UNIVERSIDAD
TORCUATO DI TELLA

Típo de documento: Tesis de Maestría

Maestría en Econometría

Preparación de Datos Para Aprendizaje Automático

Autoría: Usuga, Ivan
Año de defensa de la tesis: 2023

¿Cómo citar este trabajo?

Usuga, I.(2023) "Preparación de Datos para Aprendizaje Automático". [Tesis de Maestría. Universidad Torcuato Di Tella]. Repositorio Digital Universidad Torcuato Di Tella
https://repositorio.utdt.edu/handle/20.500.13098/12186

El presente documento se encuentra alojado en el Repositorio Digital de la Universidad Torcuato Di Tella bajo una licencia Creative Commons Atribución-No Comercial-Compartir Igual 4.0 Argentina (CC BY-NC-SA 4.0 AR)
Dirección: https://repositorio.utdt.edu

Biblioteca Di Tella

Universidad Torcuato Di Tella

Preparación de Datos Para Aprendizaje Automático

Tesis MEC: – Profesor(a): Martin Gonzalez-Rozada

Ivan Usuga
15-Junio-2023

Lista Figuras:

Resumen:

"La ciencia de datos es un conocimiento que entendemos tan bien que podemos enseñarlo a una computadora. Todo lo demás es arte" (Rosana, 2023). Mientras existan códigos que recreen un evento o datos que simulen el comportamiento de una situación, las computadoras estarán prestas aprender.

El objetivo de este ensayo es generar ciertos procedimientos a tener en cuenta en la base de datos después de haber sido recolectada, de tal modo que los datos queden a punto para realizar análisis predictivos ya sea usando técnicas estadísticas convencionales o de aprendizaje automático.

Ya que los datos son el elemento principal tanto para el análisis predictivo o la toma de decisiones es fundamental darle un manejo adecuado a la recopilación de datos relevantes, a la identificación de los datos faltantes o erróneos y a la transformación de los mismos en un formato adecuado para el procesamiento de los mismos, ya sea usando técnicas como regresiones lineales o modelos de aprendizaje automático.

La ruta de trabajo a seguir consta de dos secciones. La primera parte correspondiente al manejo de los datos para verificar anomalías que se presentan durante el proceso de recopilación de la información, las cuales necesitan ser corregidas para darles un formato correcto y así dejarlas funcionales para usos posteriores. El dataset a usar para este fin se cargará desde el repositorio de la Universidad de California correspondiente al Censo de los Estados Unidos 2016 para constatar quienes de los encuestados no tienen seguro de salud.

Basados en los datos referentes al censo 2016 USA, se buscará detectar problemas específicos dentro de ellos recurriendo tanto a un análisis analítico como gráfico con el propósito de proporcionar luego herramientas sólidas para un adecuado tratamiento de los mismos antes de alimentar cualquier técnica de predicción. Sin dicha preparación, los modelos no estarían en capacidad de recibir la información y proporcionar resultados confiables. El análisis de los datos permite garantizar la calidad y la eficacia de los modelos y extraer elementos que permitan comprender las relaciones y patrones subyacentes entre ellos brindando conocimiento valioso para la toma de decisiones estratégicas y operativas en diversas áreas tales como marketing, finanzas, salud, investigación y muchos otros campos.

La meta propuesta es poner a punto el dataset antes de generar algún pronostico y donde el analista luego pueda tener la capacidad y confianza de hacer seguimiento continuo al proyecto, monitorear el rendimiento del mismo en producción, detectar las posibles desviaciones o errores y la adaptación del modelo para mejorarlo a largo plazo. Para lo cual se requiere utilizar métricas y técnicas de validación para medir la precisión y eficacia de los resultados obtenidos partiendo de la premisa de tener un correcto formato de los datos.

Para la segunda parte del trabajo correspondiente a la estimación utilizando diferentes técnicas (regresión logística y aprendizaje automático), la base de datos a emplear proviene de la plataforma digital *GitHub* en la cual se pueden encontrar dataset previamente formateados para ser usados directamente en modelos de predicciones estadísticas. Estos

datos provienen de una entidad financiera no especificada cuyo objetivo es determinar si un cliente entrara en impago de una obligación crediticia.

En esta parte final del trabajo, partiendo del dataset default para clientes con obligaciones crediticias no cumplidas se simulan los modelos con el objetivo de mostrar lo simple de usarlos siguiendo las instrucciones del manual de referencia de cada uno de ellos, después de que los datos han sido adecuados para el uso de los algoritmos generando resultados de alta confiabilidad. Las técnicas convencionales como regresiones lineales a menudo asumen relaciones lineales entre las variables. En cambio, el aprendizaje automático está en capacidad de modelar relaciones no lineales. Un ejemplo podría ser el riesgo crediticio, donde existen múltiples factores y variables de incidencia que influyen en la evaluación del riesgo donde las relaciones entre ellas son complejas y podrían ser no lineales. El modelo de aprendizaje automático está en capacidad de extraer patrones complejos eficientemente en grandes volúmenes de información. Además, una característica adicional corresponde a la automatización de ciertos tipos de tareas de procesos sofisticados ya que los modelos inician en los datos y van aprendiendo de ellos mismos reduciendo así los tiempos y el recurso humano requerido en los análisis estadísticos generando un incremento de la productividad en las áreas donde se utilice.

El lector de este trabajo estará en capacidad de corregir errores originados tanto en la recolección de los datos como en la manipulación de los mismos y procesarlos luego con una o varias técnicas para pronosticar una variable resultado de interés.

Palabras Claves:

Procesamiento de datos, skimr, wvplots, hexbin, vtreat, dataexplorer, scales, training_set, test_set, glm, knn, random forest, xgboost, roc, auc.

Abstract

The certainty of a prediction, the reliability of a classification is the result of an adequate preparation of the database.

80% of the time spent in the preparation of a Machine Learning algorithm corresponds to the adjustment of each of the elements of the data in order to meet the requirements of the model to be executed. Being familiar with the data, knowing how to correct them in the appropriate way to the model's needs and having a prior expectation supported by reliable mathematical and statistical analysis will give the analyst and to the potential receiver of the information peace of mind.

Key Words:

Data processing, skimr, wvplots, hexbin, vtreat, dataexplorer, scales, training_set, test_set, glm, knn, random forest, xgboost, roc, auc.

Exploración de los Datos:

Partimos de la meta de este trabajo, construir un modelo para predecir quienes de los habitantes de una población cuentan con un seguro de salud o cuales clientes de una entidad financiera se encuentran en mora de pagos de un crédito suscrito con ellos

Para cualquiera de las dos predicciones el primer paso es recopilar información relacionada con la variable objetivo a predecir, la probabilidad de quienes de ellos tienen cobertura en salud o se encuentran en cesación de pagos. Los datos a recopilar pueden variar desde variables relacionadas con información personal hasta socioeconómica, la base de datos inicial requiere de un proceso adecuado de intervención antes de ser cargada a un algoritmo.

En el primer problema de clasificación, pronosticar si una persona cuenta si o no con seguro de salud se recurre al censo del 2016 correspondiente al *American Community Survey* (ACS) de los Estados Unidos. La ACS desempeña un papel fundamental en la recopilación de datos demográficos y socioeconómicos de las comunidades estadounidense y dicho censo sirve de base para la distribución de fondos, la planificación y toma de decisiones, la investigación y análisis y el monitoreo de políticas públicas acerca de la población. Las preguntas recopiladas son tales como edad, estado civil, ingreso, empleo, póliza de salud, número de vehículos, vivienda, etc. (United State Government, 2017)

La base de datos del censo 2016 USA es descargada desde el repositorio de la *Universidad de California campus Irvine* (CSU-Irvine). La CSU tiene trayectoria y reconocimiento en Inteligencia Artificial y cuenta con un repositorio creado en 1987 con aproximadamente 670 dataset sintéticos que son utilizados por la comunidad académica para el análisis empírico de algoritmos de aprendizaje automático y es usada como fuente primaria ya que permite comparar entre tipos de algoritmos (clasificación, regresión, clustering u otros), tipos de atributos (numéricos y/o categóricos), naturaleza de los datos (multivariados, series temporales, textuales, etc.) o aspectos relativos al dataset como tamaño, dimensionalidad o formato. Por claridad, el termino sintético se refiere a datos artificiales que se generan a partir de datos originales. Esto significa que los datos sintéticos y los originales arrojarían resultados similares cuando se someten al mismo análisis estadístico. (California State University – Irvine, 2007)

El segundo conjunto de datos a utilizar en este trabajo corresponde a un dataset previamente formateado para usarlo directamente en algoritmos de aprendizaje automático. El dataset se descargará directamente desde la plataforma *GitHub*, la cual se ha convertido en una herramienta esencial en el manejo de proyectos siendo uno de los portales colaborativos más populares en el ambiente de desarrolladores de software y recientemente de la enseñanza en el mundo informático. La base de datos seleccionada corresponde a una empresa financiera la cual quiere pronosticar si los créditos serán pagados o el cliente incumplirá el compromiso. (Stednick, s.f.)

R corresponde al lenguaje de programación que se usara para desarrollar el código correspondiente a la manipulación de los datos y luego el testeo en diferentes modelos estadísticos.

R fue diseñado para el análisis de datos, por lo cual rutinas comunes y necesarios ya están implementados, aunque con nombres y configuraciones extrañas. Una característica por la cual los profesionales de la ciencia de datos prefieren R en vez de otros softwares en la preparación y mantenimiento de una base de datos para usos posteriores corresponde a los siguientes ítems de acuerdo al Profesor de Ciencias de Pensilvania (USA) *Matt Dancho.* "Una comunidad más fuerte comparada con las demás, documentación disponible de altos estándares y abundante y las respuestas del *Stack Overflow* (Plataforma virtual de preguntas y respuesta para profesionales y aficionados a la ingeniería de software) más rápidas y precisas". (Dancho, 2023)

Técnicas de Predicción:

En casos a predecir como los especificados anteriormente (seguro de salud, default), la variable respuesta esperada es de tipo cualitativo lo cual invoca a variables categóricas. El proceso se denomina *clasificación* donde dicha observación esta asignada a una categoría o clase. (Gareth, 2021). Para los casos antes mencionado se podría utilizar alguna de las múltiples técnicas de clasificación ampliamente conocidas, algunas de ellas son: Naive Bayes, Análisis Discriminatorio Cuadrático, Análisis de Discriminante Lineal, Regresión Logística el cual sirve de base para desarrollar un modelo lineal generalizado. Otros métodos de clasificación computarizados y con avances más recientes se agrupan en la técnica de Aprendizaje Automático

Al final del escrito, luego de explicar diferentes técnicas para abordar situaciones específicas de los datos y mostrando caminos adecuados de solucionarlos, se ilustrará un problema de predicción pronosticando la probabilidad de no pago sobre un préstamo. El objetivo de usar esta base de datos, previamente formateada desde el origen, es mostrar que se puede predecir la probabilidad de problemas de clasificación de diferentes formas: Primero, partiendo de análisis de exploración básica de la variable objetivo (default), luego por medio de técnicas tradicionales usando específicamente regresión logística y por último por modelos específicos de aprendizaje automático.

Obtener y Examinar los Datos:

El primer paso a seguir, es poner las manos dentro de los datos y la meta es construir un modelo para conocer cuales individuos no tienen seguro médico ya sea que el interés parta por agencias gubernamental o por el departamento de algunas empresas de seguros, cargando el dataset *custdata* en R desde el repositorio de la *California State University* campus *Irvine*. (California State University – Irvine, 2007)

En la base de datos recolectada el estatus de la variable seguro de salud (*health_insurance*) de los participantes de la encuesta es conocido, ya que corresponde a una de las preguntas del cuestionario del censo. Adicionalmente, se incorporaron otros interrogantes que se consideraron como importantes a la hora de predecir la probabilidad de que un individuo

posea o no una póliza medica: Edad, estado laboral, ingresos, número de vehículos, propiedad raíz, etc.

Esta parte del proceso consume generalmente alto tiempo ya que se necesita explorar y adecuar los datos para que sean aptos en el análisis. ¿La primera pregunta a responder por el analista es cual tipo de datos están disponible?, son estos datos los adecuados para el modelo que se necesita implementar? ¿La base de datos disponible es suficiente o se requieren adicionar otras más? ¿Cuál es la calidad, confiabilidad y origen de los mismos?, Las medidas almacenadas en la base de datos responden directamente a el modelo analizar o se requiere inferir la variable?

Es importante para el programador responder estas y otras preguntas relacionadas con el modelo a implementar y en especial teniendo en cuenta que tipo de materia prima se va a usar como insumo inicial.

En esta etapa es donde inicialmente se explora, visualiza y se analiza la data. Se definen las decisiones de limpieza, reparación de errores y las transformaciones requeridas en la base de datos.

Los datos se pueden clasificar de la siguiente manera respondiendo al objetivo del modelo y cada uno de ellos tienen diferentes vías para ser abordado: (CFA Institute, 2022)

- Clasificación: Decidir si pertenece o no a una condición.
- Scoring: Estimar o predecir un valor numérico: Probabilidad, Precio, Score, etc.
- Ranking: Ordenar los elementos conforme a ciertas características
- Clustering: Agrupar acorde a una similaridad definida
- Relaciones: Encontrar relaciones inmediatas o potenciales entre los elementos
- Caracterización: Generar informe y graficas desde los datos

Una parte fundamental para un analista de datos, es fijar y delimitar claramente las expectativas del modelo. Puede suceder que una vez establecido el prototipo quienes lo ejecuten anulen las decisiones del mismo ya que el resultado es no acorde a lo esperado. En ese caso cabria la pregunta: ¿El modelo es correcto para esa decisión? o tal vez la intuición del oficial ejecutor es equivocada. Por tal motivo, el conocer profundamente la base de datos y tener expectativas sustentadas en un campo real es asertivo tanto para quien genera el modelo como para quien espera ver unos resultados según la expectativa prevista. Es fundamental clarificar las métricas a usar y definir las fortalezas y debilidades de ellas en evaluar el desempeño de la estimación. (Scheule, 2022)

Una práctica recomendable en la interacción con los datos consiste en examinarlos después de cargarlos en el R, con el propósito de entender el tipo de información almacenada y tener claro cuales procedimientos a posteriori se van a ejecutar, a continuación, se ilustran algunos comandos útiles en la identificación de la base de datos a tratar. (Kabacoff, 2015)

Figura 1: **Comandos Examinación de Datos**

Comando	Descripción
View ()	Muestra los datos tipo hoja de calculo
class ()	Indica que tipo de objeto es para R
dim ()	Numero de filas y columnas para datos tipo data-frame
head ()	Muestra en la línea superior el encabezamiento de los datos
help ()	Proporciona la documentación para una clase de dato
str ()	Indica la estructura de un objeto
summary ()	Resumen de cualquier objeto en R
print ()	Imprime todos los datos

El ejecutar un análisis previo usando estas instrucciones ayuda a identificar los datos y evita el gasto de tiempo innecesario. Crear tablas u otros mecanismos de salida es un paso fundamental para entender la información. Ya que los datos no siempre están formateados y por lo tanto necesitan ser transformados para que cumplan con los requerimientos específicos y poder usarlo en algún algoritmo determinado.

