatoria tal que $Y \sim F$ donde F es una función de distribución, ésta sigue un modelo no paramétrico si su distribución F solo aplican restricciones de regularidad y que no se puede indexar por un parámetro θ. Algunos ejemplos de método son Redes Neuronales Artificiales (ANN), Árboles de clasificación y regresión, etc. Para este trabajo se utilizarán las ANN.

Capítulo 2

Modelo paramétrico

Uno de los modelos mas usados hoy en día para estimar la probabilidad que un cliente potencial pertenezca al grupo de los clientes malos es el Modelo *Logit* o *Logístico* que al final es un modelo de elección binaria en el que la variable dependiente tomará valores 0 ó 1. El modelo *Logit* es un modelo multivariante paramétrico en el que existen variables categóricas tanto en la variable dependiente como en las explicativas, tiene la ventaja de no plantear ninguna restricción de distribución de las variables explicativas, por ejemplo, normalidad, comportamiento de varianzas o aleatoriedad. Este modelo utiliza la estimación de parámetros por el método de máxima verosimilitud

2.1. Modelo Logístico

Ya que el Modelo Logístico se enfoca en la estimación de la probabilidad de pertenecer a cierta clase $p \in (0, 1)$ para luego discriminar de la siguiente forma: Si $0.5 < \hat{p}$ entonces se clasifica como 1 y si $\hat{p} < 0.5$, se clasifica como 0. Esto es, un valor default de $\theta_0 = 0.5$, no obstante, en general no será fijo.

Dado el conjunto de datos, un muestra de $\boldsymbol{x}_i^T \in \mathbb{R}^p$ tamaño n y tiene asociada su variable de respuesta $Y_i = j$, para $j = \{0, 1\}$, con $i = 1, 2, ..., n$.

Sea $p_i = P(Y_i = 1|\boldsymbol{x}_i)$ y sea la función de cociente de probabilidades (liga):

$$\eta(p_i) = \log\left(\frac{p_i}{1 - p_i}\right). \tag{2.1}$$

Esta función (2.1) se le conoce como la transformación *Logit*, que representa una escala logarítmica la diferencia de pertenecer a ambas poblaciones, de esta manera se relaciona de forma lineal facilitando su posterior interpretación

$$\eta(p_i) = \beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}. \tag{2.2}$$

Donde $\boldsymbol{\beta}^T = (\beta_1, \beta_2, ..., \beta_p)$ son los parámetros fijos y desconocidos que se deberán estimar a través de los datos por el método de máxima verosimilitud. Así, obteniendo la inversa de η se llega a la llamada función *sigmoide* [9]

$$\sigma\left(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}\right) = p_i = \frac{\exp\left(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}\right)}{1 + \exp\left(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}\right)} = \frac{1}{1 + \exp^{-\left(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}\right)}}. \tag{2.3}$$

Por otro lado, la función de probabilidad de Y_i para cada $i = 1, 2, ..., n$ es:

$$P(Y_i = j_i) = p_i^{j_i}(1 - p_i)^{1-j_i} \text{ para } j_i = 0, 1. \tag{2.4}$$

Por lo que la conjunta es

$$P(\boldsymbol{Y}) = P(Y_1, Y_2, ..., Y_n) = \prod_{i=1}^{n} p_i^{j_i}(1 - p_i)^{1-j_i} \text{ para } j_i = 0, 1. \tag{2.5}$$

Obteniendo el logaritmos de (2.5), además utilizando las ecuaciones (2.1) y (2.2) se tiene que

$$\begin{aligned}
\log P(\boldsymbol{Y}) &= \log\left(\prod_{i=1}^{n} p_i^{j_i}(1 - p_i)^{1-j_i}\right) \\
&= \sum_{i=1}^{n}\left(\log(p^{j_i})_i + \log(1 - p_i) - j_i \log(1 - p_i)\right) \\
&= \sum_{i=1}^{n}\left(\log(1 - p_i) + j_i \log\left(\frac{p_i}{1 - p_i}\right)\right) \\
&= \sum_{i=1}^{n}\left(\log\left(1 - \frac{\exp\left(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}\right)}{1 + \exp\left(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}\right)}\right) + j_i\left(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}\right)\right) \\
&= -\sum_{i=1}^{n}\log\left(1 + \exp\left(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}\right)\right) + \sum_{i=1}^{n} j_i\left(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}\right) \\
&= -\sum_{i=1}^{n}\log\left(1 + \exp \boldsymbol{z}_i^T\boldsymbol{\beta}_*\right) + \sum_{i=1}^{n} j_i \boldsymbol{z}_i^T\boldsymbol{\beta}_* \\
&= \log P(\boldsymbol{\beta}_*) \\
&= L(\boldsymbol{\beta}_*).
\end{aligned}$$

Donde $\boldsymbol{\beta}_*^T = (\beta_0, \beta_1, ..., \beta_p)$ y $\boldsymbol{z}_i^T = (1, x_1, x_2 ..., x_p)$. Entonces para maximizar esta última ecuación de verosimilitud en función del vector de parámetros (estimadores de máxima verosimilitud),

$$\frac{\partial L(\boldsymbol{\beta}_*)}{\partial \boldsymbol{\beta}_*} = -\sum_{i=1}^{n}\left(\frac{\exp \boldsymbol{z}_i^T\boldsymbol{\beta}_*}{1 + \exp \boldsymbol{z}_i^T\boldsymbol{\beta}_*}\right) + \sum_{i=1}^{n} j_i \boldsymbol{z}_i. \tag{2.6}$$

Igualando a cero la ecuación (2.6) y hallando aquel vector $\hat{\boldsymbol{\beta}}_*$ que pueda satisfacer dicha igualdad, claramente se tiene que hallar por medio de una aproximación numérica véase en [25] el método de *Newton-Raphson*. Por tanto, se satisface que

$$\frac{\partial L\left(\hat{\boldsymbol{\beta}}_*\right)}{\partial \boldsymbol{\beta}_*} = 0. \tag{2.7}$$

y se valida que es un máximo debido a $\dfrac{\partial^2 L\left(\hat{\boldsymbol{\beta}}_*\right)}{\partial^2 \boldsymbol{\beta}_*} < 0.$

Por lo que haciendo uso a la ecuación (2.3) se puede estimar la probabilidad p_i.

$$\hat{p}_i = \frac{1}{1 + \exp^{-\boldsymbol{z}_i^T \hat{\boldsymbol{\beta}}_*}}. \tag{2.8}$$

2.2. Aplicación en Credit Scoring

De esta manera, utilizando la información de los clientes en registro de alguna financiera, se puede aplicar el proceso de construcción de un modelo Logit, dado que la ecuación (2.6) proporciona un método para estimar su vector de parámetros $\boldsymbol{\beta}$, β_0 y con ayuda del resultado (2.8) se puede estimar el riesgo asociado θ para cada cliente nuevo, es decir,

$$\hat{p} = \hat{\theta} = \hat{P}\left(Y = 1 | \boldsymbol{x}\right) = \frac{1}{1 + \exp^{-\left(\hat{\beta}_0 + \boldsymbol{x}^T \hat{\boldsymbol{\beta}}\right)}}. \tag{2.9}$$

Para posteriormente tomar la decisión aplicando la ecuación $I\left(\hat{p}\right)$ de (1.3) y así rechazar u otorgar el crédito $(\hat{Y})$ dado un *cutt-off* θ_0 óptimo.

$$I_{\theta_0}\left(\hat{p}\right) = \begin{cases} \hat{Y} = 0 & \text{Si } \hat{p} \leq \theta_0 \\ \hat{Y} = 1 & \text{Si } \theta_0 < \hat{p}\ . \end{cases} \tag{2.10}$$

En el capítulo 5 se aplicará de forma práctica este proceso siguiendo un flujo de selección de modelo, un correcto entrenamiento, la información más relevante de una base de datos real y comparando su desempeño con un modelo no paramétrico mediante métricas de clasificación.

Capítulo 3

Modelo no paramétrico

Debido al gran avance tecnológico en el procesamiento y capacidad computacional la Inteligencia Artificial (*IA*) después de casi 60 años ha alcanzado a la industria en diversos ámbitos como el procesamiento y clasificación de imágenes, detección de emociones, procesamiento natural del lenguaje, robots de servicio, predicción de contenido streaming o adquisición de productos entre muchas aplicaciones más. Gracias al desarrollo de esta disciplina se han podido resolver una gran cantidad de tareas complejas de forma automatizada y más eficiente, lo que trajo consigo muchos beneficio tales como mayor producción, mayor penetración en los mercados, mejor servicio, experiencia y eficiencia en los sectores de servicio y producción.

Como en muchas otras áreas, algunas ramas de la IA pueden ser aplicadas para el desarrollo de modelos Credit Scoring. La intención de este capítulo es presentar una herramienta de la *IA*, las Redes Neuronales Artificiales (ANN), estas pertenecen a los modelos de Aprendizaje Supervisado de los algoritmos de Aprendizaje de la Máquina, Aprendizaje Máquina o Machine Learning (ML). Centraremos nuestra atención en Modelos Credit Scoring basados en estructuras de ANN profundas o Deep Learning, tomando en cuenta teoría básica de su funcionamiento, hiperparámetros y optimización, así como identificar y tratar el sobreajuste para la selección del mejor modelo.

3.1. Acerca de la IA

Generalmente existen diversas definiciones de *IA* dependiendo en muchas ocasiones del marco de aplicación, por ejemplo, Shapiro, S. C. [29] define la Inteligencia artificial como:

"... un campo de la ciencia computacional e ingeniería ocupado con el entendimiento computacional de lo que comúnmente llamamos comportamiento inteligente, y con la creación de artefactos que exhiben dicho comportamiento".

O también de forma más sintética por Ertel W. [7]:

"El objetivo de la Inteligencia Artificial es desarrollar máquinas que se comportaran como si fueran inteligentes".

No obstante, una que es más adecuada en la dirección del Credit Scoring es la que define [7] Elaine Rich:

"... es el estudio de cómo hacer que las computadoras hagan las cosas que en este momento las personas son mejores".

Bajo este enfoque se trabajará con una rama de la *IA* que nos ayude a que las computadoras brinde de forma automatizada y eficiente un Score de Crédito para así clasificar a los clientes solicitantes de un Crédito entre buenos y malos con la mayor confianza posible.

La inteligencia artificial se divide [29] entre muchas otras ramas, por ejemplo: Lenguaje natural de Procesamiento (NLP[1]), Razonamiento, Visión, Speech Recognition (Texto-voz), robótica y Machine Learning, donde la rama de *ML* es de mayor interés para este trabajo debido a su enfoque de aprendizaje supervisado basado en los datos con un nivel mucho mayor que con una programación explícita, véase la figura 3.1.

El *ML* se basa en algoritmos que de forma iterativa ajusta sus parámetros a partir de los datos de entrada para así mejorar su desempeño en predicciones para nuevos datos. El *ML* se puede clasificar [30] en Aprendizaje supervisado, no supervisado o, por refuerzo, tal como se muestra a continuación [9].

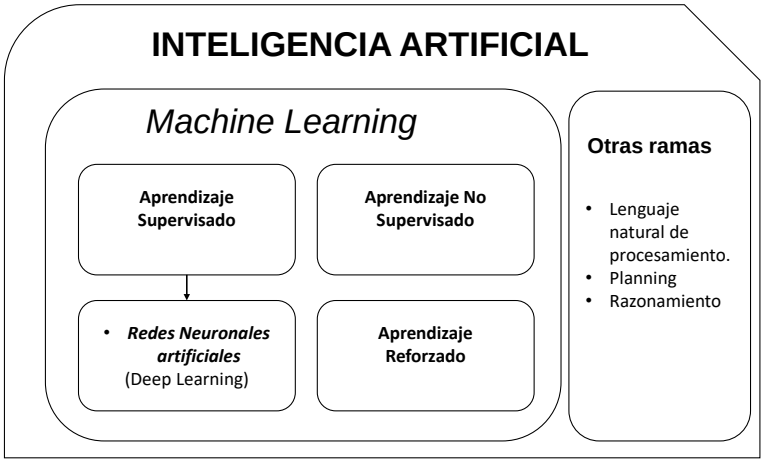


Figura 3.1: Campos de la inteligencia artificial.

Aprendizaje Supervisado: Dada la variable respuesta el algoritmo tiene que aprender a dar ese resultado. Por ejemplo,

- Máquinas de soporte de vectores (SVM[2])
- Naive Bayes
- Regresión Logística
- K-Vecinos más cercanos (k-NN[3])

Aprendizaje no Supervisado: En esta caso el algoritmo no tiene las variables y genera grupos a partir de características similares.

- K-medias
- Clusterización

Aprendizaje por Refuerzo: El algoritmo aprende dadas una serie de acciones en su entorno.

[1]Por su siglas en Inglés Natural Language Processing
[2]Por su siglas en inglés Support Vector Machine
[3]k-Nearest Neighbor

Aprendizaje profundo o Deep Learning: Es una estructura de redes neuronales artificiales multicapa usadas para encontrar relaciones entre los datos de entrada y los resultados.

- **Redes Neuronales Artificiales**
- Redes Neuronales Convolucionales
- Redes Neuronales Recurrentes
- Redes de Creencia Profunda

3.2. Aprendizaje Supervisado

La tarea principal del aprendizaje supervisado es "aprender" de la información brindada o con la que se cuenta mediante un proceso de entrenamiento, partiendo de los datos de entrada (variables independientes $\boldsymbol{x}$) mapeando este dato ($\hat{\boldsymbol{Y}}$) y contrastándolo con el resultado deseado o real ($\boldsymbol{Y}$), validando su capacidad para brindar el resultado correcto (ecuación (3.1)) ([13]). Esto se puede visualizar en la figura 3.2.

$$\hat{\boldsymbol{Y}} = f\left(\boldsymbol{x}\right) \rightarrow Y \ . \tag{3.1}$$

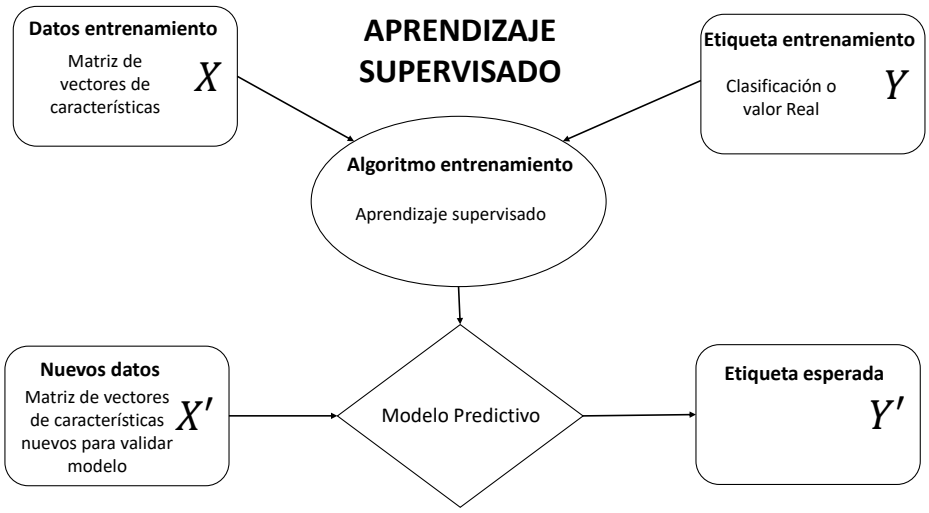


Figura 3.2: Diagrama de aprendizaje supervisado.

En resumen, el proceso de aprendizaje en los algoritmos de *ML* supervisados ocurre al reajustar los parámetros de forma iterativa del modelo, ya que dada una métrica ó función de costo J que evalúa el error entre el valor de predicción y el valor real a fin que este sea lo mínimo posible al mismo tiempo que no pierda su capacidad de predicción general, de no ocurrir esto último, sólo se habrá "memorizado" los resultados esperados para los datos de entrenamiento y dicho modelo será obsoleto.

3.3. Redes Neuronales Artificiales

En 1943, por McCulloch y Pitts [22] con sus conocimientos en neurología desarrollaron modelos de redes neuronales artificiales tipo binario, lo que posteriormente y siguiendo esta idea en 1958, desarrolló un modelo sencillo fundamentado en la corrección del error, no obstante, tenía muchas limitaciones. En 1986 Rumerlhard, Hilton y Williams brindan una idea innovadora para estas redes con el algoritmo de retropropagación[4], el cual a partir de los errores de salida de la red se reajustan los pesos pudiendo llevar a cabo tareas de clasificación sencillas. Se puede definir las Redes Neuronales Artificiales según D. Miller J. [20] como

"*... una familia de modelos inspirados por la redes neuronales biológicas y que son usadas para estimar o aproximar funciones que pueden depender de un número grande de entradas y que son generalmente desconocidas*".

En concreto, podemos pensar en estas redes como modelos matemáticos computacionales de datos que conceptualizan el cómo trabaja el cerebro humano. A su vez, éstas están compuestas o definidas por elementos [20] llamados hiperparámetros [1]. Por tanto, una ANN es una red en paralelo de neuronas artificiales interconectadas, las cuales reciben diversas entradas de otras neuronas y que dan salida cuando éstas se activan. Dicha activación es controlada por diversas funciones matemáticas [16].

3.3.1. Estructura ANN

A los parámetros que describen la estructura de una ANN se les conoce como *hiperparámetros* y tienen una gran importancia en el desempeño del proceso de aprendizaje. A diferencia de los parámetros de aprendizaje de la red, estos no se entrenan de los datos, por tanto, su definición y afinamiento para un mejor desempeño del modelo es distinto. Estos hiperparámetros son el *número de neuronas* en cada capa, el *número de capas ocultas*, el tipo de *función de activación*, el *optimizador*, la *tasa de aprendizaje*, el *número de épocas* o *epochs*, etc.

Neuronas: Calculan un valor aplicando su *función de activación* (cada neurona tiene su función de activación F) que es una suma ponderada de neuronas de capas anteriores o capa de entrada.

Capas: Se define como una colección de neuronas operando juntas a cierta profundidad. Estas capas se clasifican en

Capa de entrada También llamada capa visible y esta es por donde ingresan los datos.

Capas Ocultas Es donde se procesa la información de entrenamiento.

Capa de Salida Es donde devuelve la clasificación binaria o múltiple, como también alguna estimación requerida pudiendo ser también multidimensional.

Conexiones: Las neuronas entre capas adyacentes dentro de una estructura de Red Neuronal Artificial pueden estar conectadas todas con todas, algunas o una, etc.

Función de Activación: Aquella función F que escala los valores de paso por la neurona, comúnmente dicha escala se encuentra en los intervalos $(0, 1)$, $[-1, 1]$, $[0, \infty)$, etc. Por ejemplo, las ecuaciones (3.2), (3.3), (3.4),(3.5),(3.6),(3.7), (3.8) y (3.9) son funciones de activación comúnmente utilizadas, y en figura 3.3 se muestran sus respectivas gráficas.

[4]Back-propagation algorithm

A continuación, se presentan algunos ejemplos de las funciones de activación en capas más comunes [11] así como algunas gráficas.

Lineal: $\mathbb{R} \rightarrow \mathbb{R}$

$$F(x) = x \ . \tag{3.2}$$

Step ó Escalón: $\mathbb{R} \rightarrow \{0, 1\}$

$$F(x) = \begin{Bmatrix} 1 & \text{si } 0.5 \leq x \\ 0 & \text{Otro caso} \end{Bmatrix} \ . \tag{3.3}$$

Sigmoide: $\mathbb{R} \rightarrow (0, 1)$

$$F(x) = \frac{1}{1 + \exp^{-x}} \ . \tag{3.4}$$

Tangente Hiperbólico: $\mathbb{R} \rightarrow (-1, 1)$

$$F(x) = tanh(x) \ . \tag{3.5}$$

ReLu[5]: $\mathbb{R} \rightarrow [0, \infty)$

$$F(x) = max(0, x) \ . \tag{3.6}$$

Leaky ReLu: $\mathbb{R} \rightarrow (-\infty, \infty)$

$$F(x) = max(\alpha x, x) \quad \text{para } 0 < \alpha < 1 \ . \tag{3.7}$$

ELU[6]: $\mathbb{R} \rightarrow (-\infty, \infty)$

$$F(x) = \begin{Bmatrix} \alpha(\exp^x - 1) & \text{si } x < 0 \text{ para } 0 < \alpha \leq 1 \\ x & \text{Otro caso} \end{Bmatrix} \ . \tag{3.8}$$

SELU[7]: $\mathbb{R} \rightarrow (-\infty, \infty)$

$$F(x) = \lambda \begin{Bmatrix} \alpha(\exp^x - 1) & \text{si } x < 0 \text{ para } 0 < \alpha \leq 1 \\ x & \text{Otro caso} \end{Bmatrix} \ . \tag{3.9}$$

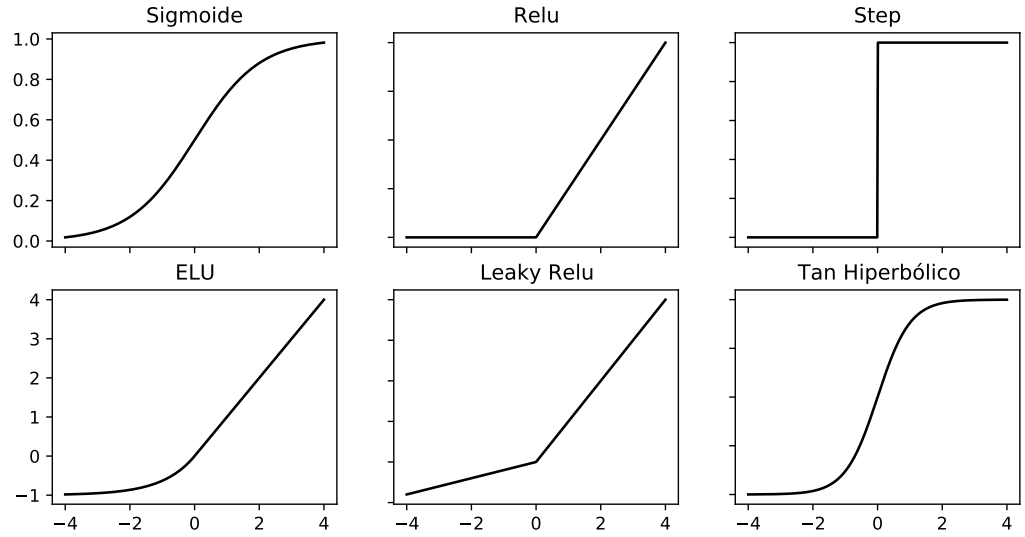


Figura 3.3: Gráficas de algunas Funciones de activación más usuales en estructuras de ANN.

[5] Rectified Linear Unit
[6] Exponential Linear Unit
[7] Scalar Exponential Linear Unit

En general, para el caso de una clasificación binaria supervisada, sea $f : \mathbb{R}^p \rightarrow \{0, 1\}$, supongamos un patrón de entrada $\boldsymbol{x}^T = (x_1, x_2, ..., x_p)$ y con salida esperada $z \in \{0, 1\}$ (ecuación 3.10). Cabe resaltar que se ha supuesto que existen patrones que permiten la clasificación para cada $\boldsymbol{x}$ entrantante desconocido.

$$f(\boldsymbol{x}) = z \ . \tag{3.10}$$

La información que se cuenta de esta función son las n parejas de entrenamiento $\left\{\boldsymbol{x}^1, z^1\right\}, ..., \{\boldsymbol{x}^n, z^n\}$, con $\boldsymbol{x}^i \in \mathbb{R}^p$ y $z^i \in \{0, 1\}$ para todo $i = 1, 2, ..., n$, es decir, un vector de valores con su respectiva clasificación real. Por ello, mediante una estructura la red neuronal artificial tendrá la tarea de aproximar esta función f desconocida a través de los patrones de entrenamiento para llegar a poder obtener un buen y óptimo desempeño en la clasificación para cada $\boldsymbol{x}$, dichas métricas se mencionarán posteriormente.

3.3.2. Perceptrón Simple y Multicapa

Una de las estructuras básicas de las redes neuronales es el *Perceptrón simple*, el cual se compone de una única capa, que mediante una regla de aprendizaje se ajustan los pesos w_i sinápticos de tal manera que se cumpla (3.10). No obstante, muchas clasificaciones no son linealmente separables (cuadro 3.2) por lo que esta estructura no es óptima o viable en tales casos ya que el perceptrón no converge en un número finito de iteraciones [22] debido a que al ser puntos no linealmente separables excede la complejidad que puede tratar esta estructura.

En la figura 3.4 se presenta un diagrama del perceptrón simple con una única neurona, en donde se espera que para un vector de datos $\boldsymbol{x}^T = (1, x_1, x_2, ..., x_p)$ con clasificación real Y_x, y mediante un proceso de entrenamiento sean hallados los pesos sinápticos $\boldsymbol{w}^T = (w_0, w_1, ..., w_p)$ de tal manera que se tenga un modelo $\hat{f}$ que aproxime a f, es decir, como se muestra en la ecuación (3.11) y (3.12):

$$\hat{f}(\boldsymbol{x}) = \hat{z} = Y_x \ . \tag{3.11}$$

$$\hat{f}(\boldsymbol{x}) = F\left(\boldsymbol{x}^T \boldsymbol{w}\right) = F\left(\sum_{i=0}^{p} x_i w_i\right) = F\left(u\right) = \hat{\boldsymbol{z}} = Y_x \ . \tag{3.12}$$

En el *Anexo A* se muestra el código en lenguaje *Python* de un algoritmo de perceptrón simple para la separación de un conjunto de datos pertenecientes a diferentes clases y que sí son linealmente separables, por lo cual se puede utilizar un perceptrón simple. Para ello se utilizó la *función signo* como función de activación y la *Regla de aprendizaje Delta* como optimizador para ajustar los pesos, dichos puntos se muestran en el cuadro 3.1. Mediante la ejecución de dicho algoritmo se pudo hallar los pesos sinápticos correspondientes a una línea que separe los puntos de forma óptima, dicha recta y puntos se visualizan en el cuadro 3.2 inciso a).

Regla de aprendizaje Delta
$w_j(k+1) = w_j + \eta\left(\hat{z} - z\right) x_j\left(k\right) \ .$
La actualización del j-ésimo peso en la k-ésima iteración con una tasa de aprendizaje η. Esta tasa controla la cuantía con la cual se modifican los pesos.

Cuadro 3.1: Regla de aprendizaje o reajuste de valor de los pesos sinápticos.

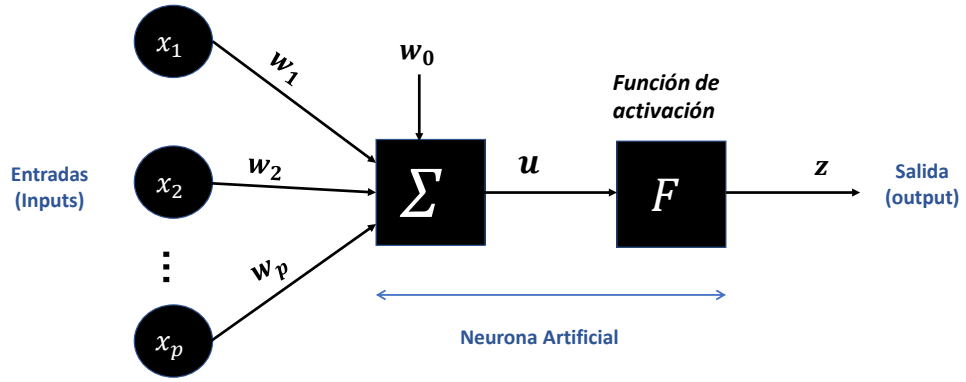
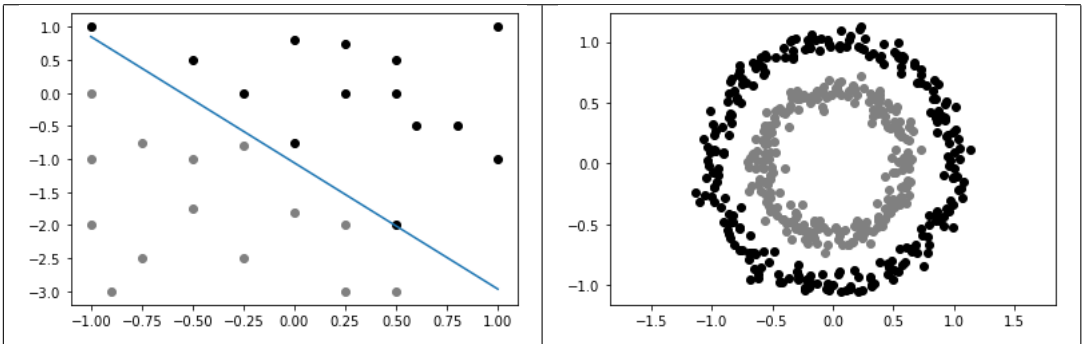


Figura 3.4: Esquema Perceptrón simple.



a) Linealmente separable.	b) No linealmente separable.

Cuadro 3.2: Separabilidad de $\boldsymbol{x}$.

Para el caso del Perceptrón Multicapa, esta estructura de ANN se compone de varias capas y que no necesariamente tiene la misma cantidad de neuronas en cada capa. La ventaja de este tipo de estructura es que se puede llevar a cabo una clasificación con un mayor grado de complejidad, ya que con este modelo se puede superar la restricción o problema de no ser el conjunto de datos linealmente separables tal como se muestra en el cuadro 3.2 inciso b). Claramente dicho conjunto de datos no se puede separar por una sola recta, sino por varias para obtener un menor error en la clasificación.

Debe notarse que estas primeras arquitecturas de ANN son de tipo *alimentación hacia adelante*[8], es decir, este tipo red funciona y reconoce patrones más complejos, ya que cada capa mantiene una conexión con la capa consecutiva hacia adelante [5]: De la *capa de entrada→capas ocultas→capa de salida*. Los pesos se ajustan por iteraciones dada una regla de aprendizaje.

En la figura 3.5 se tiene un diagrama de un modelo con una estructura de perceptrón multicapa que cuenta con los siguientes hiperparámetros:

- Una capa visible o, de entrada.
- Dos capas ocultas con r y s neuronas respectivamente.
- Función de activación F.
- Regla de aprendizaje *Delta*.
- Una capa de Salida con una sola neurona.
- Conexión de cada neurona con todas las neuronas de la capa adyacente [9].

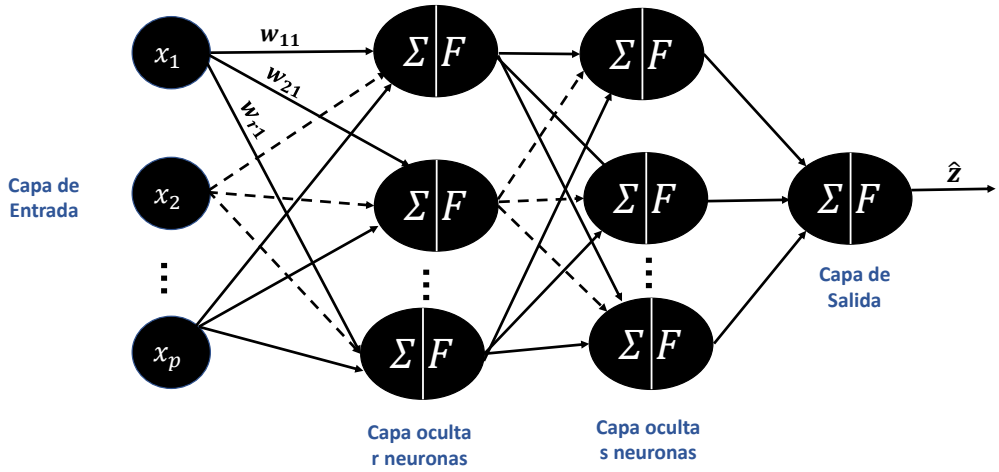


Figura 3.5: Esquema de la estructura del Perceptrón Multicapa.

Las estructuras anteriores aún tienen diversas limitaciones que se van a superar con mejores algoritmos de optimización para el aprendizaje del error. Para completar una arquitectura básica de una ANN [5], además de los elementos mencionados anteriormente, se requieren dos elementos importantes:

[8] Feedforward ANN
[9] Dense Connection

Función de pérdida o costo Es la función que define la métrica de retroalimentación basada en el error entre las predicciones obtenidas y el valor o clase correcta con la finalidad que la ANN aprenda de esta cuantificación y se minimice. Algunas de las funciones más usadas son las siguientes.

- *Error cuadrático medio*: Esta función es utilizada para predicciones de variables continuas, y al ser una función convexa tiene muchas ventajas con muchos algoritmos de optimización.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{z}^i - z^i\right)^2 \ . \tag{3.13}$$

- *Cross Entropy k-Clases*: Es una función muy utilizada como métrica del error para la tarea de clasificar en multiclases.

$$CE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{z}_k^i - z_k^i\right)^2 \boldsymbol{x}^i \ . \tag{3.14}$$

- *Log Loss*: También llamada *Binary Cross Entropy*, es el caso particular de la función de costo *Cross Entropy* cuando la clasificación es binaria, es decir, cuando $k = 2$. Esta decrece cuando dos distribuciones de probabilidad son similares y crece cuando éstas divergen [16].

$$LL = -\frac{1}{n}\sum_{i=1}^{n}\left\{z^i \log\left(\hat{z}^i\right) + \left(1 - z^i\right)\log\left(1 - \hat{z}^i\right)\right\} \ . \tag{3.15}$$

- ***Real-World-Weight Cross Entropy***: Esta función de costo la proponen *Ho Yaoshiang* y *Wookey Samuel* [33], usan la idea de la función *Cross Entropy* para el caso binario, con la diferencia que esta impacta en la penalización hacia predicciones o clasificaciones erróneas dada una ponderación de sus pesos respectivos de las categorías o clases. Los pesos, como se verá más adelante tendrán la interpretación de costos, también esta función resulta muy útil para la clasificación cuando se tienen datos desbalanceados o que existe una mayor importancia en clasificar una clase sobre la otra.

$$J_w = -\frac{1}{n}\sum_{i=1}^{n}\left\{w_1 z^i \log\left(\hat{z}^i\right) + w_0\left(1 - z^i\right)\log\left(1 - \hat{z}^i\right)\right\} \ . \tag{3.16}$$

Optimizador: Estos son algoritmos que ajustan el aprendizaje a través del entrenamiento dando una mejor convergencia de la función de costo o perdida hacia su mínimo, es decir, determina de qué forma los parámetros de la ANN continuarán reajustándose de forma óptima basado en su función de pérdida. Algunos de los algoritmos de optimización más comunes son los siguientes.

