Trabajo Fin de Máster

Técnicas de búsqueda de Causa Raíz de un problema en un entorno IT

Natalia Romina Campos Pietanesi

Máster en Técnicas Estadísticas

Curso 2022-2023

Propuesta de Trabajo Fin de Máster

Título en galego: Técnicas de procura de Causa Raíz dun problema nunha contorna IT
Título en español: Técnicas de búsqueda de Causa Raíz de un problema en un entorno IT
English title: Techniques to search for the root cause of a problem in an IT environment
Modalidad: Modalidad B
Autor/a: Natalia Romina Campos Pietanesi, Universidad de Vigo
Director/a: Javier Roca Pardiñas, Universidad de Vigo; Director/a 2, Universidad 2
Tutor/a: Xose Ramón Sousa Vázquez, OPTARE SOLUTIONS SL.; Tutor/a 2, Empresa 2
Breve resumen del trabajo: Búsqueda de técnicas de RCA (del inglés Root Cause Analysis, que significa Análisis de Causa Raíz) que permitan encontrar solución a problemas asociados con el entorno de la integración de los sistemas de información y detectar que sistema(s) son más probables que sean causa de un problema. El problema puede tener varios elementos interconectados y se tiene que estimar cuál es el origen del mismo. A partir de las métricas recogidas de la evolución normal del contexto, evaluar las anormalidades y posteriormente hacer foco en los elementos anormales para poder llegar a conocer cuál es el que provoca el problema. A razón de ello, se desarrollan sendos algoritmos de selección de variables para los modelos matemáticos de clasificación utilizados, los cuales utilizan técnicas tales como bosques aleatorios, árbol de clasificación utilizando inferencia condicional y redes neuronales.
Recomendaciones:
Otras observaciones:

Don/doña Javier Roca Pardiñas, Profesor del Departamento de Estadística e Investigación Operativa de la Universidad de Vigo, don/doña Director/a 2, categoría 2 de la Universidad 2, don/doña Xose Ramón Sousa Vázquez, Arquitecto de Software de OPTARE SOLUTIONS SL., y don/doña Tutor/a 2, cargo 2 de Empresa 2, informan que el Trabajo Fin de Máster titulado

Técnicas de búsqueda de Causa Raíz de un problema en un entorno IT

fue realizado bajo su dirección por don/doña Natalia Romina Campos Pietanesi para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En Vigo, a 6 de Julio de 2023.

ROCA PARDIÑAS JAVIER - 77592181W
Firmado digitalmente por ROCA PARDIÑAS JAVIER - 77592181W
Fecha: 2023.09.25 17:31:23 +02'00'

El/la director/a:
Don/doña Javier Roca Pardiñas

El/la director/a:
Don/doña Director/a 2

Firmado por SOUSA VAZQUEZ XOSE RAMON - ***0115** el día 26/09/2023 con un certificado emitido por AC FNMT Usuarios

El/la tutor/a:
Don/doña Xose Ramón Sousa Vázquez

El/la tutor/a:
Don/doña Tutor/a 2

Firmado por CAMPOS PIETANESI NATALIA ROMINA - ***5215** el día 20/09/2023 con un certificado

El/la autor/a:
Don/doña Natalia Romina Campos Pietanesi

Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **el/la autor/a declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,...sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Agradecimientos

Quiero agradecer en primer lugar a mi madre y hermana, por todo el apoyo, confianza y paciencia con la que me acompañaron, en los períodos más complejos de este Máster. A mis tutores Javier y Xose, por el tiempo que dedicaron a ayudarme en la realización de este TFM, y por confiar en mi, dándome la libertad suficiente para investigar y crear los algoritmos necesarios. A mi padre, por brindarme su apoyo, desde la distancia. Al profesor Rubén, por introducirme en el mundo del aprendizaje automático, y todos sus buenos consejos al respecto. A mi supervisor Fernando, por todo el apoyo y la confianza con la que me acompañó, durante todo el proyecto. A Daniel, mi primer amigo en España, quien me acompañó durante la primera parte del Máster, y me ayudó a sobrellevar el desarraigo. A Luis, por sus buenos consejos y su capacidad de escucha. A mi abuelo Pepe, porque gracias a su portátil he podido realizar este Máster, se que estaría orgulloso de que haya venido a conocer su Galicia amada. Y finalmente a la Xunta de Galicia, por invitarme a realizar un máster aquí, permitiéndome darle una nueva perspectiva a mi vida profesional y personal.