La exploración de los datos combina tanto resúmenes estadísticos (medias, medianas, varianzas, contar elementos) como visualizaciones de graficas. Los problemas de los datos se pueden identificar con los resúmenes, pero otros se detallan más claramente en la parte grafica. Por lo tanto, el manejo de ambas herramientas es prioritario en los análisis previos del dataset antes de iniciar la modelación.

La instrucción *summary ()* en R, permite el primer acercamiento a los datos y es útil para detectar problemas o inconsistencias de los mismos. Adicionalmente suministra métricas estadísticas las cuales permiten entender el alcance de cada una de las variables.

La *figura 2a* muestra en la primera línea que el archivo de datos que se están usando es de tipo *RDS* (Ray Dream Studio) asociado con formatos de imagen 3D y corresponden a archivos de tamaños grandes. En La segunda línea del resumen se visualiza que la base de datos ha sido renombrada (*clientes_data*) con el fin de resguardar la original. Se aprecia claramente que los resultados de ejecutar el comando son inicializados con el símbolo numeral doble (##).

Figura 2a: **summary () Clientes Póliza de Salud – USA 2016**

```
clientes_data = readRDS("custdata.RDS")
summary(clientes_data)
##      custid            sex           is_employed       income
##  Length:73262       Female :37837   FALSE  : 2351   Min    : -6900
##  Class:  character  Male   :35425   TRUE   :45137   1st Qu : 10700
##                                     NA's   :25774   Median : 26200
##                                                     Mean   :41764
##                                                     3rd Qu :51700
##                                                     Max    :1257000
```

En la figura 2a, *custid* corresponde a una identificación para cada uno de los clientes la cual por lo general es única. Mientras la variable *sex* indica solo dos opciones para el género (femenino/masculino) indicando la cantidad en cada uno de ellos. El primer punto a considerar corresponde a la característica si la persona en el momento de ser encuestado hacia parte del mercado laboral (*is_employed*), el resultado indica tres opciones: desempleado, empleado o *no disponible* (*NA's*: not avalaible). Los *NA's* o valores no determinados corresponde a un 33% (*25.774*) de los clientes. Lo cual podría significar que las personas no están activas en el mercado laboral, o que la información sobre el status de empleo se empezó a recolectar recientemente y los registros anteriores no los clasificaron. También podrían corresponder a estudiantes o personas que aun conviven con los padres o amas de casas o pensionados a los cuales no se tenía claro como clasificarlo por parte de los encuestadores. Es importante definir cada término del dataset en un archivo adicional que sirva a la interpretación clara y precisa de la información recolectada.

En general este tipo de entrada, *NA's*, se reconocen como valores perdidos (*missing values*). Una consideración adicional sobre los *NA's* en una variable, si la proporción no es significativa sobre el total de entradas o sobre variables no importantes en la estimación no sería un gran problema para la estructura general de la data y se podría optar por eliminar sin mucho análisis, pero en el caso de esta columna (*is_employed*) seria desechar un tercio de los datos lo cual afectaría la veracidad de los resultados. Tener en cuenta, algunos algoritmos por defecto eliminan las filas con elementos omitidos. El analista en estos casos necesita determinar qué acción tomar, si decide eliminarlas o reemplazarlas a cero o a una categoría adicional. Un punto a valorar en la decisión es el peso de la variable dentro de los datos, variables relacionadas con los empleados dentro de un algoritmo referente al cubrimiento de salud tiene más peso que columnas tales como número de vehículos, tipo de casa, dirección, hobbies, estado civil, etc.

La columna de ingreso (*income*) contiene entradas con valores negativos (*Min: -6900*) lo cual hace que dichos valores sean potencialmente considerados como inválidos y/o atípicos (*outliers values*). Es necesario analizar la información de todas las columnas o al menos de las más relevantes y chequear que los valores tengan sentido. Además, preguntarse si tiene valores atípicos o el rango si es el adecuado para la variable y en general para el modelo a desarrollar. El rango de la variable *income* se debe analizar ya que inicia (mínimo) en

valores negativos y termina con valores mayores (máximo) al millón de dólares lo cual indica una distancia demasiada amplia acarreando problemas para algunos métodos de modelación.

Figura 2b: **summary () Clientes Póliza de Salud – USA 2016**

```
##   marital_status                       health_ins
##   Divorced/Separated     :10693        Mode        :logical
##   Married          :38400              FALSE       :7307
##   Never married          :19407        TRUE        :65955
##   Widowed                :4762
##                          housing_type        recent_move         num_vehicles
##Homeowner free and clear  :16763        Mode        :logical      Min     :0.000
##Homeowner with mortgage/loan :31387     FALSE       :62418        1st Qu  :1.000
##Occupied with no rent     :1138         TRUE        :9123         Median  :2.000
##Rented                    :22254        NA's        :1721         Mean    :2.066
##NA's                      :1720                                   3rd Qu  :3.000
##                                                                  Max.  :6.000
##                                                                  NA's    :1720
##   age                    state_of_res              gas_usage
##   Min     :0.00          California    :8962       Min     :1.00
##   1st Qu  :34.00         Texas         :6026       1st Qu  : 3.00
##   Median  :48.00         Florida       :4979       Median  :10.00
##   Mean    :49.16         New York      :4431       Mean    :41.17
##   3rd Qu  :62.00         Pennsylvania  :2997       3rd Qu  :60.00
##   Max     :120.00        Illinois      :2925       Max     :570.00
##                          (Other)       :42942      NA's    :1720
```

El comando summary del R implícitamente indica el tipo de variable, en la *figura 2b* se visualiza variables binarias (falso/verdadero) tales como *health_ins* y *recent_move* reportando solo las estadísticas de conteo de cada uno de los estados. Entre las numéricas, las cuales reportan varias métricas estadísticas se encuentran *num_vehicles, age* y *gas_usage* Entre las variables categóricas se identifican *marital_status, housing_type y state_of_res.* La métrica para este tipo de variable solo corresponde al conteo de cada categoría.

La variable *health_ins* indica que el 90% de las personas encuestadas cuentan con el respaldo de una póliza de salud (*TRUE: 65.955*). Otras situaciones a tener en cuenta en esta resumen corresponden a las entradas no disponibles (*NA's*) para las variables *housing_type*, *recent_move*, *num_vehicles*, y *gas_usage* (*1720* ó *1721*), ya que la proporción respecto al total de registros (73.262) es pequeña (2.3%) podría considerarse eliminarlos especialmente si los valores perdidos corresponden a las mismas filas.

La variable edad (*age*) de las personas entrevistadas la media tiene un valor razonable (*Mean: 49.16*) pero el mínimo (*Min.: 0.00*) y el máximo (*Max.: 120.00*) son poco probables. El analista debería tener en cuenta si el error con la edad mínima podría corresponder a problemas de digitalización o un valor centinela usado para entradas desconocidas y tomar la acción apropiada de acuerdo a un criterio técnico de eliminar la fila o imputarlo por un valor útil y para el máximo decidir qué tan real puede ser.

Visualización para Identificar problemas de los Datos:

Analíticamente usando la instrucción *summary ()* del R se detectaron problemas en la data, pero para percibir otros conflictos del dataset los gráficos son más apropiados que la parte de texto. Es recomendable usar ambas propiedades, ya que de esta forma se amplía el espectro de inconvenientes en los datos y se puede decidir bien informado sobre las acciones a tomar posteriormente para solucionar dichos problemas antes de la modelación.

En la ciencia de datos se llama *Visualización* al uso de graficar para examinar la data y uno de los más reconocidos en este campo es el profesor norteamericano, de estadística y ciencia de datos, de la Universidad de Purdue (Indiana) *William Swain Cleveland* (1943, -) quien desarrollo los *Principios para la Visualización Científica*. (Swain, 1986)

El paquete principal grafico de R corresponde al *ggplot2* (Chang, s.f.). Adicionalmente, se pueden mencionar *WVPlots*, *ggpubr*, *ggstatsplot* y *lattice*. Con todos ellos se pueden generar graficas más o menos similares. A continuación, se graficarán varias de las variables analizadas en forma analítica en la sección anterior usando diferentes tipos de representaciones buscando encontrar la más adecuada y descriptiva de la situación que se desea resaltar.

Figura 3: **Densidad de la edad**

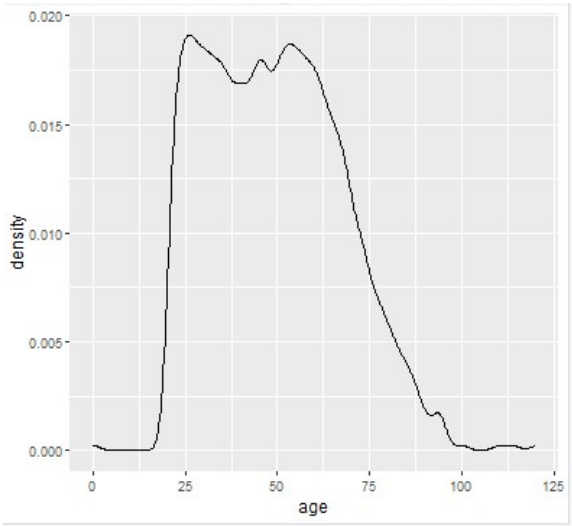


Figura 4: **Histograma del uso del gas**

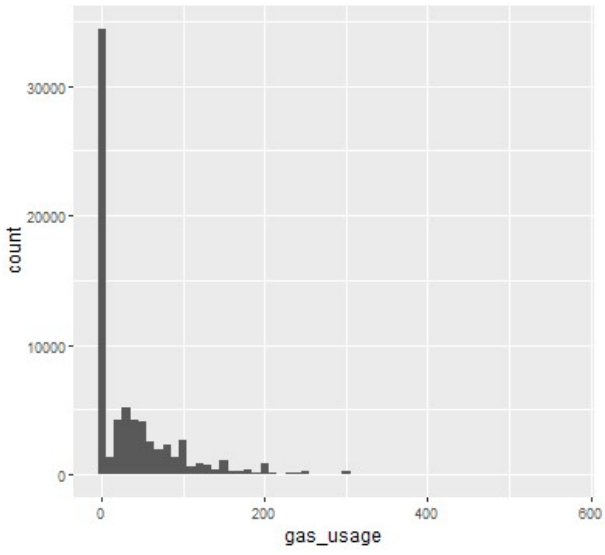


Figura 5: **Histograma Cleveland lugar de residencia**

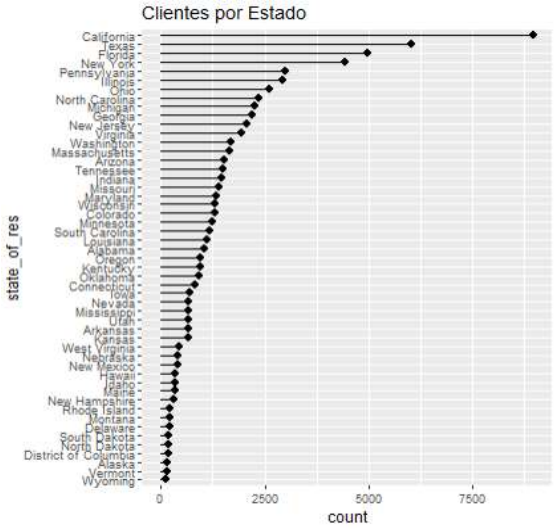


Figura 6: **Código Visualización Figuras 3, 4 y 5**

```
# Librerías requeridas para la visualización
library(ggplot2)
library(scales)
library (WVPlots)

# Grafico de densidad para la edad de los usuarios (age)
ggplot (clientes_data, aes(x=age)) +geom_density ()


# Histograma uso del gas (gas_usage)
ggplot (clientes_data, aes(x=gas_usage)) + geom_histogram (binwidth = 10)


# Grafico de barras horizontales de los estados (state_of_res)
# organizado de mayor a menor (sort=1)
ClevelandDotPlot (clientes_data,"state_of_res", sort = 1,
title = "Clientes por Estado") +coord_flip ()


ggplot (clientes_data, aes(x=income)) +
        geom_density () +scale_x_continuous (labels = dollar)
```

La figura 3 muestra la distribución de la edad de los clientes, el rango de la variable se puede determinar entre 25 a 65 años que corresponde a la altura máxima del gráfico. De la figura 3 se pueden hacer las siguientes consideraciones: Los valores antes de los 20 y después de los 100 años se podrían considerar como atípicos y esta consideración es más simple realizarla visualmente que con el resumen. En las edades, aproximadamente, de 25, 50 y 60 se muestran ciertos picos y el decrecimiento inicia alrededor de los 65. Adicionalmente alrededor de los 90 se encuentra un resalto el cual se podría considerar como un problema en la entrada de datos.

A continuación, se describirán ciertas observaciones del histograma (*figura 4*) de la variable del consumo del gas (*gas_usage*). Cuando se gestionan datos se necesita conocer el origen de los mismos y las condiciones donde las muestras se recolectan, si la base de datos incluye diccionario el hacer uso del mismo es el punto inicial antes de cualquier intervención en el dataset. El uso del gas es un buen ejemplo para tener claro de dónde vienen y cuando se recolectan los datos, esta variable tiene diferente referencia si fue cosechada en invierno para las zonas árticas o antárticas o si los datos vienen de zonas ecuatoriales o si el gas es subsidiado para cierto tipo de población o en cierta parte del año.

En los histogramas es primordial definir el tamaño del contenedor (*bin*) que sea una medida acorde a la información contenida en la variable (*binwidth = 10*). La *figura 4* muestra donde se concentran los datos y la presencia de posibles datos atípicos. De acuerdo a la gráfica la mayor concentración de usuarios es aquellos quienes pagan entre 0 – 10US por mes de factura en gas lo cual podría significar que esos clientes no usan calentadores a gas o tiene subsidios lo cual necesariamente requiere verificación en el diccionario de la base de datos.

Otro punto adicional corresponde a los clientes con altas facturación al mes (>200US) mucho más altas que la típica de los usuarios, por lo cual sería una opción dejarlos afuera del análisis.