- *Descenso del gradiente* (GD[10]): Mediante este proceso se actualizan los pesos w buscando el mínimo de la función de costo utilizando el negativo de su gradiente a pasos en magnitud del hiperparámetro *tasa de aprendizaje* η y que esta puede no ser siempre constante. Este algoritmo es muy bueno para funciones convexas (MSE) [16] (Veáse en la figura 3.6). Se actualizan los pesos w de la ANN en la iteración k, siendo $J(w)$ la función de pérdida, entonces

$$w_{k+1} = w_k - \eta\nabla_w J(w_k) \ . \tag{3.17}$$

- *Derivados GD*: Para aquellas funciones de pérdida que no son totalmente convexas y tienen mínimos locales, existen optimizadores derivados del GD con la finalidad de evadir los mínimos locales, por ejemplo, *GD Estocástico*, *Mini-batch GD*.

[10] Gradient Descent

- *Otros*:Para redes más sofisticadas, se requieren mejores optimizadores tales como *Adagrad, RMSprop,* ***Adam****, Nadam, AdaMax* [9].

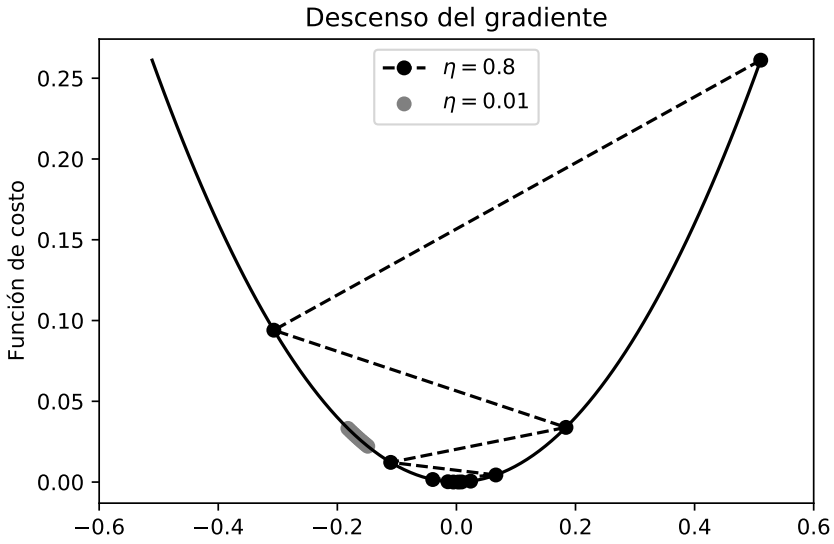


Figura 3.6: Gráfica de la aplicación del algoritmo descenso del gradiente a diferentes tasas de aprendizaje en una función de costo $J(w)$ convexa. Se observa que para la tasa de aprendizaje más pequeña se aproxima al mínimo de forma más lenta.

Claramente no hay una tasa de aprendizaje η que funcione para todos los modelos o estructuras, generalmente se podrá ir ajustando mediante un análisis del comportamiento de la función de pérdida en el entrenamiento de la red. Si la tasa es muy pequeña la convergencia requerirá muchas iteraciones para alcanzar el mínimo, y si es muy grande puede incluso diverger [16]. Se sugiere por Gerón A. [9] iniciar con una tasa del orden 10^{-3}.

3.3.3. Algoritmo de Retropropagación

Si se tiene un perceptrón multicapa con muchas capas, el cálculo del descenso del gradiente para ajuste de los pesos sinápticos de cada una de las capas por alimentación hacia adelante resultaría computacionalmente costoso y tomaría mucho tiempo, por lo que en 1986 se inventa el algoritmo de *Retropropagación* o *Backpropagation* que permite estos cálculos de forma mucho más eficiente, donde principalmente se sustenta en la regla de la cadena.

En esencia una vez que la información llega hasta la capa de salida se tiene una estimación o predicción y dada la función de pérdida se tiene una magnitud del error, de este valor se reajustan los pesos por medio la alimentación hacia atrás, para en la siguiente salida tener un menor error hasta encontrar un mínimo óptimo. A este ciclo *alimentación hacia adelante-alimentación hacia atrás* se le llama época[11], y más adelante el número de Epochs será un hiperparámetro a decidir.

[11] Epoch

Supongamos una red neuronal con $L > 2$ capas (de lo contrario sería un perceptrón simple) y sea w^l_{ji} el peso que enlaza a la i-ésima neurona de la capa $l-1$ con la j-ésima neurona de la capa l y b^l_j es el sesgo[12] de la j-ésima neurona de la capa l. Entonces la activación a^l_j de la j-ésima neurona perteneciente a la capa l y dependiente de las activaciones de la capa anterior (alimentación hacia adelante) es para $2 \leq l \leq L$:

$$a^l_j = F\left(\sum_i w^l_{ji} a^{l-1}_i + b^l_j\right) . \tag{3.18}$$

Donde F es la función de activación, claramente esta podría ser F^l, indicando que para cada capa se puede tener una diferente función de activación, incluso en cada neurona de cada capa puede ser distinta (F^l_j), no obstante, asumiremos la misma para todas las capas. Entonces podemos representar los elementos anteriores de forma matricial para la capa l con:

$$\boldsymbol{w}^l = \begin{pmatrix} w^l_{11} & w^l_{12} & \cdots \\ w^l_{21} & \ddots & \\ \vdots & & \end{pmatrix}, \boldsymbol{a}^l = \begin{pmatrix} a^l_1 \\ a^l_2 \\ \vdots \end{pmatrix}, \boldsymbol{b}^l = \begin{pmatrix} b^l_1 \\ b^l_2 \\ \vdots \end{pmatrix} . \tag{3.19}$$

De forma análoga se expresa de forma matricial la ecuación (3.18) por (3.20), notando que el resultado es la aplicación de la función de costo a un vector de sumas ponderadas $\boldsymbol{s}^l$, donde cada entrada es la suma ponderada de la neurona j de la capa l, s^l_j.

$$\boldsymbol{a}^l = F\left(\boldsymbol{w}^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l\right) . \tag{3.20}$$

Así podemos hacer,

$$\boldsymbol{s}^l = \boldsymbol{w}^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l = \begin{pmatrix} s^l_1 \\ s^l_2 \\ \vdots \end{pmatrix} . \tag{3.21}$$

Nótese que para la última capa $l = L$ se tiene que $\boldsymbol{a}^L = F\left(\boldsymbol{s}^L\right) = \hat{\boldsymbol{z}}$, es decir, la activación de la última capa es la predicción de la red. Para la función de costo J se obtiene el error de esta predicción y mediante la regla de cadena se pueden calcular el cambio del error ante variaciones en los pesos de las neuronas, para así, reajustar con algún optimizador los pesos sinápticos, tal como se mencionó anteriormente. Con ello, para la neurona j de la última capa L se tiene que:

$$\frac{\partial J}{\partial w^L_{ji}} = \frac{\partial J}{\partial a^L_j}\frac{\partial a^L_j}{\partial s^L_j}\frac{\partial s^L_j}{\partial w^L_{ji}} \text{ y } \frac{\partial J}{\partial b^L_j} = \frac{\partial J}{\partial a^L_j}\frac{\partial a^L_j}{\partial s^L_j}\frac{\partial s^L_j}{\partial b^L_j} . \tag{3.22}$$

Por otro lado, se puede simplificar un poco con la idea, cuánto cambia el costo en la neurona j al variar su entrada en la capa L, este se denota por δ^L_j

$$\delta^L_j = \frac{\partial J}{\partial s^L_j} = \frac{\partial J}{\partial a^L_j}\frac{\partial a^L_j}{\partial s^L_j} = \frac{\partial J}{\partial a^L_j} F'\left(s^L_j\right) . \tag{3.23}$$

Derivando parcialmente respecto a los pesos la ecuación (3.21), se llega a que

$$\frac{\partial s^L_j}{\partial w^L_{ji}} = a^{L-1}_i \text{ y } \frac{\partial s^L_j}{\partial b^L_j} = 1 . \tag{3.24}$$

Por tanto, de (3.22) y (3.24) se tiene que

[12] Bias: peso sináptico libre

$$\frac{\partial J}{\partial w_{ji}^{L}} = \delta_j^L a_i^{L-1} \text{ y } \frac{\partial J}{\partial b_j^L} = \delta_j^L \ . \tag{3.25}$$

De esta forma, se puede aplicar el ajuste de los pesos sinápticos por medio de algún optimizador en la capa L, por ejemplo, *El Descenso del Gradiente* de la ecuación (3.17). No obstante, solo se cuenta con la información para el reajuste de la última capa, restando $L-1$ capas. Afortunadamente, se puede proceder de forma análoga aplicando nuevamente la regla de la cadena. Así para cualquier neurona j de la capa $l < L$ se tiene

$$\frac{\partial J}{\partial w_{ji}^{l}} = \frac{\partial J}{\partial s_j^l}\frac{\partial s_j^l}{\partial w_{ji}^l} = \delta_j^l a_i^{l-1} \text{ y } \frac{\partial J}{\partial b_j^l} = \frac{\partial J}{\partial s_j^l}\frac{\partial s_j^l}{\partial b_j^l} = \delta_j^l \ . \tag{3.26}$$

pero, sabemos que

$$\begin{aligned}
\delta_j^l &= \frac{\partial J}{\partial s_j^l} \\
&= \sum_i \frac{\partial J}{\partial s_i^{l+1}}\frac{\partial s_i^{l+1}}{\partial s_i^l} \\
&= \sum_i \delta_i^{l+1}\frac{\partial s_i^{l+1}}{\partial s_i^l} \quad \text{y como} \\
\frac{\partial s_i^{l+1}}{\partial s_j^l} &= w_{ij}^{l+1} F'\left(s_j^l\right) \quad \text{por tanto} \\
\delta_j^l &= \sum_i \delta_i^{l+1} w_{ij}^{l+1} F'\left(s_j^l\right) \ .
\end{aligned}$$

Por tanto, dados los resultados anteriores se puede resumir que el aprendizaje de una ANN mediante el algoritmo de retropropagación se puede esquematizar (Chollet F. [5]) en un diagrama de flujo como se muestra en la figura 3.7.

Por tanto, el proceso anterior se puede resumir de la siguiente forma:

Estructura: Una vez definida la estructura de la ANN como número de capas ocultas y neuronas en estas. También se fija la función de costo (J), funciones de activación (F) por capa o neurona, optimizador e hiperparámetros tales como tasa de aprendizaje (η) y la cantidad de epochs (k).

Alimentación hacia adelante: Siendo $\boldsymbol{x}^T = (x_1, x_2, ..., x_p)$ la información de entrada a la red por la primera capa, esto es $\boldsymbol{x} = \boldsymbol{a}^1$, entonces para las capas siguientes $l = 2, ..., L$ y recordando que para el primer *epoch* todos los pesos son aleatorios se calcula $\left(\boldsymbol{s}^l, \boldsymbol{a}^l\right)$ mediante las ecuaciones (3.21) y (3.20) respectivamente.

Última capa: Para la última capa se obtiene el error de predicción mediante la función de costo J, es decir, $J\left(\boldsymbol{a}^L\right) = J\left(\hat{\boldsymbol{z}}\right)$. Mediante (3.25) y un optimizador se reajustan los pesos para cada neurona j de la última capa, por ejemplo, para *GD* (3.17).

$$w_{ji}^L(k+1) = w_{ji}^L(k) - \eta\delta_j^L a_i^{L-1} \ \text{ y } \ b_j^L(k+1) = b_j^L(k) - \eta\delta_j^L \ . \tag{3.27}$$

Retropropagación: Para $l = L-1, L-2, ..., 2$ se calculan los errores respectivos dadas las ecuaciones de 3.26 y de forma análoga se aplica el *GD* de la siguiente forma,

$$w_{ji}^{l}(k+1) = w_{ji}^{l}(k) - \eta \delta_{j}^{l} a_{i}^{l-1} \text{ y } b_{j}^{l}(k+1) = b_{j}^{l}(k) - \eta \delta_{j}^{l} . \tag{3.28}$$

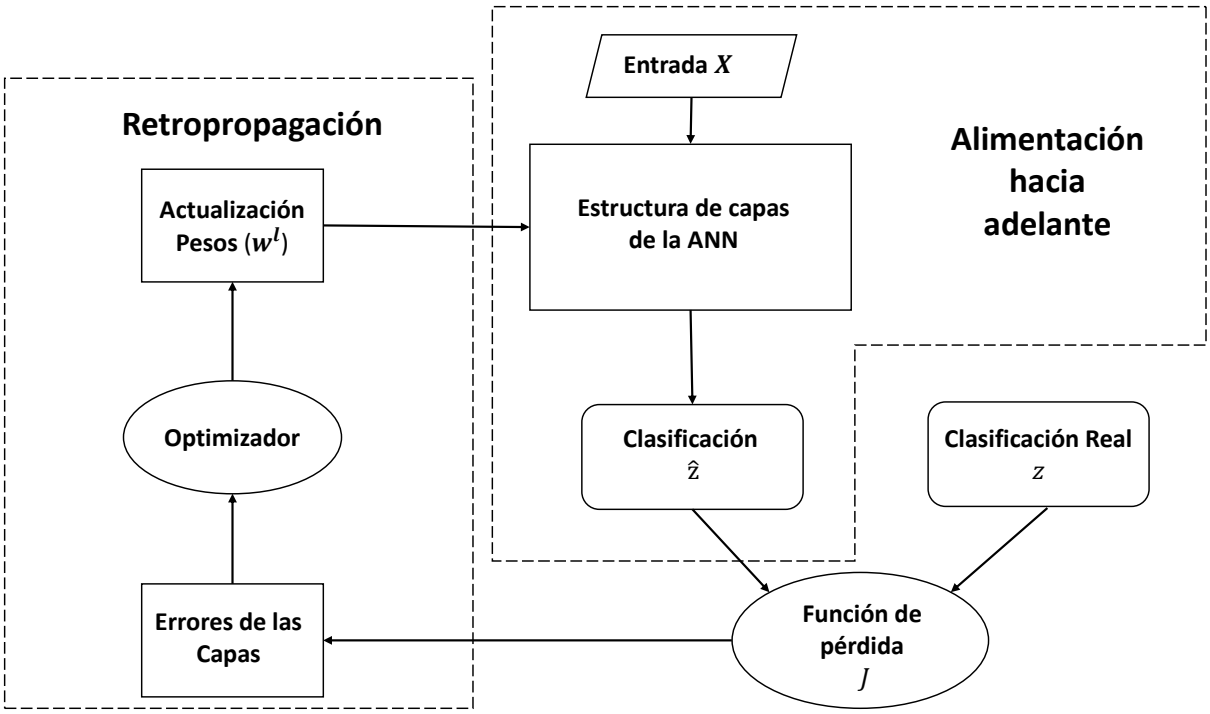


Figura 3.7: Diagrama de flujo del proceso de aprendizaje de una ANN.

3.3.4. Curva de Validación

Uno de los problemas que se presenta en muchos de los modelos de aprendizaje supervisado de Machine Learning es su capacidad de generalización en los datos, esto es, en la fase de entrenamiento de estos modelos se presenta un buen rendimiento con dichos datos, no obstante, para cuando se pone a prueba en datos nuevos dicho rendimiento ya no es aceptable. Esta situación se puede deber a dos particularidades:

Subajuste Comúnmente llamado por su nombre en inglés *Underfitting*, esto ocurre cuando el modelo es muy simple para modelar datos con comportamiento más complejo, por lo que su desempeño para los datos de prueba o validación es muy pobre, por lo que lleva a un *Sesgo*. El *Sesgo* es que la diferencia del valor de predicción y el valor real es muy grande, por lo que para un buen modelo éste será pequeño, veáse en la figura 3.8.

Sobreajuste Comúnmente llamado por su nombre en inglés *Overfitting*, se presenta cuando el modelo es muy complejo para generalizar la información de prueba, lo que lleva a una *Varianza* grande. La *Varianza* muestra qué tan dispersos están los valores predichos en comparación con los valores verdaderos, veáse en la figura 3.8.

De esta manera, un modelo y en particular una estructura ANN deberá poder generalizar sus resultados, es decir, deberá tener un desempeño muy similar con datos de entrenamiento como con los que no ha visto,

tales como datos nuevos o de validación. Estos últimos serán muy útiles para poder aproximarnos a un punto que optimice el desempeño de nuestro modelo, dicho punto estará en medio de área de sobreajuste como un subajuste y que nos valdremos de la *Curva de Validación* como se muestra, en la figura 3.9.

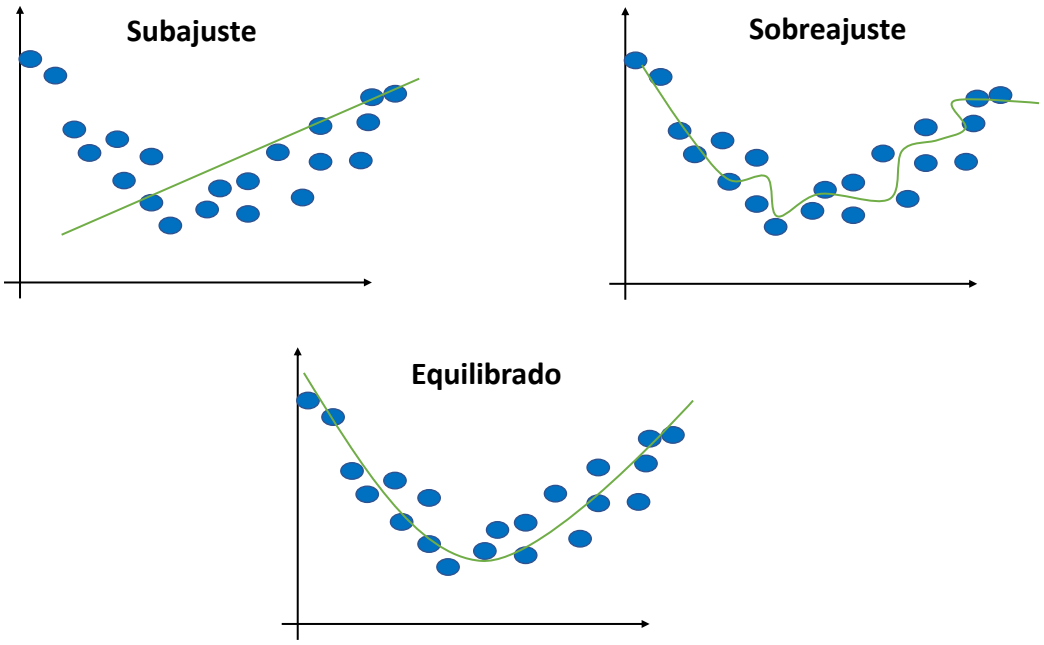


Figura 3.8: Gráfica ejemplo sobre la complejidad del modelo en unos datos.

La *Curva de Validación* nos ayuda a observar el comportamiento de nuestro modelo tanto en los datos de entrenamiento como el los datos de validación, pudiéndose observar cuando el error va disminuyendo en el área de subajuste y el momento cuando el error para los datos de validación comienza a crecer indicando que se está perdiendo la capacidad de generalización. Por tanto, el punto o región para el óptimo de complejidad en la estructura del modelo, en caso de las ANN, por ejemplo, puede ser el número de capas o el número de sus neuronas.

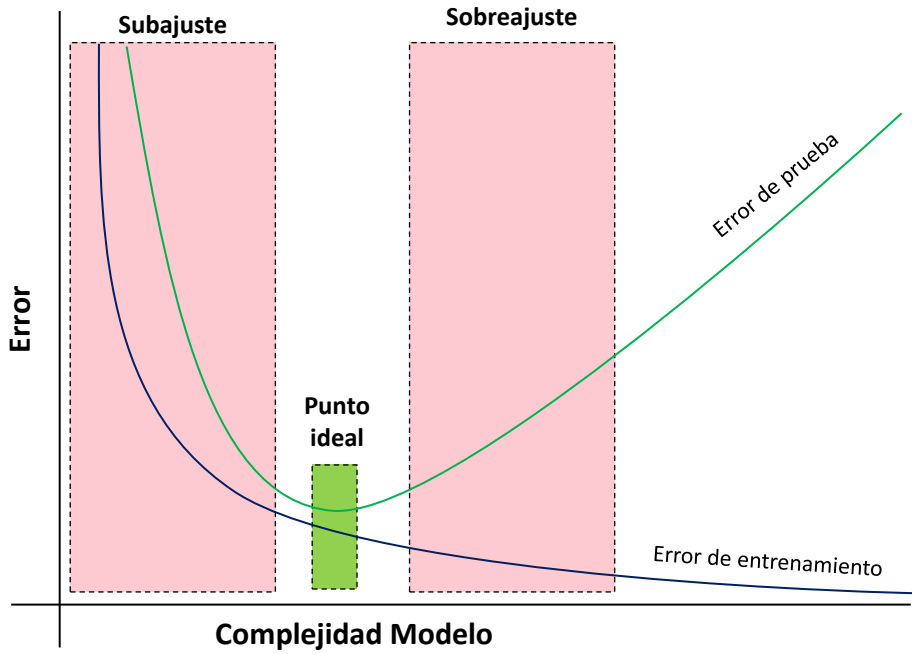


Figura 3.9: Variación del desempeño durante el entrenamiento y prueba cuando se aumenta la complejidad o estructura ANN.

3.4. ANN en Credit Scoring

Dentro del ámbito del Credit Scoring, la utilidad del uso de los modelo ANN es su capacidad de modelar funciones muy complejas simulando el cerebro humano y que ofrecen una gran habilidad de generalización en los datos, esto contrasta con técnicas estadísticas lineales clásicas. Aunque una de las desventajas de estos modelos es su interpretabilidad para cada una de las variables explicativas, su capacidad de automatización, eficiencia y generalización lo compensan.

Para este trabajo se utilizará 1 capa de entrada, de 2 a 3 capas ocultas y una capa de salida con una sola neurona que dará la estimación de riesgo $\hat{\theta}$, tal como se esquematiza en la figura 3.10, de 20-500 neuronas en cada capa, conexión densa entre las neuronas de las capas adyacentes. Como funciones de activación se tomarán en cuenta la Relu, Leaky Relu, ELU y Sigmoide que son las más utilizadas en ANN para clasificación. Para el ajuste de pesos se aplicará el algoritmo de Backpropagation con optimizador el Descenso del gradiente o descenso del gradiente estocástico y por último, la tasa de aprendizaje η será fija en un rango de 1.0001 a 0.01.

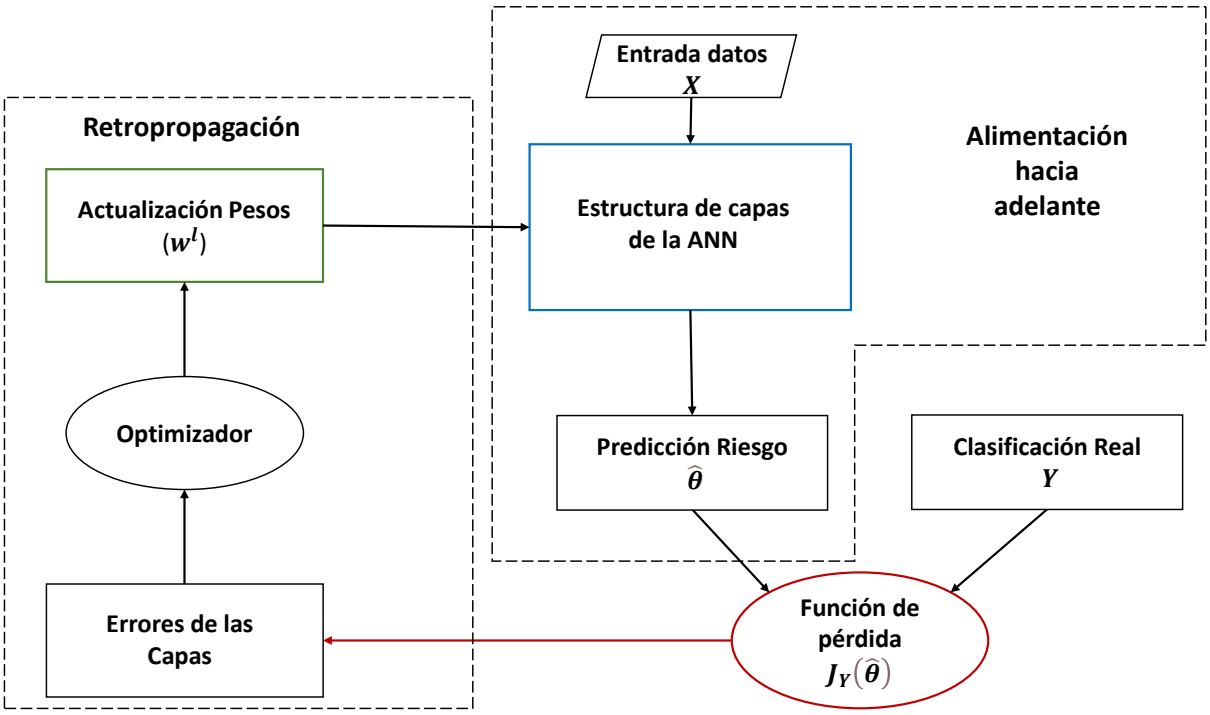


Figura 3.10: Diagrama de flujo del proceso de aprendizaje de una ANN para el concepto Credit Scoring.

De forma análoga a la aplicación del modelo paramétrico, una vez se haya estimado el riesgo asociado θ para cada cliente nuevo por el modelo de ANN h con pesos w,

$$h_w(\boldsymbol{x}) = \hat{\theta} = \hat{P}(Y = 1|\boldsymbol{x}) \ . \tag{3.29}$$

Y tomando la decisión aplicando nuevamente la ecuación $I\left(\hat{\theta}\right)$ de 1.3 para rechazar u otorgar el crédito $(\hat{Y})$ dado un *cutt-off* θ_0 óptimo.

$$I\left(\hat{\theta}\right) = \begin{Bmatrix} \hat{Y} = 0 & \text{Si } \hat{\theta} \leq \theta_0 \\ \hat{Y} = 1 & \text{Si } \theta_0 < \hat{\theta} \end{Bmatrix} \ . \tag{3.30}$$

Capítulo 4

Medidas de desempeño para modelos de clasificación

Una vez propuesto un modelo Credit Scoring debemos cuestionar qué tan bueno o malo es este, es decir, cuan bien está clasificando o distinguiendo a los clientes malos de los buenos, ya que una mala clasificación será invariablemente una pérdida, por tanto, se debe contar con métricas que pueda evaluar su desempeño, esto es, qué tan alejado están las predicciones de los valores reales o esperados tanto en la fase de entrenamiento como para poder determinar su implementación final. Estas métricas ayudarán entre otras cosas, a seleccionar el mejor modelo Credit Scoring de entre un conjunto de ellos, también para realizar ajustes tales como la selección o cambio de hiperparámetros para el caso de un ANN (tasa de aprendizaje, función de activación, umbral de categoría, etc.), encontrar un corte de clasificación óptimo dada su probabilidad de incumplimiento y/o políticas de la empresa.

En este capítulo retomaremos algunas de las medidas más comunes en la clasificación binaria tales como la *Matriz de confusión*, métricas numéricas, prueba *Kolmogórov-Smirnov*, así como la curva *Característica Operativa del Receptor* (*ROC*). Se abordará de forma breve la importancia de una correcta partición de nuestra base de datos en las fases de entrenamiento, prueba y búsqueda del mejor modelo, asimismo se mencionarán algunas técnicas útiles para tratar el problema de desbalance en la variable de interés tal como el remuestreo o penalización de costo por clasificaciones incorrectas.

4.1. Partición Dataset

Si se evaluara un modelo Credit Scoring con los mismos datos con el que se entrenó o generó se pudiera tener una idea errónea que éste es muy bueno dado que las métricas serían altas, no obstante, el modelo puede ser en su implementación real muy malo. El que las métricas fueran altas se debería a que se está intentando evaluar con los mismo datos que el modelo ya conoce e incluso puede ocurrir que se haya modelado su ruido (*Overfitting*) debido a errores muy cercanos a cero. Claramente si se implementara con datos que en el modelo no conozca es muy probable que su desempeño sea muy malo, por tanto, se debe evaluar con datos que no haya visto durante el entrenamiento con la finalidad de valorar su capacidad de generalización.

Con esta idea, varios autores sugieren dividir toda la información disponible para el entrenar un modelo con la mayor capacidad de generalización posible, la división consiste en tres principales segmentos (como se muestra en la figura 4.1): segmento o Set de *Entrenamiento* (*Training Set*), set de *Evaluación* (*Evaluation Set*), y set de *Prueba* (*Test Set*), con esta partición se confía lograr a un modelo óptimo y más confiable, por ejemplo, Yuxi[16] sugiere una partición del 80 %, 10 % y 10 % respectivamente.

Set de Entrenamiento: Es el segmento de información con mayor proporción para el modelo, ya que de este se aprenderán las características y patrones para la correcta predicción en la clasificación.

Set de Evaluación: Una vez que el modelo ha aprendido del Set de Entrenamiento, su desempeño hasta ese momento es evaluado con este set, con oportunidad ajustar o recalibrar el modelo de forma repetida hasta alcanzar un mejor rendimiento. La última versión del modelo debe ser aquella que ajuste este set, no obstante, no es la evaluación final del desempeño. Este set es de suma importancia, ya que además de proveer de estimaciones insesgadas, ayuda a evitar el sobreajuste en el modelo[13].

Set de Prueba: Una vez que se cuenta una propuesta de modelo para su versión final del modelo, la evaluación definitiva del rendimiento deberá ser con este último set. Notando que en este set ya no hay entrenamiento, ajustes o actualizaciones para modelo después de que sus métricas han sido calculadas. De tener unas buenas métricas puede considerarse para su implementación real.

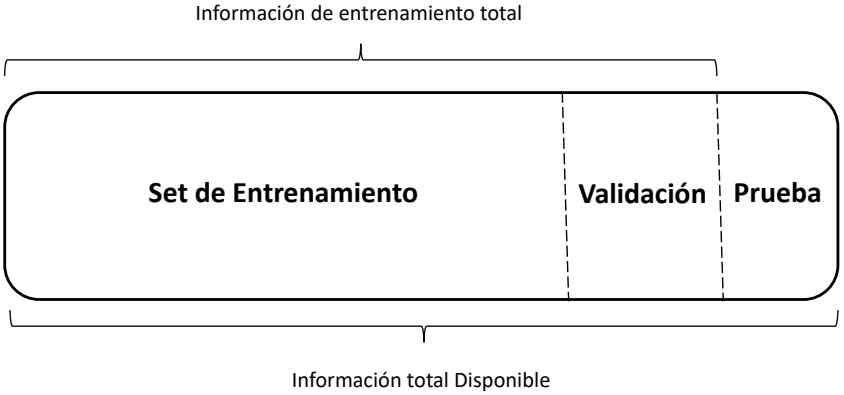


Figura 4.1: Partición DataSet.

Utilizando de esta manera la información disponible para el modelo Credit Scoring se evita sobreajuste, un mejor panorama para la recalibración de hiperparámetros y así facilitar el elegir el mejor modelo para su implementación.

4.2. Matriz de confusión

A priori, si de antemano sabemos que un cliente es malo entonces el modelo Credit Scoring debe dar una probabilidad de incumplimiento $\hat{\theta}$ alta (*probability of default*), por lo que su clasificación sería malo ($\hat{Y} = 1$), es decir, si su clasificación real es 1 entonces se espera que el modelo lo clasifique como 1, de hecho, si sabemos la clasificación real de n clientes, se espera que la mayor parte de estos clientes sean clasificados se forma correcta.

Supongamos que nuestro modelo Credit Scoring brinda las estimaciones de la probabilidad de impago para n solicitantes $\hat{\boldsymbol{\theta}}^T = \left(\hat{\theta}_1, \hat{\theta}_2, ..., \hat{\theta}_n\right)$, y además de saber su clase verdadera correspondiente $\boldsymbol{Y} = (Y_1, Y_2, ..., Y_n)$, este último es un vector de 0's (clientes buenos) y 1's (clientes malos). De lo anterior podemos visualizar la distribución de los scores dada su clase real como se muestra en la figura 4.2. Así, para un riesgo dispuesto fijo θ_0 (*Cut-off*), se utiliza la función de clasificación (1.3) para estimar la clase a la cual pertenece cada cliente, es decir, obtendremos un vector $\hat{\boldsymbol{Y}}$ también compuesto de 0's y 1's tal como se expresa en la ecuación (4.1).

$$\hat{\boldsymbol{Y}}^T = \left(I_{\theta_0}\left(\hat{\theta}_1\right), I_{\theta_0}\left(\hat{\theta}_2\right), ..., I_{\theta_0}\left(\hat{\theta}_n\right)\right) = \left(\hat{Y}_1, \hat{Y}_2, ..., \hat{Y}_n\right) \ . \tag{4.1}$$

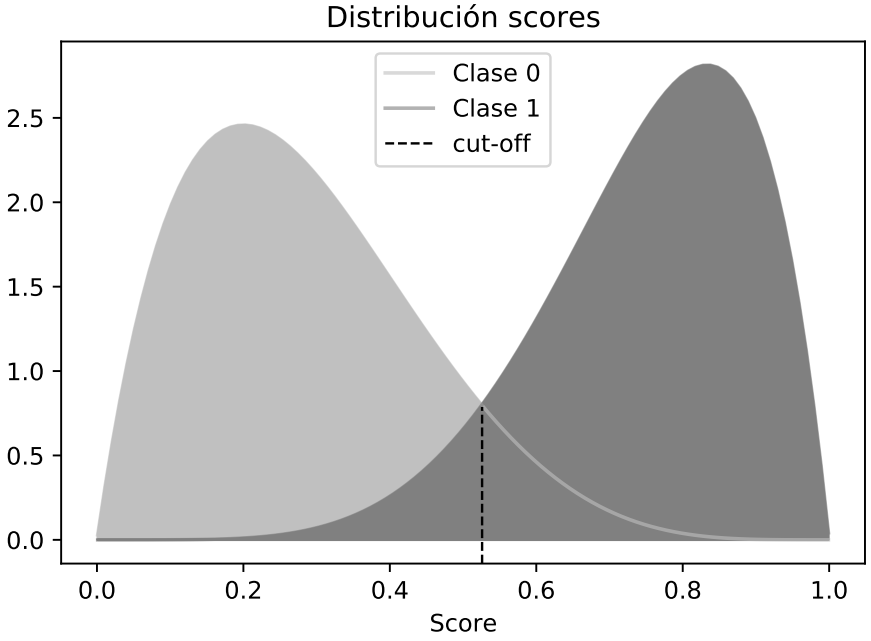


Figura 4.2: Distribución predicción modelo Credit Scoring.