Índice general

Resumen

Resumen en español

En el presente trabajo se realiza una búsqueda de técnicas de RCA (del inglés Root Cause Analysis, que significa Análisis de Causa Raíz) que permitan encontrar solución a problemas asociados con el entorno de la integración de los sistemas de información y detectar que sistemas, o sistema, son mas probables que sean causa de un problema. El problema puede tener varios elementos interconectados y se tiene que estimar cual es el origen del mismo. Luego, a partir de las métricas recogidas de la evolución normal del contexto, se debe evaluar las anormalidades y posteriormente hacer foco en los elementos anormales para poder llegar a conocer cual es el que provoca el problema. A razón de ello, y pensando en el hecho de que la empresa requiere re-calcular los modelos para los distintos data-set, ya que trabajan brindando soluciones a diferentes clientes, es que se desarrollaron sendos algoritmos de selección de variables para los modelos matemáticos de clasificación utilizados, los cuales utilizan técnicas tales como bosques aleatorios, árbol de clasificación a partir del p-valor y redes neuronales.

English abstract

Search for RCA (Root Cause Analysis) techniques that allow find solutions to problems associated with the integration environment of the information system and to detect which system or systems are the most probable cause for a problem. The problem could have several interconnected elements and we have to estimate which is the origin for it. Based on the metrics governed by the normal evolution of the context, we have to evaluate the abnormalities and then focus on the abnormal elements in order to get to know which one is causing the problem. Because of this, and thinking about the fact that the company needed to recalculate the models for different data-sets, because their work is to produce technological solutions for their different clients so, tree selection algorithms are developed for the mathematical classification models used, which use techniques such as random forest, classification tree based on p value and basic neural networks.

Prefacio

Este Trabajo de Fin de Máster (TFM), analiza datos pertenecientes a una red de telecomunicaciones. Dicha red tiene diferentes elementos interconectados, a lo largo de gran variedad de espacios geográficos. Se pueden encontrar, entonces, elementos como servidores, antenas, transmisores, y otros componentes que permiten establecer una conexión exitosa entre un servidor y un cliente final. Se dice que la conexión es exitosa cuando, tanto la comunicación de ida con el pedido del cliente, como la de vuelta, con la respuesta del servidor, han llegado en tiempo y forma.

La red que se está analizando es un sistema verdaderamente complejo, por varios motivos:

1. la gran cantidad de zonas de cobertura de cada red;
2. el hecho de que las redes brindan servicio a las operadoras;
3. el enorme número de clientes finales a los que se brinda servicio;
4. la cantidad de elementos vinculados entre sí, que la componen;
5. el tipo de estructura de red que vincula sus elementos. Esto hace que varios elementos dependan de determinados transmisores, box o grupos. Elementos que, a su vez, se encuentran interconectados entre sí para formar la estructura de red, tal como se verá más adelante en el desarrollo de este Trabajo de Fin de Máster.

Para poder controlar dicha estructura, existe un sistema de monitoreo, que basándose en las señales que van emitiendo los distintos elementos, genera alertas dentro de un registro. Dichas alertas reportan el estado de la alarma, y las principales características que presenta el elemento en ese momento. Estos datos, son recogidos por las variables que componen el data-set. Dentro de ellas se encuentra una, que recoge dichos valores, enumerados como Normal, Crítico, Mayor, etc. Pero el problema, es que estos valores, por sí solos, no determinan que exista un problema en el elemento alarmado. Otro problema es que, existe un gran número de alarmas que reportan un estado Normal, con lo cual, en realidad, no estarían indicando que haya un problema tampoco.

Este sistema, además, genera un registro de incidencias, en donde constan los servicios y clientes afectados por determinadas fallas reportadas. Pero estas incidencias no siempre son problemas a resolver, por un técnico especializado. Muchas veces, una incidencia se resuelve sola o de modo remoto, no requiriendo entonces intervención especializada.