Una gráfica adicional que es importante tener en cuenta en los análisis corresponde a las barras de forma horizontal y ordenados (mayor a menor en este caso) para variables categóricas con muchos niveles y valores disimiles entre el máximo y mínimo (*figura 5*) los cuales son más legibles que el grafico de barras verticales. En este caso, se usará la aplicación se *William Cleveland* (*library (WVPlots))* desarrollada en *The Elements Of Graphing Data* con la variable correspondientes a los estados de la USA (*state_of_res*) donde indica el número de usuarios de la póliza de salud.

La *figura 6*, enlista el código generado en R para visualizar las variables de edad, uso del gas y el estado de residencia de los usuarios de póliza de salud.

Las *figuras 7* y *8* corresponde a la representación de la misma variable de ingreso (*income*) con diferentes escalas: continua y logarítmica. Si la diferencia entre los valores es muy grande o el rango es muy amplio, la distribución se concentra fuertemente en uno de los lados lo cual dificulta distinguir características de la gráfica. En estos casos, la escala recomendada en la visualización es una logarítmica, la cual usa el logaritmo del intervalo del ingreso, en lugar del propio valor de la variable en ese rango lo cual reduce los valores a una gama mucho más manejable. El código correspondiente a la implementación de la escala logarítmica se puede observar en la *figura 10*.

En la *figura 8* correspondiente a los ingresos con escala $\log_{10}$ hay varias partes de la gráfica a analizar:

- Los valores con una densidad muy cercana a 0.00 se pueden considerar atípicos (ingresos alrededor de 100 US.).
- La mediana indica que la mayoría de las personas tienen un ingreso alrededor de los 10.000 US.
- El rango más amplio en la curva de distribución se ubica entre los 10.000 y 100.000 lo cual indica que la mayoría de los clientes se encuentran en esos niveles de ingreso indicado con el símbolo de doble flecha en la gráfica.
- El punto máximo de ingreso es aproximadamente unos 40.000 y hay un sobresalto que se podría atribuir a la digitación en los alrededores de 500.000 el cual se omitiría o reemplazaría.

Figura 7**: Income – Escala Continua**

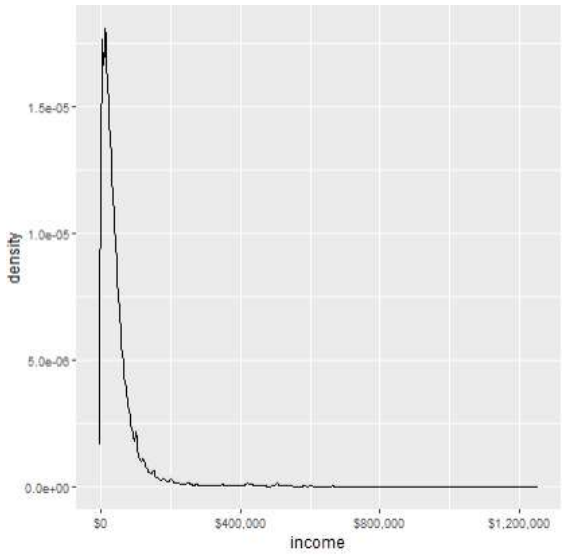


Figura 8**: Income – Escala Log**

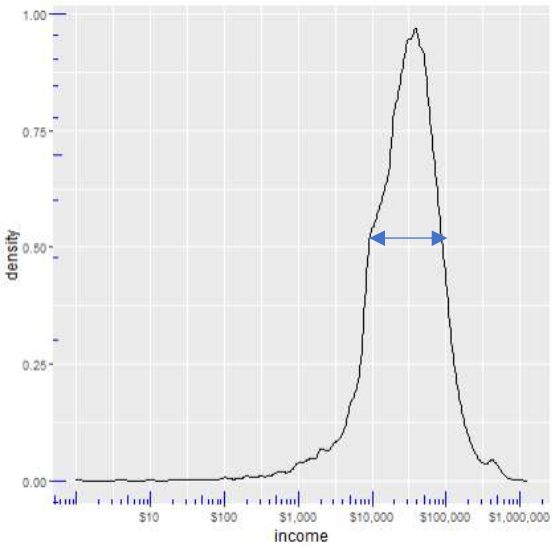


Figura 9: **Correlación: Ingreso – Edad**

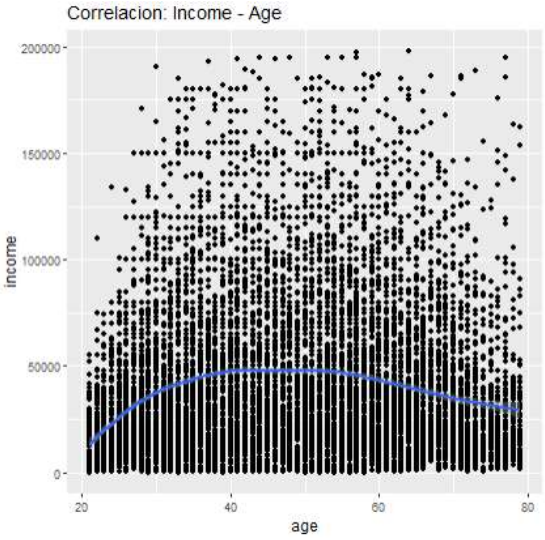


Figura 10: **Código Figuras 7, 8, 9**

```
# Densidad de Ingreso (income) - Escala Continua
ggplot (clientes_data, aes(x=income)) +geom_density () +
scale_x_continuous (labels = dollar)

# Densidad de Ingreso (income) - Escala Logarítmica base 10
ggplot (clientes_data, aes(x=income)) +geom_density () +
scale_x_log10(breaks=c (10,100,1000,10000,100000,1000000), labels = dollar) +
annotation_logticks (color = "blue")

# Correlación: Income - Age
# Submuestra (subset) de clientes_data
clientes_data2 <- subset (clientes_data, 20<age & age<80 & 0<income & income<200000)
cor (clientes_data2$age, clientes_data2$income)

# Grafico de Correlación
clientes_data_muestra <- dplyr::sample_frac (clientes_data2, size = 0.2, replace = FALSE)
ggplot (clientes_data_muestra, aes (x=age, y=income)) + geom_point () +
ggtitle ('Correlación: Income - Age') +geom_smooth ()
```

La *figura 9* corresponde a la correlación entre las variables edad e ingreso para un rango adecuado y una muestra del 20% del tamaño total. El resultado de la correlación es positiva cercana al 0.6 significando que existe una relación entre la edad y el ingreso como seria presuntivamente asumido pero la idea grafica da un concepto concreto que el valor numérico. Adicionalmente se agrega una curva suave la cual muestra que el ingreso tiende a incrementarse con la edad de las personas. Los códigos para las gráficas 7, 8 y 9 se encuentran en la figura 10.

La *figura 11* corresponde a una dispersión hexagonal (*Package Hexbin*) la cual muestra la relación entre dos variables (*age*; *income*) muy útil para datos densos evitando superposición, los cuales son divididos dentro de contenedores y la cantidad de datos contenidos en cada uno de ellos se representa con un color y una tonalidad respectiva.

La *figura 12*, grafica el cubrimiento de la póliza de salud como una función de la edad, mostrando que la probabilidad de tener aseguramiento medico en los Estados Unidos se incrementa con la edad. Para usar este grafico de dispersión en R, la variable lógica (False, True) necesita ser convertida a numérica (0/1) y por medio de la curva se visualiza la relación entre la variable booleana (*health_ins*) y la continua (age).

La variable *health_ins* toma el valor de uno (*1*) cuando el individuo posee póliza y cero (*0*) en caso contrario. El *eje y* contiene valores entre 0 a 1 para la variable respectiva dando la impresión de no ser informativa, pero la curva muestra el valor promedio de los 0/1 de las personas que cuentan con póliza de salud en función de la edad. Por ejemplo, a los 20 años el 80% aproximadamente cuentan con póliza de salud y a la edad de los 50 alrededor del 90% ya están asegurados.

Siguiendo con otro ejemplo para visualizar dos variables, se usa la variable del estatus marital (*marital_status*) y la póliza de salud. Por medio de barras verticales se cuenta la cantidad de personas aseguradas en salud de acuerdo al estado civil reportado. La *figura 13*, barras lado a lado, con la cual se puede comparar fácilmente el número de personas con y sin seguro según el estado marital. Este tipo de grafico cuenta entre las debilidades que no se puede saber el total de individuos en cada categoría.

La *figura 14*, muestra el código para generar las tres graficas anteriores. Tenga en cuenta instalar los paquetes (*install.packages ()*) *ggplot, WVPlots* y *Hexbin* y cargar las librerías (*library ()*) correspondientes. Tomar tiempo para examinar los datos antes de sumergirse en la modelación es un punto importante, ya que le suministra una idea clara de la relación entre las variables y la visualización óptima para tal propósito. En caso de que una gráfica no llene las expectativas, es recomendable explorar otras alternativas y compararlas.

Las aplicaciones graficas tienen el margen de permitir personalizar la gráfica a la necesidad del usuario. Tareas como delimitar los rangos, potenciar valores, definir el tamaño de la muestra, configurar los parámetros de color, grilla, ejes y elementos de visualización, agregar curvas adicionales con el fin de resaltar alguna relación especifica, agregar anotaciones dentro del grafico con el fin de documentarlo. La única forma de saber cuál es el grafico correcto en la presentación es explorarlo o realizar búsquedas en páginas como *GitHub* o *Kaggle*. Otra ayuda importante para tener en cuenta corresponde a *Stack Overflow*.

Stack Overflow es una fuente de respuesta a preguntas sobre *ggplot2*. Igualmente se puede solicitar ayuda por medio del foro, luego de haber generado un ejemplo que ilustre el problema al cual se está enfrentando y que otros usuarios lo puedan reproducir.

Figura 11: **Dispersión Hexagonal: Ingreso - Edad**

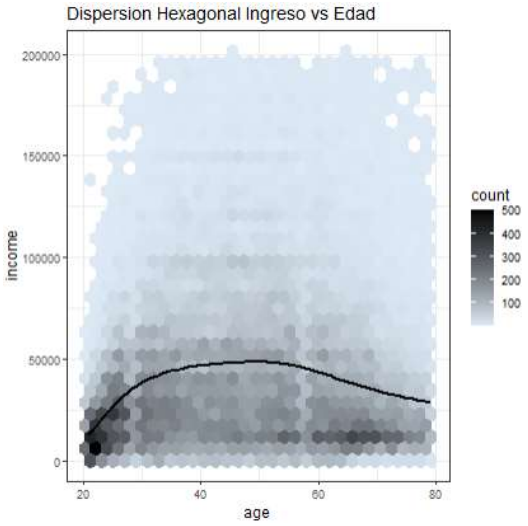


Figura 12: **Probabilidad: Póliza de Salud -Edad**

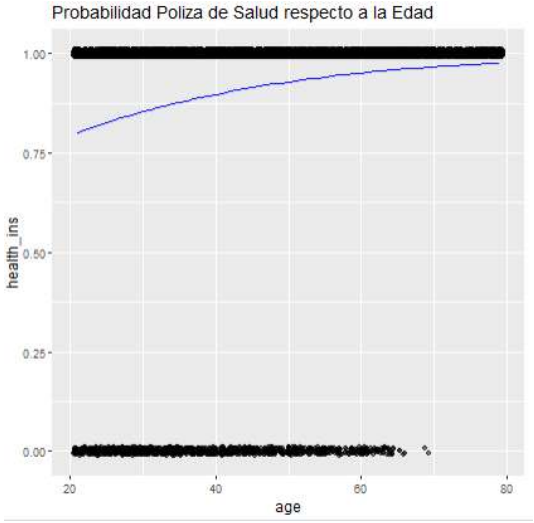


Figura 13: **Probabilidad**

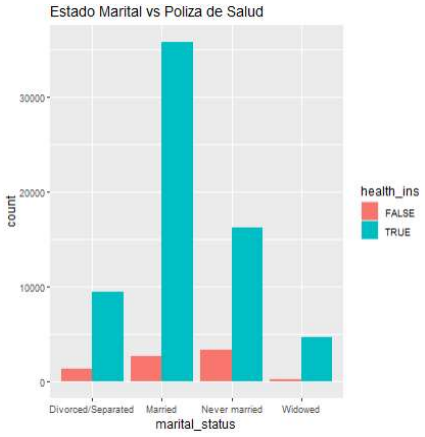


Figura 14: **Código Figuras 11, 12, 13**

```
# Dispersión Hexagonal de Ingreso vs Age
HexBinPlot (clientes_data2,"age","income", "Dispersión Hexagonal Ingreso vs Edad")
+geom_smooth (color="black", se=FALSE)

# Probabilidad de Póliza de Salud por Edad
BinaryYScatterPlot (clientes_data_muestra,"age","health_ins", title = "Probabilidad Poliza
de Salud respecto a la Edad")

# Barras Lado a Lado Estado Marital vs Poliza de Salud
ggplot (clientes_data, aes (x=marital_status, fill=health_ins)) +geom_bar (position =
"dodge") +ggtitle ('Estado Marital vs Poliza de Salud')
```

Mejorar Calidad de los Datos:

Antes de iniciar un modelaje es necesario adecuar los datos en una forma estadísticamente valida. Por lo tanto, después de la exploración y descubrir diferentes anomalías relacionados con los datos o la interacción entre ellos es necesario tomar ciertas medidas teniendo en cuenta el origen y la finalidad de los mismos como se analizará a continuación. La base de datos a usar como ya se mencionó previamente se descargó desde el repositorio de la Universidad de California en la parte correspondiente a Machine Learning y en este caso se usará el dataset denominada *custdata.rds*. (California State University – Irvine, 2007)

El primer paso a tratar es el manejo adecuado de los valores ausentes (*missing values*) los cuales aparecen en la data muy comúnmente y los procedimientos para tratarlos básicamente son intervenir y convertirlos en algún valor significativo o eliminar las filas donde se encuentran. La decisión necesita pasar por un criterio valido ya que ambos procedimientos afectan en general la base de datos.

La *figura 15* suministra la dimensión del dataset por medio de la instrucción *dim ()* con la cual se viene trabajando, indicando que contiene aproximadamente 73.200 clientes y doce (12) variables y se usara de base para comparar con los *NA´s* presente en cada columna. El paquete a usar corresponde a *Skimr ()*, el cual suministra las estadísticas de resumen que el usuario puede hojear rápidamente para entender los datos. La versión actual del paquete se puede descargar directamente desde el CRAN. (CRAN, 2022).

La figura 16, muestra el resultado de usar la función *skimr ()* indicando el número de missing para cada variable. Una característica importante de esta librería corresponde a que se puede modificar fácilmente y usar *pipeline* (*%>%*).