De la función (4.1), se pueden tener cuatro casos al contrastar las predicciones de clase contra su clasificación correcta, sea n el número de predicciones a contrastar, además n_0 y n_1 la cantidad de real de categorías 0 y 1 respectivamente ($n = n_0 + n_1$), entonces:

Verdaderos Positivos(TP) y Verdaderos Negativos(TN): En general, son aquellas predicciones que coinciden con su clasificación real, particularmente en el caso de la clasificación binaria los TP son la cantidad de clasificados por el modelo como clase 1 y que su clase real es 1. Análogamente para los TN con clase 0:

$$TP = \sum_{i=1}^{n} \mathbb{1}_{\{Y_i=1\}}\left(\hat{Y}_i\right) \ . \tag{4.2}$$

$$TN = \sum_{i=1}^{n} \mathbb{1}_{\{Y_i=0\}}\left(\hat{Y}_i\right) \ . \tag{4.3}$$

Falsos Positivos(FP): Es la cantidad de clasificados como 1 por el modelo pero que su clase real es 0. Este también es conocido como el error tipo 1.

$$FP = n_1 - \sum_{i=1}^{n} \mathbb{1}_{\{Y_i=1\}}\left(\hat{Y}_i\right) \ . \tag{4.4}$$

Falsos Negativos(FN): De forma análoga, es la cantidad de clasificados como 0 pero que su clase real es 1. Este también es conocido como el error tipo 2.

$$FN = n_0 - \sum_{i=1}^{n} \mathbb{1}_{\{Y_i=0\}}\left(\hat{Y}_i\right) . \tag{4.5}$$

Esto se puede resumir en la *Matríz de Confusión* que se muestra en la figura 4.3. De esta matriz surgen algunas métricas que nos indica qué tan bien está clasificando el modelo: *Exactitud*, *Precisión*, *Recall* y *Score* F_β.

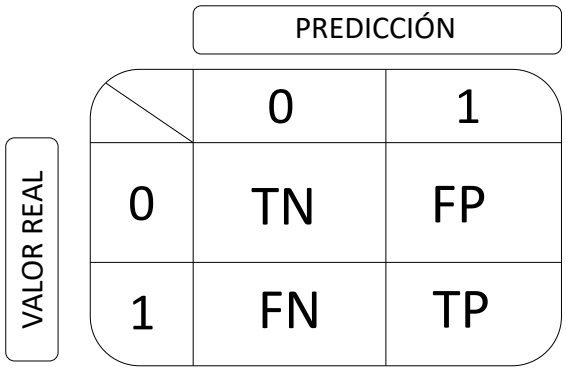


Figura 4.3: Matriz confusión.

Exactitud: Es la proporción de predicciones correctas del total de las predicciones hechas.

$$Ex = \frac{TP + TN}{TP + TN + FP + FN} . \tag{4.6}$$

Precisión: Indica la proporción del total clasificados como positivos que realmente son positivos.

$$Pres = \frac{TP}{TP + FP} . \tag{4.7}$$

Recall: También llamada sensibilidad o tasa de positivos verdaderos indica la tasa de verdaderos positivos clasificados de forma correcta.

$$Re = \frac{TP}{TP + FN} . \tag{4.8}$$

Especificidad: Es la tasa de verdaderos negativos.

$$Esp = \frac{TN}{TN + FP} . \tag{4.9}$$

Score F_β: Ante el costo de decisión entre las métricas $Pres - Re$, se tiene una función real dependiente del *Recall* y la *Precisión* $F_\beta : (Pres, Re) \rightarrow (0, 1)$ que sintetiza ambas métricas en una y que al estar más cerca del 1 es mejor. Esta medida se basa en una media Harmónica entre la *Precisión* y el *Recall*. Dado $\beta > 0$, se debe tomar en cuenta que si además $\beta < 1$ la métrica se enfocará en la *Precisión*, $\beta > 1$ en el *Recall* y para clases balanceadas $\beta = 1$ [10].

$$F_\beta = (1 + \beta)\, \frac{Pres \cdot Re}{\left(\beta^2 \cdot Pres\right) + Re} . \tag{4.10}$$

Es importante señalar que aun cuando la *Exactitud* brinda una aceptable referencia sobre la capacidad clasificación y predicción del modelo debido a que refleja una proporción generalizada de todas las clasificaciones correctas, en general esta métrica no es buena cuando en los datos hay mucho mayores ejemplos de una clase respecto a otra u otras a causa de que los modelos tienden a aprender más de la clase predominante, a esto le decimos sesgo de clasificación. Por lo que para tales casos las métricas *Precisión* y *Recall* proporcionan un mejor panorama o referencia de clasificación dado que la métrica *Recall* se enfoca en medir la correcta clasificación de la clase positiva 1, pues penaliza los FN, y de forma análoga la *Precisión* para la clase negativa.

Nótese que la función I_θ (1.3) se ha mantenido constante, no obstante, si se varía el valor de corte entonces las métricas (4.6)-(4.9) brindarían diferentes resultados. Esta variación permite tener un panorama del desempeño de un modelo con las métricas de mayor interés y que en este caso es *Recall* y *Precisión*. Supongamos tenemos un modelo f y sean $\{p_1, p_2, ..., p_k\} \subset (0, 1)$ con $p_l < p_m$ para $l < m$, donde $l = 1, 2, ..., k-1$ y $m = 2, ..., k$ los diferentes cortes.

De esta manera se pueden apreciar en la gráfica 4.4 el comportamiento de las métricas *Precisión* y *Recall* a lo largo de diferentes *cut-offs* .

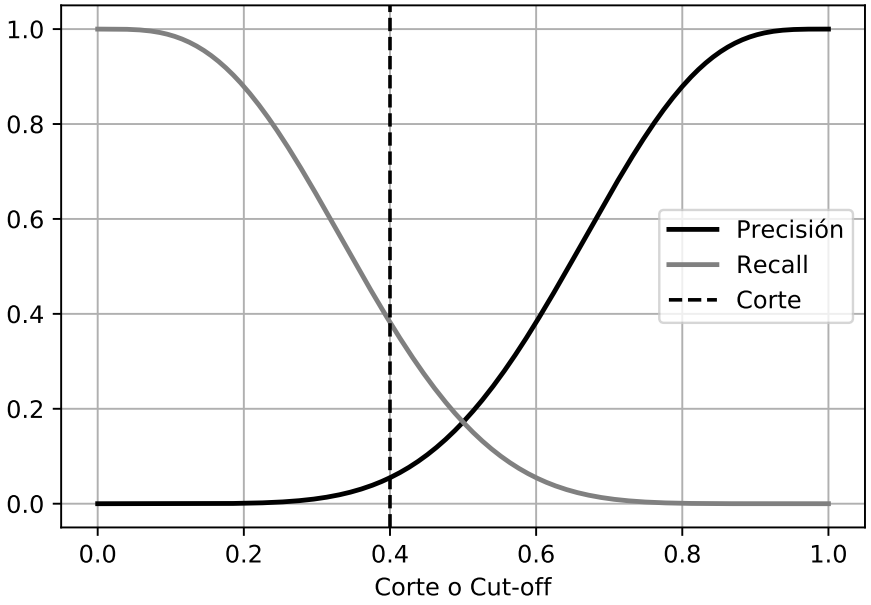


Figura 4.4: Métricas de un modelo de clasificación para diferentes cortes.

En la misma gráfica por ejemplo, para un cut-off $p_k = 0.4$ implica una *Precisión* del 5 % y un *Recall* de 40 %. Nótese que si el corte es más hacia la izquierda de 0.50 se enfocará en el *Recall* y para la derecha en la *Precisión*. En nuestro contexto, para evitar pérdidas debemos enfocamos un poco más en el *Recall* puesto dado que se requiere minimizar los Falsos Negativos (evitar clasificar un cliente malo como bueno), por tanto, el corte debe estar más a la izquierda tomando en cuenta además otros factores como costos.

Por otro lado, directamente se pude relacionar la *Precisión* con el *Recall* en una gráfica tal como se muestra en la figura (4.5), de esta forma, al fijar algún valor para una métrica, se puede saber el valor de la otra. Por ejemplo, si se requiere una *Precisión* del 82 %, se debe buscar el mínimo corte p_k que brinde al menos esa *Precisión*, por lo que el coste de decisión es un *Recall* del 70 %, tal como se muestra en la línea

punteada de la figura.

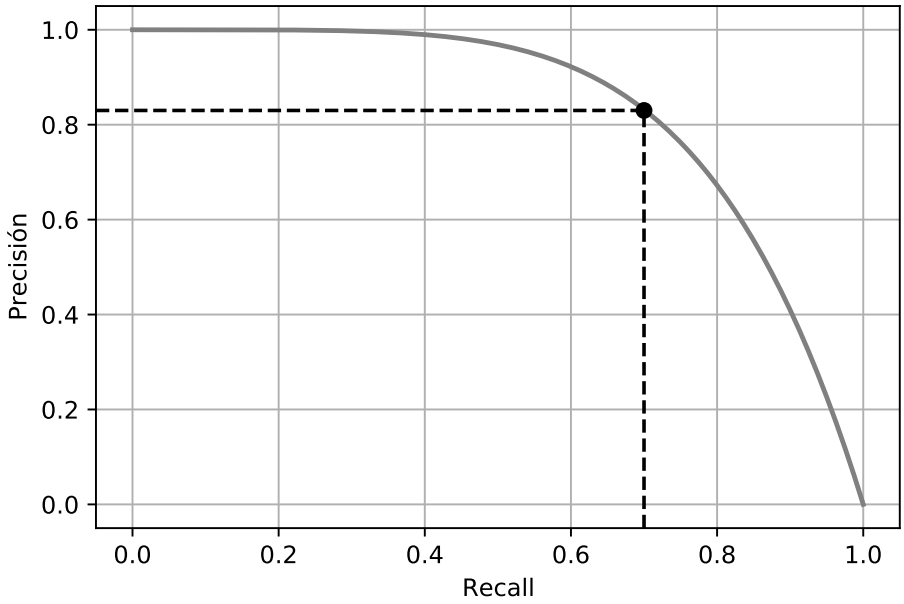


Figura 4.5: Gráfica relación de métricas.

4.3. Curva ROC

Una herramienta muy útil en la clasificación binaria y particularmente en el Credit Scoring es la Curva Característica Operativa del Receptor[1] (Curva ROC por su siglas en inglés) y que también es otra forma de visualizar la gráfica 4.4 y que evalúa clasificadores. En la curva ROC principalmente se visualiza la contraposición entre las tasas de positivos verdaderos y falsos positivos, por tanto, determina qué tasa de instancias serán clasificadas de forma correcta para una tasa de positivas incorrectas dada.

En la figura 4.6 se puede ver que un excelente modelo Credit Scoring tendrá un punto de la curva ROC en $(0, 1)$ o muy cerca de éste, esto es, clasificará de forma de forma correcta tanto las clases positivas como las negativas. De forma análoga, un pésimo modelo tendrá un punto de Curva ROC en $(1, 0)$. Para un modelo que no tiene capacidad de predicción, su curva ROC es la línea de la función identidad en el dominio $(0, 1)$ [10]. Por tanto, esta curva un buen método para encontrar un clasificador óptimo, ya que dado un conjunto de curvas ROC y fijado una tasa de falsos positivos, se elige aquella con mayor Tasa de verdaderos positivos. Debe tenerse cuidado con esta curva en aquellos modelos entrenados con datos desbalanceados, ya que para estos casos brinda información sesgada.

Una métrica adicional muy utilizada en las curvas ROC es el área debajo de éstas, un valor para comparar clasificadores, mayor área implica mejor modelo, esto también se puede apreciar en la gráfica 4.6. Una forma de estimar esta área es [10]: Dados n_0 , n_1 puntos de las clases 0 y 1 respectivamente, además siendo S_0 la suma de los rangos de la clase 0, la ecuación es la siguiente:

$$AUC_{ROC} = \frac{2S_0 - n_0(n_0 + 1)}{2n_0n_1} \, . \tag{4.11}$$

[1] Receiver Operating Characteristic

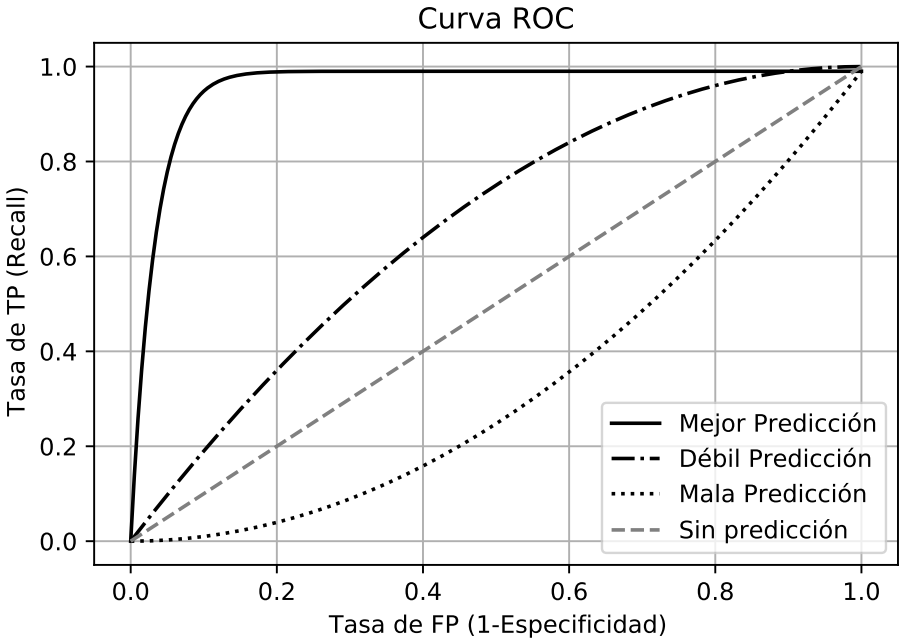


Figura 4.6: Curva ROC.

4.4. Estadístico KS

En el Credit Scoring, un estadístico muy utilizado incluso hasta en nuestros días es el *Kolmogórov-Smirnov* (KS), el cual se basa en el cálculo de la distancia entre funciones de distribución empírica (fde) de los datos de las muestras a comparar, es decir, mide cuan alejados se encuentran dos funciones de distribución empírica, que en desde nuestro enfoque son las de los buenos y malos clientes. Este estadístico se enfoca principalmente en la mayor diferencia absoluta entre estas dos funciones (D_{ks}), donde $0 < D_{ks} < 1$.

Sea $\{x_1, ..., x_n\}$ una muestra aleatoria simple de la variable aleatoria X, donde $X \sim G$, entonces para $x \in (0, 1)$ fijo la función de distribución empírica es:

$$G_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{(-\infty, x]}(x_i) \ . \tag{4.12}$$

Por tanto, con $p_0 \in (0, 1)$ las funciones de distribución empírica P_0 y P_1 de las clases 0 y 1 respectivamente, así

$$D_{ks} = \max_{p_0} \{P_0(p_0) - P_1(p_0)\} \ . \tag{4.13}$$

En la figura 4.7 se puede apreciar las fdae de las clases 0 y 1, además se indica el estadístico $D_{ks} = 0.4$ donde se tiene el 80 % y 20 % de las clases 0 y 1 respectivamente. De esta manera se observa que la máxima distancia que separa las distribuciones se obtiene con $p_0 = 0.4$.

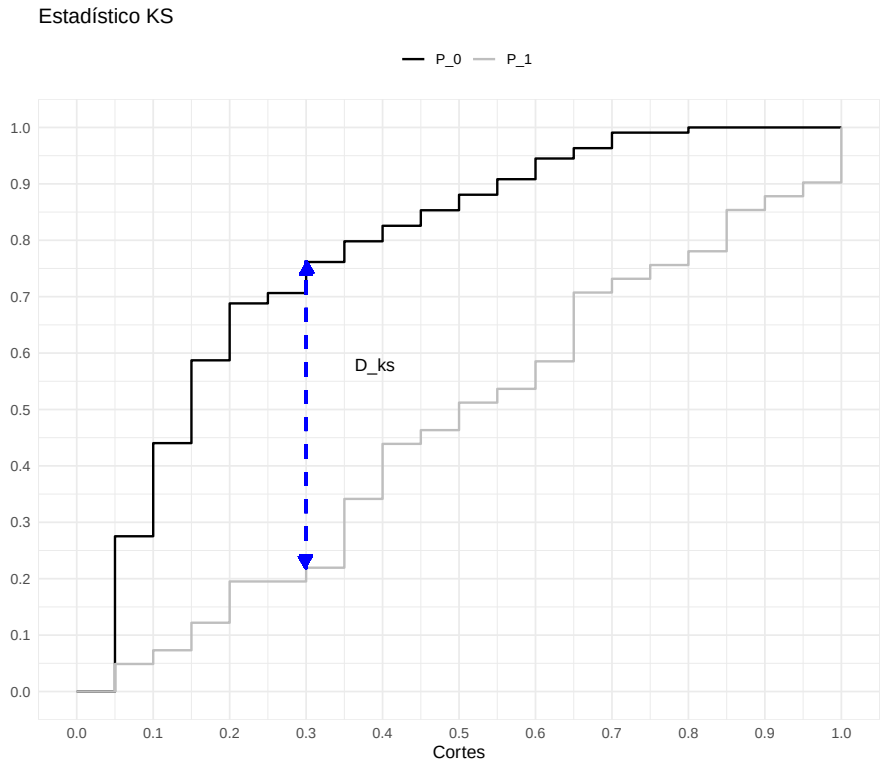


Figura 4.7: Funciones de distribución acumulativas empíricas de dos clases e indicando el estadístico KS.

4.5. Datos Desbalanceados

Se dice que hay desbalance en un conjunto de datos si se tiene una desproporción significante en los ejemplos de las clases de interés, así, alguna clase está sobrerrepresentada, como se muestra en la figura 4.8. Con la presencia de desbalance los modelos de clasificación son no son tan directos de medir, definir límites o fronteras de decisión, esto debido a que tienden a tener una mayor exactitud a la clase mayoritaria y un mal desempeño en clasificar de forma correcta la clase minoritaria (Fernández A. [28]), es decir, existe sesgo de clasificación en el modelo. El grado de desbalance (IR) es la proporción que existe entre la clase mayoritaria y la minoritaria tal como se muestra en la ecuación (4.14). El IR puede ser moderado (IR =10) o extremo (IR >1000) y por ello, se complica el sesgo de clasificación o error que es mayor para clasificar la clase minoritaria.

$$IR = \frac{\#\text{ Clase Mayoritaria}}{\#\text{ Clase Minoritaria}} > 1\ . \tag{4.14}$$

Una ejemplo claro de una métrica errónea para cuando hay presencia de desbalance es el siguiente: Supongamos un clasificador o modelo con un $IR = 16$ en la variable de interés de dos clases explicada por dos variables continuas, digamos *Atributo 1* y *Atributos 2*, cuya métrica de exactitud ha sido del 98 % (figura 4.9), aparentemente esta exactitud puede brindar la idea de ser bueno, no obstante, si se verificara la tasa de verdaderos positivos (*Recall*) muy probamente este sea bajo (en este ejemplo 50 %) debido al sesgo de clasificación hacia la clase mayor, es decir, un mayor número de falsos negativos, de esta manera, es necesario contar con herramientas que ayuden a tratar con esta problemática, ya que a mayor desbalance menor *Recall*.

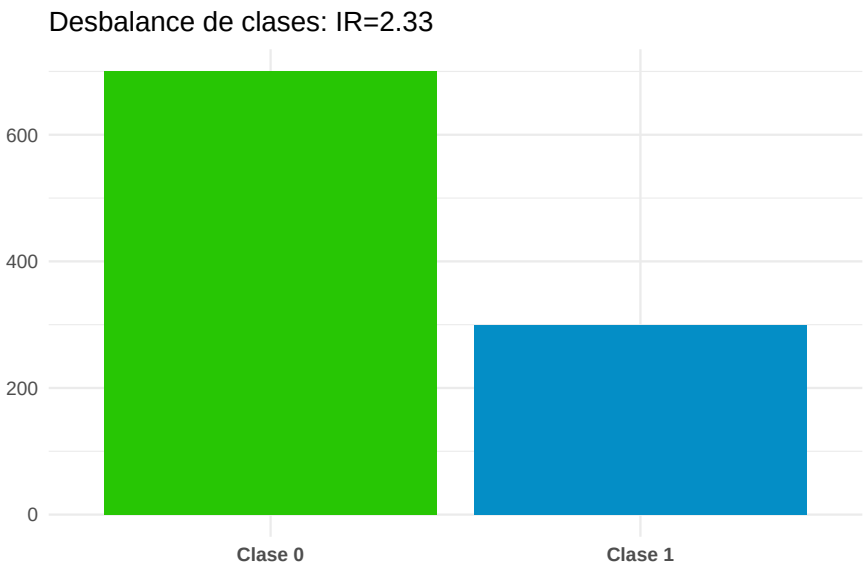


Figura 4.8: Gráfica desbalance las clases, donde se observa claramente la sobrerrepresentación de la clase 0 sobre la clase 1 (80 % y 20 % respectivamente).

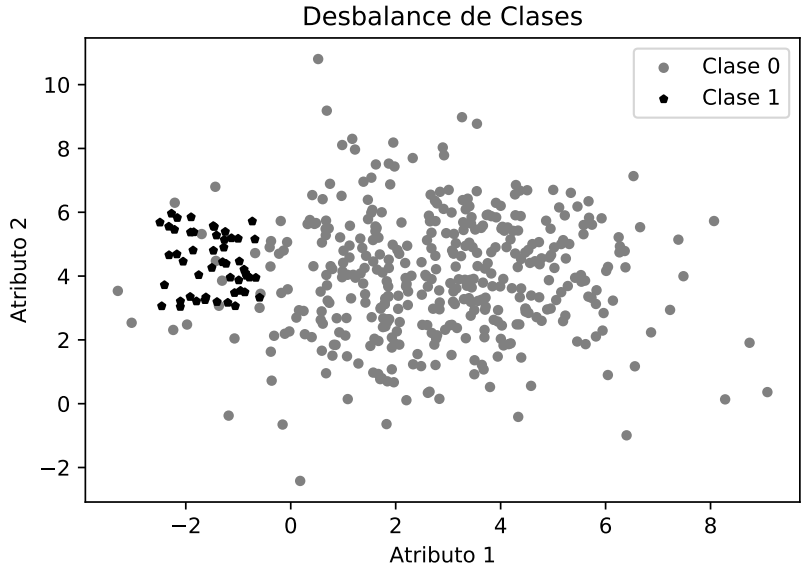


Figura 4.9: Gráfica desbalance de clases con $IR = 16$ con métrica de exactitud del 98 % y Recall del 50 % debido a el desbalance de clases en la variable a explicar.

Para el tratamiento del problema de desbalance en la variable de interés con la información disponible, existen diversos métodos, por ejemplo, dos muy usados son: *Aprendizaje Costo Sensitivo* y *Preprocesamiento de Nivel de Datos* (remuestreo). Donde en este último se trata de balancear de forma artificial las clases, sin embargo, en este trabajo, utilizaremos el *Aprendizaje Costo Sensitivo*.

4.5.1. Aprendizaje Costo Sensitivo

El aprendizaje de Costo Sensitivo penaliza severamente la clasificación errónea (si es de gran interés) en la clase minoritaria, de esta manera se da mayor preponderancia a su correcta clasificación durante el entrenamiento del modelo, es decir, un costo mayor. Entonces, se buscará minimizar el costo de clasificación tanto en el entrenamiento para hallar el mejor modelo como para medir su desempeño final. Básicamente hay dos puntos de vista para este enfoque [10]:

Costos por atributos: Se asigna un vector de costos al vector de atributos de la base de datos, y evaluando el rendimiento del modelo por el mínimo costo. Claramente se puede asumir el enfoque de costo como valor monetario o incluso un beneficio.

Costos por clases: Se asigna costo más alto a aquellas clasificaciones erróneas. Claramente en el entrenamiento del modelo se enfocará en una correcta predicción de la clase con mayor costo. En el caso de un modelo Credit Scoring, se enfocará por predecir de forma correcta la clase positiva, es decir, los clientes malos.

Los costos se expresan mediante una matriz de costos tal como se muestra en la figura 4.10, la cual se puede definir mediante experiencia y/o conocimiento de un experto o se puede estimar mediante los datos de la base de entrenamiento. Esto es, los costos de un Positivo o Negativo Verdadero, o un falso Positivo o Negativo (para mayor detalle véase en [6], [10], y [14]).

		PREDICCIÓN	
		0	1
VALOR REAL	0	C_{TN}	C_{FP}
	1	C_{FN}	C_{TP}

Figura 4.10: Matriz costo para el modelo de clasificación.

Una recomendación para la obtención de una matriz de costos por medio de los datos es la que recomienda Fernández A. [28]: en la cual asigna un costo de 0 para las clasificaciones correctas, un costo de 1 para los *Falsos positivos* y el valor IR para los *Falsos Negativos* como se muestra en la figura 4.11.

		PREDICCIÓN	
		0	1
VALOR REAL	0	*0*	*1*
	1	*IR*	*0*

Figura 4.11: Matriz costo dada por los datos.

Así, para un modelo se puede obtener según Elkan C.[6] el corte óptimo, o bien, el corte del costo mínimo dada la matriz de la figura 4.11 como se muestra en la figura de costo de clasificación 4.12. El método Costo Sensitivo también se puede aplicar a las ANN como función de costo o pérdida, con lo cual su aprendizaje es distinto, presentando cuatro métodos en este, véase en Korolenko I. [14].

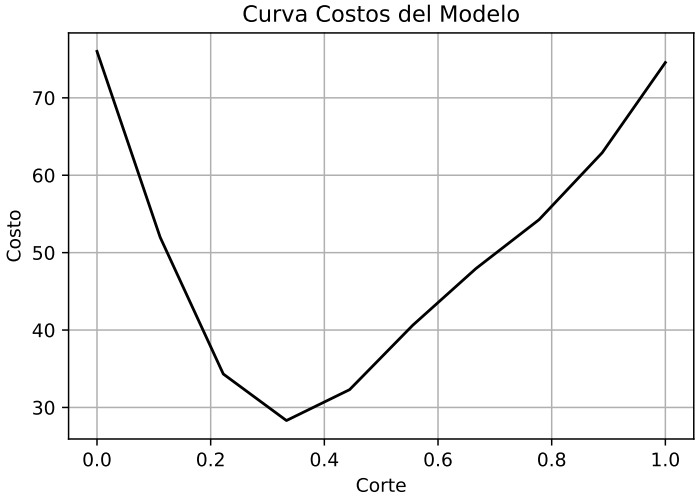


Figura 4.12: Curva costo para cada corte entre $(0, 1)$.

Ante un desbalance la función ajustada para el aprendizaje binario costo-sensitivo para *Log Loss* (ecuación 3.15) es la función *Real-World-Weight Cross Entropy* (ecuación 3.16), quedaría de la siguiente forma si definimos los pesos como los costos de falsos positivos y su costo de acuerdo a su desbalance:

$$\begin{aligned} J_w &= -\frac{1}{n}\sum_{i=1}^{n}\left\{w_1 z^i \log\left(\hat{z}^i\right) + w_0\left(1-z^i\right)\log\left(1-\hat{z}^i\right)\right\} \\ &= -\frac{1}{n}\sum_{i=1}^{n}\left\{C_{FN}\cdot z^i \log\left(\hat{z}^i\right) + C_{FP}\cdot\left(1-z^i\right)\log\left(1-\hat{z}^i\right)\right\} \\ &= -\frac{1}{n}\sum_{i=1}^{n}\left\{IR\cdot z^i \log\left(\hat{z}^i\right) + 1\cdot\left(1-z^i\right)\log\left(1-\hat{z}^i\right)\right\} \\ &= -\frac{1}{n}\sum_{i=1}^{n}\left\{IR\cdot \theta_i \log\left(\hat{\theta}_i\right) + (1-\theta_i)\log\left(1-\hat{\theta}_i\right)\right\} \\ &= J_\theta^{IR}\left(\hat{\boldsymbol{\theta}}\right) . \end{aligned}$$

Por tanto,

$$J_\theta^{IR}\left(\hat{\boldsymbol{\theta}}\right) = -\frac{1}{n}\sum_{i=1}^{n}\left\{IR\cdot \theta_i \log\left(\hat{\theta}_i\right) + (1-\theta_i)\log\left(1-\hat{\theta}_i\right)\right\} . \tag{4.15}$$

Está última función de costo, será de suma importancia en el modelo de Redes Neuronales artificiales debido a que penalizará con un costo distinto las clasificaciones incorrectas, mayormente a nuestra clase 1 de interés, mejorando con ello el aprendizaje de la red, obteniendo el mejor aprendizaje al menor costo.

Capítulo 5

Aplicación Modelos en Credit Scoring

Una vez que se tiene la teoría necesaria acerca de los modelos que se quieren comparar, revisaremos brevemente el tema del proceso de selección de un modelo *Credit Scoring*, un modelo no paramétrico o de estructura de una Red Neuronal Artificial y uno paramétrico o modelo logístico. Dicho proceso toma en cuenta el preprocesamiento de datos, entrenamiento y primera evaluación a fin de recalibrar su estructura o hiperparámetros en el caso de ANN o selección de variables explicativas en base a su evidencia estadística para el caso logístico, esto para su prueba final y la decisión de implementar o no el modelo en la fase final.

Una vez se tengan los mejores modelos representativos de cada enfoque se realizará su comparación analizando su desempeño final, su complejidad de construcción, la interpretabilidad y su implementación en datos reales.

5.1. Proceso modelo Credit Scoring

El Proceso de entrenar un modelo estadístico para Credit Scoring es una importante metodología que brinda los pasos o etapas para hallar el mejor modelo posible en base a los ***Datos Históricos*** internos con los que cuenta la entidad financiera. Esta información pasa una etapa llamada ***Preprocesamiento de datos***, esta fase es muy importante ya que la robustez de un modelo se basa en la calidad de sus datos de entrenamiento, por ello, cuenta con varias etapas información ([28] y [26]):

Selección Es donde se selecciona la información de las diversas fuentes además de la interna para su análisis preliminar, además de definir el tipo de variables con las que se cuenta, es decir, *cuantitativas* o *cualitativas*. Además de realizar las primeras visualizaciones, se debe detectar la información que no ayuda al modelo y tratar los valores perdidos, outliers, etc.

Preparación: En esta parte se analiza el sentido lógico de relación entre los atributos a fin de detectar errores, correlaciones altas, información repetida.

Transformación Dado que cada algoritmo trabaja con diferentes formatos o rangos de información se debe ajustar o transformar la o las variables para su óptimo resultado, por ejemplo, en el caso de las ANN deben tener rasgos pequeños las variables, de lo contario, estos atributos con rangos cortos tienen pesos sesgados.

Reducción Con la finalidad de contar con un algoritmo y entrenamiento más eficiente se reduce la cantidad de variables explicativas que aportan nula o muy poca información a la variable de interés, para ello se puede hacer uso de pruebas estadísticas, por ejemplo, la prueba χ^2 para dependencia entre variables categóricas.