Por ejemplo, cuando se llama al servicio técnico de la empresa operadora porque algo no anda bien en la red, y desde atención al cliente se indica como realizar el reset; y allí se resuelve el conflicto. En este caso, no es requerido que un técnico encargado de atender la red deba intervenir, el problema se resuelve desde la operadora. En este mismo sentido, una falla que se resuelve sola, puede ser aquella vinculada a una tormenta que genera interferencias en la red, dicha falla cesará cuando pase la tormenta.

Todas estas incidencias, o fallas, se van presentando en una pantalla, donde los encargados de controlar, deben verificar una por una, utilizando su experiencia y conocimientos sobre el tema, para dirimir cuál es un problema a resolver y cuál no. Una vez determinadas aquellas incidencias que son un problema, deben buscar entre las alarmas de elementos y servicios, cuáles pueden corresponderse con dichos incidentes, para rastrear la causa de los mismos e intentar resolver el problema.

Además, muchas veces suele suceder que el elemento que tiene el inconveniente, no genera alarmas, sino que lo hacen otros vinculados a él, en estos casos el problema puede deberse a una falla en la conexión entre los elementos alarmados y aquel que tiene el inconveniente. Cabe aclarar, que las redes para abarcar grandes espacios geográficos, se comunican a través de cables subterráneos. Esta situación complica aún más la posibilidad de ubicar la raíz del problema.

Más aún, la dimensión de estas bases de datos es tal, que una de ellas, para una zona determinada, generó unas 400 alarmas en apenas un día. Esto se vuelve mucho más complejo cuando se trata de varias zonas a la vez, o si la alarma afecta a varios elementos y servicios. Cabe destacar que las alarmas originadas en elementos se encuentran en un data-set separado, de las originadas en servicios. Por su parte, las alarmas de servicios, reportan alertas relacionadas con el funcionamiento de un servicio.

Este tipo de situaciones, hace de suma utilidad tener un modelo, que pueda colaborar con una visualización clara de la estructura de red que se tiene en frente, y de cuáles pueden ser los componentes "padre del alarmado" que estén generando el problema. Una vez que se haya definido si la alarma corresponde a un problema o no. De esta forma, se facilita la tarea de quien está controlando, al indicarle las posibles causas del inconveniente. Dicho modelo, se basa en los datos históricos con los que ha sido entrenado, para analizar la información brindada por la alerta a considerar.

Se busca, entonces, dilucidar cuáles son los componentes causales de las incidencias que se han presentado en el pasado. Relacionarlos con el hecho presente en estudio, y exponer así, aquellos elementos que tienen la mayor probabilidad de ser los responsables del incidente, con sus probabilidades asociadas.

Este tema es más complejo aún, debido a la gran cantidad de datos faltantes o duplicados, que fueron detectados luego de realizar un proceso de depuración de los mismos. Esto hace que dicha información, sea poco profunda, y escasa en cantidad, para una tarea de tal magnitud. Por ende, se ha decidido apoyar el estudio sobre bases de datos sintéticas, generadas por la empresa a partir de la información real, pero con mayor profundidad (más niveles de interrelación entre los componentes de la red), y más volumen también.

En resumen, con este trabajo de fin de máster, lo que se busca es poner de manifiesto la red existente, para poder tener una mejor gestión de las alarmas. Explorar en profundidad los datos para encontrar información relevante que colabore con la toma de decisiones. Y con ello colaborar con la mejora en la gestión de los recursos.

Mediante el modelado de los datos, se busca realizar predicciones sobre cuál puede ser el componente que origina la alerta. Esto es de suma importancia, porque si finalmente se clasificó la incidencia como un problema a resolver, esta información hará mucho más fácil el siguiente paso, que es ubicar la **Causa Raíz** del problema. Esto último, se hará sobre la base de datos que se espera obtener en un futuro, que informará cuál fue la **Causa Raíz** de cada incidencia. Esto permitirá, analizar como fue la gestión de esos problemas, y sus correspondientes estados posteriores, para controlar si dichas respuestas fueron exitosas. Completando de esta forma, todos los pasos recomendados para un correcto estudio de **Causa Raíz**.