Figura 15: **Dimensión Base de Datos**

```
> # Determinar Variables con missing values

> dim(t(clientes_data))
      Rows  Columns
[1]  73262       12
```

Figura 16: **Variable con NA´s**

```
> skim(clientes_data) %>%
+ dplyr :: select (skim_variable, n_missing)
# A tibble: 12 × 2
  skim_variable  n_missing
  <chr>          <int>
1 housing_type   1720
2 is_employed    25774
3 recent_move    1721
4 num_vehicles   1720
5 gas_usage      1720
```

Al analizar la *figura 16*, los *NA´s* de las variables *housing_type*, *recent_move*, *num_vehicles* y *gas_usage* corresponden a un 2.3% del total de clientes, pero los *NA´s* de *is_employed* son aproximadamente un 35% lo cual indica una afectación diferente en la muestra al intervenir esas variables en mención. El manejo de esta situación específica depende de los analistas quienes deben decidir cuales valores son inválidos y que procedimiento hacer con ellos, en este caso se mostraran diferentes alternativas.

Ya que la variable *is_employed* tiene cerca de un tercio del total de los clientes como entrada ausente, es pertinente preguntarse: ¿cuál decisión tomar en este caso? Eliminarla causaría una reducción significativa en el número de registro totales y al no tener claro la razón por la cual ellos aparecen en la base de datos sería más adecuada conservarlos y crear una nueva categoría que los identifique en el dataset.

La solución sería crear una nueva categoría y denominar a los *NA´s* como missing, ya que es el nombre con el cual se reconocen dichas entradas. Pero los analistas deberían profundizar cual es la razón de la alta proporción de valores ausentes en la data, identificarlas ayudaría a un mejoramiento intrínseco en la misma y las razones de que se

presente podrían ser múltiples desde un mal registro de datos hasta problemas de semántica pasando por personas no activas como estudiantes, amas de casa, retirados, personas que buscan no aparecer como empleados, etc. En este caso, a los valores missing se denominarán *no activos* en la fuerza laboral sin ninguna discriminación de si son estudiantes, retirados, etc. Un punto a tener en cuenta en la programación, cuando se generan variables categóricas, como en este caso, partiendo de tipo factor el tipo de variable para la nueva categórica es *string*, por lo tanto, se requiere convertir nuevamente a factor para poder usar funciones tales como *summary ()*.

La *figura 17*, muestra el código para crear la nueva entrada para la variable *is_employed* llamándola *no activo* para las entradas NA´s.

Figura 17: **Reasignación – is_employed**

```
> nueva_base1 <- clientes_data
> nueva_base1$is_employed_fix <- ifelse(is.na(nueva_base1$is_employed
                "no activo", ifelse (nueva_base1$is_employed==TRUE,
                "empleado", "desempleado"))
```

Figura 18: **Resumen – is_employed_fix**

```
> summary (as. factor(nueva_base1$is_employed_fix))
Desempleado        empleado       no activo
2351               45137          25774
```

El crear nueva base de datos y variable incluida (*figura 17*) corresponde a una decisión de la persona a cargo de generar el código. Entre los pros se encuentra conservar intacta la base de datos original y en el caso de una decisión diferente o deshacer los cambios por alguna consideración se puede regresar al inicio de la transformación sin mayores contratiempos. En los contras se tiene que al tener dos bases de datos y dos variables casi idénticas en el nombre de la columna deja la puerta abierta a usarla indebidamente por equivocación.

La *figura 18* describe el resumen de la nueva variable categórica donde se definen tres niveles para la variable: si el cliente está o no empleado en el momento de originarse los datos y la nueva etiqueta de *no activo* correspondiendo a los *NA´s* originales de la base de datos.

Analicemos el caso de la variable *income,* donde el mínimo ingreso para ciertos clientes es negativo como se analizó previamente y el cual se puede considerar fuera del rango e invalido. Por lo tanto, valores negativos de la variable ingresos se reemplazarán por *NA´s* y quienes tenga valor de cero (0) se conservará ya que podrían ser estudiantes, amas de casa, etc.

El siguiente caso corresponde a la edad variable *age*, donde el valor mínimo es cero (0) y el máximo 120. La entrada del mínimo se considerará nula y se reemplazara por *NA´s* y el máximo se mantendrá, aunque sería poco probable, pero al no tener una evidencia de la razón de ese valor es preferible ser conservativo.

La *figura 19* muestra el código usado para transformar las entradas que se clasificaron invalidas por NA´s usando la función *mutate ()* y el condicional *ifelse ()*, cabe recordar que

para hacer uso de esta función se requiere el paquete *dplyr ()* permitiendo utilizar pipeline (*%>%*). Adicionalmente, enlista el código usado para cambiar los mínimos de las variables ingreso y edad como se muestra en el resumen de cada una de ellas.

Mientras tanto, la *figura 20* indica la aparición de los NA´s usando la función *skim ()* para dichas variables debido al reemplazo de entradas consideradas nulas por valores no disponibles.

Figura 19: **Código -Invalido Valores**

```
# Definir valores inválidos (NAs) de edad (age) e ingreso (income)
clientes_data <- clientes_data %>% mutate (age = na_if(age,0),
                 income=ifelse (income<0, NA, income))

> summary(clientes_data$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  21.00   34.00   48.00   49.22   62.00   120.00      77

> summary(clientes_data$income)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
      0   10700   26300   41793   51800 1257000      45
```

Figura 20: **Resumen de Age e Income**

```
> skim(clientes_data) %>%
+ dplyr :: select (skim_variable, n_missing)
# A tibble: 12 × 2
  skim_variable n_missing
  <chr>             <int>
9 income               45
11 age                 77
```

Los datos no validos pueden considerarse como *random* (aleatorios) o *sistemáticos*. Los datos aleatorios se consideran independientes de la situación o de otros datos y se presentan por fallos en la recolección de los mismos. En este caso, se pueden reemplazar o imputar por un estimado razonable ya sea la media o la mediana. La *imputación* consiste en reemplazar el valor omitido por la media o mediana de la columna dependiendo de la decisión del analista, si dentro de la variable a imputar existen entradas consideradas como atípicas (*outliers*) es recomendable usar la mediana. Ya que, outliers tienen mayor efecto sobre la media. En el caso de imputar alguna entrada es recomendable agregar algún indicador adicional o crear una nueva variable para mantener la pista de cuales puntos fueron alterados. La *figura 21* muestra la imputación de los NA's en la variable income usando una media de US41.793 para un total de 45 entradas modificadas.

Figura 21: **NA´s – Aleatorios**

```
nueva_base2 <- clientes_data
summary(nueva_base2$income)
Min   1st Qu    Median    Mean    3rd Qu     Max        NA's
0     10700     26300     41793   51800    1257000    45

# media de income removiendo los NA, de lo contrario mean=NA
media_income <- mean (nueva_base2$income, na.rm = TRUE)
# Nueva variable income.fix
nueva_base2$income.fix <-ifelse (is.na (nueva_base2$income),
                                 media_income, nueva_base2$income)
summary(nueva_base2$income.fix)
Min   1st Qu    Median    Mean    3rd Qu     Max
0     10700     26300     41793   51700    1257000
```

Figura 22: **NA´s – Sistemáticos**

```
nueva_base3 <- clientes_data

# Seleccionar cortes de la variable income según conveniencia
cortes <- c (0,10000,50000,100000,250000,1500000)
>
# Nueva variable income. grupos
nueva_base3$income.grupos <- cut (nueva_base3$income,
                    breaks = cortes, include. lowest = TRUE)

summary(nueva_base3$income.grupos)
[0,1e+04]  (1e+04,5e+04]  (5e+04,1e+05]  (1e+05,2.5e+05]  (2.5e+05,1.5e+06]
17738          36725          13117           4671                966
NA's
45
```

Reemplazar los valores ausentes con la media o con otro valor determinado previamente asume que los clientes con una entrada en el ingreso (*income*) de *NA´s* corresponde a una situación aleatoria en la recolección de datos. Pero es posible que esas entradas reflejen

una condición especial de los clientes y sean sistemáticas, podrían corresponder a personas que no tienen ingresos ya que no están activos en el mercado laboral. Por lo tanto, imputar el ingreso de ellos con un determinado valor seria equivocado y lo adecuado sería implementar otra alternativa para dichas entradas.

Es fundamental tener claridad si los NA´s son aleatorios o sistemáticos y esta clasificación determinara el modelo a seguir para solucionarlos. En el caso de la variable ingreso imputar los missing con algún valor lideraría a conclusiones equivocadas en el modelo. Por lo tanto, en la posibilidad de no saber con claridad si el origen de los datos no validos es aleatorio o sistemático es recomendable tomar la segunda opción (*sistemático*) en vez de sustituir los valores NA´s de las variables tomando suposiciones erróneas que conllevaran a resultados falsos.

La *figura 22* muestra el código a utilizar, el cual consiste en generar una nueva variable (*income.grupos*) y crear varios subgrupos dependiendo del ingreso. Luego, por medio de la instrucción resumen (*summary ()*) se analiza el comportamiento de los *NA´s* en cada una de las subdivisiones de la variable. Se observa claramente que los valores ausentes se encuentran concentrados en el primer rango entre cero a 10.000 (0 – 10.000) dólares como ingreso. Por lo tanto, el haberlos sustituidos por la media o mediana de la variable hubiese sido un error, en este caso se reemplazarán por el valor cero (0).

Una buena práctica es mantener registro de los cambios que se realicen a la base de datos. En este caso, se creará una nueva variable (*income.NA´s*) para diferenciar los ceros (*0*) originales de la columna ingresos y los nuevos ceros (*0*) provenientes del reemplazo de los NA´s, como se muestra en la figura 23.

Figura 23a: **NA´s – Diferenciar Ceros**

```
# Diferenciar Ceros entre los originales de la DB y los reemplazo de las NAs
> nueva_base3$income.NAs <- is.na(nueva_base3$income)
> summary(nueva_base3$income.NAs)
Mode        FALSE       TRUE
Logical     73217       45
```

Figura 23b: **NA´s – Sustituir por Ceros**

```
> # Reemplazar NAs sistemáticos por cero (0)
> nueva_base3$income.fix <- ifelse(is.na(nueva_base3$income),
                                   0, nueva_base3$income)
> summary(nueva_base3$income.fix)
Min     1st Qu      Median      Mean      3rd Qu      Max
0       10700       26200       41767     51700       1257000
```

Tratamiento Automático de los NA´s:

Después de un análisis detallado de las variables consideradas como predictores de probabilidad, *i.e. income*, para un aseguramiento en salud y darles un tratamiento adecuado ya sea modificando los *NA´s* o conservándolos, se puede hacer uso de uno de los paquetes de *R* para tratamiento automático de datos.

La *PhD* española en Ciencia de Datos *Rosana Ferrero* (@RosanaFerrero) recomienda usar el paquete *vtreat* para la manipulación del dataset, ya que los parámetros de la función son asequibles y se puede volver a repetir el proceso cuantas veces se requiera. (Mount, 2021)

El *vtreat* acondiciona la base de datos para modelos de *Machine Learning* (ML) o M*odelos Predictivos* de una manera estadísticamente sólida. En forma concisa, el paquete transforma todas las variables explicativas a numéricas sin valores perdidos (*missing*) tomando los datos desordenados del mundo real y preparándolos de forma fiable, repetible para los modelos ML. El incorporar *vtreat* en el modelo de estimación permite trabajar a mayor velocidad con datos estructurados diversos ya que las sustituciones son documentadas mediante una columna indicadora, a la cual denomina con el nombre de la variable y el prefijo *isBAD* (*variable_isBAD*) donde una entrada con un valor igual a *uno* (1) indica el reemplazo de un valor en esa fila.

La *figura 24* muestra el proceso de crear y aplicar la función, en el cual se genera un plan base donde se guarda toda la información necesaria para repetir el proceso de tratamiento de datos. El primer paso es definir cuales columnas de la data son las variables de entrada, en este caso todas excepto *health_ins* la cual corresponde al resultado a predecir y *custid* correspondiente a la identificación de cada cliente. La función *setdiff (x, y)* cuyo resultado corresponde a los elementos de "*x*" pero no los de "*y*" es la encargada de seleccionar las columnas que servirán de origen para el tratamiento base de eliminación o sustitución de los *NA´s* mediante *vtreat*.

El data frame generado (*no_missing_plan*) crea unas nuevas variables como se indicó anteriormente etiquetadas con ***_isBAD* con lo cual identifica claramente las columnas a las cuales se le aplico el tratamiento (*figura 25*) y el resumen muestra el listado de las variables y la verificación de que el número de valores ausentes (*missing*) presentes después de aplicar *vtreat* es igual a cero.

Figura 24: **Tratamiento Automático NA´s**

```
# Tratamiento Automático de NAs (vtreat)
# Variable no incluidas en el reemplazo automático
lista_vbles <- setdiff(colnames(clientes_data),
                       c("custid","health_ins"))

# Plan Base
no_missing <- design_missingness_treatment
                (clientes_data, varlist = lista_vbles)
no_missing_plan <- prepare (no_missing, clientes_data)
```

Figura 25: **Variables sin NA´s**

```
> # Determinar Variables con missing values después de vtreat
> skim(no_missing_plan) %>%
+   dplyr::select (skim_variable, n_missing)
# A tibble: 18 × 2
```

skim_variable	n_missing	skim_variable	n_missing
<chr>	<int>	9 income	0
		10 income_isBAD	0
1 custid	0	11 recent_move	0
2 sex	0	12 recent_move_isBAD	0
3 marital_status	0	13 num_vehicles	0
4 housing_type	0	14 num_vehicles_isBAD	0
5 state_of_res	0	15 age	0
6 health_ins	0	16 age_isBAD	0
7 is_employed	0	17 gas_usage	0
8 is_employed_isBAD	0	18 gas_usage_isBAD	0

Igualmente se verificará que la función *vtreat ()* haya reemplazado los NA´s y para este caso se realizará con dos variables. La primera edad (*age*) la cual entre sus entradas contaba con elementos missing (*figura 26*). Y la otra variable corresponde a número de vehículos (*num_vehicles)* la cual en el dataset original no tenía registros con NA´s. En la

figura 26 se filtran algunos clientes por edad y números de vehículos que poseen en el momento de la encuesta

Figura 26: **Código Verificación NA´s**

```
# Verificar el tratamiento
age_missing <- which(is.na(clientes_data$age))
vbles_verificar_original <- c ("custid","age", "num_vehicles")

> clientes_data [age_missing, vbles_verificar_original] %>% head ()
Custid      age      num_vehicles
823         NA       1
1773        NA       0
2332        NA       3
3304        NA       0
4048        NA       3
7108        NA       2
```

Figura 27: **Variables Age y num_vehicles**

```
> vbles_verificar_tratado <- c ("custid","age","age_isBAD","num_vehicles",
                                "num_vehicles_isBAD")
> no_missing_plan [age_missing, vbles_verificar_tratado]
                                %>% head ()
Custid   age      age_isBAD  num_vehicles   num_vehicles_isBAD
823      49.21647 1          1              0
1773     49.21647 1          0              0
233      49.21647 1          3              0
3304     49.21647 1          0              0
4048     49.21647 1          3              0
7108     49.21647 1          2              0
```

Como se observa en la *figura 27* para los mismos clientes, la variable age fue imputada por la media de la columna edad. El valor de la media se puede calcular mediante la siguiente instrucción: *mean (clientes_data$age, na.rm = TRUE)* dando un resultado de *49.21647*. En cambio, el número de vehículos no se alteró después de la transformación.