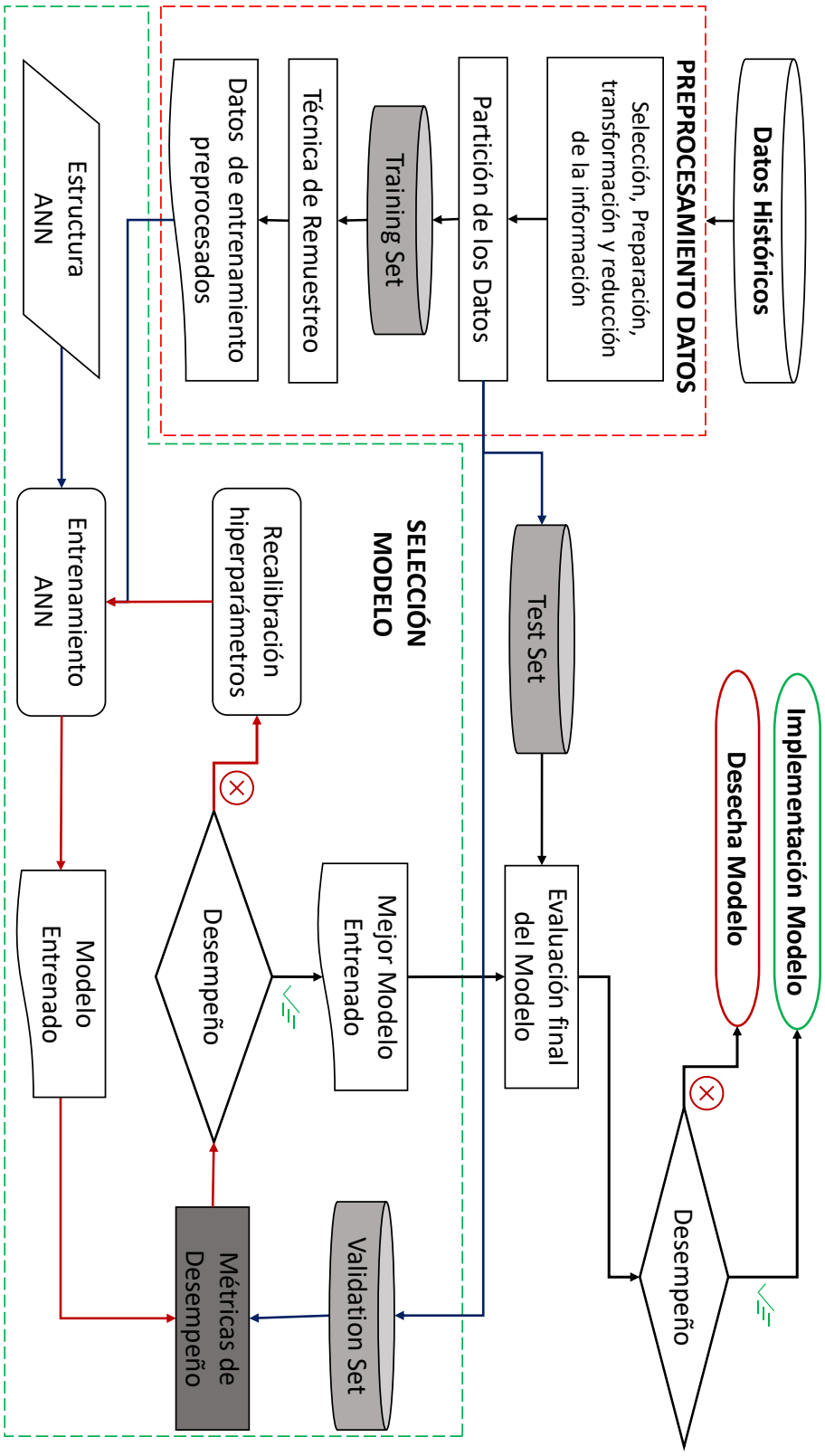


Figura 5.1: Diagrama proceso de modelación Credit Scoring por ANN.

Retomando lo mencionado en la sección 4.1, la segmentación de la totalidad de los datos también se

realiza en la etapa de ***Preprocesamiento de datos*** y se continua con el *set de Entrenamiento* para aplicarle técnicas de remuestreo ante el desbalance en la variable de interés para así pasar a la etapa de ***Selección del Modelo***.

En la etapa ***Selección del Modelo*** se definirá por ejemplo la estructura (ANN) o variables significativas (Logit) con la que se estimarán los parámetros de los modelos para posteriormente evaluar su desempeño preliminar con el *set de Validación* y de no ser éste satisfactorio se recalibrarán los hiperparámetros (ANN), teniendo así un ciclo de entrenamiento y aprendizaje del modelo. Cuando se obtengan resultados preliminares satisfactorios pasará la última etapa: ***Implementación***.

La fase final ***Implementación*** es donde se evalúa el modelo Credit Scoring de forma definitiva su capacidad predictiva generalizada utilizando el *set de Prueba*, de no obtener buenas métricas, dicho modelo ya no se recalibrará sino se descartará de forma definitiva, en caso contrario, se tomará en cuenta para su implementación real. En la figura 5.1 se esquematiza el flujo del proceso de modelación para estos enfoques paramétrico y no paramétrico. En las siguientes secciones se aplicará dicho proceso a cada modelo, por lo que habrá varios candidatos que están en la última fase, descartando a muchos.

5.2. Base de datos

Se usará la base de datos *German Credit Data*, la cual está disponible en la red[1], esta base tiene 1,000 registros con 21 variables: 3 cuantitativas, 17 cualitativas y una dicotómica de respuesta. De esta manera se cuenta con 20 variables explicativas (72 atributos en total), la variable de interés cualitativa *Clase de cliente* Y: que toma el valor de 0 si se considera al cliente *"Bueno"* (700 registros) y 1 cuando es *"Malo"* (300 registros), las demás variables registran datos sociodemográficos, financieros, bancarios, etc. A continuación, se describen las variables, su tipo, rango y nombre:

Nombre de la Variable	**Tipo de variable**	**Espacio muestra**l
X_1: Balance de cuenta	Categórica	Mayor o igual a 0.
X_2: Duración del Crédito meses	Cuantitativa	1 mes-10años.
X_3: Historial de Crédito	Categórica	Sin deuda-Cuentas crítica.
X_4: Propósito del Préstamo	Categórica	Coche(nuevo o usado), negocio, etc.
X_5: Monto del Crédito	Cuantitativa	100-10,000,000.
X_6: Monto en la cuenta de ahorro	Categórica	Mayor o igual a 0.
X_7: Tiempo empleo actual	Categórica	Desempleado, mayor igual 1 año.
X_8: % ingreso para pagar	Cuantitativa	Mayor a 0.
X_9: Sexo y Estado Civil	Categórica	Masculino, Femenino, Casado, etc.
X_{10}: Garantes/Deudores	Categórica	Ninguno, cliente, etc.
X_{11}: Años residencia actual	Cuantitativa	mayor igual 1 año.
X_{12}: Posesiones	Categórica	Inmuebles, automóvil, etc.
X_{13}: Edad	Categórica	Años cumplidos.
X_{14}: Otros créditos	Categórica	Banco, tiendas, ninguno.
X_{15}: Alojamiento	Categórica	Alquiler, propio, gratis.
X_{16}: No. Créditos en el banco	Cuantitativa	Mayor o igual a 0.
X_{17}: Trabajo	Categórica	Empleado, desempleado, etc.
X_{18}: No. Beneficiarios	Cuantitativa	Mayor o igual a 0.
X_{19}: Teléfono	Categórica	Ninguno, sí a nombre cliente.
X_{20}: Trabajador extrajeron	Categórica(binaria)	Sí, No.
Y: Clase Cliente	Categórica(binaria)	Bueno o Malo.

Cuadro 5.1: Data Set *German Credit Data*.

[1] `http://archive.ics.uci.edu/ml/datasets/Statlog`

5.2.1. Variables de la base de datos

Cabe mencionar que esta base cuenta con un proceso previo de *selección*, *preparación* y algunas por *transformación*, por lo que solo se transformarán las variables numéricas para reducir su rango; *monto del crédito*, *edad* y *duración del crédito*. En las figuras (5.2-5.21) se muestran las gráficas de distribución de las variables contenidas en la base de datos. En cada gráfica se puede apreciar que la variable de interés tiene desbalance, $IR = \frac{700}{300} = \frac{7}{3} = 2.33$, por lo que se aplicará la función de costo *Real-World-Weight Cross Entropy* (ecuación 4.15) de tipo *Aprendizaje Costo Sensitivo* afrontar este problema.

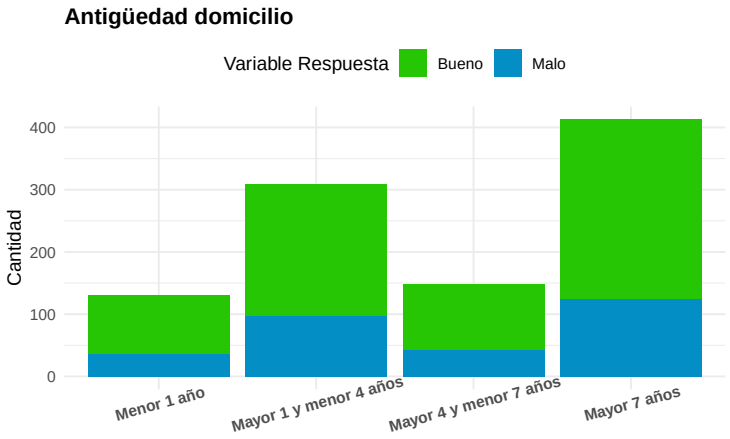


Figura 5.2: Gráfica variable Antigüedad domicilio.

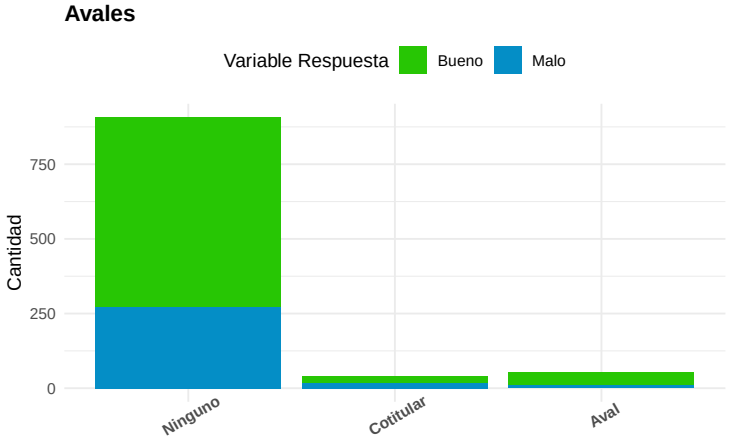


Figura 5.3: Gráfica variable Avales en solicitud.

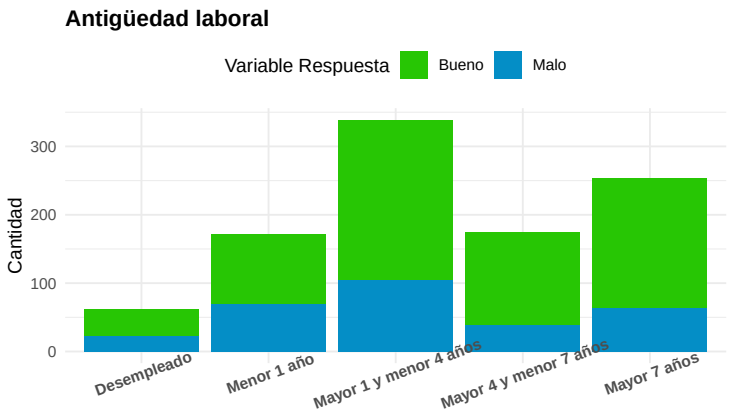


Figura 5.4: Gráfica variable Antigüedad laboral.

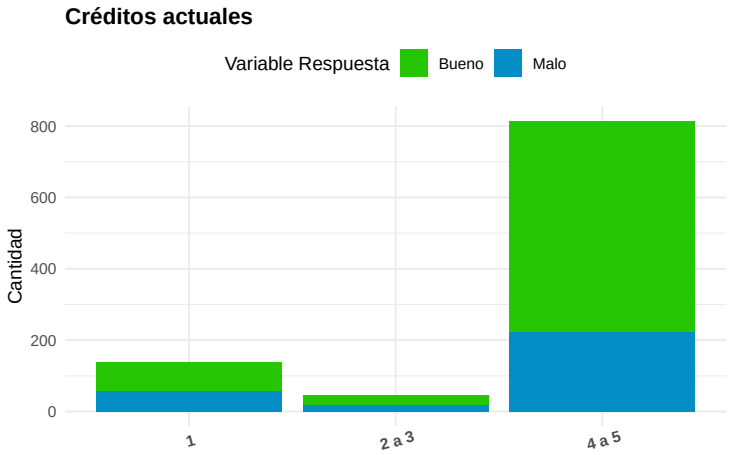


Figura 5.5: Gráfica variable Créditos actuales.

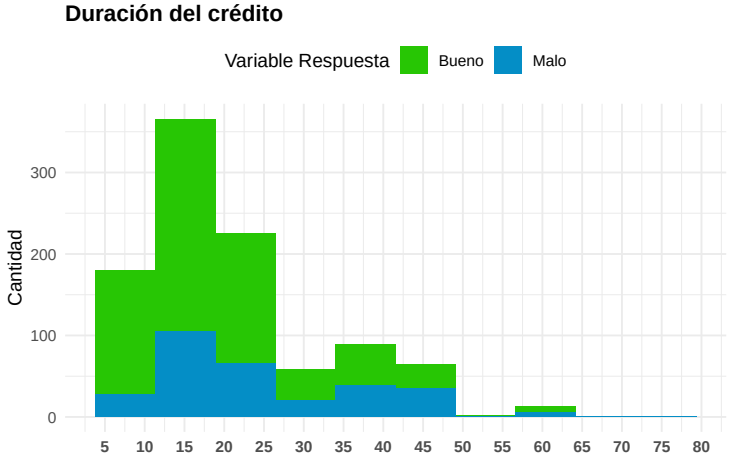


Figura 5.6: Gráfica variable Duración del crédito.

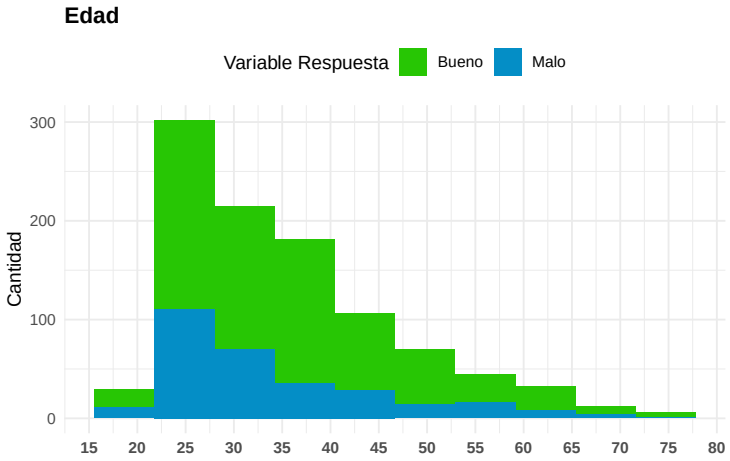


Figura 5.7: Gráfica variable Edad.

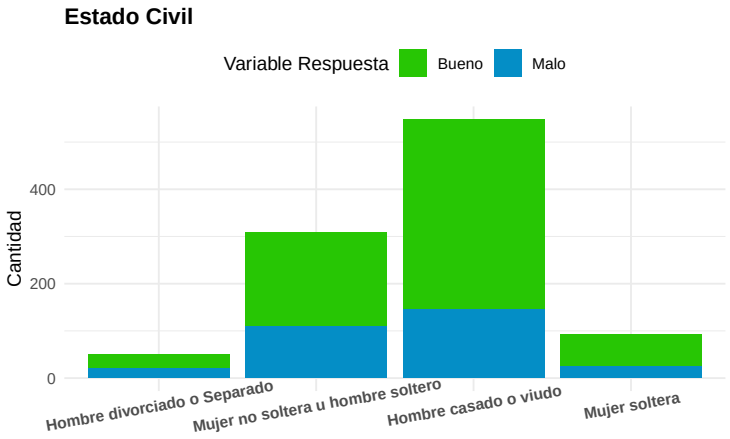


Figura 5.8: Gráfica variable Estado Civil.

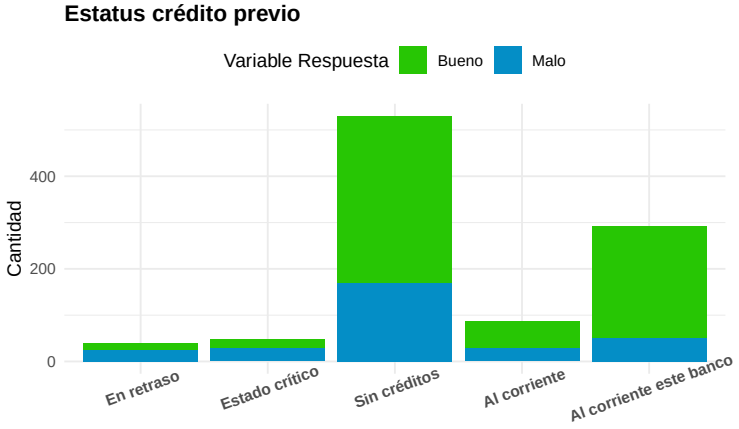


Figura 5.9: Gráfica variable Estatus crédito previo.

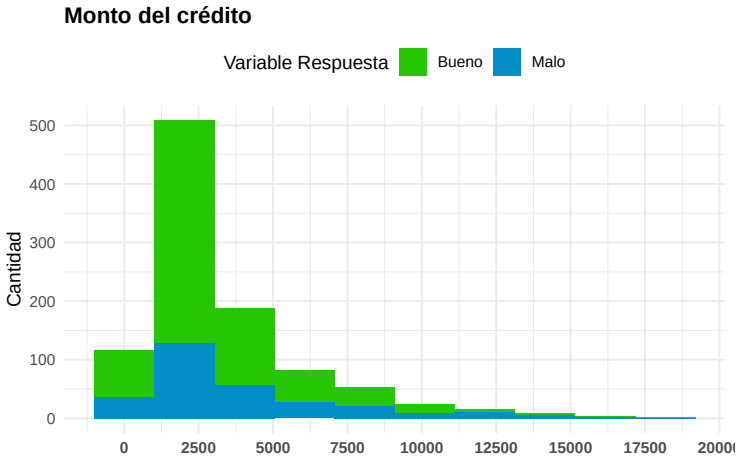


Figura 5.10: Gráfica variable Monto del crédito.

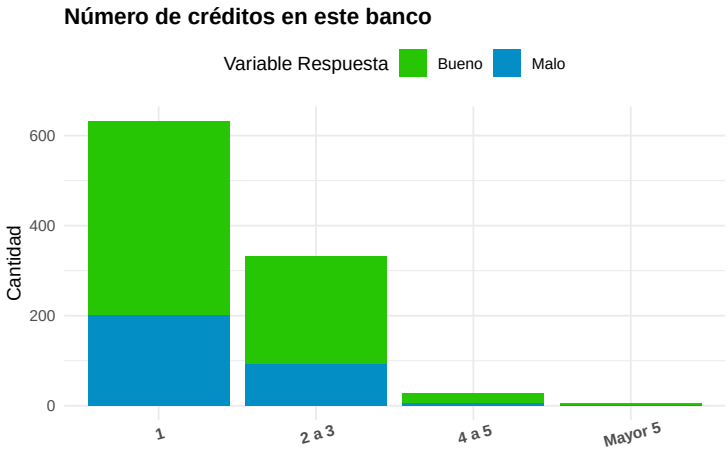


Figura 5.11: Gráfica variable Número de créditos en este banco.

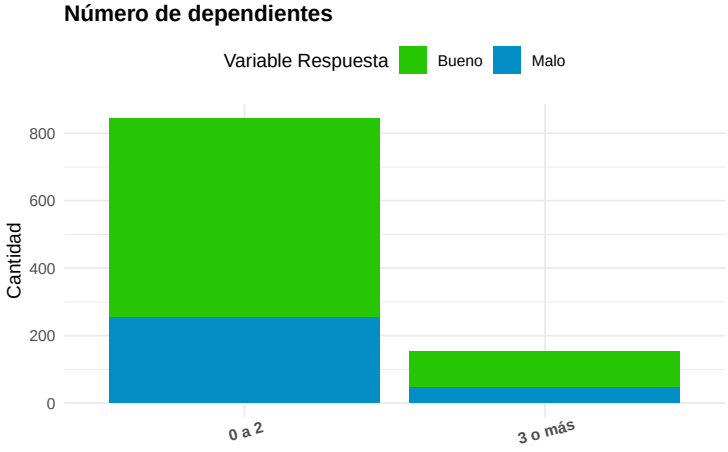


Figura 5.12: Gráfica variable Número de dependientes.

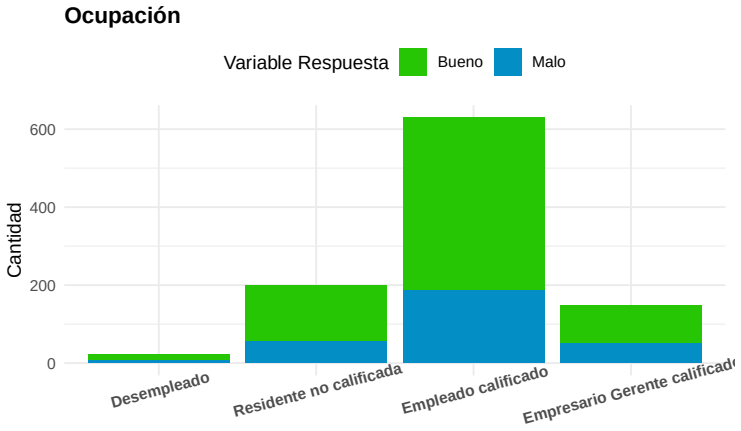


Figura 5.13: Gráfica variable Ocupación.

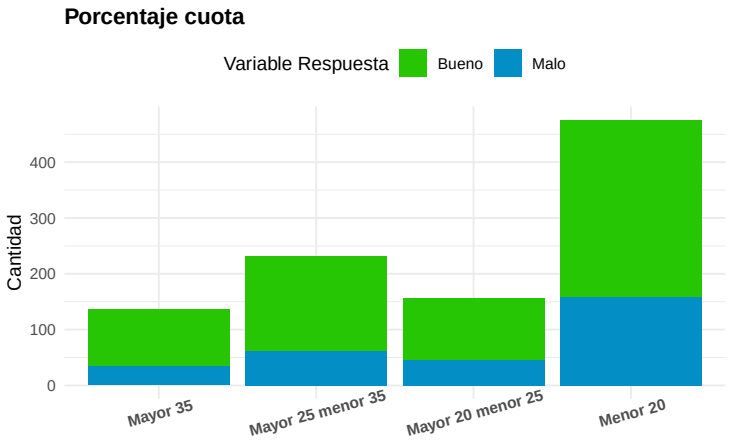


Figura 5.14: Gráfica variable Porcentaje cuota.

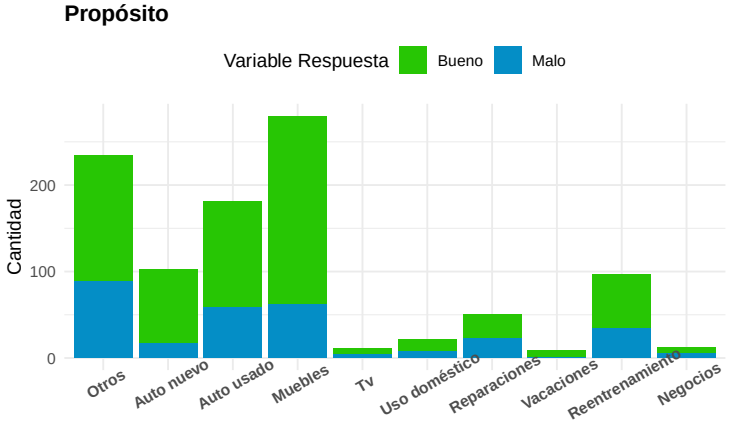


Figura 5.15: Gráfica variable Propósito.

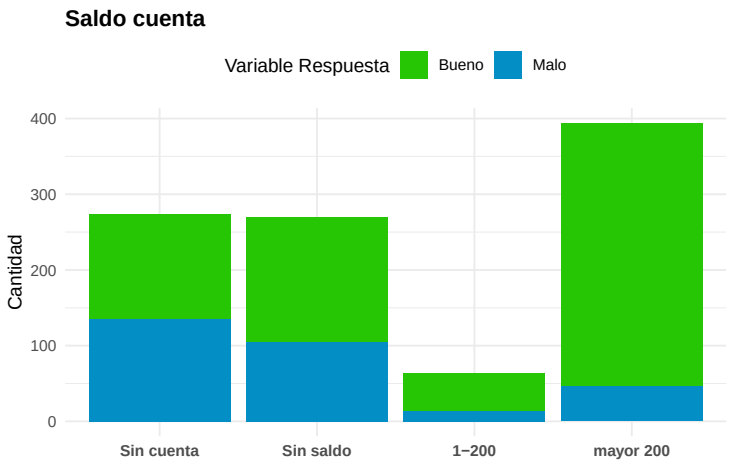


Figura 5.16: Gráfica variable Saldo cuenta.

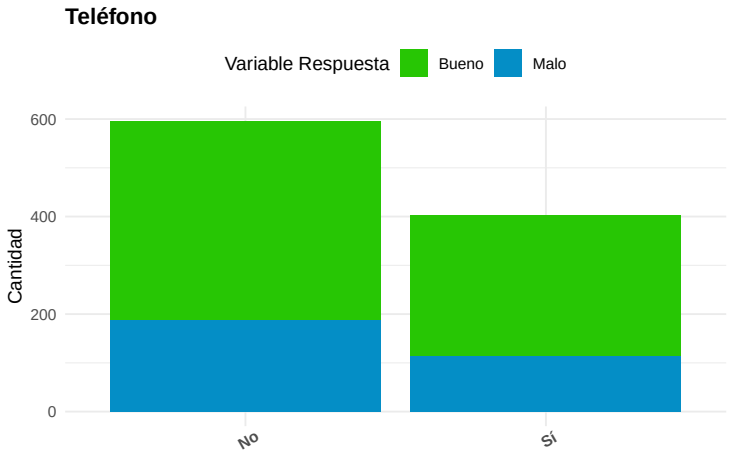


Figura 5.17: Gráfica variable Teléfono.

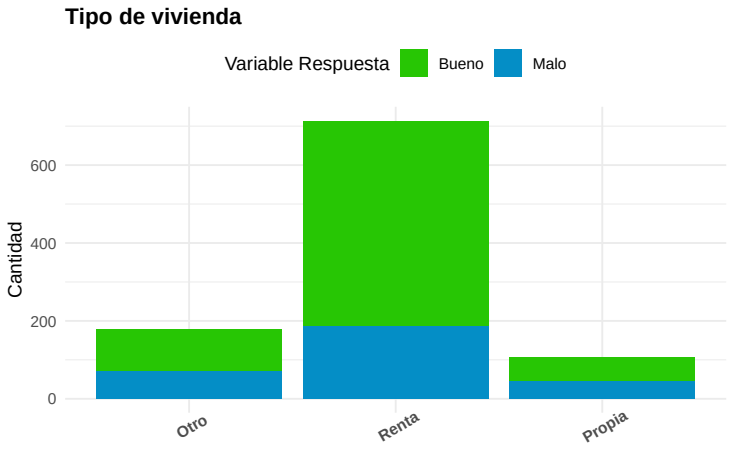


Figura 5.18: Gráfica variable Tipo de vivienda.

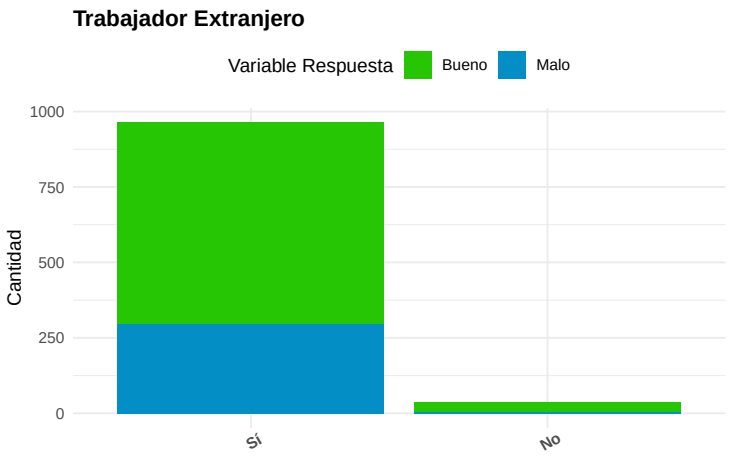


Figura 5.19: Gráfica variable Trabajador Extranjero.

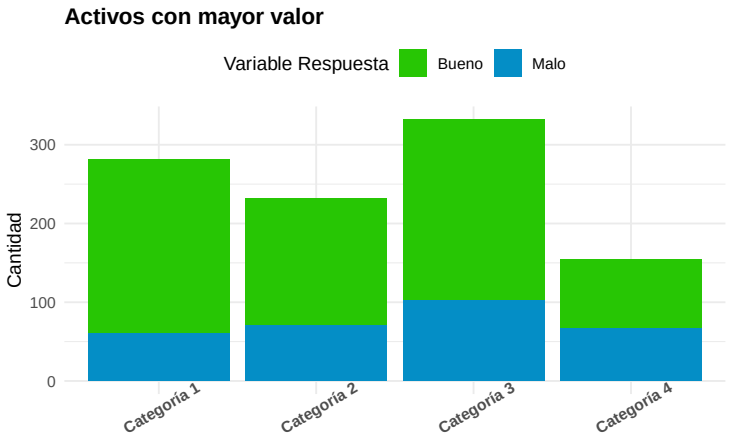


Figura 5.20: Gráfica variable Activos con mayor valor.

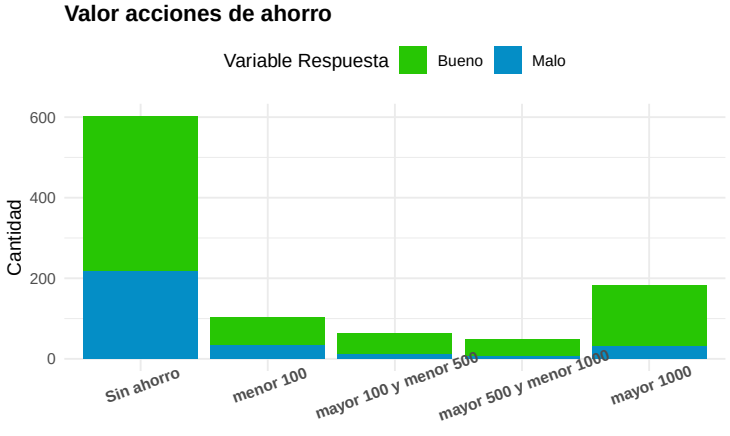


Figura 5.21: Gráfica variable Valor acciones de ahorro.

5.2.2. Partición Base de datos

Como se mencionó en la sección *4.1*, es necesario la partición del dataset para entrenar, validar y seleccionar cualquier modelo Credit Scoring, por lo que para la comparación de modelo logístico y por redes neuronales artificiales es primordial realizar dicha segmentación. Para el entrenamiento, validación y selección de los modelos se segmentará de forma aleatoria y estratificada sobre la variable de interés de la base antes mencionada de la siguiente forma:

Entrenamiento 70 %.

Validación 15 %.

Prueba 15 % .

Recuérdese que para esta base se debe tomar en cuenta que $IR = 2.33$, de esta manera se tiene la siguiente matriz de costo definida por los datos:

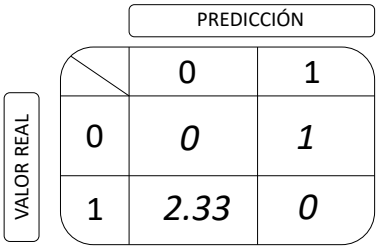


Figura 5.22: Matriz de costo experta, con base al grado de desbalance $IR = 2.33$.

Por otro lado , se tiene de antemano la siguiente distribución de la variable de interés una vez realizada la partición:

Entrenamiento $Y = 0$: 486 y $Y = 1$: 214.

Validación $Y = 0$: 105 y $Y = 1$: 45.

Prueba $Y = 0$: 109 y $Y = 1$: 41.

De lo anterior, claramente se observa el grado de desbalance (IR) se mantiene aproximadamente con esta segmentación, reto que tendrá que enfrentar cada modelo entrenado en cada fase de prueba.

5.3. Aplicación Modelo Logístico a Credit Scoring

Siguiendo el esquema del proceso de modelación Credit Scoring (5.1), se proponen varios modelos logísticos que se han entrenado con *set de entrenamiento* utilizando el Software libre ***R***. Dado que para un modelo logístico se prefiere el más parsimonioso, se toma como primer filtro aquellos modelos con menor *Criterio de información de Akaike* (AIC) tal como se muestra en la tabla 5.2, estos posteriormente serán evaluados en cuanto a su desempeño con las métricas del capítulo anterior con el *set de validación*. En dicha tabla indica el número de variables de cada modelo y su correspondiente número de atributos.

De los modelos de la tabla (5.2), en la tabla (5.3) se indican las métricas de evaluación de los modelos utilizando el *set de validación*: *Área debajo de la curva ROC* (AUC), el estadístico *KS*, el *Costo mínimo* de clasificación, así como su correspondiente *corte óptimo* y la métrica de *Exactitud*. Posteriormente, en la tabla 5.4 se desglosa su capacidad de predicción individual de las clases mediante las métricas: *Precisión*,

Recall y F_β Score. Cabe mencionar que el *cut-off* es el obtenido de la fase de entrenamiento y está siendo aplicado en la fase de validación y será el mismo que se aplicará en la fase de test o prueba. Estos primeros resultados ya brindan un panorama de desempeño de cada uno de estos modelos.

Entrenamiento

Entrenamiento: Modelo Logístico	No. Variables	No. Atributos	AIC
log_ cs	20	68	714
log_ cs2	19	64	709.85
log_ cs3	18	60	705.17
log_ cs4	17	58	703.21
log_ cs5	16	56	703.74
log_ cs6	15	52	701.28
log_ cs7	14	49	699.96
log_ cs8	13	46	699.03
log_ cs9	12	41	699.15
log_ cs10	12	41	699.15
log_ cs11	12	29	827.23

Cuadro 5.2: Se indica el nombre del modelo (compuesto: log=logístico, cs=Credit Scoring , n=versión), estos son los 11 modelos propuestos de la fase de entrenamiento con menor *AIC*, también se indica el número de variables explicativas y el número total de atributos de las variables categóricas asociadas.