Las variables binarias *_isBAD* agregadas por la función *vtreat* para indicar si hubo alguna modificación en la columna original. El digito igual a uno (*1*) indicara cambios en la entrada y el valor de cero (*0*) en caso contrario. La columna referente a la edad todas las entradas corresponden a uno, mientras en la variable para el número de vehículos el valor es cero. Cabe aclarar las *figuras 26 y 27* corresponden a una muestra parcial con el objetivo de mostrar los resultados pertinentes.

Antes de continuar con las transformaciones cuyo propósito es adecuar la data para un asertivo modelaje, el analista debe ser consciente de que los cambios incluidos van a incrementar el consumo de recursos del equipo, la simulación de modelos de aprendizaje automático está limitada a la capacidad de cálculo del ordenador. Mediante el paquete *DataExplorer* el cual permite la manipulación y visualización de la data y suministra información del consumo de recursos del equipo por parte del dataset. (Cui,2020)

La *figura 28* muestra el consumo total de la base de datos original discriminada por el tipo de variable, el espacio ocupado por las missing y las observaciones completas. Mientras la *figura 29* muestra un incremento de aproximadamente 4 Mb después de aplicar el procedimiento *vtreat* a la data.

Figura 28: **Memoria - Dataset Original**

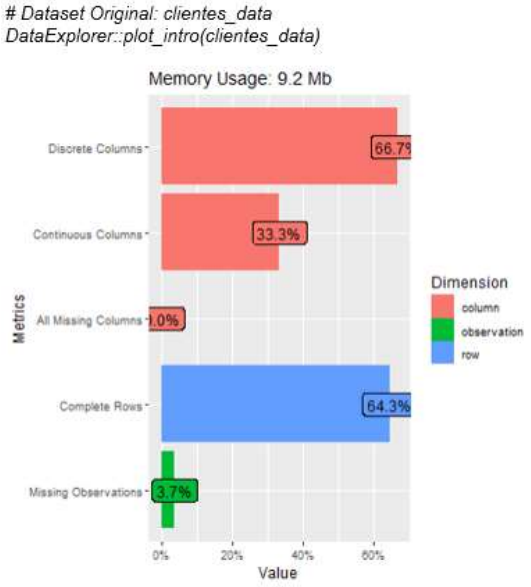


Figura 29: **Memoria – Dataset Modificada**

Transformación de los Datos:

El analista necesita estar claro tanto en el conocimiento de los datos como el modelo a usar para realizar las transformaciones requeridas en el dataset. Por ejemplo, si el modelo corresponde a una regresión ya sea lineal o logística la cual es una relación aproximadamente lineal entre las variables de entrada y salida y donde la varianza de las variables respuesta es constante. Por lo tanto, para lograr este objetivo probablemente se necesitarían realizar ciertas transformaciones como normalización, centrado y escalado y/o transformaciones tipo log.

La *Normalización* se usa cuando cantidades absolutas son menos significativas que la relativas respecto a la misma variable. Para contextualizar el concepto pensemos en los ingresos de los individuos de la base de datos en relación con el amparo de una póliza de salud: Que es más importante para el analista, el ingreso promedio de la base de datos o el ingreso promedio dependiendo del estado donde reside ya que es claro que en muchos países los ingresos por salarios para igual función dependen de la zona geográfica donde desarrolle la actividad, en especial en países como Estados Unidos origen de esta base de datos. Otro ejemplo significativo es la edad: Que sería más importante lo joven o adulto que un cliente sea respecto a mi apreciación o que tanto lo es con el promedio de la muestra.

En el caso de ingresos y para la base de datos *custdata.rds* (dataset original) se requeriría la media por estados para normalizar la variable *income,* dicho archivo no se encuentra disponible en el repositorio. Para la edad, con la media del dataset se podría normalizar la variable como se muestra en la *figura 30*.

Figura 30a: **Normalizar - edad**

```
> # Resumen Variable Edad
> summary(no_missing_plan$age)
Min    1st Qu   Median   Mean   3rd Qu   Max
21.00   34.00   48.00    49.22   62.00  120.00

> # Media Variable edad
> media_edad <- mean(no_missing_plan$age)
> print(media_edad)
[1] 49.21647
```

Figura 30b: **Normalizar – edad**

```
> # Variable edad normalizada por la media
> edad_normalizada <- no_missing_plan$age/media_edad

> # Resumen Variable Edad Normalizada
> summary(edad_normalizada)
Min      1st Qu   Median   Mean   3rd Qu   Max
0.4267  0.6908  0.9753  1.0000  1.2597  2.4382
```

De acuerdo a la *figura 30a* la edad media de la población es aproximadamente 50 años, después de normalizar la variable (*age*) los valores menores a uno (*1*) se consideran jóvenes y los valores mayores a uno (*1*) adultos (*figura 35b*).

Esta transformación no permite cuantificar que tan joven o adulto es un individuo de la base de datos respecto a la edad media, en este caso se recurre a la desviación estándar y en termino de ella se expresa una edad relativa.

Los datos se pueden *escalar* nuevamente usando la desviación estándar como unidad de distancia. En este caso, quienes se encuentren dentro de una desviación estándar son considerados ni más jóvenes ni más adultos del promedio de la muestra. Cuando el cliente se encuentre a una distancia mayor a una desviación se considera más adulto o en su defecto más joven que el promedio de la muestra. Un mecanismo para un entendimiento simple de la edad relativa seria centrar los datos, en este caso el valor medio de edad tendría un valor centrado de cero (0). La *figura 31* muestra el código para generar el rango de edad promedio y centrar los datos usando el valor de media como origen.

Figura 31: **Variable Edad - Centrar & Escalar**

```
> # Desviación Estándar
> desviacion_edad <- sd(no_missing_plan$age)
>
> # Rango para una desviación estándar
> print (media_edad + c (-desviacion_edad, desviacion_edad))
[1] 31.20407 67.22886
>
> # Escalado y Centrado, Origen = Media
>no_missing_plan$edad_escalada <- (no_missing_plan$age-media_eda
                                  / desviacion_edad
```

Figura 32: **Rangos – Desviación Estándar**

```
> # Clientes en el rango de edad promedio la edad escalada < 1
> no_missing_plan %>% filter(abs(age-media_edad) <desviacion_edad)
                    %>% select (age, edad_escalada) %>% head ()
age          edad_escalada
67             0.9872942
54             0.2655690
61             0.6541903
64             0.8207422
57             0.4321210
55             0.3210864
>
> # Clientes en el rango de edad promedio la edad escalada > 1
> no_missing_plan %>% filter(abs(age-media_edad)>desviacion_edad)
                    %>% select (age, edad_escalada) %>% head ()
age          edad_escalada
24             1.399951
82             1.820054
31             1.011329
93             2.430745
76             1.486950
26             1.288916
```

La *figura 31* indica que el rango promedio para la población corresponde a las edades comprendidas entre 31 a 67 años, el cual se obtiene de sumar o restar una desviación estándar a la media. Igualmente, los datos se centran restándole el valor de la media y se escalan dividiendo el resultado anterior por la desviación.

La *figura 32* muestra los clientes que se ubican en el rango promedio de edad menor a una desviación estándar, y el valor escalado es menor a uno (*1*) significando personas cuya edad se encuentra entre los 31 a los 67 años. Para valores escalados mayores a uno (*1*) son clientes más adultos o más jóvenes que el rango promedio calculado.

El procedimiento anterior de escalado y centrado es de forma manual, el R por medio de la función *scales ()* permite automatizar y extender el procedimiento a múltiples variables. como se muestra a continuación en la *figura 33*. Luego, se ejecutarán ambas funciones para las cuatro variables como se enlista el código en la *figura 34*. (Becker, s.f.)

Figura 33: **Resumen – Múltiples Variables**

```
> # Centrado y Escalado Múltiple
> data_multiple_center <- no_missing_plan
                  [, c("age","income","num_vehicles","gas_usage")]
> summary(data_multiple_center)
Age              income          num_vehicles   gas_usage
Min:   21.00   Min:      0     Min:   0.000   Min:     1.00
1st Qu: 34.00  1st Qu:  10700   1st Qu:  1.000  1st Qu:   3.00
Median: 48.00  Median: 26300   Median: 2.000  Median: 20.00
Mean:   49.22  Mean:   41793   Mean:  2.066   Mean:   41.17
3rd Qu:  62.00  3rd Qu:  51700  3rd Qu:.3.000  3rd Qu:  50.00
Max:   120.00  Max:   1257000  Max:   6.000   Max:   570.00
```

Figura 34: **Múltiples Variables - Scale () & Center ()**

```
> # Escalado Múltiple
> data_multiple_center_escalar <- scale (data_multiple_center,
                center = TRUE, scale = TRUE)
> summary(data_multiple_center_escalar)
Age              income          num_vehicles      gas_usage
Min:    1.56650  Min:    -0.7193  Min:     -1.78631  Min:     -0.6447
1st Qu: -0.84478  1st Qu:  -0.5351  1st Qu:  -0.92148  1st Qu:   -0.6126
Mean:   -0.06753  Median: -0.2666  Median: -0.05665  Median: -0.3398
Mean:    0.00000  Mean:    0.0000  Mean:    0.00000  Mean:    0.0000
3rd Qu:  0.70971  3rd Qu.: 0.1705  3rd Qu:   0.80819  3rd Qu:   0.1417
Max:    3.92971  Max:    20.9149  Max:     3.40268  Max:     8.4872
```

En la *figura 33*. se genera una submuestra con el objetivo de solo usar unas pocas variables numéricas para ilustrar el uso de la función *scale ()*. Se muestra el resumen estadístico de dichas variables con el objeto de tener un parámetro de comparación después de ejecutar la función.

La *figura 34* indica lo simple que es el uso de la función y ambas operaciones, *escalar* y *centrar*, se pueden ejecutar en la misma instrucción. Una de las ventajas de realizar escalamiento múltiple es que ahora todas las variables escaladas tienen rangos similares y más compatibles lo cual se puede percibir fácilmente al comparar ambos resúmenes en las *figuras 33* y *34*.

Con un fin didáctico se comparan dos variables, *edad* e *ingreso*. Por ejemplo, una diferencia de 20 entre un dato y otro no tiene el mismo efecto en edad que en la variable ingreso, es obvio que no tiene el mismo significado 20 años de edad en diferencia a 20 dólares en un ingreso. Después de escaladas y centradas, la *media* con valor igual a cero (*0*) tiene igual significado en ambas variables. Por ejemplo, el valor de *1,5* respecto a la edad o al ingreso se interpreta como una persona con *1,5 desviaciones estándar* mayor que el promedio de edad o de ingreso.

Cuando se usan medidas métricas tales como medias, medianas y desviaciones estándar para la transformación de los datos antes de modelar es buena costumbre mantener esos valores para usarlos nuevamente en el caso de que haya nuevos registros en el dataset. A continuación, las *figura 35* y *36* muestra las instrucciones para recuperar dichos valores almacenados como atributos, *center* y *scaled*, luego de ejecutar la función scale ().

Figura 35: **Atributo – Media**

```
> (media_data_multiple_center <- attr (data_multiple_center_escalar,
                                       "scaled: center"))
Age          income       num_vehicles     gas_usage
49.21647     41792.51062      2.06550      41.17021
```

Figura 36: **Atributo – Desviación**

```
> (sd_data_multiple_center <- attr (data_multiple_center_escalar,
                                    "scaled: scale"))
Age          income       num_vehicles     gas_usage
18.012397    58102.481410     1.156294     62.308948
```

Con el objetivo de valorar porque es importante guardar los valores usados en las transformaciones, solo pensar en nuevos registros en la base original daría relevancia ya que las métricas recuperadas se aplicarían a la nueva base de datos incluyendo por supuesto la transformación de *NA´s*. A continuación, las *figuras 37* y *38* muestran un hipotético caso donde se hubiesen ingresado nuevos registros a la data.

Figura 37: **Nueva - Data Scaled**

```
> library(vtreat)
> > # Nueva base de datos (opcional)
> data_nueva <- clientes_data
>
> # Aplicar Tratamiento de NAs usando el anterior caso
> tratamiento_data_nueva <- prepare (no_missing, data_nueva)
> data_nueva_filtrada <- tratamiento_data_nueva
                    [, c("age","income","num_vehicles","gas_usage")]

> # Escalar la nueva base de datos tratada de NAs
> data_nueva_escalada <- scale (data_nueva_filtrada,
center=media_data_multiple_center, scale=sd_data_multiple_center)
```

Figura 38: **Nueva Data - Resumen**

```
> summary(data_nueva_escalada)
Age                income           num_vehicles       gas_usage
Min.   :-1.56650   Min.   :-0.7193  Min.   :-1.78631   Min.   :-0.6447
1st Qu.:-0.84478   1st Qu.:-0.5351  1st Qu.:-0.92148   1st Qu.:-0.6126
Median :-0.06753   Median :-0.2666  Median :-0.05665   Median :-0.3398
Mean   : 0.00000   Mean   : 0.0000  Mean   : 0.00000   Mean   : 0.0000
3rd Qu.: 0.70971   3rd Qu.: 0.1705  3rd Qu.: 0.80819   3rd Qu.: 0.1417
Max.   : 3.92971   Max.   :20.9149  Max.   : 3.40268   Max.   : 8.4872
```

La *figura 37* muestra el código para crear una nueva base de datos, aplicar el tratamiento automático para entradas ausentes (*NA´s*) y luego se escala y centra usando las métricas almacenadas de un ejercicio previo. La instrucción estandariza el nuevo dataset usando la función *scale ()* el cual convierte cada valor original en un *z-score* correspondiendo al número de desviaciones estándar de la media desde un valor. La *figura 38* enlista el resumen de las variables.