Validación

Validación: Modelo Logístico	Est. KS	AUC	Costo Min.	Cut-off	Exactitud
log_ cs	0.37168	0.70222	68.27	0.4545	0.71333
log_ cs2	0.36868	0.70646	68.28	0.3838	0.68667
log_ cs4	0.35921	0.70307	67.60	0.4848	0.72667
log_ cs3	0.35837	0.70434	68.60	0.4848	0.72000
log_ cs7	0.35038	0.69376	70.94	0.4040	0.68667
log_ cs5	0.34206	0.69333	68.95	0.3434	0.67333
log_ cs8	0.34003	0.69693	66.61	0.4343	0.70667
log_ cs6	0.33174	0.69503	70.60	0.4747	0.70667
log_ cs9	0.33132	0.69587	69.62	0.3333	0.66000
log_ cs10	0.33132	0.69587	69.62	0.3333	0.66000
log_ cs11	0.17986	0.62899	78.27	0.3434	0.64667

Cuadro 5.3: Métricas de desempeño en set de Validación. Ordenando por la métrica KS, el modelo log_ cs tiene el mayor KS combinado con un aceptable AUC y un costo de clasificación menor a la media (69.76). Por otro lado, el peor modelo identificable es log_ cs11 debido a su bajo KS 0.17986 y su alto costo de clasificación 78.27, sin mencionar baja exactitud.

```
#########################   MODELO log_cs   ############################



#Se carga el set de entrenamiento
df_train<-leerCSV("df_trainlog")

#Se transforman a variables categoricas
for(j in 1:18){
  df_train[,j]<-as.factor(df_train[,j])
}
```

```
log_cs<-glm(Y~.,family=binomial(link ="logit") ,data=df_train)
summary(log_cs)
```

```
##
## Call:
## glm(formula = Y ~ ., family = binomial(link = "logit"), data = df_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3831  -0.6652  -0.3475   0.5974   2.8112
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                         0.96813    1.22912   0.788 0.430893
## Account_Balance2                   -0.25394    0.27594  -0.920 0.357442
## Account_Balance3                   -0.62604    0.45992  -1.361 0.173452
## Account_Balance4                   -1.61636    0.29167  -5.542 3.00e-08 ***
## Payment_Status_of_Previous_Credit1 -0.01010    0.73113  -0.014 0.988983
## Payment_Status_of_Previous_Credit2 -0.76326    0.53446  -1.428 0.153263
## Payment_Status_of_Previous_Credit3 -0.97040    0.57752  -1.680 0.092902 .
## Payment_Status_of_Previous_Credit4 -1.52072    0.53023  -2.868 0.004130 **
## Purpose1                           -1.49773    0.45447  -3.296 0.000982 ***
## Purpose2                           -1.03033    0.31799  -3.240 0.001195 **
## Purpose3                           -1.11970    0.31058  -3.605 0.000312 ***
## Purpose4                           -1.47790    1.05583  -1.400 0.161588
## Purpose5                            0.22513    0.66960   0.336 0.736708
## Purpose6                            0.55111    0.47636   1.157 0.247306
## Purpose8                          -15.49814  490.22318  -0.032 0.974780
## Purpose9                           -0.89828    0.41444  -2.167 0.030200 *
## Purpose10                          -1.29345    0.96390  -1.342 0.179629
## Value_Savings_Stocks2              -0.02775    0.35075  -0.079 0.936940
## Value_Savings_Stocks3              -0.45215    0.52170  -0.867 0.386113
## Value_Savings_Stocks4              -1.61672    0.61402  -2.633 0.008464 **
## Value_Savings_Stocks5              -1.12807    0.32238  -3.499 0.000467 ***
## Length_of_current_employment2      -0.07741    0.53142  -0.146 0.884190
## Length_of_current_employment3      -0.15504    0.50107  -0.309 0.756998
```

```
## Length_of_current_employment4       -0.99375    0.55297  -1.797 0.072316 .
## Length_of_current_employment5       -0.25093    0.49635  -0.506 0.613175
## Instalment_per_cent2                 0.92591    0.39423   2.349 0.018840 *
## Instalment_per_cent3                 1.18781    0.43565   2.727 0.006400 **
## Instalment_per_cent4                 1.75737    0.40191   4.372 1.23e-05 ***
## Sex_Marital_Status2                 -0.42758    0.46203  -0.925 0.354739
## Sex_Marital_Status3                 -1.12564    0.45037  -2.499 0.012442 *
## Sex_Marital_Status4                 -0.69909    0.56080  -1.247 0.212544
## Guarantors2                          0.43748    0.50639   0.864 0.387626
## Guarantors3                         -1.41710    0.57767  -2.453 0.014162 *
## Duration_in_Current_address2         0.85386    0.36932   2.312 0.020777 *
## Duration_in_Current_address3         0.55499    0.41670   1.332 0.182902
## Duration_in_Current_address4         0.34260    0.38146   0.898 0.369112
## Most_valuable_available_asset2       0.45628    0.31938   1.429 0.153112
## Most_valuable_available_asset3       0.05970    0.29688   0.201 0.840625
## Most_valuable_available_asset4       0.78222    0.51963   1.505 0.132239
## Concurrent_Credits2                 -0.26977    0.53418  -0.505 0.613552
## Concurrent_Credits3                 -0.43512    0.30135  -1.444 0.148771
## Type_of_apartment2                  -0.48821    0.29196  -1.672 0.094489 .
## Type_of_apartment3                  -0.62655    0.59268  -1.057 0.290444
## No_of_Credits_at_this_Bank2          0.32930    0.30030   1.097 0.272823
## No_of_Credits_at_this_Bank3          0.36698    0.75477   0.486 0.626814
## No_of_Credits_at_this_Bank4          0.25795    1.08339   0.238 0.811804
## Occupation2                          0.19267    0.88999   0.216 0.828610
## Occupation3                         -0.08211    0.85479  -0.096 0.923476
## Occupation4                         -0.31487    0.86355  -0.365 0.715395
## No_of_dependents2                    0.56296    0.30587   1.840 0.065698 .
## Telephone2                           0.04575    0.25119   0.182 0.855476
## Foreign_Worker2                     -1.01769    0.68906  -1.477 0.139697
## duration                             1.48600    0.80639   1.843 0.065360 .
## credit_amoun                         2.79988    1.02275   2.738 0.006189 **
## edad                                -1.50759    0.66824  -2.256 0.024066 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 861.88  on 699  degrees of freedom
## Residual deviance: 604.40  on 645  degrees of freedom
## AIC: 714.4
##
## Number of Fisher Scoring iterations: 14
```

Validación: Modelo Logístico	Precisión	Recall	F_β Score
log_ cs	0.52000	0.57778	0.71738
log_ cs2	0.48333	0.64444	0.69700
log_ cs4	0.54348	0.55556	0.72751
log_ cs3	0.53191	0.55556	0.72170
log_ cs7	0.48214	0.60000	0.69507
log_ cs5	0.46875	0.66667	0.68556
log_ cs8	0.50909	0.62222	0.71400
log_ cs6	0.51020	0.55556	0.71006
log_ cs9	0.45588	0.68889	0.67369
log_ cs10	0.45588	0.68889	0.67369
log_ cs11	0.43333	0.57778	0.65832

Cuadro 5.4: Rendimiento de predicción de las clases en el set de Validación. En esta tabla se puede observar a mayor detalle qué modelos tienen el mejor rendimiento en la clasificación correcta de la variable de interés: log_ cs10 y log_ cs9, debido a su resultado en el Recall, sin embargo, presenta baja precisión, esto quiere decir que se están clasificando muchos buenos como malos. Por otro lado, el modelo log_ cs4 presenta un mejor balance entre el Recall y Precisión.

Test

De la etapa de validación se toman los 5 modelos como mayor estadístico *KS*, estos se evalúan con el *Set de Prueba o test* y en la tabla 5.5 se registran los resultados de las principales métricas de interés. En dicha tabla se puede directamente observar que las métricas en general no son malas dado que el AUC, KS, Exactitud y F_β están alrededor del 0.80, 0.40, 0.75 y 74 respectivamente, además de un costo no mayor a 65 unidades, es decir, ligeramente mayor a los resultantes en la fase de validación.

Lo anterior se puede ver mejor en la tabla 5.6, donde se muestran las diferencias proporcionales entre el desempeño en el set de prueba respecto al de validación. Se puede ver con mayor claridad que mantuvo aproximadamente el mismo rendimiento en la mayoría de las métricas, no obstante, para 4 de estos modelos donde ha disminuido el Recall hasta en un 17 %. Por otro lado, se muestran incrementos ligeros en el KS, lo que indica un buen modelo. El modelo log_ cs destaca porque además de tener el mejor KS, también presentó el mejor Recall, dado que tuvo el mayor incremento, (35.08 %), por lo que se puede proponer como el mejor modelo.

Modelo	KS	AUC	Costo Real	Exactitud	Precisión	Recall	F_β Score
log_ cs	0.44121	0.79705	62.60	0.74667	0.52459	0.78049	0.75871
log_ cs2	0.42550	0.80533	61.94	0.74667	0.53488	0.56098	0.74851
log_ cs4	0.41947	0.80756	64.93	0.75333	0.55556	0.48780	0.74809
log_ cs3	0.41407	0.80868	64.93	0.75333	0.55556	0.48780	0.74809
log_ cs8	0.39515	0.81517	60.60	0.77333	0.60000	0.51220	0.76742

Cuadro 5.5: Rendimiento de los modelos con el set de Prueba (Test Set), resultados ligeramente superiores a los de la fase de Validación.

Modelo	d-KS	d-AUC	d-Costo Real	d-Exactitud	d-Precisión	d-Recall	d-F_β Score
log_ cs	0.1870	0.1350	0.0830	0.0467	0.0088	0.35080	0.0576
log_ cs2	0.1541	0.1399	0.0928	0.0873	0.1066	-0.1290	0.0739
log_ cs4	0.1677	0.1486	0.0394	0.0366	0.0222	-0.1210	0.0282
log_ cs3	0.1554	0.1481	0.0534	0.0462	0.0444	-0.1210	0.0365
log_ cs8	0.1621	0.1696	0.0902	0.0943	0.1785	-0.1760	0.0748

Cuadro 5.6: Modelos con mayor KS en la fase de validación y se muestran las diferencias proporcionales entre el desempeño en el set de prueba respecto al de validación.

Modelo: *log_cs*

A continuación, revisaremos de forma gráfica En la figura 5.23 se muestra la gráfica de la distribución distinguida entre malos y buenos de las predicciones del modelo ***log_ cs***, de esta manera, sabiendo que el corte óptimo para el mínimo costo obtenido de la fase de entre entrenamiento fue 0.4545, se puede tener una idea de las clasificaciones: verdaderos positivos y negativos, así como falsos positivos y negativos. En la figura 5.24 se muestra la gráfica del comportamiento

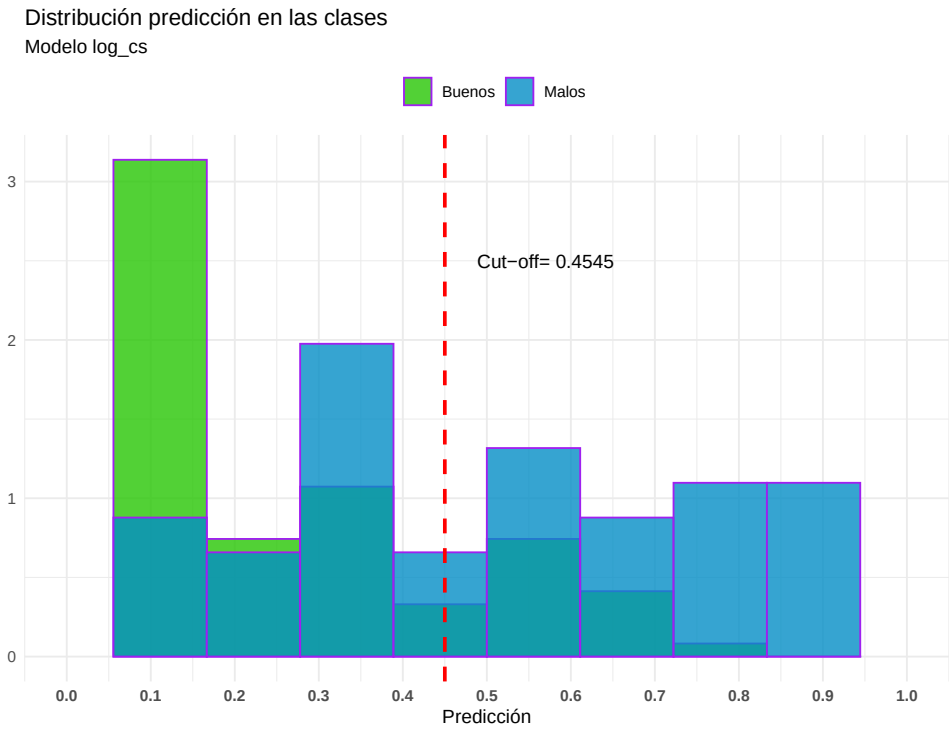


Figura 5.23: Histograma de distribución de la predicción del modelo ***log_ cs***, distinguiendo por color la etiqueta verdadera de buen o mal cliente. Para este modelo se indica el corte 0.4545, éste es donde se encuentra el costo menor de predicción. Aquellos malos a la izquierda de dicho corte serían los falsos negativos, de forma análoga, los buenos de lado derecho serían los falsos positivos.

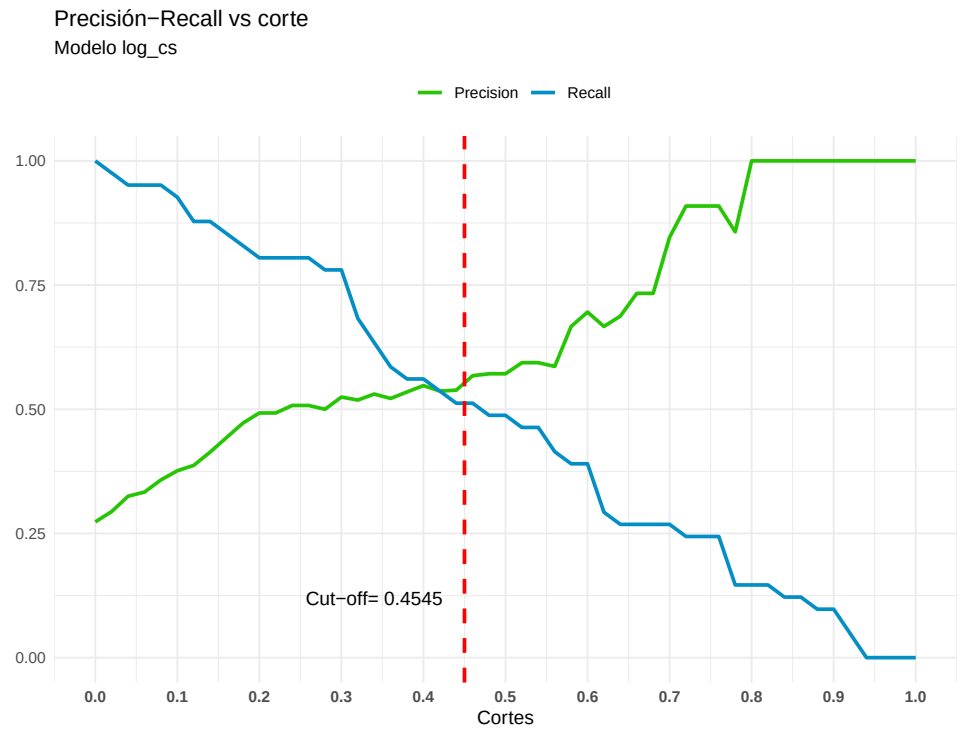


Figura 5.24: Gráfica Precisión-Recall del modelo ***log_cs***, en ella se observa el comportamiento de las métricas Precisión y Recall para cada corte.

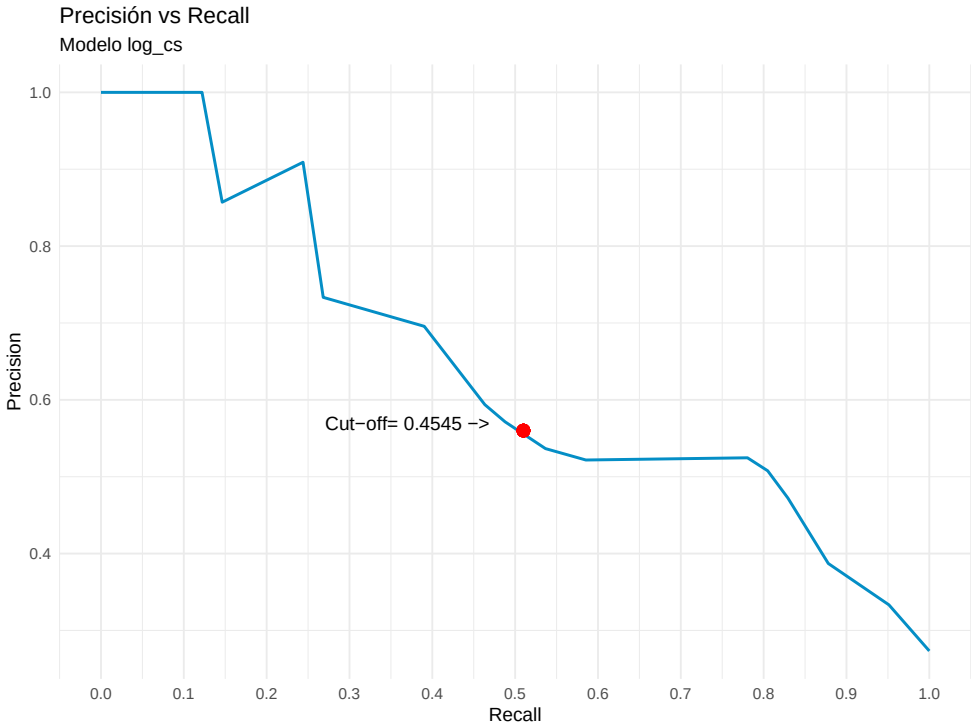


Figura 5.25: La predicción del modelo ***log_cs*** , el punto (Recall, Precision) para cada corte.

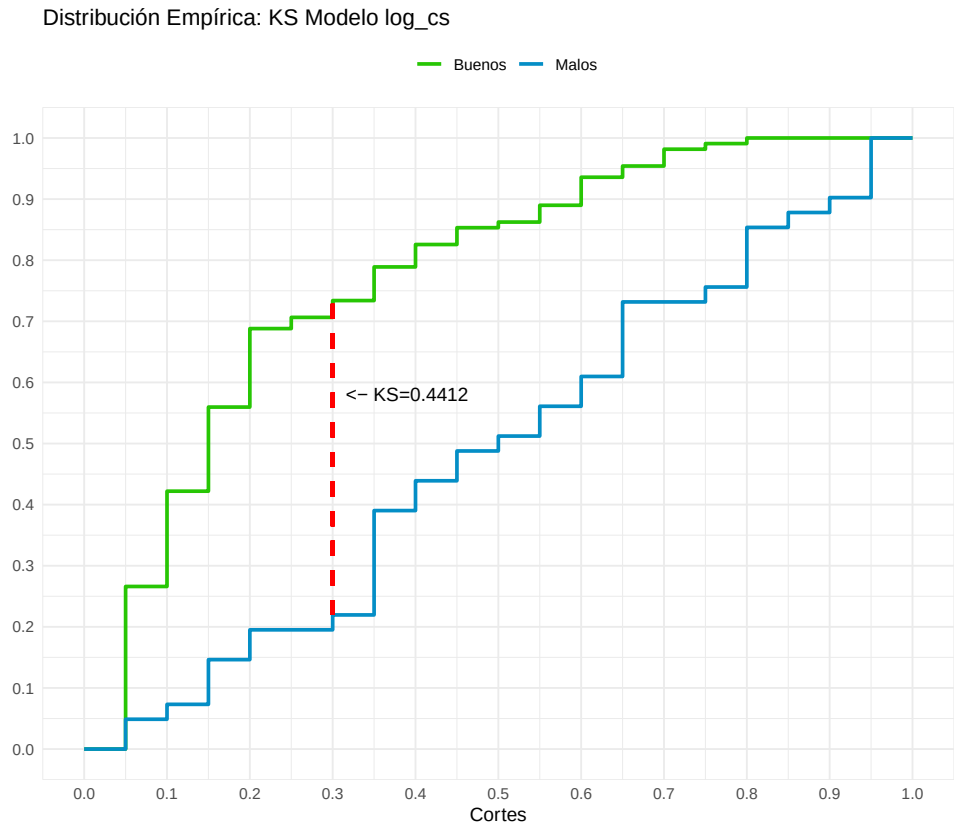


Figura 5.26: Estadístico KS del modelo ***log_cs*** , donde ser observa la distancia máxima y el corte correspondiente.

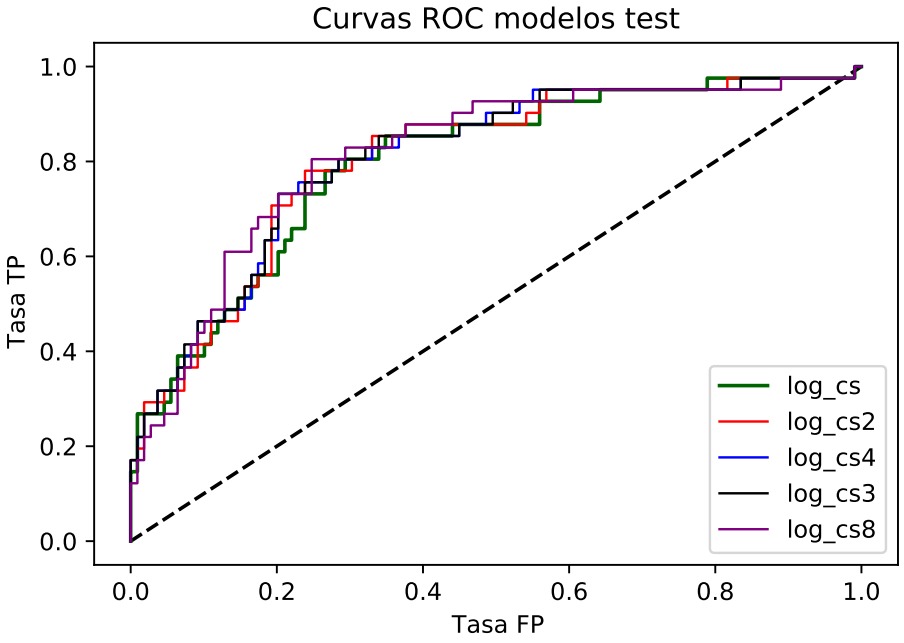


Figura 5.27: Curvas ROC de los modelos de Validación.

Por tanto, analizando las gráficas como los resultados en las métricas de prueba se ha decido elegir los primeros 3 modelos de la tabla 5.6, estos modelos logísticos serán comparados con las ANN.

Modelo 1: log_ cs.

Modelo 2: log_ cs2.

Modelo 3: log_ cs4.

5.4. Aplicación Modelo ANN a Credit Scoring

En esta sección se implementa la búsqueda de algunos modelos no paramétricos que también clasifiquen de forma óptima a los clientes. Para ello se aplicará una malla de hiperparámetros que brindará información para encontrar los mejores y así elegir un mejor modelo de redes neuronales artificiales (estructura ANN), dichas redes son tipo secuenciales y cada capa con la misma cantidad de neuronas. Se utiliza el lenguaje de programación *Python* con las librerías principales *TensorFlow* y *Keras*. El modelo tiene la forma

$$c_E - h_1 - h_2 - ... - h_n - c_S \ .$$

Donde c_E, c_S son las capas de entrada y salida respectivamente, y h_i es la i-ésima capa oculta. Por tanto, la capa c_E recibirá los datos y en la capa c_S se utilizará una sola neurona de salida con la función de activación sigmoide de la ecuación (3.4), la cual se interpretará como la probabilidad de incumplimiento. Los hiperparámetros que se utilizarán en dicha malla son:

Número de capas: Se validará la capacidad de predicción generalizada para 1-4 capas ocultas.

- $c_E - h_1 - sig_S$.
- $c_E - h_1 - h_2 - sig_S$.
- $c_E - h_1 - h_2 - h_3 - sig_S$.
- $c_E - h_1 - h_2 - h_3 - h_4 - sig_S$.

Función de activación: Se utilizará las funciones de activación más comunes para clasificación por ANN.

- *Relu* (3.6).
- *Leaky Relu* (3.7).
- *Elu* (3.8).
- *Selu* (3.9).

Número de neuronas en cada capa: En la cuestión de número de neuronas se tomarán los siguientes valores, de menos complejo a más compleja.

- *71* neuronas.
- *90* neuronas.
- *100* neuronas.
- *120* neuronas.
- *150* neuronas.
- *200* neuronas.
- *250* neuronas.

Tasa de aprendizaje: Por último se tomarán únicamente tres tasas:

- *0.0001* .
- *0.0003* .
- *0.0005* .

5.4.1. Estructura ANN

Una vez ejecutado los modelos con los hiperparámetros de la malla anterior y con el objetivo de hallar estructuras óptimas para algunos modelos de redes neuronales artificiales se realizan los siguientes resultados preliminares. En la figura 5.28 se puede observar que la mejor capacidad de predicción generalizada se encuentra para cuando se tiene de 1 a 3 capas ocultas, por lo que para 2 capas ocultas se podría tener un mejor desempeño. Se puede observar que para cuando la red neuronal tiene 4 capas ocultas el error de validación ya es muy divergente al error de entrenamiento, de esta manera, se puede descartar el tener ésta última estructura, por lo que en adelante solo se realizará el análisis para 1 a 3 capas ocultas.

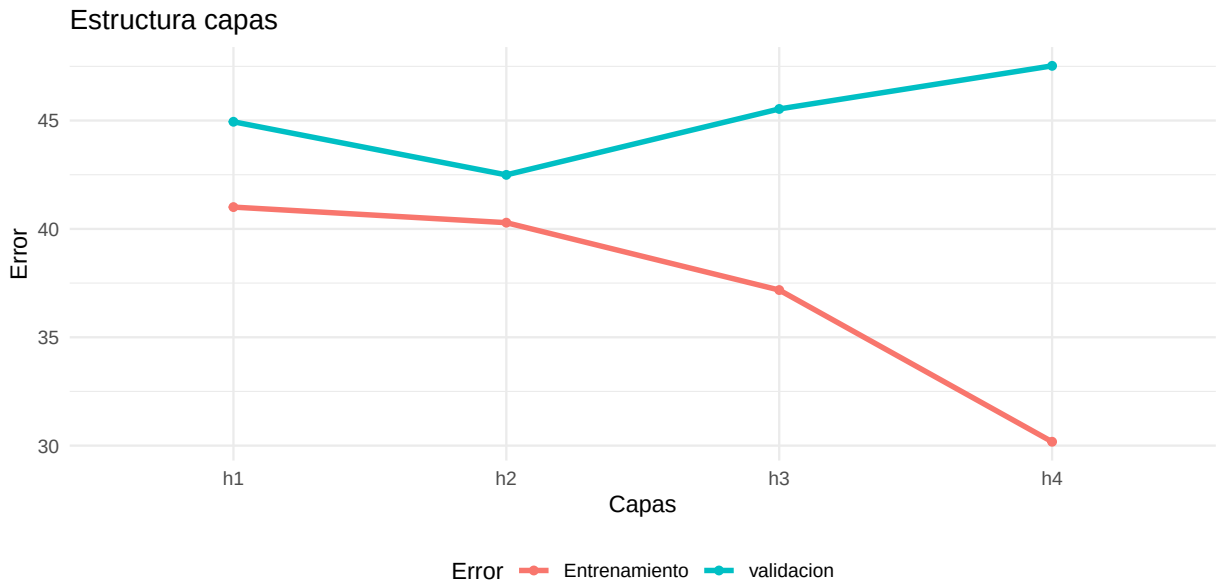


Figura 5.28: Aproximación del Error general dada la cantidad de capas ocultas.

Funciones de activación

Con la finalidad de también tener un panorama sobre qué función de activación tiene un mejor desempeño, se grafica de esa malla el comportamiento del error de entrenamiento y validación en 5.29, donde no parece observarse una diferencia significativa a priori, no obstante, parece que se tiene un mejor rendimiento para cuando existen dos capas ocultas en la estructura. Por otro lado, si en análisis posteriores no existe evidencia contundente sobre el rendimiento en las funciones de activación, entonces se puede aplicar aquella con menor costo computacional.

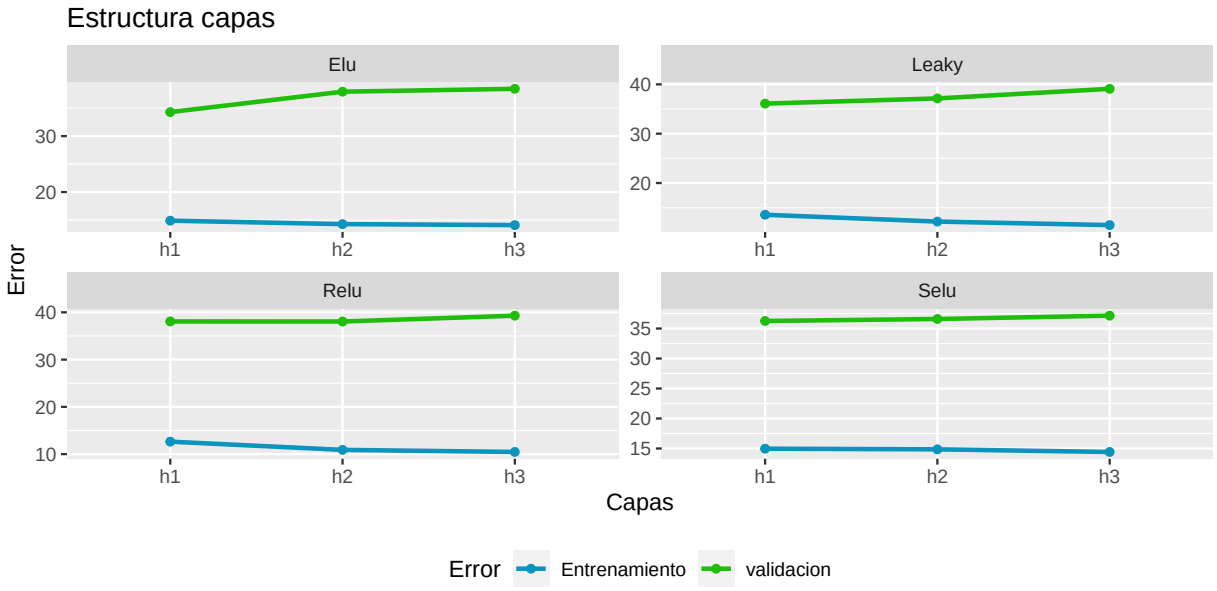


Figura 5.29: Aproximación del Error general dada la cantidad de capas ocultas para las diferentes funciones de activación.

Neuronas por función de activación

A continuación, en las figuras 5.30 y 5.31 se puede observar el rendimiento de cada función de activación contra el número de neuronas para cada caso del número de capas ocultas 1, 2 y 3 (h_1,h_2 y h_3). De estas gráficas se puede apreciar una comparativa directa entre cada capa para cada función y además hacer una comparativa directa entre las mismas funciones de activación, de esta manera también se comparará el error. En tabla 5.7 se resume el aproximado de neuronas a utilizar:

F. act vs Ocultas	h_1	h_2	h_3
Leaky	120-150	alrededor de 150	alrededor de 71
Relu	alrededor de 100	alrededor de 100	71 neuronas
Elu	71	alrededor de 120	alrededor de 120
Selu	alrededor de 100	71 neuronas	alrededor de 100

Cuadro 5.7: Resumen sobre el número de neuronas para cada función de activación y capa.

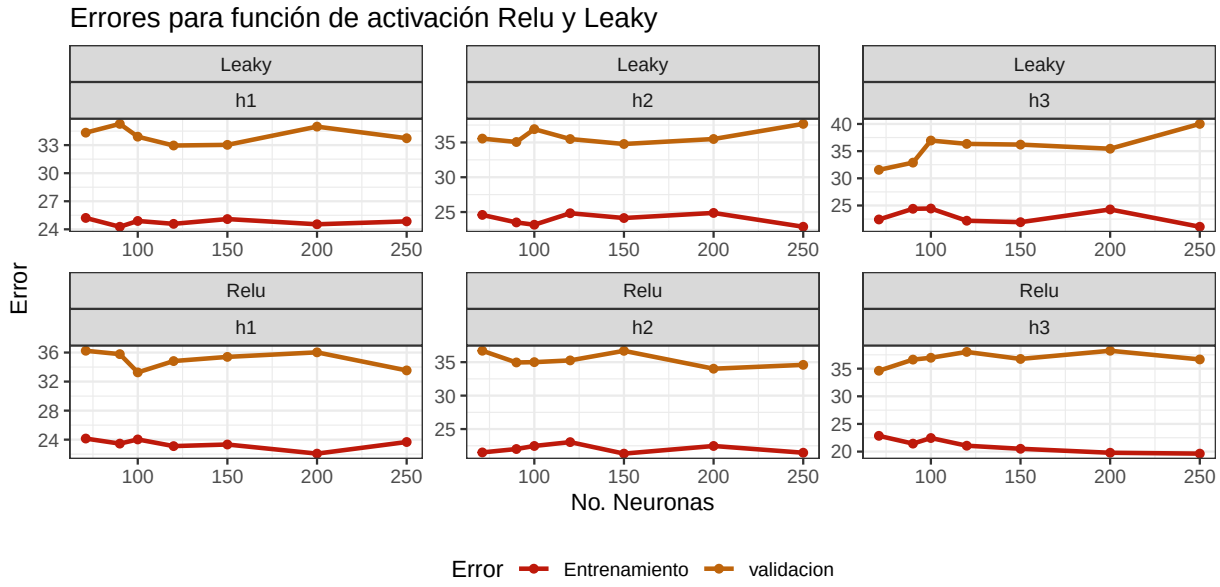


Figura 5.30: Comparativo desempeño de las funciones de activación *Relu* y *Leaky* vs número de neuronas y número de capas.

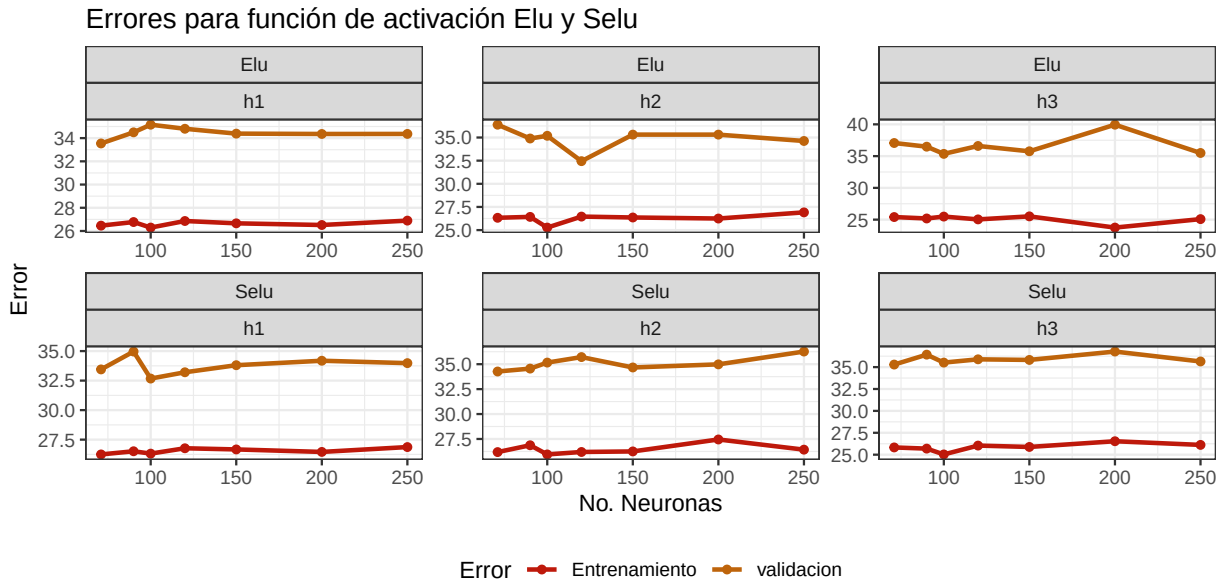


Figura 5.31: Comparativo desempeño de las funciones de activación *Elu* y *Selu* vs número de neuronas y número de capas.

Tasa de aprendizaje

Como último hiperparámetro de la malla de hiperparámetros en estos modelos de redes neuronales artificiales es la tasa de aprendizaje, en las figuras 5.32, 5.33, 5.34 y 5.35 se tiene un panorama más general para cada función de activación en dicha malla, donde se aprecian los cambios en los errores para diferentes tasas en cada número de neuronas y que se podría tener un panorama de la mejor estructura para cada función. En la tabla 5.8 se resume la posible mejor estructura a priori de cada modelo de red neuronal con

función de activación y número de capas ocultas. En comparativa con las observaciones anteriores, para esta parte se puede definir de forma un poco más clara la mejor estructura.

F. act vs Capas ocultas	h_1	h_2	h_3
	No. neuronas-tasa	*No. neuronas-tasa*	*No. neuronas-tasa*
Relu	100-0.0001	100-0.0001	71-0.0003
Leaky	120-0.0001	150-0.0003	71-0.0001
Elu	71-0.0005	120-0.0001	120-0.0001
Selu	100-0.0005	100-0.0005	71-0.0005

Cuadro 5.8: Resumen general de cada estructura de red neuronal artificial.

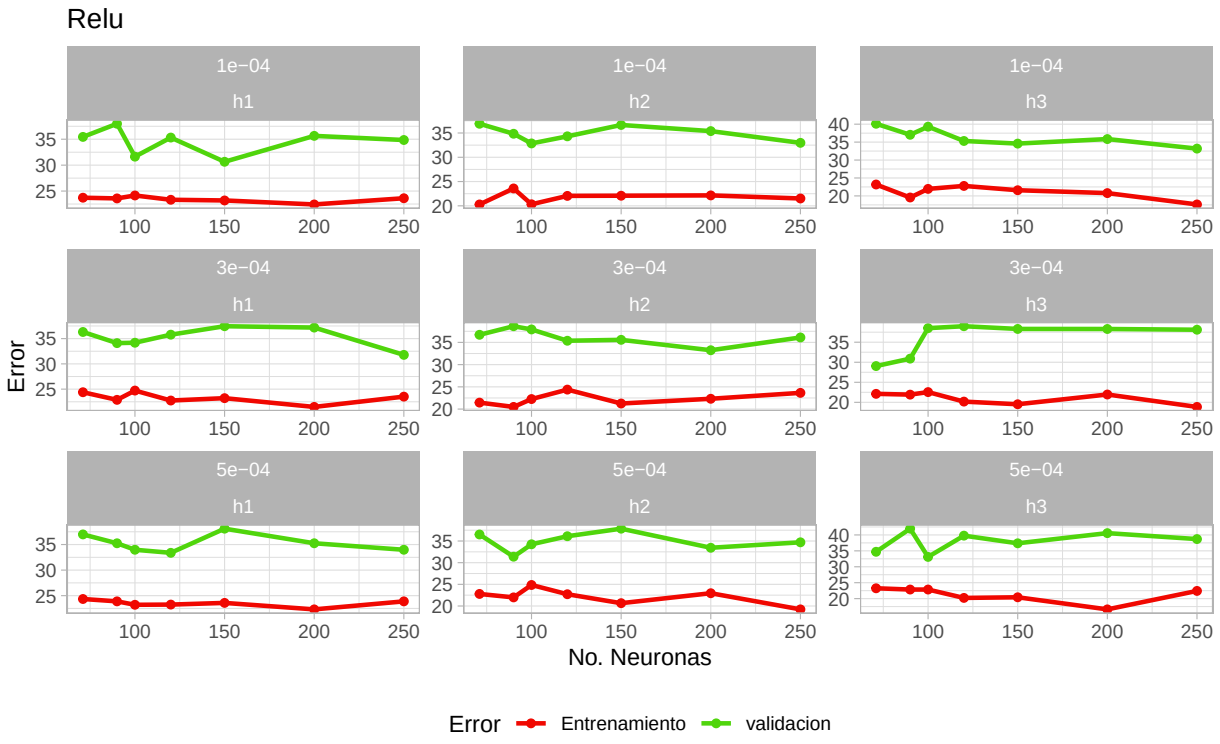


Figura 5.32: Panorama a priori del rendimiento de la función de activación *Relu* en la malla de hiperparámetros.

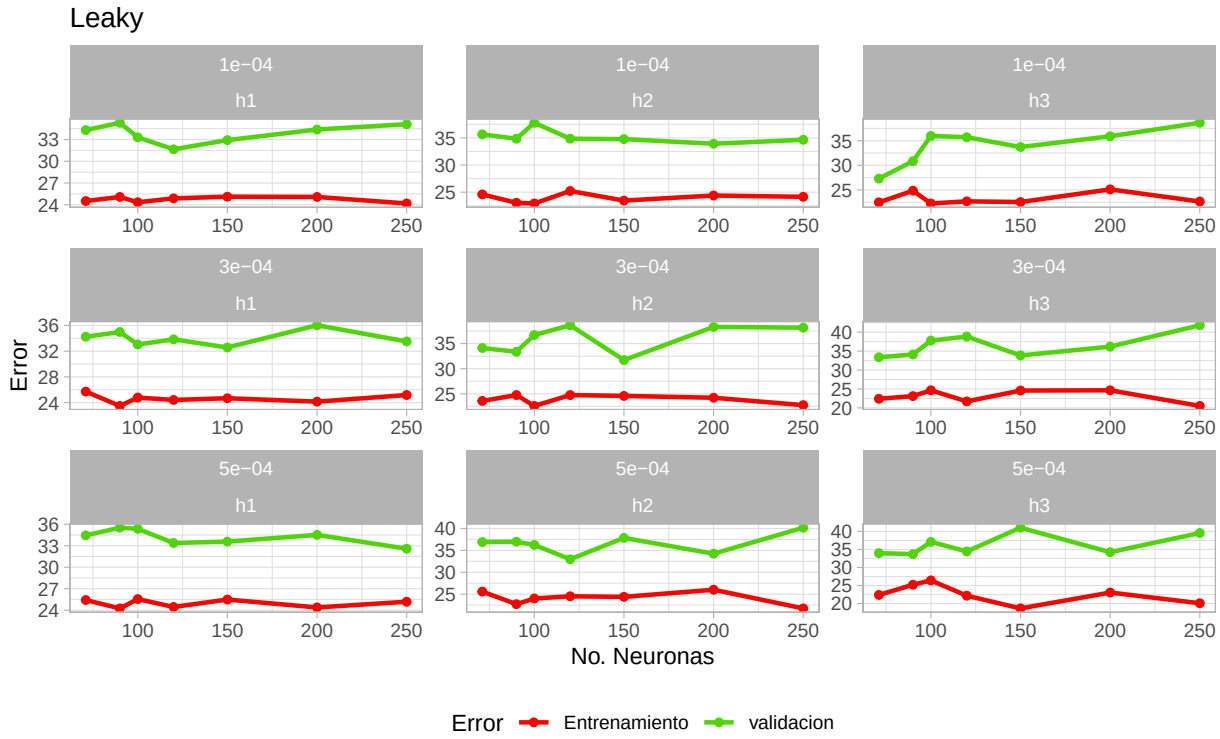


Figura 5.33: Panorama a priori del rendimiento de la función de activación *Leaky* en la malla.

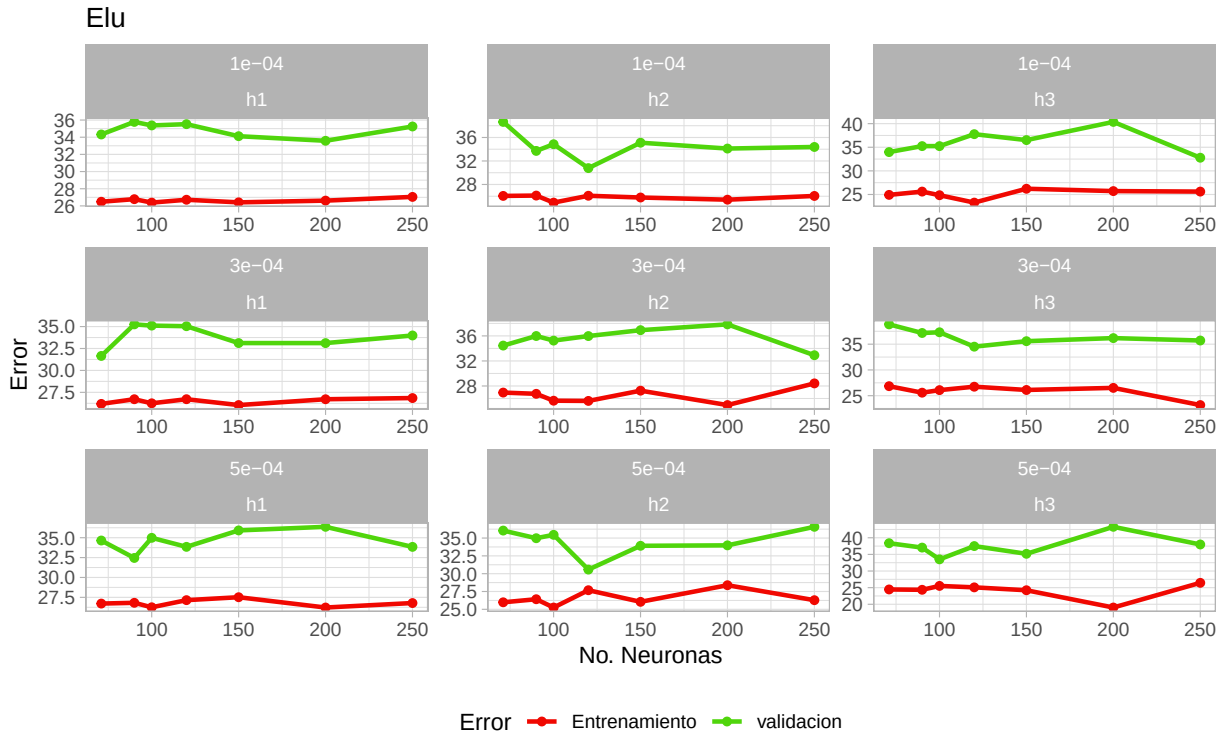


Figura 5.34: Panorama a priori del rendimiento de la función de activación *Elu* en la malla.

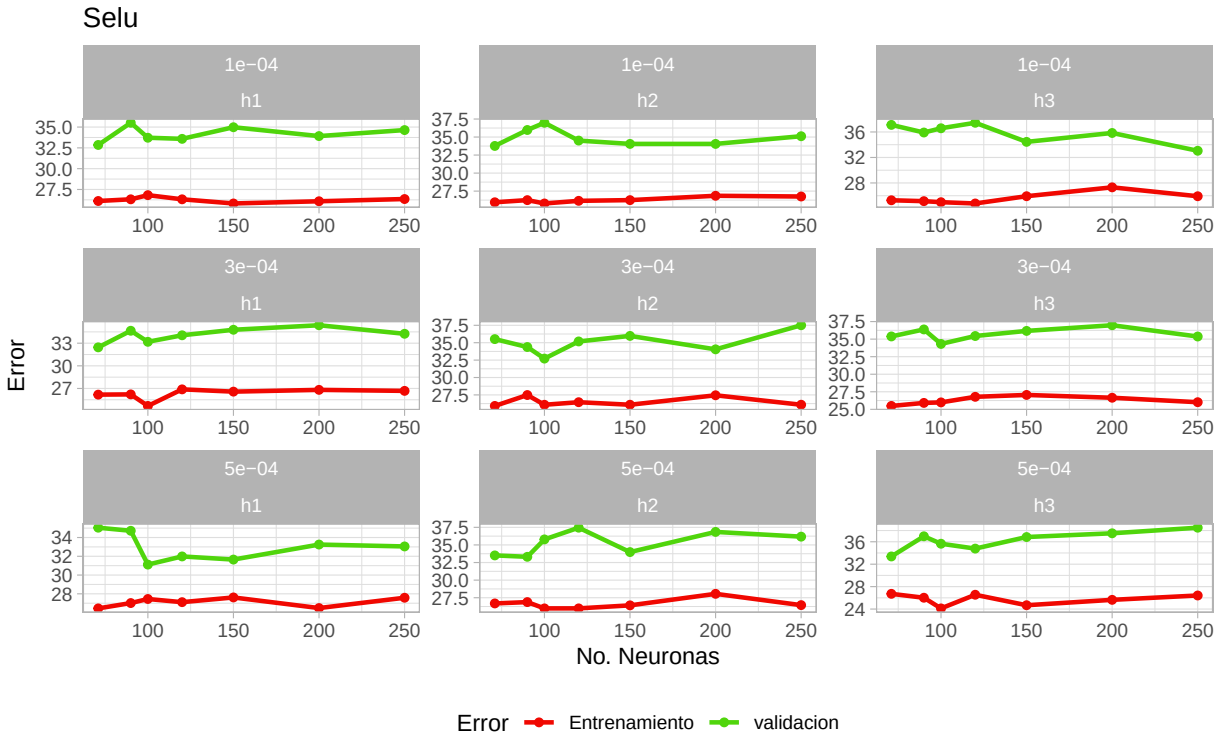


Figura 5.35: Panorama a priori del rendimiento de la función de activación *Selu* en la malla.

5.4.2. Metricas Validación

Una vez que se conocen las posibles estructuras óptimas, se entrenan estos modelos y serán comparados en forma general en las métricas de desempeño ya mencionadas: Estadístico KS, área bajo la curva, métrica-F, Recall, precisión y costo. Esto se visualiza en las gráficas 5.36-5.39. De forma análoga se realiza el análisis, pero más a detalle de estas métricas por el número de capas ocultas en las figuras 5.40-5.43.

Metricas generales

En la gráfica 5.36 brinda indicios que la mejor función de activación es Elu, pero también la función Selu brinda buen aspecto a priori, lo mismo se puede decir para el caso de área bajo la curva de la figura 5.37. Por otro lado, en la distribución de las métricas F, Recall, precisión, así como la de costo parece no haber una diferencia significativa en el desempeño por la función de activación.

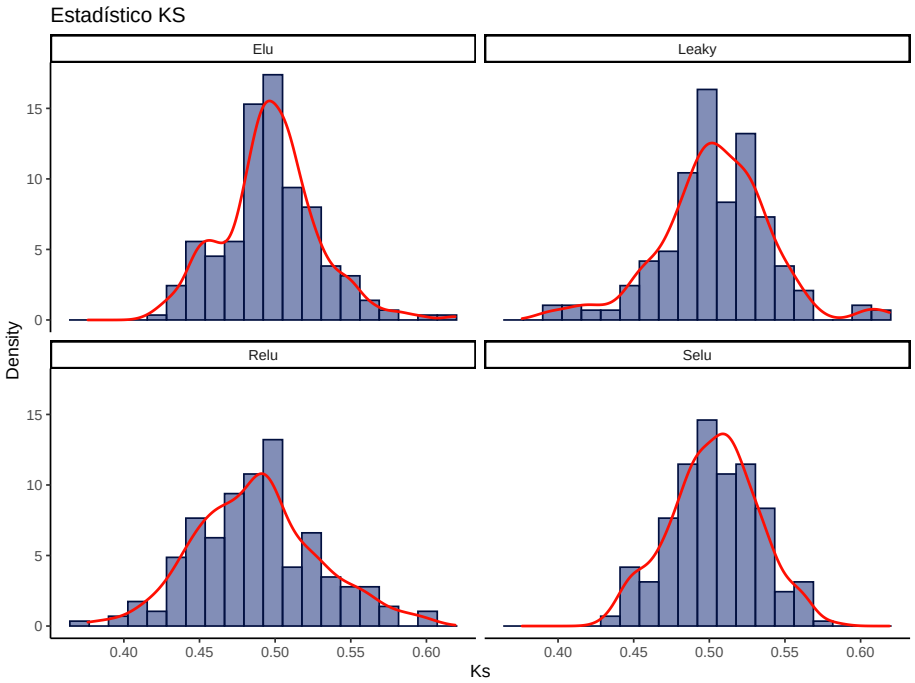


Figura 5.36: Distribución estadístico KS para las cuatro funciones de activación.

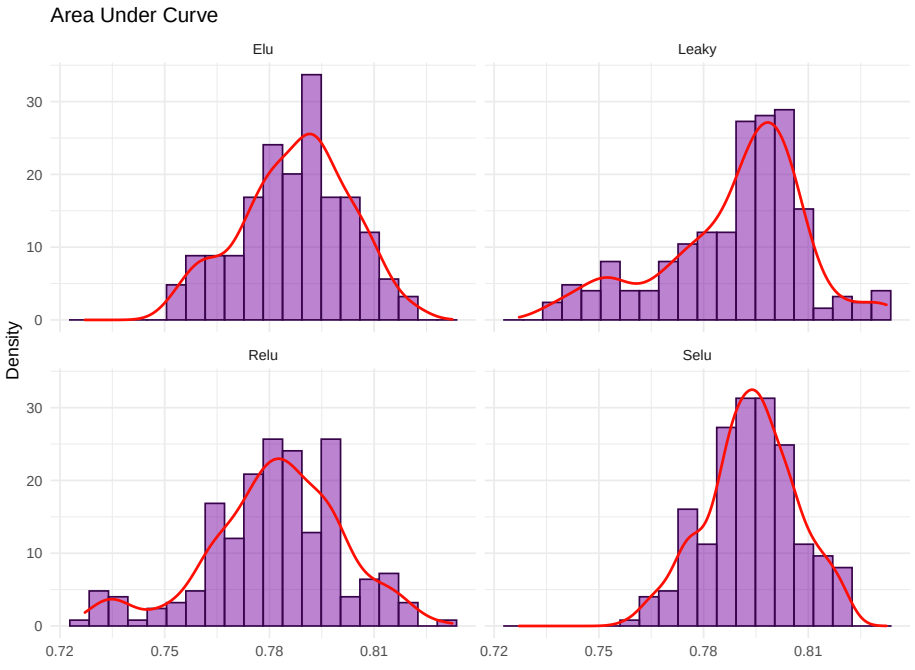


Figura 5.37: Distribución AUC para las cuatro funciones de activación.

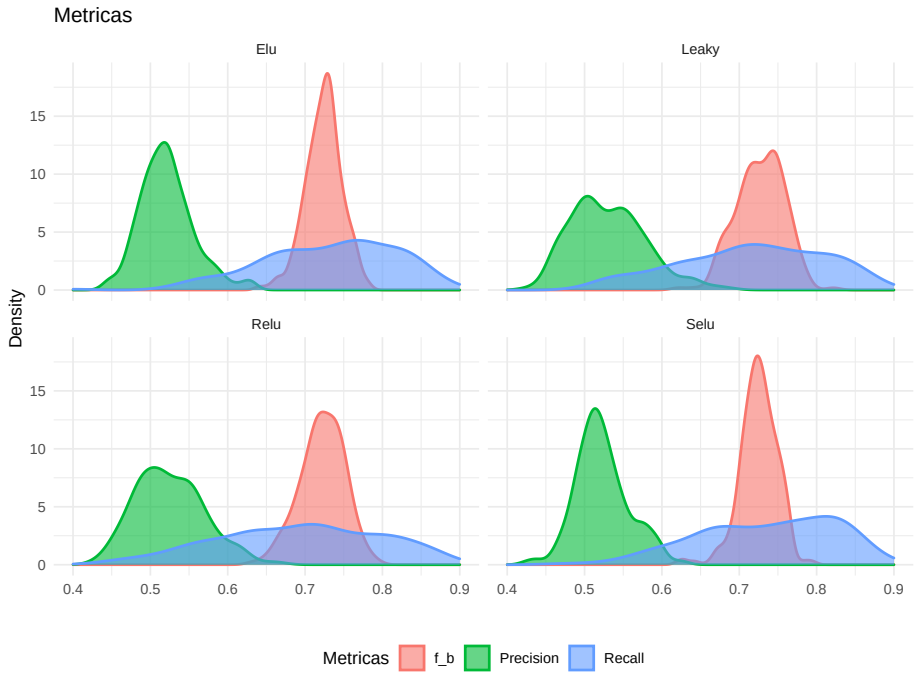


Figura 5.38: Distribución métricas para las cuatro funciones de activación.

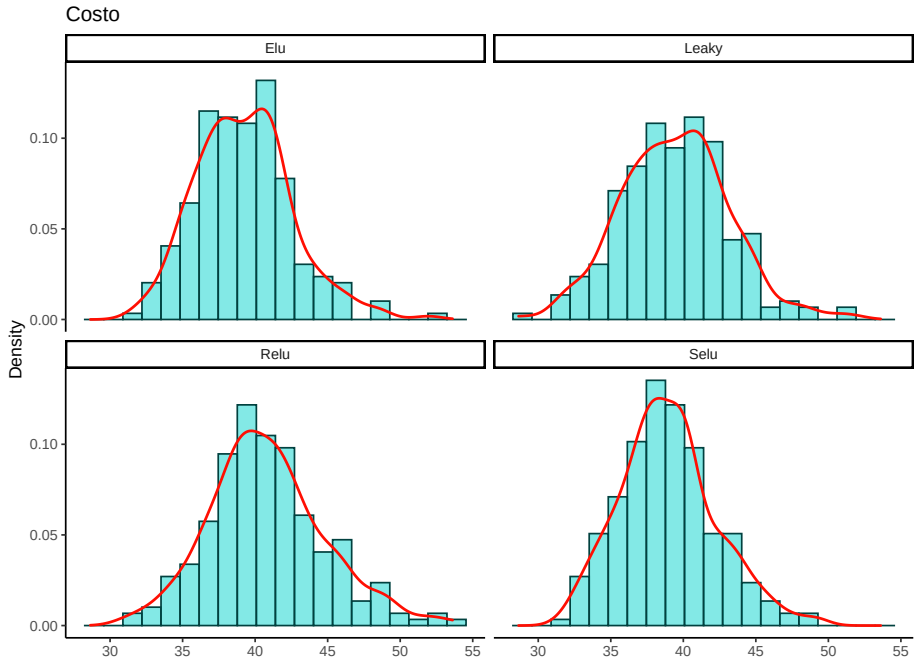


Figura 5.39: Distribución costo para las cuatro funciones de activación.

Metricas por capas

De las figuras 5.40-5.43 se analiza el cruce por el número de capas y función de activación en el desempeño de cada una de las métricas. No se debe olvidar que estamos en la etapa de validación, por lo que

posteriormente se evaluará el desempeño con el set de prueba. A continuación, se describe brevemente las observaciones en estas comparaciones.

Figura 5.40: Análsis comportamiento KS h_i-F.

- *Elu*: Para cuando se tiene una capa existe menos varianza, no obstante, se alcanzan mayores valores KS para dos y tres capas.

- *Leaky*: Se observa un mejor desempeño para h_2, mayores valores y varianza no tan alta como para h_3.

- *Relu*: Existe un mejor resultado para cuando h_2 es aplicado, ya que para h_3 tiene una varianza muy alta.

- *Selu*: De igual manera para h_2 brinda mejores valores KS, no obstante, su desempeño es ligeramente mejor para cuando se tiene una capa.

- *Comparativa*: En general, estos resultados indican que se tiene un mejor desempeño para h_2, y con la función de activación Selu.

Figura 5.41: Área bajo la curva.

- *Elu, Leaky, Relu y Selu* : Tienen mejor resultado con una capa oculta, métrica con menos varianza.

- *Comparativa*: La gráfica indica que la mejor función de activación es la Selu.

Figura 5.42: Métricas: F_β, Precisión y Recall.

- *Elu y Leaky*: Para cuando se tiene h_2 el Recall mejora ligeramente.

- *Relu*: Parece que sus resultados Recall son bajos.

- *Selu*: Para h_2 y h_3 brinda mejores resultados para Recall y precisión, por tanto, F_β.

- *Comparativa*: La mejor función de activación ha sido Selu en general.

Figura 5.43: Costos.

- *Elu, Leaky,Relu y Selu*: En esta métrica de costo, parece que tienen resultados similares, no obstante, la función de costo Selu parece tener una ligera mejora.

- *Comparativa*: Selu presenta resultados ligeramente mejores.

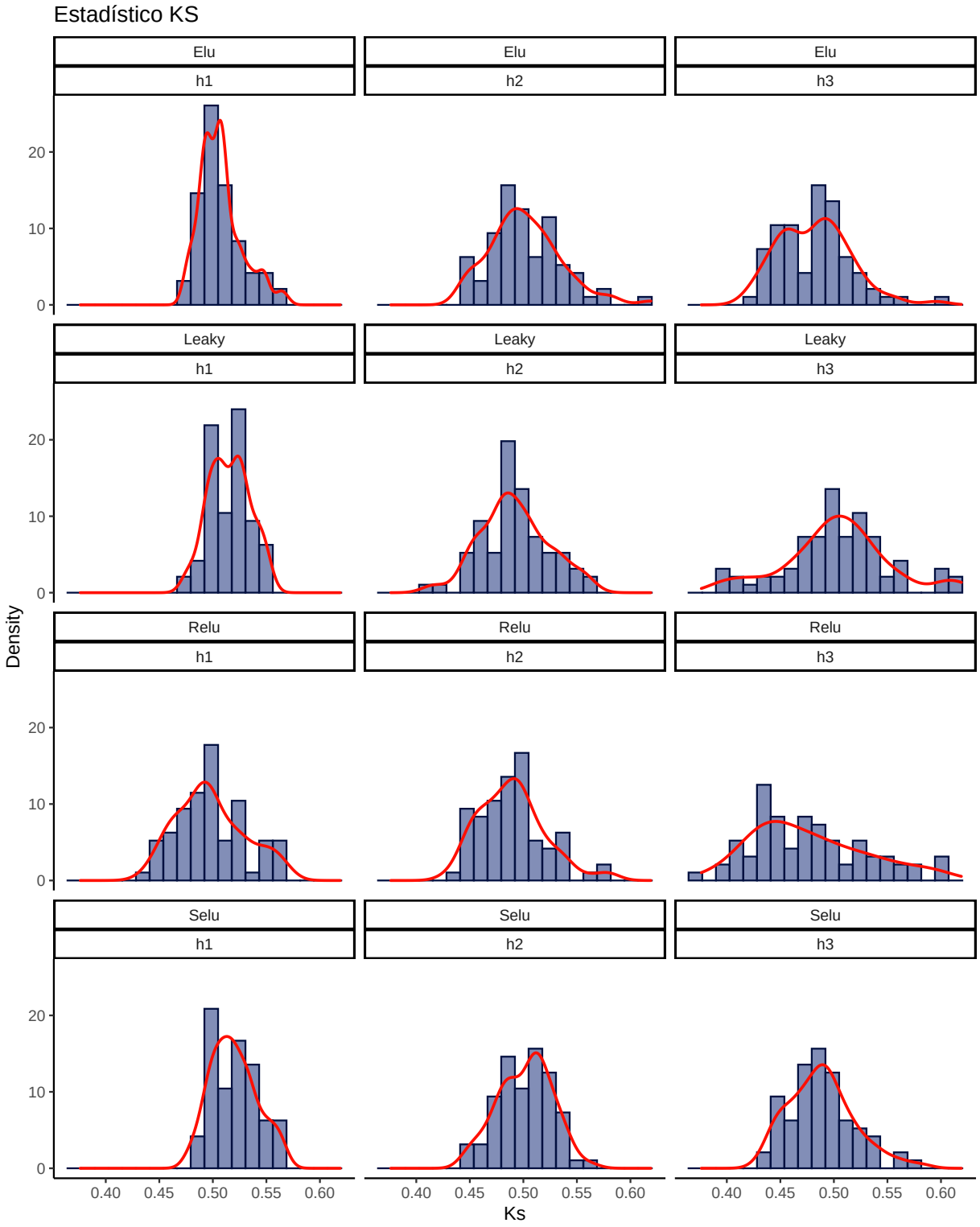


Figura 5.40: Distribución del estadístico KS: Funciones de activación vs número capas.

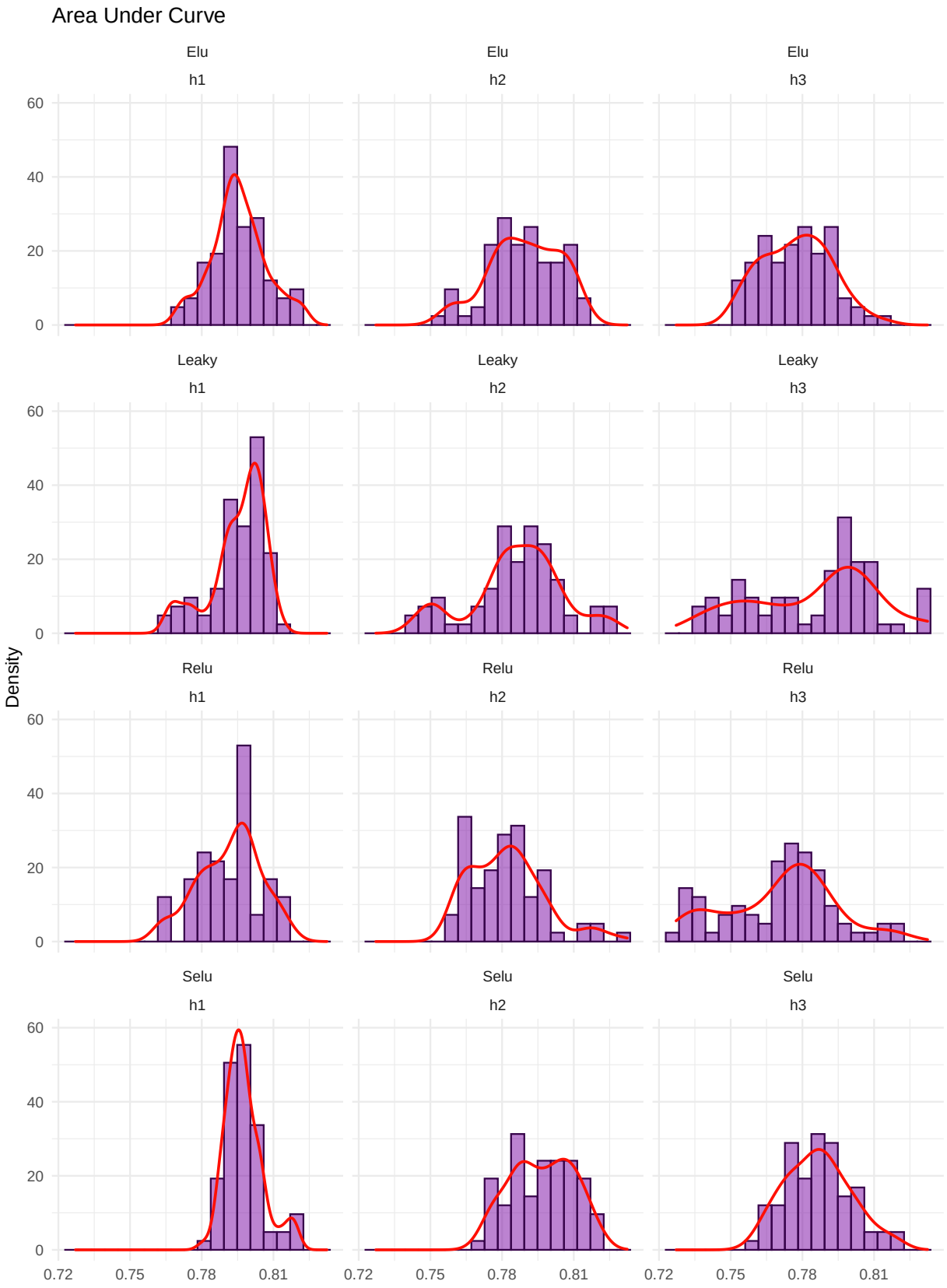


Figura 5.41: Distribución del AUC: Funciones de activación vs número capas.