Submuestra para Entrenamiento (Train) y Testeo (Test):

El objetivo final del aprendizaje automático es poder generar un modelo para ejecutar predicciones y que estas sean correctas con nuevos datos. Por lo tanto, es imperativo crear submuestras dentro de la base de datos con el fin de que una de ellas sirva de entrenamiento y la otra de testeo del modelo.

La submuestra es el proceso de seleccionar una parte de los datos que represente correctamente toda la población de la data. Con el objeto de testear, depurar y visualizar una muestra más pequeña antes de realizarlo en toda la base de datos. Las submuestras son la base para generar la data de *entrenamiento* (training) y de *prueba* (test).

En la literatura a la submuestra de *entrenamiento* se le llama *training set*, mientras a la de *prueba* se le denomina *test set*. Cabe aclarar, las publicaciones referentes a los avances científicos en la ciencia de datos y la inteligencia artificial traen consigo definiciones en inglés y al traducirlas al español muchas de ellas pierden la generalidad de lo que realmente significan.

El *training set* corresponde a los datos que se utilizan en la construcción del modelo con el fin de ajustarlo y lograr conseguir la mejor predicción de la variable resultado. Mientras tanto, *test set* corresponde a los datos que se usaran en el modelo resultante para verificar que las predicciones generadas sean precisas con datos nuevos. Algunos modelos requieren un set adicional de datos para calibración, *calibration set*, el cual se usa básicamente para establecer los parámetros de ajuste requeridos por el algoritmo, la sugerencia de los programadores consiste en caso de necesitarse datos de calibración realizar una submuestra del *training set*.

Es importante que las submuestras de la base de datos correspondan a una correcta interpretación de toda la muestra seleccionando los datos aleatoriamente. En R, existen varias formas de crear las muestras aleatoriamente, la instrucción *sample ()* incorporada en el programa es una de ellas. El paquete *dplyr* trae consigo las funciones *sample_n ()* y *sample_frac ()* y otra forma seria crear una columna con un numero aleatorio generado uniformemente por medio de la función *runif ()*, el cual será el método a describir a continuación ya que las dos anteriores opciones el manual de ayudas de R es didáctico para entender como ejecutar cada una de ellas

Sea cual sea el mecanismo de programación a usar para generar tanto el training set como el test set se necesita garantizar que la muestra extraída cada vez que se haga sea igual, y en especial cuando se está en el proceso de depuración. Solo pensar que el código se bloquee y cada vez la muestra aleatoria sea diferente, no hay certeza si el error se corrigió o en qué momento volverá aparecer, definir una semilla es primordial (*seed*).

En este caso, se genera una columna y con un numero uniforme entre *cero* (0) y *uno* (1) y a partir de este índice se define la proporción arbitrariamente a usar para el *training* como para el *test set*.

La *figura 39a* muestra como primer paso fijar una semilla arbitraria para garantizar que se genere igual muestra aleatoria cada vez. Esto es conveniente para la depuración del código o comparación de algoritmos. De otro modo, sin semilla aleatoria, sería demasiado complicado la certeza si el código se puede recrear nuevamente para evaluar reparaciones o si las conclusiones de los modelos comparados se pueden corroborar con nuevos datos. La segunda fila de código de la figura corresponde a la columna de índice creada con una muestra aleatoria uniforme.

El código descrito en la *figura 39b* corresponde a la generación de las submuestras tanto para entrenamiento como testeo. El umbral determinado, a priori, para el *training set* se definió como una décima parte del total de la muestra (10%) y el restante para *test set* (90%), se puede generar repetidamente la muestra aleatoria del tamaño que se desee solamente variando el valor del umbral. La instrucción dimensión (*dim ()*) del R indica el tamaño de cada una de las muestras, indicando igual número de columnas y la diferencia de registro de acuerdo a la partición realizada.

Figura 39a: **Training & Test Set**

```
# Crear semilla arbitraria
> set.seed (260269)

> # Crear columna de índices (index)
> clientes_data4$index <- runif (nrow (clientes_data4))
```

Figura 39b: Training & Test Set

```
> # Submuestras Training & Test Set
> clientes_data4_training <- subset (clientes_data4, index <=0.1)
> clientes_data4_test <- subset (clientes_data4, index >0.1)

> dim (clientes_data4_training)
[1] 7391   18
> dim(clientes_data4_test)
[1] 65871    18
```

Métodos de Modelación:

La *Predicción* es una inquietud, la cual ha acompañado a la humanidad desde los inicios de la vida misma, pero si nos remontamos a la asociación de predecir usando recursos estadísticos se puede referenciar al humanista y racionalista ruso americano *Isaac Asimov*. (Wikipedia,s.f.)

En una de sus obras, *Serie de la Fundación*, donde el matemático *Hari Seldon* desarrollo un campo científico llamado *Psicohistoria*, la cual es la combinación de la Historia, Sociología y Matemáticas permitiendo así la predicción en términos probabilísticos. De esta forma Seldon, personaje principal de la obra, crea una sociología matemática, con la cual utilizando las leyes estadísticas de acción de masas puede predecir el futuro de grandes poblaciones.

De igual forma en otro texto del mismo autor del año 1950, el cual luego fue la base creativa para un álbum musical, capítulo del programa los Simpson y película, "*Yo, Robot*" en uno de sus apartes dos hombres dialogan con *Cutie*, quien es un robot especializado en análisis de datos en la estación espacial donde habitan, ellos le suministran datos buenos y malos sobre la tierra para que luego él los use en sus algoritmos matemáticos para la toma de decisiones.

En el trabajo de tesis, hasta ahora, se explicó el análisis previo de la base de datos e identificar los datos anómalos y diferentes métodos de solución antes de incursionar en modelación. La manipulación previa de los datos permite clarificar las preguntas que se desean responde y el alcance del problema que se pretende solucionar. Ya que meta final es solucionar un problema concreto: aumentar márgenes de ganancia, identificar transacciones fraudulentas, predecir perdidas de un portafolio, etc.

El problema a analizar a continuación, corresponde a conocer si un individuo quien solicita un préstamo bancario de cualquier índole puede caer en situación de impago. Para conseguir esta respuesta se recurre a potentes métodos de clasificación tanto estadísticos como de aprendizaje automático, cada método tiene ventajas y desventajas para una meta propuesta con sus respectivas restricciones.

La elección entre una técnica tradicional como la regresión logística o algún modelo especifico de aprendizaje automático depende del contexto, los datos disponibles y las características del problema que se quiere solucionar.

La regresión logística es valiosa en termino de inferencia estadística, simplicidad, interpretabilidad y preferida si las relaciones son lineales. Mientras, el aprendizaje automático maneja alta dimensionalidad, eficiente en relaciones no lineales, capacidad de aprender automáticamente y alto rendimiento predictivo en problemas de clasificación complejos.

La base de datos a utilizar en la predicción usando una técnica estadística como la regresión logística y modelos de aprendizaje automáticos, corresponde a una publicada por el consultor y analista de datos de la universidad Johns Hopkins, Zach Stednick (Stednick, s.f.), quien las comparte en su Web Page, la cuales ya está previamente formateada para usar directamente con modelos de predicción, además son gratuitas, sugeridas por analistas de datos y de acceso libre.

Regresión Logística (Logit):

El modelo Logit es utilizado para predecir la probabilidad de que un evento binario suceda: 0/1, éxito/fracaso ó si/no. La función *Logit* modela la relación lineal entre las variables independientes o predictoras y las variables resultado o dependientes representando la proporción de las probabilidades de éxito o fracaso mediante una transformación logarítmica de los odds ratio.

El modelo estima los coeficientes de la regresión utilizando máxima verosimilitud o técnicas de optimización. Dichos coeficientes cuantifican el efecto de las variables predictoras en la probabilidad de ocurrencia del evento de interés y con ellos se pueden hacer nuevas predicciones con nuevos datos.

Aprendizaje Automático o Machine Learning (ML):

El concepto de Aprendizaje Automático (*Machine Learning*) corresponde a un enfoque de Inteligencia Artificial (*AI*) que permite a la computadora aprender automáticamente sin ser programada explícitamente para cada tarea específica. La máquina aprende de los datos directamente y busca parámetros y relaciones entre ellos con el objetivo de predecir o tomar decisiones. El elemento fundamental del ML son lo datos, los cuales pueden ser numéricos, textuales, imágenes, audio, etc. Estos necesitan cumplir ciertas características propias

como se ha explicado previamente con el fin de garantizar una alta fiabilidad en los resultados. Los datos se dividen principalmente en entrenamiento (*training set*) y los de prueba (*test set*), como se mencionó en la sección anterior, los cuales se usan tanto para entrenar el modelo como para generar las predicciones o tomar decisiones.

Los algoritmos de Machine Learning son los encargados de ejecutar los modelos cuyos datos cumplan ciertas características previas para la ejecución de la rutina de dicho modelo especifico. Cabe anotar, la literatura de cada uno de los algoritmos es profusa tanto física como digital, donde explican claramente cada parámetro involucrado, las limitaciones y alcances de cada modelo. Además, el ejecutarlos después de ajustar adecuadamente los datos y cumplir los requisitos requeridos por el modelo solicitado es relativamente simple.

ML corresponde a un proceso iterativo, el modelo se ajusta a los datos de entrenamiento y aprende hacer predicciones, después de entrenar el modelo se evalúa su rendimiento utilizando el conjunto de datos de prueba. La evaluación ayuda a determinar la generalización del modelo con datos nuevos, en caso de no cumplir las expectativas se realizan los ajustes necesarios en el algoritmo, los datos o parámetros hasta conseguir el rendimiento óptimo.

Ejercicio de Predicción (Probabilidad de Default): Logit y Modelos de Aprendizaje Automático

A continuación, *figura 40a*, muestra la instrucción requerida para cargar directamente desde la web la base de datos a trabajar indicando el *url* donde se aloja el dataset y el separador de datos, en este caso son las comas (,). Adicionalmente, la *figura 40b* indica las librerías fundamentales requerida del R para el procesamiento de datos, para la visualización gráfica, para los modelos del ML y para generar las curvas ROC.

Figura 40a: **Dataset – Probabilidad de Default**

```
> # Cargar datos desde el URL
> # Base de datos: GitHub de Zach Stednick
>url <- "https://raw.githubusercontent.com/stedy/
        Machine-Learning-with-R-datasets/master/credit.csv"
> data <- read.csv (url, sep = ",")
```

Figura 40b: **Librerías – Probabilidad de Default**

```
> library(caret)
> # Gráficos y Procesamiento de Datos
> library(tidyverse)
> # ML XGBoost
> library(xgboost)
> # Análisis de Curvas ROC
> library(pROC)
> # Modelos ML
> library(randomForest)
> library(class) # Modelo kNN
> library(xgboost)
> # Generación de Gráficos
> library(ggplot2)
```

Regresión Logística: La *figura 41a* muestra el código para la variable cualitativa *default*, la cual se codifica para dos valores 0 y 1 para asegurar que el rango de probabilidades este entre [0,1]. Mientras la *figura 41b* se muestra el modelaje de la probabilidad de impago (*default*) en función de la cantidad pendiente de pago (*amount*).

Figura 41a: **Logit – codificación [0,1]**

```
> # Base de datos para regresión Logística
> data_logit <- data
>
> # Se recodifican los niveles No, Yes a 1 y 0
> data_logit <- data_logit %>%
                select (default, amount) %>%
                mutate (default = recode (default,
                                          "1" = 0,
                                          "2" = 1))
```

Figura 41b: **Logit – Default-amount**

```
> head(data_logit)
  default amount
1       0   1169
2       1   5951
3       0   2096
4       0   7882
5       1   4870
6       0   9055
```

La función *Logit* transforma el resultado de una regresión lineal empleando una función (sigmoide o máxima verosimilitud) cuyo resultado este siempre comprendido entre 0 y 1. La *figura 42a* corresponde al código para la función Logit y el resultado generado en R. La *figura 42b* corresponde a la representación gráfica, la cual muestra que las probabilidades se encuentran obligatoriamente entre 0 y 1.

Figura 42a: **Logit - GLM**

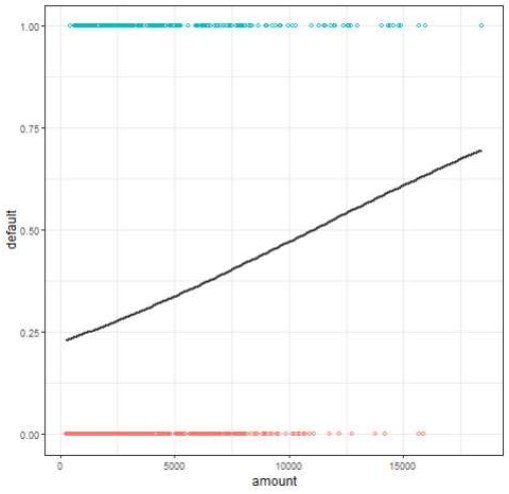
```
> # Ajuste de un modelo logístico.
> modelo_logistico <- glm (default ~., data = data_logi
                           family = "binomial")
> modelo_logistico

Call:  glm (formula = default ~., family = "binomial",
        data = data_logit)

Coefficients:
(Intercept)              amount
1.2293749                0.0001119

Degrees of Freedom: 999 Total (i.e. Null); 998 Residu
Null Deviance: 1222
Residual Deviance: 1199      AIC: 1203
```

Figura 42b: **Logit - Grafica**



La *figura 43a* indica que la variable de interés *default* se renombro a *target* ya que va hacer la primera variable a predecir y se quiere tener control sobre ella. Cabe aclarar, la columna default del dataset original define dos valores: uno (1) para clientes en default o dos (2) en

no-default. Luego, dicha variable se transforma a factor para que represente una variable categórica con las dos opciones para la predicción (*default (1)*, *no-default (2)*) y se suministra la dimensión de la base de datos, comando *dim ()*, indicando que el dataset contiene 1.000 registros con 21 variables. La *figura 43b* corresponde a la instrucción para generar la distribución de la variable objetivo (*target*), la cual indica que el 70% de los usuarios se encuentran clasificados con cesación de pagos y el 30% en no-default.