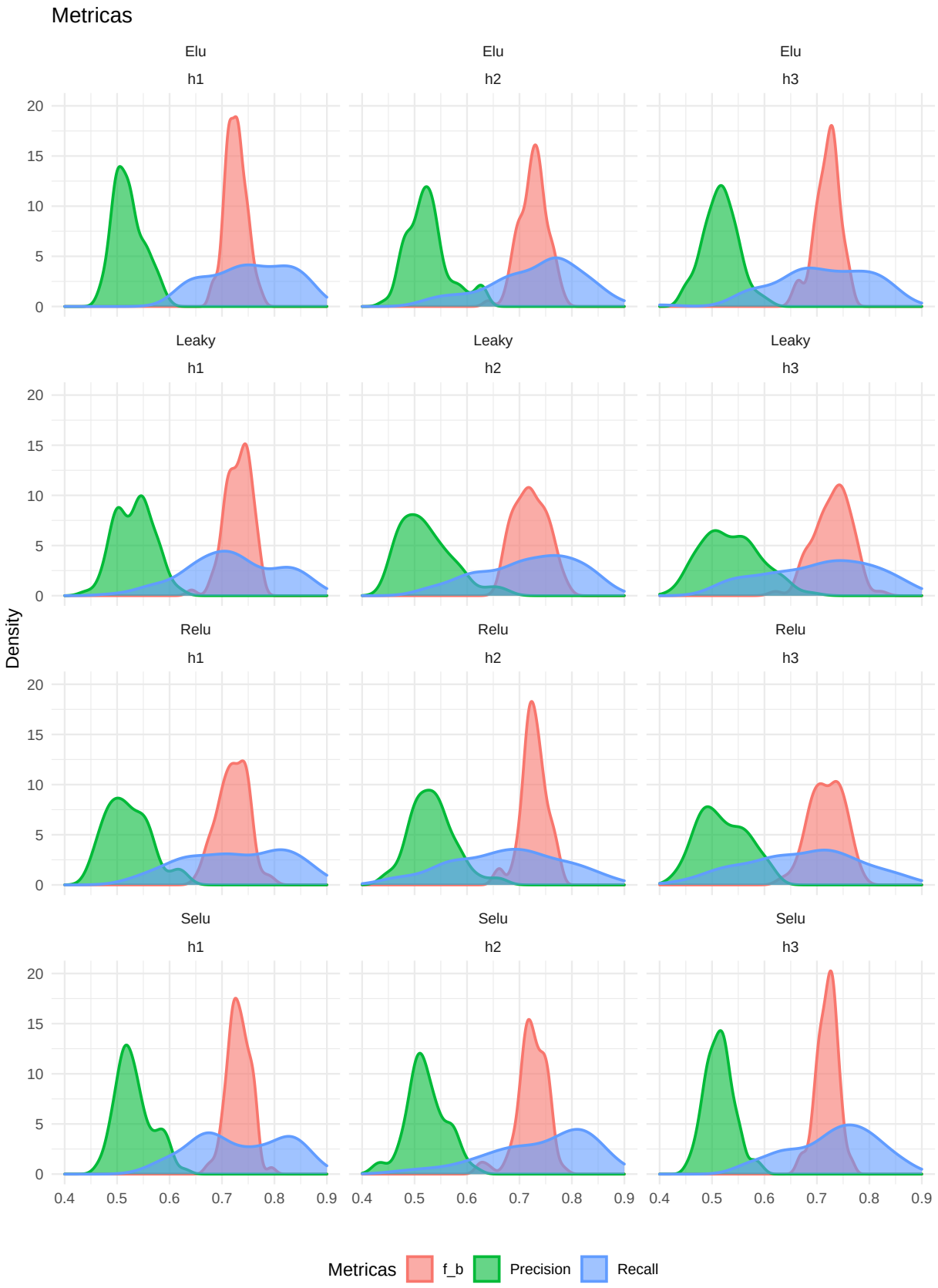


Figura 5.42: Distribución de las métricas F, Recall y Precisión: Funciones de activación vs número capas.

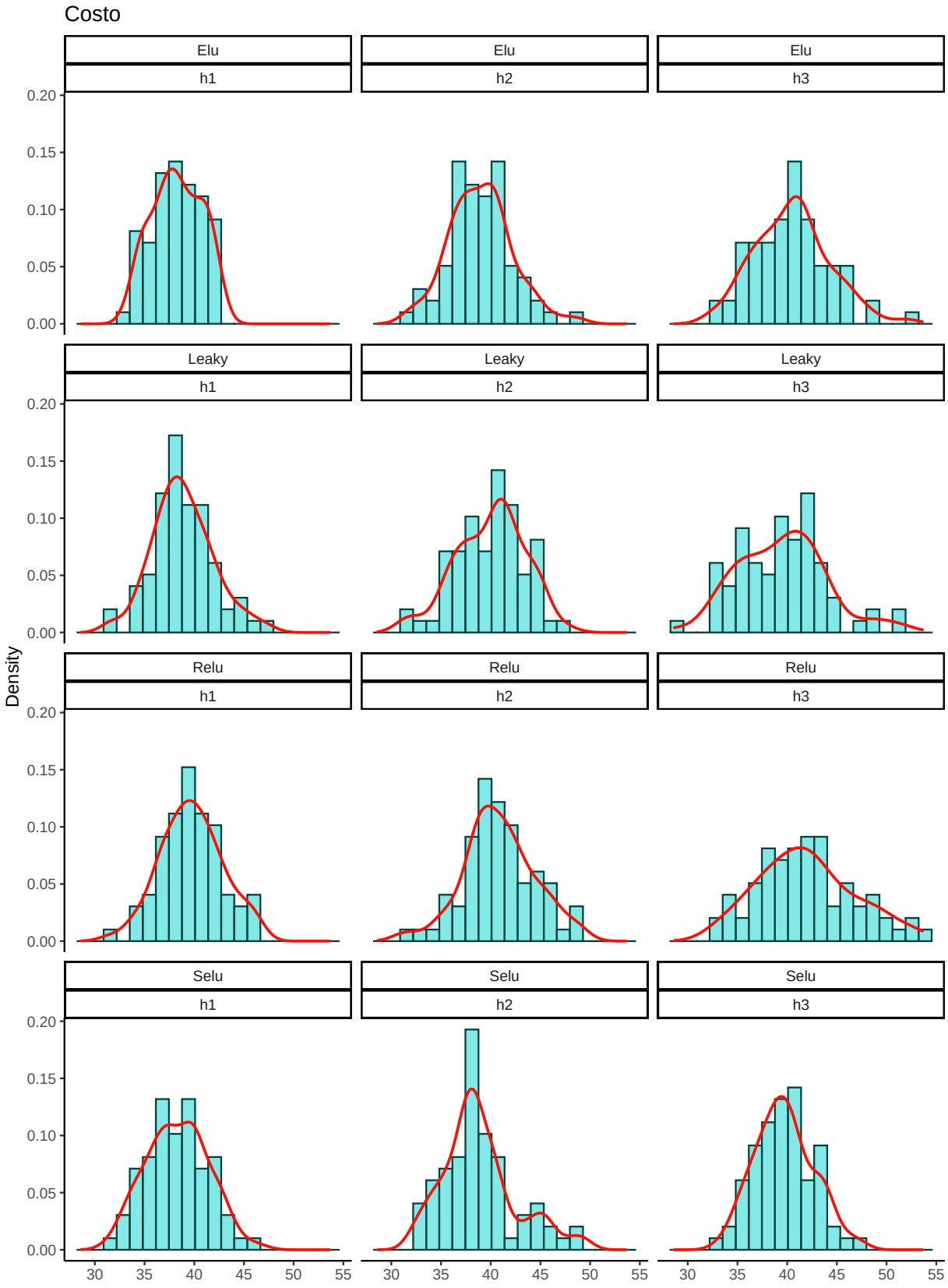


Figura 5.43: Distribución del costo: Funciones de activación vs número capas.

A continuación, se muestra en el cuadro 5.9 se muestran los mejores modelos dadas sus métricas individuales, donde en cada columna se muestra el valor de desempeño por cada métrica de interés. Esta tabla está ordenada por el *Costo* de forma descendente, de esta forma se tomarán para la siguiente parte, aplicación al set de prueba y comparar los resultados.

Resultado Métricas de Validación							
Modelo	Costo	KS	AUC	Exactitud	Precisión	Recall	F_β
leaky3_ h2_ 90_ 0.0001	31.64	0.61429	0.82063	0.776	0.60332	0.76666	0.78224
leaky5_ h3_ 71_ 0.0001	32.97	0.60952	0.83152	0.774	0.60471	0.74	0.77976
relu3_ h1_ 150_ 0.0005	33.31	0.56031	0.81016	0.746	0.55896	0.8	0.75572
leaky3_ h3_ 150_ 0.0001	33.41	0.59048	0.82079	0.776	0.60812	0.72222	0.78142
elu3_ h1_ 71_ 0.0001	33.53	0.55079	0.81413	0.753	0.56753	0.77778	0.76178
elu3_ h1_ 250_ 0.0001	33.64	0.54762	0.81254	0.770	0.59795	0.73333	0.77577
selu3_ h2_ 90_ 0.0001	33.97	0.60476	0.81841	0.766	0.59449	0.73333	0.77199
selu3_ h2_ 200_ 0.0001	34.31	0.56349	0.81651	0.750	0.56615	0.76667	0.75777
relu3_ h2_ 71_ 0.0001	34.31	0.56667	0.80143	0.736	0.55082	0.8	0.74582
relu5_ h3_ 90_ 0.0003	34.37	0.5581	0.81524	0.744	0.55485	0.78	0.75306
elu3_ h1_ 100_ 0.0003	34.416	0.55714	0.82048	0.766	0.59286	0.72222	0.77245
leaky3_ h2_ 71_ 0.0001	34.42	0.54286	0.81079	0.753	0.56697	0.75556	0.76142
relu3_ h3_ 150_ 0.0003	34.52	0.54286	0.82206	0.770	0.60324	0.71111	0.77485
elu5_ h1_ 71_ 0.0003	34.57	0.54762	0.81952	0.742	0.5545	0.78	0.75075
selu5_ h1_ 100_ 0.0005	34.57	0.55524	0.81543	0.742	0.55806	0.78	0.7498
relu3_ h2_ 250_ 0.0005	34.75	0.55714	0.80842	0.750	0.56638	0.75556	0.75813
selu3_ h1_ 120_ 0.0005	34.75	0.54603	0.80429	0.7	0.56257	0.75556	0.75797
leaky3_ h2_ 250_ 0.0001	34.86	0.54762	0.8046	0.766	0.59623	0.71111	0.77139
selu3_ h1_ 250_ 0.0001	34.86	0.54127	0.82048	0.75333	0.57843	0.74444	0.75993
selu3_ h1_ 71_ 0.0005	34.97	0.53968	0.80556	0.74333	0.55787	0.76667	0.75149
selu5_ h1_ 71_ 0.0003	34.97	0.53619	0.79857	0.738	0.54643	0.78	0.74725
relu5_ h2_ 90_ 0.0005	35.04	0.55143	0.81953	0.74	0.55761	0.77334	0.74739
elu3_ h2_ 71_ 0.0003	35.08	0.54286	0.80953	0.74667	0.55757	0.75556	0.7553
selu3_ h1_ 250_ 0.0005	35.08	0.57619	0.81333	0.74667	0.56096	0.75555	0.75384
elu3_ h3_ 100_ 0.0005	35.09	0.54762	0.81397	0.73333	0.54398	0.78889	0.74307
relu3_ h2_ 100_ 0.0003	35.20	0.57619	0.8146	0.73667	0.56293	0.77778	0.74428
relu3_ h2_ 120_ 0.0001	35.20	0.50953	0.82302	0.73667	0.54376	0.77778	0.74643
leaky3_ h1_ 90_ 0.0003	35.30	0.54762	0.82349	0.76667	0.59547	0.7	0.77135
leaky3_ h3_ 90_ 0.0005	35.42	0.50952	0.78857	0.74333	0.55749	0.75555	0.75149

Cuadro 5.9: Tabla modelos de métricas de Validación.

5.4.3. Metricas Test

En esta tabla 5.10 muestra los resultados de las métricas de interés al aplicar el set de prueba a cada uno de los modelos antes mencionados manteniendo el mismo orden. El desempeño en general disminuyó como se tenía previsto, no obstante, se analiza cuales se mantuvieron alrededor de los mismos resultados, de esta manera, se puede elegir cuál es el mejor modelo; aquel cuyo resultado sea similar al obtenido en la fase de validación.

5.4.4. Comparativa Validación-Prueba

En la tabla siguiente 5.11 se presenta una comparativa directa sobre el resultado en las métricas principales de las fases validación y prueba, refiriendo su diferencia proporcional. Esta tabla ayuda a identificar

Resultado Métricas de Prueba							
Modelo	Costo	KS	AUC	Exactitud	Precisión	Recall	F_β
leaky3_ h2_ 90_ 0.0001	43.96333	0.44762	0.76222	0.70667	0.51673	0.63334	0.71375
leaky5_ h3_ 71_ 0.0001	48.092	0.40286	0.75505	0.684	0.48372	0.58667	0.69152
relu3_ h1_ 150_ 0.0005	42.63333	0.45556	0.77429	0.70667	0.51351	0.66667	0.71567
leaky3_ h3_ 150_ 0.0001	40.63	0.46667	0.77682	0.74	0.55896	0.63333	0.7442
elu3_ h1_ 71_ 0.0001	42.29667	0.48889	0.77095	0.72333	0.53519	0.63333	0.72806
elu3_ h1_ 250_ 0.0001	42.63	0.4619	0.77111	0.72	0.52894	0.63334	0.72613
selu3_ h2_ 90_ 0.0001	45.73333	0.48095	0.77095	0.72	0.5354	0.55556	0.72016
selu3_ h2_ 200_ 0.0001	42.19	0.46508	0.76762	0.70667	0.51394	0.67778	0.71569
relu3_ h2_ 71_ 0.0001	45.52	0.40635	0.73587	0.68667	0.48742	0.64444	0.69543
relu5_ h3_ 90_ 0.0003	37.24	0.49048	0.76162	0.734	0.54287	0.73333	0.7429
elu3_ h1_ 100_ 0.0003	45.51333	0.48412	0.7754	0.71333	0.52204	0.57778	0.71726
leaky3_ h2_ 71_ 0.0001	42.85333	0.42064	0.77079	0.71333	0.51781	0.64445	0.72091
relu3_ h3_ 150_ 0.0003	43.52	0.48889	0.77079	0.70667	0.50925	0.64444	0.71411
elu5_ h1_ 71_ 0.0003	44.898	0.45905	0.75905	0.692	0.49265	0.64667	0.69984
selu5_ h1_ 100_ 0.0005	41.168	0.51333	0.78057	0.716	0.52503	0.68	0.72357
relu3_ h2_ 250_ 0.0005	44.40667	0.44762	0.75238	0.70667	0.50819	0.62222	0.71281
selu3_ h1_ 120_ 0.0005	44.07333	0.48413	0.75905	0.71	0.52081	0.62222	0.7164
leaky3_ h2_ 250_ 0.0001	43.74	0.45714	0.77587	0.71333	0.51835	0.62222	0.71938
selu3_ h1_ 250_ 0.0001	40.74333	0.45714	0.77476	0.73	0.54838	0.65555	0.73561
selu3_ h1_ 71_ 0.0005	42.07667	0.49524	0.76889	0.71667	0.52393	0.65555	0.72335
selu5_ h1_ 71_ 0.0003	44.366	0.49238	0.75943	0.692	0.49344	0.66	0.70101
relu5_ h2_ 90_ 0.0005	48.158	0.41143	0.72381	0.686	0.48404	0.58	0.69201
elu3_ h2_ 71_ 0.0003	45.96	0.4	0.74302	0.7	0.50026	0.6	0.70691
selu3_ h1_ 250_ 0.0005	43.74	0.46667	0.76429	0.71333	0.52446	0.62222	0.71841
elu3_ h3_ 100_ 0.0005	42.08	0.48254	0.76381	0.70333	0.50735	0.68889	0.71226
relu3_ h2_ 100_ 0.0003	48.29667	0.39365	0.72937	0.66333	0.46149	0.63333	0.6727
relu3_ h2_ 120_ 0.0001	47.96	0.40952	0.73682	0.68	0.47258	0.6	0.68857
leaky3_ h1_ 90_ 0.0003	44.62333	0.47619	0.77968	0.72667	0.54423	0.56667	0.72815
leaky3_ h3_ 90_ 0.0005	39.52667	0.46825	0.76286	0.72	0.52729	0.71111	0.72882

Cuadro 5.10: Tabla modelos métricas Test.

rápidamente aquellos modelos en los cuales dicha diferencia fue menor, es decir, lo modelos que mejor generalizan a la población.

En la tabla 5.12 es el filtro de los mejores modelos de la tabla anterior (5.11), tomando como prioridad la estabilidad de la métrica *KS*, es decir, que esta no haya tenido un cambio relativamente significativo de la validación a la de prueba. Se puede observar que el modelo $leaky3\_h3\_90\_0.0005$ es el modelo con la mayor estabilidad en el *KS*, ya que solo tuvo una pérdida en un 8.1 % en la fase de prueba respecto al de validación, de forma análoga disminuyó la métrica *AUC* en un 3.3 % y aumentó el *Costo* en un 11.6 %. Por tanto, el mejor modelo es $leaky3\_h3\_90\_0.0005$, nótese que para los otros modelos tuvieron una disminución del *KS* hasta en un 15.5 %, aun cuando algunos tuvieron una disminución menor en el *Costo* o *AUC*.

Para fines de comparación con los modelos Logit, se toman los siguientes modelos:

Modelo 1 ANN leaky3_ h3_ 90_ 0.0005

Modelo 2 ANN selu5_ h1_ 100_ 0.0005

Modelo 3 ANN selu3_ h1_ 71_ 0.0005

5.5. Comparativa de Resultados

Finalmente se cuenta con los mejores modelos paramétricos y no paramétricos como Credit Scoring en los datos, ya que se hizo una comparativa en cada caso, por lo que ya se puede hacer una comparativa para

Validación				Prueba			Diferencia		
Modelo ANN	**Costo**	**KS**	**AUC**	**Costo**	**KS**	**AUC**	**Costo**	**KS**	**AUC**
leaky3_ h2_ 90_ 0.0001	31.643	0.614	0.821	43.963	0.448	0.762	0.389	-0.271	-0.071
leaky5_ h3_ 71_ 0.0001	32.974	0.61	0.832	48.092	0.403	0.755	0.458	-0.339	-0.092
relu3_ h1_ 150_ 0.0005	33.313	0.56	0.81	42.633	0.456	0.774	0.28	-0.187	-0.044
leaky3_ h3_ 150_ 0.0001	33.417	0.59	0.821	40.63	0.467	0.777	0.216	-0.21	-0.054
elu3_ h1_ 71_ 0.0001	33.533	0.551	0.814	42.297	0.489	0.771	0.261	-0.112	-0.053
elu3_ h1_ 250_ 0.0001	33.64	0.548	0.813	42.63	0.462	0.771	0.267	-0.157	-0.051
selu3_ h2_ 90_ 0.0001	33.973	0.605	0.818	45.733	0.481	0.771	0.346	-0.205	-0.058
selu3_ h2_ 200_ 0.0001	34.31	0.563	0.817	42.19	0.465	0.768	0.23	-0.175	-0.06
relu3_ h2_ 71_ 0.0001	34.313	0.567	0.801	45.52	0.406	0.736	0.327	-0.283	-0.082
relu5_ h3_ 90_ 0.0003	34.378	0.558	0.815	37.24	0.49	0.762	0.083	-0.121	-0.066
elu3_ h1_ 100_ 0.0003	34.417	0.557	0.82	45.513	0.484	0.775	0.322	-0.131	-0.055
leaky3_ h2_ 71_ 0.0001	34.42	0.543	0.811	42.853	0.421	0.771	0.245	-0.225	-0.049
relu3_ h3_ 150_ 0.0003	34.527	0.543	0.822	43.52	0.489	0.771	0.26	-0.099	-0.062
elu5_ h1_ 71_ 0.0003	34.578	0.548	0.82	44.898	0.459	0.759	0.298	-0.162	-0.074
selu5_ h1_ 100_ 0.0005	34.578	0.555	0.815	41.168	0.513	0.781	0.191	-0.075	-0.043
relu3_ h2_ 250_ 0.0005	34.753	0.557	0.808	44.407	0.448	0.752	0.278	-0.197	-0.069
selu3_ h1_ 120_ 0.0005	34.753	0.546	0.804	44.073	0.484	0.759	0.268	-0.113	-0.056
leaky3_ h2_ 250_ 0.0001	34.86	0.548	0.805	43.74	0.457	0.776	0.255	-0.165	-0.036
selu3_ h1_ 250_ 0.0001	34.863	0.541	0.82	40.743	0.457	0.775	0.169	-0.155	-0.056
selu3_ h1_ 71_ 0.0005	34.977	0.54	0.806	42.077	0.495	0.769	0.203	-0.082	-0.046
selu5_ h1_ 71_ 0.0003	34.978	0.536	0.799	44.366	0.492	0.759	0.268	-0.082	-0.049
relu5_ h2_ 90_ 0.0005	35.044	0.551	0.82	48.158	0.411	0.724	0.374	-0.254	-0.117
elu3_ h2_ 71_ 0.0003	35.087	0.543	0.81	45.96	0.4	0.743	0.31	-0.263	-0.082
selu3_ h1_ 250_ 0.0005	35.087	0.576	0.813	43.74	0.467	0.764	0.247	-0.19	-0.06
elu3_ h3_ 100_ 0.0005	35.09	0.548	0.814	42.08	0.483	0.764	0.199	-0.119	-0.062
relu3_ h2_ 100_ 0.0003	35.2	0.576	0.815	48.297	0.394	0.729	0.372	-0.317	-0.105
relu3_ h2_ 120_ 0.0001	35.2	0.51	0.823	47.96	0.41	0.737	0.362	-0.196	-0.105
leaky3_ h1_ 90_ 0.0003	35.303	0.548	0.823	44.623	0.476	0.78	0.264	-0.13	-0.053
leaky3_ h3_ 90_ 0.0005	35.42	0.51	0.789	39.527	0.468	0.763	0.116	-0.081	-0.033

Cuadro 5.11: Tabla métricas de modelos comparativa Validación-Prueba.

Validación				Prueba			Diferencia		
Modelo ANN	**Costo**	**KS**	**AUC**	**Costo**	**KS**	**AUC**	**Costo**	**KS**	**AUC**
relu5_ h3_ 90_ 0.0003	34.378	0.558	0.815	37.24	0.49	0.762	0.083	-0.121	-0.066
leaky3_ h3_ 90_ 0.0005	35.42	0.51	0.789	39.527	0.468	0.763	0.116	-0.081	-0.033
selu3_ h1_ 250_ 0.0001	34.863	0.541	0.82	40.743	0.457	0.775	0.169	-0.155	-0.056
selu3_ h1_ 71_ 0.0005	34.977	0.54	0.806	42.077	0.495	0.769	0.203	-0.082	-0.046
elu3_ h3_ 100_ 0.0005	35.09	0.548	0.814	42.08	0.483	0.764	0.199	-0.119	-0.062
selu5_ h1_ 100_ 0.0005	34.578	0.555	0.815	41.168	0.513	0.781	0.191	-0.075	-0.043

Cuadro 5.12: Tabla métricas de modelos comparativa Validación-Prueba.

el tipo de modelo con base a las mismas métricas que se han evaluado. A continuación, se analizarán las tablas 5.13, 5.14 y 5.15 donde se puede directamente comparar su desempeño las métricas *KS*, validar su *Costo* y su *AUC*, y así llegar a la conclusión sobre qué tipo de modelo es el mejor para este caso.

En la primera tabla (5.13) señala el tipo de modelo en cuestión, observamos que está ordenado de mayor a menor *KS* obtenido en la fase de validación, en la siguiente columna el *KS* de la fase test, donde para la tercera columna refleja en términos porcentuales al aumento o disminución de éste. De esta última columna se puede comparar directamente la estabilidad del modelo: el modelo ANN selu5_ h1_ 100_ 0.0005 tuvo la mayor estabilidad ya que solo se redujo en un 7.50 %, por otro lado, el modelo logístico log_ cs_ v2 aumentó

en un 54.93 %, es decir, el menos estable. También se puede observar que, en todos los casos, los modelos ANN fueron significativamente mucho más estables que los logísticos. Por tanto, en este rubro, los modelos ANN superaron a los logísticos.

Estadístico KS				
Tipo	Nombre del Modelo	Validación	Test	Diferencia
ANN	selu5_ h1_ 100_ 0.0005	0.555	0.513	-7.50 %
ANN	selu3_ h1_ 71_ 0.0005	0.540	0.495	-8.20 %
ANN	leaky3_ h3_ 90_ 0.0005	0.510	0.468	-8.10 %
LOGIT	log_ cs	0.371	0.441	18.70 %
LOGIT	log_ cs2	0.368	0.425	15.41 %
LOGIT	log_ cs4	0.359	0.419	16.77 %
LOGIT	log_ cs_ v2	0.271	0.420	54.93 %

Cuadro 5.13: Comparativa de modelos paramétricos y no paramétricos en la métrica KS.

Para el caso de la tabla 5.14 donde se compara el costo (ordenada de menor a mayor en la fase de validación), es evidente que también en todos los casos los modelos ANN tuvieron mejores valores reales en los costos por clasificación tanto en la fase de validación como en la de prueba; alrededor de un 50 % menos costoso, es decir, mejor desempeño de clasificación. Aun cuando los modelos logísticos tuvieron un mejor desempeño en la fase de prueba respecto a la de validación (disminución de costo hasta en un 23.80 %), esto es, su costo de clasificación fue menor, no fue mejor que los resultados en los modelos ANN. De esta manera, se tiene que los modelos ANN fueron mejores para este caso dado que el mejor modelo es leaky3_ h3_ 90_ 0.0005.

Costo				
Tipo	Nombre del Modelo	Validación	Test	Diferencia
ANN	selu5_ h1_ 100_ 0.0005	34.57	41.16	19.10 %
ANN	selu3_ h1_ 71_ 0.0005	34.97	42.07	20.30 %
ANN	leaky3_ h3_ 90_ 0.0005	35.42	39.52	11.60 %
LOGIT	log_ cs4	67.60	64.93	-3.94 %
LOGIT	log_ cs	68.27	62.60	-8.30 %
LOGIT	log_ cs2	68.28	61.94	-9.28 %
LOGIT	log_cs_v2	75.60	57.60	-23.80 %

Cuadro 5.14: Comparativa de modelos paramétricos y no paramétricos en el Costo.

Por último, en la tabla 5.15 que está ordenada de mayor a menor *AUC* en la fase de validación, de donde se puede observar que los modelos ANN también superaron a los logísticos, por otro lado, para la fase de prueba ya no cumple debido a que, en algunos casos, los modelos Logit superaron a los ANN, sin embargo, para la diferencia porcentual de éstos aparentemente fue más estable para los modelos ANN. Entonces, el mejor modelo en este rubro es selu5_ h1_ 100_ 0.0005, perteneciente a los modelos no paramétricos.

AUC				
Tipo	Nombre del Modelo	Validación	Test	Diferencia
ANN	selu5_ h1_ 100_ 0.0005	0.815	0.781	-4.30 %
ANN	selu3_ h1_ 71_ 0.0005	0.806	0.769	-4.60 %
ANN	leaky3_ h3_ 90_ 0.0005	0.789	0.763	-3.30 %
LOGIT	log_ cs2	0.706	0.805	13.99 %
LOGIT	log_ cs4	0.703	0.807	14.86 %
LOGIT	log_ cs	0.702	0.797	13.50 %
LOGIT	log_cs_ v2	0.667	0.833	24.85 %

Cuadro 5.15: Comparativa de modelos paramétricos y no paramétricos en la métrica AUC.

Por otro lado, se debe tener claro aquel mejor modelo de cada tipo para así facilitar la elección entre el mejor modelo ANN y el mejor logístico. Para la elección de éste, sería de la siguiente forma,

Caso paramétrico: El mejor modelo para el caso paramétrico es log_ cs2, ya que además de tener la mayor estabilidad del estadístico *KS*, es quien mejor costo brindó, es decir, la mejor clasificación con la mayor estabilidad. Notése que, aunque el modelo log_cs_ v2 brinda el mejor costo de clasificación y métrica *AUC*, ya que mejoraron en un 54.94 % y 24.86 % respectivamente, es aquel modelo con la menor estabilidad *KS*.

Caso no paramétrico: En este caso, el mejor es leaky3_ h3_ 90_ 0.0005, debido a que si bien no tiene la mejor estabilidad con 8.10 % *KS*, no es significativamente diferente al mejor con 7.50 %, sin embargo, sí tuvo un mayor rendimiento en costo de clasificación con solo un aumento del 11.60 % respecto a la fase de validación. No obstante, como sugerencia el modelo selu5_ h1_ 100_ 0.0005 también pudiera ser implementado y monitoreado a la par con el anterior, y observar o precisar aquel con mayor exactitud en todas las métricas.

Finalmente, al comparar los mejores de cada tipo claramente el modelo ANN es superior: El valor *KS* final para el logístico y ANN fue 0.4255 y 0.468 respectivamente, donde este último es significativamente superior en estabilidad. Además, en cuanto a costo de clasificación el modelo ANN fue 36.19 % menos costoso respecto al costo del modelo logístico. Para la última métrica *AUC*, aunque el modelo logístico fue mayor, tuvo menor estabilidad.

Por tanto, en este trabajo y para estos datos, el mejor modelo fue el tipo no paramétrico leaky3_ h3_ 90_ 0.0005, es decir, una red neuronal artificial con 3 capas ocultas y en cada una con 90 neuronas aplicando una tasa de aprendizaje 0.0005, teniendo la función de activación *Leaky*, y una capa de salida con función de activación *Sigmoide*.

Capítulo 6

Conclusiones

Una vez que se ha concluido qué tipo de modelo Credit Scoring tuvo el mejor desempeño con los datos referidos, se pueden observar muchas cosas particulares, algunas que podrían ser generales y/o útiles para cuando se intenta modelar con otros datos. Es muy difícil generalizar muchos aspectos para estos tipos de modelos, ya que la modelización utilizada en este trabajo muy probablemente no sea eficiente o no aplicara si se implementara para otros tipos de datos, aun cuando se traten también para un Credit Scoring. Por ello, en esta parte se marcarán precisamente aquellos aspectos particulares y algunos que podrían ser generales para la modelización de un Credit Scoring con los modelos con base en redes neuronales artificiales y/o modelos logísticos.

Para una mejor comparación de los modelos, los datos tipo numérico en la base utilizada solo se transformaron de tal manera que su rango fuera más pequeño, esto para evitar que fuera preponderante su coeficiente o pesos en los modelos para algunas variables, manteniendo así el menor tratamiento posible en la información. Lo anterior fue para comparar qué modelo "aprende" mejor de los datos puros, donde claramente, los modelos ANN fueron mucho mejores. Se tiene que mencionar que también además, que se hizo un modelo logístico en el cual los datos fueron tratados mediante un análisis bivariante para mejorar su rendimiento y poder de predicción, y que en este trabajo se muestra las reagrupaciones referidas como variables artificiales (se utilizó el prefijo a_), de esta manera se mejoró el *AIC* en comparación con los modelos logísticos anteriores, se mejoró la parsimonia, los costos, *KS* y *AUC* en la fase de prueba, no obstante, no se obtuvieron mejores métricas que los modelos ANN. Lo anterior podría inferir que las bien entrenadas y estructuradas ANN pueden "aprender" mejor de los datos brutos que los modelos paramétricos tradicionales, ya que estos requieren diversas transformaciones para cumplir algunos o muchos supuestos sobre la distribución de los datos y que sus resultados sean válidos.

Uno de los retos enfrentados fue la situación de desbalance en la información de la variable de interés, donde a través de una correcta selección de la función de costo, en este caso de *Real-World-Weight Cross Entropy* se obtuvo un mejor resultado respecto a otras funciones tal como puramente la *Binary Cross Entropy*. Por otro lado, no es el caso de los modelos logísticos cuyo costo de clasificación fue significativamente diferente a los modelos ANN, también se observó que en éstos para todos los casos hubo una mayor estabilidad en las métricas de desempeño, lo cual implica un mejor aprendizaje de las estructuras de los datos a modelar, no obstante, no se puede validar por cuanto tiempo continuará dicha estabilidad dado que la información de este tipo, en general, su comportamiento es muy dinámico.

Una clara ventaja de los modelos logísticos sobre los ANN es la interpretabilidad en los coeficientes de las variables, esto es, se puede observar de cada variable ya sea por su magnitud numérica, presencia o ausencia de algún atributo, el peso o puntaje en la que contribuye en la estimación del cálculo de la probabilidad de impago. Evidentemente, no es el caso para un modelo ANN, ya que sus pesos son múltiples, están conectados a pesos adyacentes y sobre todo ocultos, lo cual, tiene no da cabida a una interpretación directa. Algo que se puede hacer para intentar mitigar el problema, es analizar directamente casos particulares de

cada vector de variables, y observar la contribución de estos casos específicos; cómo se va acumulando los puntajes de dichas variables en la estimación de incumplimiento.

Por otro lado, los modelos no paramétricos de redes neuronales artificiales con una adecuada estructura se pueden aprender de patrones muy complejos en los datos o de datos masivos, lo cual potencializa el nivel de aprendizaje de estos, por el contrario, un modelo logístico puede (no necesariamente) quedar en una situación de subajuste en este tipo de casos. No obstante, existe una limitante importante para el entrenamiento de una ANN con datos masivos en bruto, y este es la capacidad de procesamiento de cómputo, ya que a más variables demandará más de éste, por el otro extremo, para un modelo paramétrico generalmente se realizan análisis estadísticos que ayudan a disminuir la dimensión en los datos al seleccionar solo variables estadísticamente relacionados con la variable de interés. Esto no quiere decir que no se pueda realizar dicho análisis previo para una ANN, sino que existe para este caso una disyuntiva *reducción dimensionalidad-procesamiento de cómputo*.

Finalmente, se concluye que para este trabajo de modelización y utilizando datos sin preprocesar, que el modelo no paramétrico de redes neuronales artificiales supero en casi todos los aspectos al modelo paramétrico logístico, desde las métricas de desempeño en aprendizaje hasta el tratamiento de datos desbalanceados para su correcta clasificación, por tanto, las redes neuronales artificiales pueden ser una gran herramienta para la modelización para un Credit Scoring en sus diversas formas.

Anexos

A: Clasificación Perceptrón Simple

Se cargan las librerías necesarias, se definen los puntos y se grafican.

```
#%% Librerias necesarias

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import time
from scipy import stats
from scipy.stats import ks_2samp
#from sklearn.utils import class_weight
#%%
# liberias para la red neuronal
import tensorflow as tf
from tensorflow import keras

from sklearn.model_selection import StratifiedKFold
# Libreria para guardar
#%%
#from scipy.stats import ks_2samp
#from tensorflow.keras.datasets import cifar10
from imblearn.over_sampling import SMOTE
from imblearn.combine import SMOTETomek
from imblearn.combine import SMOTEENN
from imblearn.over_sampling import SMOTEN
from imblearn.under_sampling import TomekLinks
from imblearn.under_sampling import EditedNearestNeighbours
from imblearn.over_sampling import SMOTENC

A=np.array([[-1,-1,-1],[1,-1,1],[1,1,1],[-1,1,1],[-1/2,-1,-1],[-3/4,-3/4,-1],
 [-.9,-3,-1],[1/2,1/2,1],[0,-3/4,1],[-1/2,1/2,1],[1/4,3/4,1],[.6,-1/2,1],[-1/4,0,1],
 [-3/4,-2.5,-1],[0,-1.8,-1],[1/2,-2,1],[-1,0,-1],[-1/2,-7/4,-1],[-1,-2,-1],
 [1/4,-2,-1], [1/2,-3,-1],[-1/4,-2.5,-1],[-1/4,-.8,-1],[0,.8,1],[1/4,0,1],
 [1/4,-3,-1],[1/2,0,1], [.8,-1/2,1]])
z=A[:,-1:]
X=A[z[:,0]==1,0]
X2=A[z[:,0]==1,1]
X11=A[z[:,0]==-1,0]
X22=A[z[:,0]==-1,1]
plt.scatter(X,X2,c="black")
plt.scatter(X11,X22,c="gray")
plt.show()
Xs=A[:,:-1]
z=A[:,-1:]
X=Xs
Y=z
```

Se definen las función signo , corrección de pesos.

```
# Funcion activacion
def fsigno(x):
    if x<0:
        return -1
    elif 0<=x:
        return 1

# Funcion correccion pesos sinapticos
def correccionWs(w,x,z,y,lr):
    w[0]=w[0]+lr*(z-y)*x[0] #z[0]
    w[1]=w[1]+lr*(z-y)*x[1]
    w[2]=w[2]+lr*(z-y)*x[2]
    return w
```

Primera iteración con pesos iniciales aleatorios.

```
# Tasa de aprendizaje
ta=.25
iteraciones=50 # Numero de iteraciones en entrenamiento

# Se generan los pesos y el bias incial

w=np.random.rand(2)*2-1 # pesos iniciales aleatorios(2)
b=np.random.rand(1)*2-1 # bias inicial aleatorio(1)

# Vectores de pesos y valores de x

ws=np.array([w[0],w[1],b[0]]) # vector pesos
print("Primer vector pesos:\n",ws)
recta=lambda x: (ws[2]-ws[0]*x)/(ws[1])  # lambda es funcion anonima
```

Iteraciones de aprendizaje.

```
for k in range(iteraciones):

    for l in range(len(z)):
        xs=np.array([Xs[l,0],Xs[l,1],-1]) # se mueve el renglon
        y=fsigno(ws@xs)
        if z[l]-y!=0:
            j=0
            while z[l]-y!=0:
                ws=correccionWs(ws,xs,z[l],y,ta)
                y=fsigno(ws@xs)
                j=j+1
                if j>100:
                    break
    W=np.array([[ws[0]],[ws[1]]])
    recta=lambda x: (ws[2]-ws[0]*x)/(ws[1])
    _x=np.linspace(-1,1,100)
    time.sleep(0.5)
    plt.plot(_x,recta(_x))
    plt.scatter(X,X2,c="black")
    plt.scatter(X11,X22,c="gray")
    plt.show()
    clear_output(wait=True)
print("Vector de pesos encontrado pesos:\n",ws)
```

B: Algoritmo ANN

El siguiente código es de la generación de una Red Neuronal Artificial, pudiéndose aplicar cualquier estructura o topología.

Código que define la clase capa:

```
import pandas as pd
import numpy as np

class neural_layer():
    def __init__(self, n_conn ,n_neur,act_f):
        self.act_f=act_f
        self.b=np.random.rand(1,n_neur)*2-1 #bias
        self.W=np.random.rand(n_conn,n_neur)*2-1 #weights
```

Código función de activación sigmoide:

```
sign=(lambda x: 1/(1+np.e**(-x)),lambda x: x*(1-x))
```

Código que genera la red neuronal dada una estructura

```

# CREA TOPOLOGIA DE RED NEURONAL
def create_nn(topology,act_f): #crea neural network
    nn = [] # lista vacia
    for l,layer in enumerate(topology[:-1]):
        nn.append(neural_layer(topology[l],topology[l+1],act_f))
    return nn
```

Función de costo: ECM

```
f_cost=(lambda Yp,Yr: np.mean((Yp-Yr)**2),lambda Yp,Yr: Yp-Yr)
```

Función de entrenamiento y predicción dados los datos y la estructura dado la matriz de datos X, el vector de la variable dependiente Y, función costo, tasa de aprendizaje y el objeto de estructura de red.

```

def train(neural_net, X, Y, f_cost, lr=0.05, train=True):

    #Alimentacion hacia adelante: vector de entrada y pasarlo capa por capa
    out=[(None,X)]  #salidas de las capas y se inicia
    for l,layer in enumerate(neural_net):
        #Notamos [-1](ultimo par)[1](elemento a que es la salida)
        z=out[-1][1]@neural_net[l].W + neural_net[l].b
        a=neural_net[l].act_f[0](z)
        out.append((z,a))
    if train:
        #Backpropagation
        deltas=[]
        #bucle de adelante hacia atras
        for l in reversed(range(0,len(neural_net))):
            z=out[l+1][0]
            a=out[l+1][1]

            if l==len(neural_net)-1:
                #como queremos el ingreso de las deltas al primer elemento
                deltas.insert(0,f_cost[1](a,Y)*neural_net[l].act_f[1](a))
            else:
```

```
                #calcular delta respecto a capa previa
                deltas.insert(0,deltas[0]@_W.T * neural_net[l].act_f[1](a))

            _W=neural_net[l].W

           #Descenso del gradiente: optimiza los parametros de la red.
            neural_net[l].b=neural_net[l].b-np.mean(deltas[0],axis=0,keepdims=True)*lr
            neural_net[l].W=neural_net[l].W-out[l][1].T@deltas[0]*lr

    return out[-1][1]
```

Algoritmo que ejecuta el entrenamiento dado el número de atributos, los epoch deseados y la estructura.

```
import time
from IPython.display import clear_output

n_atributos=75
epochs=3000
topology=[n_atributos,76,38,17,1] #Estructura
neural_n=create_nn(topology,sign)
loss=[]
print(epochs/50)
for i in range(epochs+1):

    #Primer valor de prediccion
    pY=train(neural_n, X, Y, f_cost, lr=0.07)

    if i%50==0:
        #print(i)
        loss.append(f_cost[0](pY,Y))
        plt.plot(range(len(loss)),loss)
        plt.show()
        time.sleep(0.5)
        if np.array(loss).min()<0.005:
            print("Error limite alcanzado")
            break
```

C: Algoritmo ANNs para diferente función de activación

Función generadora estructura ANN

```
def ann_estructura(topo, f_act, ta=0.001,nombre="Model"):

    ker="he_normal"
    if f_act=="leaky":
        f_act=tf.keras.layers.LeakyReLU(alpha=0.2)
    if f_act=="selu":
        ker="lecun_normal"

    #Generar modelo
    mod=keras.models.Sequential(name=nombre)
    mod.add(keras.layers.Dense(topo[1],input_shape=(topo[0],),
                      activation=f_act,kernel_initializer=ker))
    for l,neuronas in enumerate(topo[2:-1]):
        etiqueta_l="oculta_"+str(l+2)
        mod.add(keras.layers.Dense(neuronas, activation=f_act,
                                   kernel_initializer=ker,
                                   name=etiqueta_l))
    mod.add(keras.layers.Dense(1, activation="sigmoid",name="Score"))

    #Metricas a mostrar en el entrenamiento
    metricas = [
    #keras.metrics.Precision(name="precision"),
    #keras.metrics.FalseNegatives(name="FN"),
    #keras.metrics.TruePositives(name="TP"),
    #keras.metrics.BinaryAccuracy(name='binary_accuracy', threshold=0.35),
    keras.metrics.Recall(name="Recall")]

    #Se termina la estructura de la ANN
    mod.compile(loss="binary_crossentropy",
              optimizer=keras.optimizers.SGD(lr=ta),
              metrics=metricas)

    return mod
```

Algortimo entrenamiento ANN

```

name_model="prueba_smote_hibrido"
topologia=[63,600,600,1]
modelo=ann_estructura(topologia, f_act="selu", ta=0.001,nombre=name_model)
modelo.summary()


#%% Punto de guardado y evitar overfitting
ptoG = keras.callbacks.ModelCheckpoint(name_model+".pb",save_best_only=True)
parar = keras.callbacks.EarlyStopping(patience=10, restore_best_weights=True)

history = modelo.fit(X_train, y_train, epochs=100,
                     validation_data=(X_val,y_val),
                     callbacks=[ptoG, parar])

# Grafica Error vs Epochs
plot_train=pd.DataFrame(history.history)
plot_train.plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.legend(loc="lower right",ncol=2)
plt.show()
```

D: Algoritmo prueba χ^2

Algoritmo prueba χ^2: Independecia entre variables y de respuesta dada una tabla de contigencia de variables categóricas

```
def statchi(table,vari ,alfa=0.05,info=True,rastreo=False,depend=False):
    #De la tabla de contingencias se obtienen las sumas.
    n_atributos =len(table.iloc[1,:])
    sa =table.sum()
    sr =table.T.sum()
    sT =table.sum().sum()
    print("-----------------------------------------")
    print("VARIABLE:",vari+1)
    print(" ")
    print("n_atributos=",n_atributos)

    if info:
        print(" ")
        print("SUMA ATRIBUTOS: ",sa)
        print(" ")
        print("SUMA RENGLONES: ",sr)
        print(" ")
        print("SUMA TOTAL: ",sT)
    #Se obtiene el estadistico ji-cuadrada Xc
    Xc=0
    for j in range(n_atributos):
        for i in range(2):
            fe=(sa[j]*sr[i])/sT    #frecuencia esperada
            Xc+=((table.iloc[i,j]-fe)**2)/fe
            if rastreo:
                print("atributo=",j+1," Y=",i," Xc=",round(Xc,5))
    #Se realiza la prueba de hipotesis
    glibertad=n_atributos-1
    cota=stats.chi2.ppf(1-alfa, glibertad)
    if info:
        print(" ")
        print("g.l.=",glibertad)
    print(" ")
    if cota<Xc:
        if depend:
            return print(1)
        else:
            return print("***depend*** Xc=", round(Xc,4),"> Cota=",round(cota,4))
    else:
        if depend:
            return print(0)
        else:
            return print("independ Xc=", round(Xc,4),"> Cota=",round(cota,4))
```

Código donde se generan las tablas de contigencia y se aplica para cada variable explicativa.

```
# Se buscan las variables en las que Y dependa de X_i

for k in range(20):                  #Las 20 variables
    tabla=suma.iloc[:,indices[k]]    #tabla auxiliar
    statchi(tabla,k,alfa=0.1, info=False, rastreo=False,depend=True)
```

E: Algoritmos Medidas de desempeño

Código de Matriz confusión y Métricas

```
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score,cohen_kappa_score

def predecir_clase(y_proba,corte=0.5):
    clase=[]
    for j in y_proba:
        if j<corte:
            clase.append(0)
        else:
            clase.append(1)
    return np.array(clase)

def display_metricas(y_hat_class,y_test_class):
    pres=precision_score(y_test_class, y_hat_class)
    recall=recall_score(y_test_class, y_hat_class)
    g_measure=np.sqrt(pres*recall)
    MC=confusion_matrix(y_test_class, y_hat_class)
    P_N=MC@[1,1]
    tpr=MC[0][0]/P_N[0]
    tnr=MC[1][1]/P_N[1]
    G_mean=np.sqrt(tpr*tnr)
    bac=(tpr+tnr)/2
    print("=========================================")
    print("Matriz de confusion:\n fila=True: 0,1 col=Pred: 0,1\n",MC )
    print("=========================================")
    print("Exactitud",round(accuracy_score(y_test_class, y_hat_class),5))
    print("=========================================")
    print("Precision",round(pres,5))
    print("=========================================")
    print("Recall",round(recall,5))
    print("=========================================")
    print("f beta score",round(f1_score(y_test_class, y_hat_class,average="weighted")
    ,5))
    print("=========================================")
    print("kappa_metrica",round(cohen_kappa_score(y_test_class, y_hat_class),5))
    print("=========================================")
    print("G_measure",round(g_measure,5))
    print("=========================================")
    print("G_mean",round(G_mean,5))
    print("=========================================")
    print("BAC",round(bac,5))
    print("=========================================")
```

Código Estadístico KS

```

def ks_estadistico(yhat,ytest,n_cortes=400,plot=False,error=0.04):
    ymalos=yhat[ytest==1]
    ybuenos=yhat[ytest==0]
    cortes=np.linspace(0, 100, n_cortes)
    p_m=np.percentile(ymalos,cortes)
    p_b=np.percentile(ybuenos,cortes)
    #diferencias
    ks=np.abs(p_m-p_b)
    #maxima dif= est ks
    max_dis=ks.max()
    #se obtiene el corte de maxima distancia
    indice=[]
```

```
    corte=[]
    for i in range(n_cortes):
        if ks[i]==max_dis:
            indice.append(i)
            corte.append(cortes[i])

    if plot==True:
        plt.plot(cortes,p_m,color="darkred",label="Malos",linestyle="solid",linewidth
    =2)
        plt.plot(cortes,p_b,color="darkgreen",label="Buenos",linestyle="solid",
    linewidth=2)
        plt.axvline(corte,p_b[indice]+error-0.01,p_m[indice]+error+0.02,
                    label="Est. KS "+ str(round(max_dis,4)),color="silver",linestyle="
    --")
        plt.axvline(corte,0,0.15,label="Corte "+ str(round(corte[0],2)),color="black")
        plt.legend(loc='upper left')
        plt.grid()

    if plot==False:
        return round(max_dis,5), round(corte[0],5)
```

Curva ROC y AUC

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score


def curva_ROC(yhat, ytest,area=False):
    if area==False:
        FPR, TPR, cortes = roc_curve(ytest, yhat,pos_label=1)
        plt.plot(FPR, TPR, linewidth=2, color="black", label="Curva ROC")
        plt.plot([0, 1], [0, 1], 'k--') # Dashed diagonal
        plt.xlabel("Tasa FP")
        plt.ylabel("Tasa TP")
        plt.legend(loc='lower right')
        plt.title("Curva ROC")
    else:
        return round(roc_auc_score(ytest,yhat),5)


def plot_precision_recall_vs_cortes(yhat, ytest ,p_vs_r=False, area=False):
    precisions, recalls, cortes = precision_recall_curve(ytest, yhat, pos_label=1)

    if p_vs_r==False:
        plt.plot(cortes, precisions[:-1], "b--", label="Precision",color="green")
        plt.plot(cortes, recalls[:-1], "g-", label="Recall",color="blue")
        plt.legend(loc='lower center')
        plt.grid(b=True, which='major', color='gray', linestyle='-')
        plt.minorticks_on()
        plt.grid(b=True, which='minor', color='black', linestyle='-', alpha=0.2)
        plt.xlabel("Cortes")
    else:
        if area==True:
            return round(auc(recalls , precisions),5)
        else:
            plt.plot(recalls[:-1],precisions[:-1],color="blue")
            plt.xlabel("Recall")
            plt.ylabel("Precision")
            plt.axvline(.8, label="80% recall",color="gray",linestyle='--')
            plt.legend(loc='lower center')
            plt.grid()

def plot_costo(yhat, ytest, datos=False,IR=2.33,num_cortes=100):
```

```
    n_cortes=np.linspace(0, 1, num_cortes)
    costos=[]
    for i in n_cortes:
        MatC=confusion_matrix(ytest, predecir_clase(yhat,corte=i))
        cost_i=MatC[0][1]+IR*MatC[1][0]
        costos.append(cost_i)
        #print(cost_i)

    min_cost=np.min(costos)
    media_cost=np.mean(costos)
    corte_min=n_cortes[costos==min_cost]
    if datos:
        return print("costo min= ", round(min_cost,2)," media=",round(media_cost,2),
                     " corte costo min= ", corte_min )
    plt.plot(n_cortes,costos,c="blue")
    #plt.plot(n_cortes,costos,c="black")
    plt.axvline(corte_min[0],0,0.1,
                label="Cut-off costo min ",
                color="red",linestyle="dashed")
    plt.legend(loc="upper center")
    plt.title("Curva Costos del Modelo")
    plt.xlabel("Corte")
    plt.ylabel("Costo")
    plt.grid()

def costos_display(yhat, ytest, IR=2.33, num_cortes=100):
    n_cortes=np.linspace(0, 1, num_cortes)
    costos=[]
    for i in n_cortes:
        MatC=confusion_matrix(ytest, predecir_clase(yhat,corte=i))
        cost_i=MatC[0][1]+IR*MatC[1][0]
        costos.append(cost_i)
        #print(cost_i)

    min_cost=np.min(costos)
    corte_min=n_cortes[costos==min_cost]
    return round(min_cost,2), round(np.min(corte_min),4)
```

Función Real Weight Cross Entropy

```
def tf_brwwce(y_true, y_pred):
    log_0=tf.math.log(1-y_pred)
    log_1=tf.math.log(y_pred)
    #Pesos: los costos
    W_FN=tf.constant(2.334,dtype=tf.float32)
    W_FP=tf.constant(1.0,dtype=tf.float32)
    uno=tf.constant(1.0,dtype=tf.float32)
    #res=-tf.reduce_mean(y_true* log_1 +(1-y_true)*log_0)
    res=-tf.reduce_mean(W_FN*tf.cast(y_true,dtype=tf.float32)*log_1+
                        W_FP*(uno-tf.cast(y_true,dtype=tf.float32))*log_0)
    #se convierte a tensor para correcto funcionamiento como funcion error
    return res
```

F: Algortimo de Partición Dataset y técnica de remuestreo

Partición de la información

```
semilla=10

# Particion: .1 test y .9 train
X_train_full, X_test, y_train_full, y_test = train_test_split(df.iloc[:,:-1],
                    df.iloc[:,-1:], test_size=.1, random_state=semilla,
                    stratify=df.iloc[:,-1:])
X_sm, X_val, y_sm, y_val= train_test_split(X_train_full, y_train_full,
                    test_size=.12, random_state=semilla, stratify=y_train_full)
```

Algoritmos remuestreo

```

# Oversampling SMOTE
smote= SMOTE(random_state=semilla)
X_train, y_train =smote.fit_resample(X_sm,y_sm)
remuestreo="SMOTE"

# Hibrido SMOTE+Tomek Link (implementado).

smote_tomek = SMOTETomek(sampling_strategy="minority",random_state=semilla)
X_train, y_train = smote_tomek.fit_resample(X_sm, y_sm)
remuestreo="SMOTE+Tomek Link"

#  SMOTE+ ENN
smote_enn = SMOTEENN(sampling_strategy="minority",random_state=semilla)
X_train, y_train = smote_enn.fit_resample(X_sm, y_sm)
remuestreo="SMOTE+ ENN"

print("Tecnica de remuestreo: ", remuestreo)
print("dimension X_train")
print(X_sm.shape)
print("dimension  remuestreo X_train")
print(X_train.shape)
print("Se valida balance")
print(y_train.value_counts(normalize=True)*100)
```

G: Algortimo de Partición Dataset y técnica de remuestreo

Algoritmos Regresión Logística

```

#liberia y funcion de carga

library(tidyverse)

leerCSV<-function(namefile){
  etiqueta3<-paste(namefile,"csv",sep=".")
  myfile<-paste(".../logit_cs/",etiqueta3,sep = "")
  read.csv(myfile, stringsAsFactors = FALSE)
}

#funcion reclasificacion de las categorias
df_train<-leerCSV("df_trainlog")
class_Purpose<-function(x){
  if(x==10 | x==6){ return(1)}
  else{
      if(x==5 | x==0){ return(2)}
      else{
          if(x==9 | x==2 | x==4){ return(3)}
          else {if(x==1 | x==3 | x==8) return(4)}
         }
      }
}#fin_funcion

#Transformacion a variables categoricas
for(j in 1:18){
  df_train_v2[,j]<-as.factor(df_train_v2[,j])
}

df_train_v3<-df_train_v2 %>%
  select(-Occupation, -Telephone, -a_Occupation,
  -a_Length_of_current_employment,
  -a_Most_valuable_available_asset,
  -Foreign_Worker,
  -a_No_of_Credits_at_this_Bank,
  -No_of_dependents)


log_cs<-glm(Y~.-1,family=binomial(link ="logit"),
          data=df_train_v3)


summary(log_cs)
```

H: Definición funciones ANN

Funciones Métricas

```
 # plot y estadistico ks

def ks_estadistico(yhat,ytest,n_cortes=400,plot=False,error=0.04):
    ymalos=yhat[ytest==1]
    ybuenos=yhat[ytest==0]
    cortes=np.linspace(0, 100, n_cortes)
    p_m=np.percentile(ymalos,cortes)
    p_b=np.percentile(ybuenos,cortes)
    #diferencias
    ks=np.abs(p_m-p_b)
    #maxima dif= est ks
    max_dis=ks.max()
    #se obtiene el corte de maxima distancia
    indice=[]
    corte=[]
    for i in range(n_cortes):
        if ks[i]==max_dis:
            indice.append(i)
            corte.append(cortes[i])

    if plot==True:
        plt.plot(cortes,p_m,color="darkred",label="Malos",linestyle="solid",linewidth
    =2)
        plt.plot(cortes,p_b,color="darkgreen",label="Buenos",linestyle="solid",
    linewidth=2)
        plt.axvline(corte,p_b[indice]+error-0.01,p_m[indice]+error+0.02,
                    label="Est. KS "+ str(round(max_dis,4)),color="silver",linestyle="
    --")
        plt.axvline(corte,0,0.15,label="Corte "+ str(round(corte[0],2)),color="black")
        plt.legend(loc='upper left')
        plt.grid()

    if plot==False:
        return round(max_dis,5), round(corte[0],5)


def copiar_v(vec):
    nvo_vec=[]
    for vec_i in vec:
        nvo_vec.append(vec_i)
    return nvo_vec

def histograma_prob(yhat,ytest,bines=10):
    bads=copiar_v(yhat[ytest==1])
    goods=copiar_v(yhat[ytest==0])
    bins = np.linspace(0, 1, bines)
    plt.hist(goods, bins, density= True, rwidth=.9,
          alpha = 0.8, label="Buenos", color="darkgreen")
    plt.hist(bads, bins, density= True, rwidth=.9,
          alpha = 0.8, label="Malos", color="darkred")
    plt.legend(loc='upper right')
    plt.title("Distribucion prediccion en las clases")
    plt.xlabel("Probabilidad incumplimiento")



from sklearn.metrics import precision_recall_curve, auc

```

```
def plot_precision_recall_vs_cortes(yhat, ytest ,p_vs_r=False, area=False):
    precisions, recalls, cortes = precision_recall_curve(ytest, yhat, pos_label=1)

    if p_vs_r==False:
        plt.plot(cortes, precisions[:-1], "b--", label="Precision",color="green")
        plt.plot(cortes, recalls[:-1], "g-", label="Recall",color="blue")
        plt.legend(loc='lower center')
        plt.grid(b=True, which='major', color='gray', linestyle='-')
        plt.minorticks_on()
        plt.grid(b=True, which='minor', color='black', linestyle='-', alpha=0.2)
        plt.xlabel("Cortes")
    else:
        if area==True:
            return round(auc(recalls , precisions),5)
        else:
            plt.plot(recalls[:-1],precisions[:-1],color="blue")
            plt.xlabel("Recall")
            plt.ylabel("Precision")
            plt.axvline(.8, label="80% recall",color="gray",linestyle='--')
            plt.legend(loc='lower center')
            plt.grid()
```

Algoritmo final ANN

```

#Se define el Grid de hiperpametros

guardar=".../grid_mods_original/"

fs_act=["relu","leaky","elu","selu"]
nns=[71,90,100,120,150,200,250,300]
ts_a=[0.00001, 0.00005,0.0001,0.0005]
dim_input=df.shape[1]-1
estr="_hl3_" #estructura ANN
print("Tecnica de remuestreo: ", remuestreo, "\n")
print(" Dimension set: ",dim_input,"\n",
      "Funciones activacion: \n", fs_act,"\n",
      "Tasas de aprendizaje: ", ts_a,"\n",
      "No. Capas ocultas: ",estr[3],"\n",
      "No. Neuronas: \n", nns)


#%% Se ejecutuva el Grid de Modelos

nom_models_pb=[]
nom_models_csv=[]
for f_a in fs_act:
    for nn in nns:
        for tap in ts_a:
            n_mod=eti_remu+f_a+estr+str(nn)+"_"+str(tap)
            n_mod_s=n_mod+".pb"
            n_mod_csv=n_mod+".csv"
            nom_models_pb.append(n_mod_s)
            nom_models_csv.append(n_mod_csv)
            topology=[dim_input,nn,nn,nn,1]
            #print(n_mod)
            #print(topologia)
            #print(n_mod_s)
            modelo=ann_estructura(topology, f_act=f_a, ta=tap, nombre=n_mod,
                                  balanceado=False)
            pG = keras.callbacks.ModelCheckpoint(filepath=guardar+n_mod_s,
                                                 save_best_only=True, verbose=0)
            stop = keras.callbacks.EarlyStopping(patience=6,
```

```
                                                    restore_best_weights=True)
            historia = modelo.fit(X_train, y_train, epochs=100,
                     validation_data=(X_val,y_val),
                     callbacks=[pG, stop])
            pd.DataFrame(historia.history).to_csv(guardar
                                    +n_mod_csv,index=False)

pd.DataFrame(nom_models_pb).to_csv(guardar+eti_remu+estr+"Mods_pb.csv",
                                   index=False,
                                 header=["Modelo"])
pd.DataFrame(nom_models_csv).to_csv(guardar+eti_remu+estr+"Mods_csv.csv",
                                    index=False,
                                 header=["Modelo_csv"])
```

Bibliografía

[1] Tarek Amr. *Hands-on Machine Learning with scikit-learn and Scientific Python Toolkits*. 2.ª ed. Packt, 2010.

[2] Raymond Anderson. *The Credit Scoring Toolkit*. 1.ª ed. Oxford,university press, 2007.

[3] Carlos Serrano-Cinca y Begoña Gutiérrez-Nieto. «The use of profit scoring as an alternative to credit scoring systems in peer-to-peer (P2P) lending». En: *ELSEVIER* (2016).

[4] Christian Bluhm. *Introduction to Credit Risk Modeling*. 2.ª ed. Springer, 2010.

[5] Francois Chollet. *Deep Learning with Python*. 1.ª ed. Manning Publications Co., 2018.

[6] Charles Elkan. «The Foundations of Cost-Sensitive Learning». En: *Department of Computer Science and Engineering* ().

[7] Wolfgang Ertel. *Introduction to Artificial Intelligence*. 2.ª ed. Springer, 2017.

[8] David Trujillo Fernández. «Aplicación de Metodologías Machine Learning a Gestión de Riesgo de Crédito». Director: F. Águeda Mata Hernández. Tesis de mtría. Madrid, España: Universidad Politécnica de Madrid, 2017.

[9] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow*. 2.ª ed. O'REILLY, 2019.

[10] HAIBO HE y YUNQIAN MA. *Imbalanced Learning: Foundation, Algorithms, and Applications*. 1.ª ed. IEEE PRESS, 2013.

[11] Jeff Heaton. *AIFH, Volume 3: Deep Learning and Neural Networks*. 1.ª ed. Heaton Research, Inc., 2015.

[12] Paul G. Hoel. *Introduction to Probability Theory*. 1.ª ed. Houghton Mifflin Company Boston, 1971.

[13] Benjamin Johnston e Ishita Mathur. *Applied Supervised Learning with Python*. 1.ª ed. Packt, 2019.

[14] Igor Kononenko. «Cost-Sensitive Learning with Neural Networks». En: *Jhon Wiley and Sons, Ltd.* (1998).

[15] Edward M. Lewis. *An Introduction to Credit Scoring*. 1.ª ed. Athena Press, 1994.

[16] Yuxi (Hayden) Liu. *Hands-On Deep Learning Architectures with Python*. 1.ª ed. Packt, 2019.

[17] Cuicui Luo. «A deep learning approach for credit scoring using credit default swaps». En: *ELSEVIER* (2016).

[18] David Lovelock y Marilou Mendel. *An Introduction of Mathematics of Money*. 1.ª ed. Springer, 2000.

[19] Loretta J. Mester. «What's the Point of Credit Scoring». En: *Federal Reserve Bank of Philadelphia* (1997), págs. 3-16.

[20] James D. Miller. *Statistics for Data Science*. 1.ª ed. Packt, 2017.

[21] Douglas C. Montgomery. *Introduction to Linear Regression Analysis*. 5.ª ed. WILEY, 2012.

[22] Alba Morera Munt. «Introducción a los modelos de redes neuronales artificiales, el perceptón y multicapa». Director: Tomás Alcalá Nalvaiz. Tesis de mtría. España: Universidad de Zaragoza, 2018.

[23] Soraida Nieto Murillo. «Crédito al consumo: La estadística aplicada a un problema de Riesgo Crediticio». Asesora Dra. Blanca R. Pérez S. Tesis de mtría. Distrito Federal, México: Universidad Autónoma Metropolitana, 2010.

[24] Kevin W. Bowyer Nitesh V. Chawla. «SMOTE: Synthetic Minority Over-sampling Technique». En: *Journal of Artificial Intelligence Research 16* (2002).

[25] Daniel Peña. «Análisis de datos multivariantes». En: Inbook, 2002. Cap. Discriminación logística y otros métodos de clasificación.

[26] Maria Luisa Puertas Medina RosaMartí Selva. «ANÁLISIS DEL CREDIT SCORING». Español. En: *RAE - Revista de Administração de Empresas* (2013). ISSN: 0034-7590. URL: https://www.redalyc.org/articulo.oa?id=155127485011.

[27] Brandon Reagen. *Deep Learning for Computer Architects*. 1.ª ed. SYNTHESIS LECTURES ON COMPUTER ARCHITECTURE, 2017.

[28] Alberto Fernández y Salvador García. *Learning from Imbalanced Data Sets*. 1.ª ed. Springer, 2018.

[29] S. C. Shapiro. *Artificial Intelligence*. 2.ª ed. New York, Encyclopedia of Artificial Intelligence, 1994.

[30] Sandro Skansi. *Introduction to Deep Learning, from logical calculus to artificial intelligence*. 1.ª ed. Springer, 2018.

[31] Lyn C. Thomas. *Credit Scoring and its applications*. 1.ª ed. SIAM monographs on mathematical modeling y computation, 2002.

[32] Dennis L. Wilson. «Asymptotic Properties of Nearest Neighbor Rules Using Edited Data». En: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-2.3 (1972), págs. 408-421.

[33] Samuel Wookey Yaoshiang Ho. «The Real-World-Weight Cross-Entropy Loss Function: Modeling the Cost of Mislabeling». En: *Thinky AI Research* ().

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

ACTA DE EXAMEN DE GRADO

No. 00227
Matrícula: 2193803504

Credit Scoring: Un comparativo de un modelo paramétrico y un no paramétrico.

DAVID HERNANDEZ GOMEZ
ALUMNO

REVISÓ

MTRA. ROSALIA SERRANO DE LA PAZ
DIRECTORA DE SISTEMAS ESCOLARES

En la Ciudad de México, se presentaron a las 11:00 horas del día 28 del mes de noviembre del año 2022 en la Unidad Iztapalapa de la Universidad Autónoma Metropolitana, los suscritos miembros del jurado:

DRA. HORTENSIA JOSEFINA REYES CERVANTES
DRA. BLANCA ROSA PEREZ SALVADOR
DR. ALBERTO CASTILLO MORALES

Bajo la Presidencia de la primera y con carácter de Secretario el último, se reunieron para proceder al Examen de Grado cuya denominación aparece al margen, para la obtención del grado de:

MAESTRO EN CIENCIAS (MATEMÁTICAS APLICADAS E INDUSTRIALES)

DE: DAVID HERNANDEZ GOMEZ

y de acuerdo con el artículo 78 fracción III del Reglamento de Estudios Superiores de la Universidad Autónoma Metropolitana, los miembros del jurado resolvieron:

Aprobar

Acto continuo, la presidenta del jurado comunicó al interesado el resultado de la evaluación y, en caso aprobatorio, le fue tomada la protesta.

DIRECTOR DE LA DIVISIÓN DE CBI

Roman Linares Romero
DR. ROMAN LINARES ROMERO

PRESIDENTA

DRA. HORTENSIA JOSEFINA REYES CERVANTES

VOCAL

DRA. BLANCA ROSA PEREZ SALVADOR

SECRETARIO

DR. ALBERTO CASTILLO MORALES