Figura 43a: **Variable Objetivo - Target**

```
> # Renombrar vble objetivo (default) por target
> data_ML <- data_ML %>%
                rename (target=default) %>%
                mutate (target=factor(target))
>
> # Resumen Base de Datos
> dim (data_ML)
[1] 1000   21
```

Figura 43b: **Variable Target - Distribución**

```
> # Analizar resumen y distribución variable target
> summary (data_ML$target)
      1 (default)      2 (no – default)
      700              300
> prop.table (summary (data_ML$target))
      1 (default)      2 (no – default)
      0.7              0.3
>
> # Semilla fija para los modelos ML
> set.seed (260269)
```

El paso a seguir es generar una partición de la base de datos entre training and test set como ya se explicó previamente, en esta ocasión usaremos una función directamente con el fin de ampliar las opciones a la hora de decidir el método a seguir. Cabe recordar, fijar una semilla (*set.seed*) para asegurar luego replicar el modelo con una selección aleatoria de índices. Para generar la partición, se usará una función del paquete *caret* (*createDataPartition*), la cual indicando los parámetros requeridos genera ambas submuestras como lo muestra la *figura 44a. (Zach. S.f.*)

Figura 44a: **Crear Partición**

```
> library(caret)
> # Crear Particion
> index <- createDataPartition (data_ML$target, times = 1,
                p=0.2, list = FALSE)
> # https://www.statology.org/createdatapartition-in-r/
        > # y: Vector, DataBase
        > # times: Numero de particiones a crear
        > # p: Porcentaje de datos usado en el set
        > # list: Almacenar en lista los resultados; si o no
```

Figura 44b: **Training_set; Test_set**

```
> # Train_set correspondiente al 80% de la muestra
> train_set <- data_ML[-index,]
> dim(train_set)
[1] 800    21

> # Test_set correspondiente al 20% de la muestra
> test_set <- data_ML [index,]
> dim(test_set)
[1] 200    21
```

La *figura 44b* indica que el training set corresponde al 80% de la muestra, esto es 800 registros para 21 variables. mientras, el test set son 200 usuarios para igual número de columnas.

Antes de iniciar los algoritmos de aprendizaje automático, es importante un análisis exploratorio de predicción de la variable objetivo para establecer una referencia. En orden de realizar dicho análisis se recurre a la función *sample ().* (Becker. s.f.)

En la *figura 45* se muestra la aproximación aleatoria más simple correspondiente a la variable *target*, el resultado de referencia no es indicativo ya que la predicción resultante fue aproximadamente 50%. Ahora se tratará de evaluar si la precisión se incrementa usando como predictor una variable especifica. Candidatas viables podrían ser *credit_score*, *credit_history*, etc. En este caso se usará *credit_history* la cual es un referente importante para los analistas financieros en el caso de solicitud de créditos para determinar el nivel de pago del potencial cliente.

Figura 45: **Predicción - Target**

```
> # Prediccion de referencia para la variable objetivo (target)
> # sample (x, size, replace = FALSE, prob = NULL)
> # https://www.rdocumentation.org/packages/base/
                versions/3.6.2/topics/sample
                > # x: vector o elemento o
                      entero positivo a seleccionar
                > # size: número de ítems a seleccionar,
                      entero positivo
                > # replace: ¿la muestra debería ser reemplazada?
                > # prob: pesos de probabilidad para
                      los elementos de la muestra
> target_predict1 <- sample (c (1,2), length(index), replace = TRUE)
                %>% factor (levels = levels(test_set$target))
>
> # Media de la prediccion de la variable objetivo
> mean(target_predict1==test_set$target)
[1]     0.435
```

Figura 46: **Predicción -credit_history**

```
> data_ML %>% group_by (credit_history) %>%
                summarise (mean (target == 2))
# A tibble: 5 × 2
  credit_history        `mean (target == 2)
  <chr>                              <dbl>
1 critical                           0.171
2 delayed                            0.318
3 fully repaid                       0.625
4 fully repaid this bank             0.571
5 repaid                             0.319
>
> # Prediccion de referencia con predictor especifico (credit_history)
> # if_else (condition, true, false, missing = NULL)
> target_predict2 <- if_else (test_set$credit_history
                %in% c ("fully repaid", "fully repaid this bank"),2,1)
> # Media de la Prediccion de referencia
        con predictor especifico (credit_history)
> mean(target_predict2==test_set$target)
[1] 0.74
```

En la *figura 46* se solicita el resumen de los registros no-default agrupados por la variable referente a la historia crediticia. Se observa que los créditos totalmente pagados son casi el total de la muestra. El uso de una variable especifica (*credit_history*) como predictor muestra un significativo avance ya que paso del casi un 50% a algo más del 70%. Se podría seguir haciendo el ejercicio con otras variables, pero en este caso se pasará a usar algunos modelos de aprendizaje automático.

Los modelos ML a usar corresponden a *GLM*, *kNN*, *Random Forest* y *XGBoost*. Para los cuales se comparan las predicciones de probabilidad y el desempeño de ellos usando la métrica del área bajo la curva (*AUC*). A continuación, se describirá cada uno de ellos en forma genérica recurriendo a un website recomendado por la literatura llamado ciencia de datos (Amat. s.f.). En caso de requerir especificaciones detalladas de cualquier modelo recurrir a las ayudas respectivas en la documentación R.

GLM: El modelo de Regresión Lineal Generalizada (*GLM*) corresponde a una clase de ML el cual generaliza el concepto de regresión lineal permitiendo manejar varios tipos de variables respuesta tales como binarias, reconteo y continuas. Las distribuciones de error,

combinando ya sea un predictor lineal o una función lineal o una distribución de probabilidad, lo cual permite la modelización de los predictores y la variable respuesta.

GLM se entrena maximizando la función de verosimilitud, *likelihood*, estimando los parámetros que mejor se ajustan a los datos dada la distribución de probabilidad y la función de enlace elegida por el modelo, la cual se usa para la predicción y la inferencia.

La *figura 47* muestra el código para ejecutar la función *train ()* del R, con la cual se ejecutarán los algoritmos de aprendizaje automático GLM y kNN. Se observa claramente que los parámetros a definir son simples: Variable objetivo (*target*), predictores (todas las variables excepto la objetivo), el dataset preparado previamente (*train_set*) y el algoritmo de machine learning a usar (*glm*). (Kuhn, 2008)

Figura 47: **Machine Learning – modelo GLM**

```
> ml_glm <- caret::train (target ~., train_set, method = "glm")
> # https://www.rdocumentation.org/packages/caret/
      versions/4.47/topics/train
      > # x: Vector factor conteniendo resultado de la muestra
      > # Data: Data set
      > # method: Especificar regresión o clasificación a usar
```

Figura 48: **GLM – Predicción & Matriz de Confusión**

```
> # Prediccion
> mean (test_set$target == predict (ml_glm, test_set))

> # Matriz de Confusión (mat_conf) para modelo glm
> mat_conf_glm <- confusionMatrix (predict (ml_glm, test_set),
        as.factor (test_set$target))
```

La *figura 48*, suministra la media de la predicción de la variable target dada por el modelo GLM. Adicionalmente, la matriz de confusión para este modelo. Ambos resultados se tabularán al final de ejecutar los cuatro modelos con el fin de tener una comparación.

La matriz de confusión es una medida de rendimiento para la clasificación de ML donde las salidas pueden ser dos o más clases. En este caso, el enfoque estará dado solamente para los siguientes tres resultados: *Accuracy*, *Precision*, *Recall*.

Accuracy corresponde al porcentaje de predicciones correctas realizadas por el modelo. Precision son los verdaderos positivos (verdaderos positivos y falsos positivos) obtenidos en el resultado de la predicción. Recall es la proporción de los verdaderos positivos respecto al resultado total de positivos.

***kNN*:** El modelo de vecinos cercanos (*k-Nearest Neighbors*) corresponde a un algoritmo de aprendizaje automático usado tanto para clasificación como regresión, cuyas predicciones se basan en la similitud de los datos de entrada con los de entrenamiento. El modelo incluye básicamente cuatro fases: Entrenamiento, Calculo de la distancia, seleccionar los k vecinos cercanos y la predicción.

Para el modelo de vecinos cercanos se usa igualmente la función *train ()* incorporándole un control para ella (*trainControl*) en el cual se necesita especificar el método a usar, en este caso se usará validación cruzada (*CV*) indicando el número de veces. Adicionalmente, se requiere especificar el ajuste (*tuneGrid*) para k, se usará desde tres hasta 71 en pasos de

tres. El valor del ajuste no se ciñe algún criterio especifico en este caso, solamente se tomó igual a los valores de algún ejercicio previo.

La *figura 49* lista el código usado para el modelo kNN donde se especifica cada uno de los parámetros los cuales ya se habían determinado previamente para el algoritmo anterior. Mientras la *figura 50* corresponde al cálculo de la predicción y la matriz de confusión.

Figura 49: **Machine Learning – modelo kNN**

```
> ml_knn <- train (target ~., train_set,
                   method = "knn",
                   trControl = trControl,
                   metric    = "Accuracy",
                   tuneGrid = data.frame (k = seq (3, 71, 3)))
```

Figura 50: **kNN – Predicción & Matriz de Confusión**

```
> # Prediccion
> mean (test_set$target==predict (ml_knn, test_set))

> # Matriz de Confusion (mat_conf) para modelo knn
> mat_conf_knn <- confusionMatrix (predict (ml_knn, test_set),
                                   as.factor (test_set$target))
> mat_conf_knn$byClass
```

***Random Forest*:** Bosque aleatorio (*Random Forest*) corresponde a un modelo de amplio reconocimiento en ML para clasificación y/o regresiones. El método de aprendizaje combina múltiples arboles de decisión para realizar la predicción.

La primera fase del algoritmo consiste en generar el árbol, el cual es construido usando submuestras aleatoriamente tanto del entrenamiento como las variables, con lo cual se busca aumentar la diversidad y reducir el *overfitting*. En la segunda fase, el modelo selecciona para cada nodo una submuestra de las columnas de datos con el fin de mejorar la división de los ramales. La tercera fase corresponde al entrenamiento, el cual se realiza con una submuestra del training set donde cada árbol crece en forma independiente en función de las variables hasta llegar al nodo final (*leaf nodes*). La última fase corresponde al método que el modelo usa para predecir. En el caso de clasificación, la predicción final se ejecuta por votación en la cual la clase que recibe más alta votación en los árboles de decisión es la determinante. Mientras tanto, para regresión se promedian las predicciones de todos los árboles.

El modelo *Random Forest* surge como una modificación de los modelos *bagging* construyendo una cantidad de árboles no correlacionados agregándole aleatoriedad al proceso de crecimiento de los árboles y tratando de disminuir la varianza mejorando así el desempeño del modelo. (Singh, 2018).

Los parámetros a definir para el algoritmo (*figura 51*) corresponde a la variable respuesta (*target*) y los predictores corresponderán a las demás variables, la base de datos a usar es la de entrenamiento (*training_set*). Para el ajuste (*tuning*), el número de árboles (*ntree*) se seleccionará un valor igual a 100 solo por limitante computacional siendo 500 el valor por defecto. El parámetro de numero de variables muestreadas aleatoriamente en cada división (*mtry*) se toma una secuencia (*seq*) entre uno a siete.

Igualmente, como en los modelos anteriores la *figura 52* ejecuta el código de la media de probabilidad para el algoritmo en cuestión y la matriz de confusión, cuyos resultados se tabularán y se mostrarán después del cuarto modelo de aprendizaje automático.

Figura 51: **Machine Learning – modelo** *Random Forest (rf)*

```
> library (randomForest)
> # https://datascienceplus.com/random-forests-in-r/
> ml_rf <- train (target ~., train_set,
        method = "rf",
        ntree = 100,
        tuneGrid = data.frame (mtry = seq (1:7)))
> # Predictor: target y todas las demás variables
> # Data: train_set
> # method: Random Forest
> # ntree: Numero de árboles (100)
> # seq: Ajuste en una secuencia de 1 a 7
```

Figura 52: **RF – Predicción & Matriz de Confusión**

```
> # Prediccion
> mean (test_set$target == predict (ml_rf, test_set))

> # Matriz de Confusion (mat_conf) para modelo knn
> mat_conf_rf <- confusionMatrix (predict (ml_rf, test_set),
                as.factor (test_set$target))
> mat_conf_rf$byClass
```

XGBoost: El modelo del impulso extremo del gradiente (*Extreme Gradient Boosting*) hace parte de la familia de algoritmos *gradient boosting*, a los cuales los programadores de aprendizaje automático los reconocen como de alto rendimiento y escalabilidad que se utiliza para resolver problemas de predicción de valores o la clasificación de datos. Suele utilizarse cuando se dispone de muchos datos y se quiere hacer predicciones o tomar decisiones precisas.

La primera fase del *XGBoost* corresponde a aprendices débiles, los cuales mediante arboles de decisión se realizan predicciones y el modelo se entrena iterativamente corrigiendo los errores en las anteriores predicciones. La segunda fase usa el modelo gradient boosting para minimizar la función de perdida añadiendo iterativamente aprendices débiles como en el caso anterior para ajustar los pesos de la función de perdida con respecto a la predicción. La tercera fase realiza un submuestreo de las variables similar al descrito para el modelo random forest para introducir aleatoriedad y reducir la correlación entre los árboles mejorando así la generalización. La cuarta fase, el modelo incluye técnicas de regularización con el propósito de controlar el overfitting lo cual lo logra con penalidades a la función de perdida. En la última fase del modelo, controla la contribución de cada aprendiz débil, buscando tasas de aporte bajas con el fin de robustecer el modelo. Además, para de recibir contribuciones de los aprendices cuando el rendimiento del modelo deja de mejorar.

La descripción detallada del algoritmo se puede encontrar en el artículo del año 2016 de los profesores de la Universidad de Washington *Tianqi Chen* y *Carlos Guestrin* donde describe el nuevo modelo *XGBoost*. (Chen, 2016)

El algoritmo requiere una implementación inicial. El primer paso consiste en convertir los datos de entrenamiento a un tipo de matriz denominada *Dmatrix*, para lo cual se usará la función *xgb.Dmatrix* incorporada en el paquete XGBoost, para información referente a

xgb.Dmatriz recurrir a las ayudas de R mediante la siguiente instrucción - *??xgb.train*. Una consideración importante que resalta la ayuda en R del paquete en cuestión, la matriz no puede incorporar la variable objetivo, de lo contrario la precisión obtenida en el set de entrenamiento será inútil para datos nuevos. En la *figura 53* se observa que la variable objetivo (*target*) se excluye de la matriz generada, lo cual se logra mediante la siguiente instrucción - (*! names (train_set_xgb) %in% c("target")*).

Después de realizar el proceso de la generación de la matriz, se pasa al entrenamiento del modelo predictivo. (Mendoza, 2018)

La función de entrenamiento a utilizar corresponde *xgboost ()* incluida en el paquete con igual nombre. El primer paso es definir los hiper-parámetros a utilizar, en este caso se detallará a los que se ajustaran en el ejercicio:

- Objective; Tipo de tarea de clasificación a realizar, en este caso binary: logistic para regresión binaria y salida de probabilidad
- Eval_metric: Métrica de evaluación para los datos, en este caso se usará área bajo la curva (AUC) para la evaluación de clasificación.
- Max_depth: Máxima profundidad del árbol o número de nodos en los árboles de decisión usados para el entrenamiento, por defecto el parámetro es igual a seis.
- Subsample: corresponde a la proporción de la submuestra referente al entrenamiento, el valor configurado le indica al *xgboost* recoger aleatoriamente dicho parámetro para hacer crecer los árboles evitando así el overfitting. El valor por defecto es igual a uno.
- Colsample_bytree: Corresponde a la proporción de las columnas de la submuestra para construir los árboles. El valor por defecto es igual a uno.
- Data: corresponde al set de entrenamiento, acepta solamente xgb.Dmatrix
- Verbose: Activa la función callback para el periodo igual a uno.
- Nrounds: Máximo número de iteraciones del impulso (boosting)

Figura 53a: **XGBoost- Preprocesamiento datos**

```
> library(xgboost)
> train_set_xgb <- train_set%>%
                   mutate_if(is.character,as.factor)%>%
                   mutate_if(is.factor,as.numeric)%>%
                   mutate (target = if_else(target==1,0,1))
> # Preprocesar test set
> test_set_xgb <- test_set%>%
                   mutate_if (is.character,as.factor) %>%
                   mutate_if (is.factor,as.numeric) %>%
                   mutate (target = if_else(target==1,0,1))
```

Figura 53b: **XGBoost- Matrix xb.Dmatrix & Parámetros**

```
> train_matrix_xgb <- xgb.DMatrix(
                   data = as.matrix (train_set_xgb [,! names(train_set_xgb)
                   %in% c("target")]),
                   label = train_set_xgb$target )

> parameters_xgb <- list (objetive = "binary: logistic", eval_metric = "auc",
                   max_depth = 11, eta = 0.071, subsample = 0.99,
                   colsample_bytree = 0.85)
```

En la *figura 54* muestra la ejecución del modelo de ML después del preprocesamiento previo y el alistamiento de los parámetros respectivos. En la *figura 55* se ejecuta el código para calcular la predicción media del algoritmo XGBoost y la respectiva matriz de confusión.

Figura 54: **Machine Learning – modelo XGBoost**

```
> ml_xgb <- xgb.train (params = parameters_xgb,
                       data = train_matrix_xgb, verbose = 1,
                       nrounds = 155)
```

Figura 55: **XGBoost – Predicción & Matriz de Confusión**

```
> # Prediccion
> xgb_prediccion <- ifelse (predict (ml_xgb, newdata = test_matrix_xgb)
                            >0.5,1,0)
> mean (test_set_xgb$target == xgb_prediccion)

> # Matriz de Confusion (mat_conf) para modelo XGBoost
> mat_conf_xgb <- confusionMatrix (as.factor (xgb_prediccion),
                                   as.factor (test_set_xgb$target))
> mat_conf_xgb$byClass
```

La *figura 56* recolecta las predicciones tanto de los modelos de aprendizaje automático como el análisis exploratorio de la variable objetivo (*target*) y usando una variable especifica como predictor (credit_history. La precisión se movió en todos los casos en un rango entre 74% y 78% excepto para el caso más simple de predicción, usando solamente la variable target.

Figura 56: **Predicción – Resumen**

	Precision
Target	0.44
Credit_History	0.74
GLM	0.75
kNN	0.73
Random Forest	0.78
XGBoost	0.76

Figura 57: **Matriz de Confusión**

	Matriz de Confusion		
	Accuracy	Precision	Recall
GLM	0.79	0.80	0.85
kNN	0.70	0.72	1.00
Random Forest	0.78	0.78	0.94
XGBoost	0.79	0.79	0.89

En la matriz de confusión, *figura 57*, indica que los modelos con más alto promedio de *accuracy* corresponde al GLM y XGBoost significando que el 79% de todos fueron correctamente clasificados sin importar si ellos fueron positivos o negativos. En el caso de *precisión*, los scores más altos corresponden a los modelos GLM, Random Forest y XGBoost indicando que entre todos los casos predichos como positivos entre el 78% al 80% fueron correctamente clasificados. En el caso del *recall*, la tasa de recuperación de casos reales positivos alcanzo el 100% en el modelo kNN y luego Random Forest con el 94%.

Finalmente, se realizará un análisis de las curvas ROC y AUC con el objeto de seleccionar el mejor modelo para el análisis de clasificación de los créditos default.

La Curva ROC (*Receiver Operating Characteristic*) es una métrica útil de comparación para modelos binarios de clasificación donde se muestra la relación entre la tasa de verdaderos

positivos (*Sensibilidad*) y la de falsos positivos (*Especificidad*) a medida que el umbral varie. Adicionalmente el AUC (*Area Under Curve*) de la ROC es una medida que se usa para comparar los modelos igualmente en forma cuantitativa. El *área bajo la curva* representa la probabilidad de que el resultado de un ensayo para un caso positivo seleccionado supere el resultado de un caso negativo elegido aleatoriamente. Por ejemplo, cuando el *AUC* es igual al 0.5% el modelo no tiene la capacidad de discriminación para distinguir entre clase positiva y clase negativa. Cuando *AUC* es aproximadamente cero el modelo predice la clase negativa y cuando *AUC* es el 0.7 significa que existe el 70% de probabilidades de que el modelo pueda distinguir la clase positiva de la clase negativa.

La *figura 58* enlista el código pertinente para solicitarle al R la curva *ROC* usando el paquete *pROC* para cada uno de los cuatro modelos de ML analizados.

Figura 58a: **ROC – GLM & kNN**

```
> # Modelo glm
> roc_glm <- roc (as.numeric (test_set$target),
                  as.numeric (predict (ml_glm, test_set)))
Setting levels: control = 1, case = 2
Setting direction: controls < cases

> # Modelo knn
> roc_knn <- roc (as.numeric(test_set$target),
                  as.numeric (predict (ml_knn, test_set)))
Setting levels: control = 1, case = 2
Setting direction: controls < cases
```

Figura 58b: **ROC – RF & XGB**

```
> # Modelo rf
> roc_rf <- roc (as.numeric (test_set$target),
                 as.numeric(predict (ml_rf, test_set)))
Setting levels: control = 1, case = 2
Setting direction: controls < cases

> # Modelo xgb
> roc_xgb <- roc (as.numeric (test_set_xgb$target),
                  xgb_prediccion)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

A continuación, se procede a combinar la data generada con el código de *figura 58* en un solo data frame (*figura 59*) y luego usando el paquete grafico *ggplot* se obtienen las respectivas curvas ROC. (figura 60)

Figura 59: **Combinación Modelos**

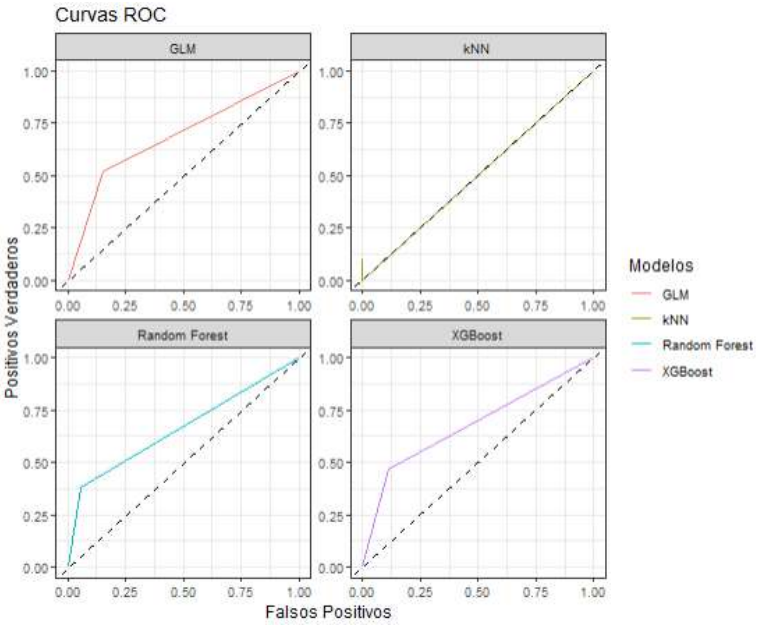
```
> roc_comb <- rbind (
        data.frame (Modelos = "GLM", roc_glm %>% coords ()),
        data.frame (Modelos = "kNN", roc_knn %>% coords ()),
        data.frame (Modelos = "Random Forest", roc_rf %>% coords ()),
        data.frame (Modelos = "XGBoost", roc_xgb %>% coords ()))
```

Figura 60: **ROC – Plots Código**

```
> ggplot (roc_comb, aes (x = 1 - specificity, y = sensitivity)) +
        geom_line (aes (color = Modelos)) +
        geom_abline (slope = 1, intercept = 0, linetype = "dashed") +
        labs (title = "Curvas ROC", x = "Falsos Positivos",
              y = "Positivos Verdaderos") +
        facet_wrap (~ Modelos, scales = "free") +
        theme_bw ()
```

Las curvas ROC se muestran en la *figura 61* para los modelos de aprendizaje automático: GLM, kNN, RF y XGB. En una clasificación binaria las curvas es una manera fácil de entender el desempeño de los modelos, entre más alta sea la gráfica indica mejor rendimiento. En los modelos analizados las curvas con mejor perfil corresponden a los modelos GLM y XGBoost.

Figura 61: **Curvas ROC – Plots**



El otro elemento cuantitativo analizar corresponde al área bajo la curva ROC (AUC) el cual determina que el modelo ML clasifique un real positivo elegido aleatoriamente más alto que una instancia negativa igualmente elegida al azar, la siguiente referencia corresponde a un estudio detallado de este tipo de gráficas, *análisis ROC*. (Fawcett, 2006)

La *figura 62* muestra el código para compilar el *AUC* de los modelos y tabularlos mediante la función *kable ()* incorporada en el paquete *knitr ()*, el cual permite darle formato de salida a la tabla y ordenarla en forma descendente. (Quick, 2010)

Figura 62a: **AUC – Código Tabla**

```
> tabla_auc <- data.frame (Modelos = c ("XGBoost", "GLM",
        "Random Forest", "kNN"), auc = c (roc_xgb$auc,
        roc_glm$auc, roc_rf$auc, roc_knn$auc))

> knitr::kable (digits = 2, arrange (tabla_auc, -auc),
        align = "lc", format = "rst")
```

Figura 62b: **AUC - Tabla**

Modelos	auc
GLM	0.68
XGBoost	0.68
Random Forest	0.66
kNN	0.55

REFERENCIAS

Amat, J. (s.f.). Machine Learning con R. *Ciencia de Datos*. https://www.cienciadedatos.net/machine-learning-r

Becker, R. (s.f.). Scale Version 3.6.2. *Cran – r. Avalaible Packages.* *https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/scale*

Becker, R. (s.f.). Random Samples and Permutations. *Cran – r. Avalaible Packages.* https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/sample

California State University – Irvine. (2007). Car Evaluation Data Set. *UCI Machine Learning Repository*. http://archive.ics .uci.edu/ml/custdata.RDS

CFA Institute. (2022). Machine Learning in Finance using Python. *https://financetrain.com/c/127*

Chang, W. (s.f.). ggplot2 Function Reference. *Ggplot2 3.4.2*. https://ggplot2.tidyverse.org/

Chen, T., Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. https://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf

CRAN. (2022). Using Skimr. *Cran – r. Avalaible Packages*. https://cran.r-project.org/web/packages/skimr/vignettes/skimr.html

Cui, B. (2020). Introduction to Data Explorer. *Cran – r. Avalaible Packages.* https://cran.r-project.org/web/packages/DataExplorer/vignettes/dataexplorer-intro.html

Dancho; M. [@mdancho84]. (15 de mayo 2023). *R. One letter that can change your life, forever*. [tuit]. Twitter. https://twitter.com/mdancho84

Exponentis. (2019). Uso de la función mutate () de dplyr junto a un condicionante ifelse en R. *Exponentis Blog*. http://exponentis.es/uso-de-la-funcion-mutate-de-dplyr-junto-a-un-condicionante-ifelse-en-r

Fawcett, T. (2006). An Introduction to ROC Analysis. *Science Direct*. https://www.sciencedirect.com/science/article/abs/pii/S016786550500303X

Gareth, J., Daniela, W., et alii. (2021). *An Introduction to Statistical, Learnig with Application in R*. Springer.

Kabacoff, R. (2015). *R in Action, Data Analysis and Graphics with R*. Manning.

Kuhn, J. (2008). Train: Fit Predictive Models over Different Tuning Models. *RDocumentation*. https://www.rdocumentation.org/packages/caret/versions/4.47/topics/train

Lantz, B. (2013). *Machine Learning* with R. Packt

Mendoza, J. (2018). XGBoost en R. *RPubs*. https://rpubs.com/jboscomendoza/xgboost_en_r

Peng, R. (2015). *R Programming for Data Science*. Leanpub

Quick, D. (2010). APA Making Tables and Figures. *Colorado State University*. https://web.cortland.edu/hendrick/APA%20Making%20Tables%20and%20Figures.pdf

Rosero, R. [@RosanaFerrero]. (04 de mayo, 2023*). La ciencia es un conocimiento que entendemos tan bien que podemos enseñarlo a una computadora.* [Tuit]. Twitter. https://twitter.com/RosanaFerrero/status/1654003177111314433

Scheule, H., Rösch, D. (2022). Deep Credit Risk, Machine Learning in R.

Singh, A. (2018). Random Forest in R. *Data Science+*. https://datascienceplus.com/random-forests-in-r/

Stednick, Z. (s.f.). Machine Learning with R Datasets. *GitHub Page*. https://github.com/stedy/Machine-Learning-with-R-datasets

Swain, C. (1986). *The Elements Of Graphing Data*. University of Illinois Press

United State Government. (2017). Explore Census Data. *United States Census Bureau*. https://data.census.gov/

Vaughan, D. (s. f.) Introduction to dplyr. *Tidyverse*. https://dplyr.tidyverse.org/articles/dplyr.html

Wickham, H. (s.f.). Style Guide. *Advanced R. http://adv-r.had.co.nz/Style.html*

Wikipedia. (s.f.). Isaac Asimov, *Wikipedia la Enciclopedia Libre*. https://es.wikipedia.org/wiki/Isaac_Asimov

Zach, B. (2022). How to Use Create Data Partition. *Statology*. https://www.statology.org/createdatapartition-in-r/

Zumel, N., Mount, J. (2014). *Practical Data Science with R*. Manning

Zumel, N., Mount, J. (2021). Vtreat Package. *Cran – r. Avalaible Packages.*

https://cran.r-project.org/web/packages/vtreat/vignettes/vtreat.html