Es importante destacar que la variable respuesta es categórica, nominal, no ordinal, con una gran cantidad de categorías dentro. Estas categorías indican el nombre de cada elemento que compone la red. Las variables explicativas, en quienes podríamos apoyarnos, son también del tipo categórica, algunas ordinales y otras no, es por esto que se buscó utilizar modelos, que permitieran manejar grandes cantidades de información de este tipo.

Por todo lo expuesto, es que interesa realizar un análisis de **Causa Raíz**, (del inglés Root Cause Analysis, RCA). Dicho análisis, se encuentra dividido en más de una etapa, debido a la complejidad de los datos, la cantidad de elementos involucrados, y la dificultad para obtener la información requerida.

Dentro de este trabajo de fin de máster, se efectúa un análisis preliminar con herramientas de RCA, para extraer información relevante para el estudio.

Se procede a una búsqueda exhaustiva de las variables óptimas para generar un modelo de bosques aleatorios, y se genera dicho modelo.

En este punto surge, desde la empresa, el pedido de confeccionar varios modelos para confrontarlos entre sí, y evaluar su desempeño. Se hicieron, entonces, dos modelos más, ejecutando nuevamente el análisis de variables a considerar, hiperparámetros importantes y demás cuestiones.

Al presentar los modelos mencionados, se plantea la necesidad de probarlos, no solo en ese conjunto de datos, sino, en otros conjuntos de datos generados por diferentes fuentes. Es conveniente destacar que, cada modelo es apto para el tipo de datos que fue entrenado, pero no existen, aún, modelos que sean universalmente buenos para todos los datos.

Dicha inquietud, provoca la idea de generar un algoritmo que, de manera automática, seleccione las variables predictoras más adecuadas para cada modelo. Con dicho algoritmo, se podrían generar, mucho más rápido, los distintos modelos sobre los diferentes conjuntos de datos a modelar, para luego evaluar y comparar su desempeño. Esto permitiría ahorrar mucho tiempo y esfuerzo en esa primera fase, de selección de las variables, para obtener un mejor primer ajuste. Pudiendo, incluso, invertir más tiempo en realizar un refinamiento de parámetros del modelo, y en la evaluación del mismo, para mejorar su calidad de ajuste. La calidad del modelo es fundamental, en todos los ámbitos. Con la construcción de un modelo se busca predecir los sucesos, pero además interesa, que dicho modelo sea fiable y robusto.

En este caso, el modelo final, permite saber cuál es el elemento padre del alarmado, quien probablemente estaría relacionado con la **Causa Raíz** que esté originando el problema, y generando la incidencia.

En síntesis, mediante la creación de este algoritmo de búsqueda automatizada, que no existía para este tipo de modelos, se logran estudiar más combinaciones de variables en menor tiempo. Esto mejora considerablemente la eficiencia del estudio, y permite contar con más tiempo para perfeccionar la calidad del modelo en cuestión.

Capítulo 1

Conociendo el entorno de trabajo

A lo largo de este trabajo, se presentarán diferentes herramientas y algoritmos, que fueron de utilidad para analizar la estructura de los datos. De esta manera, se busca sentar las bases para un correcto análisis de **Causa Raíz**, de un problema en un entorno IT.

Primeramente, se deben definir algunos conceptos y herramientas, que sirven de base para este estudio. Se los explicará, sobre el caso puntual en el que se ha estado trabajando. También, se expondrán los datos considerados, y sus principales características.

1.1. Sobre Los Datos

Inicialmente, se contaba con bases de datos reales, pertenecientes a una operadora relacionada con el mundo de las telecomunicaciones. El contacto con la operadora, se realiza a través de una empresa intermediaria, encargada de facilitar todos los recursos necesarios para realizar este TFM. Esta empresa es una consultora, que se dedica a generar soluciones de software para sus clientes, en el ámbito IT.

Aquellos primeros datos, comprendían una franja temporal muy amplia, entre enero del 2018 y mayo del 2022. Dichos datos son generados por un sistema de monitoreo, encargado de vigilar el correcto funcionamiento de la red de telecomunicaciones. Dicho data-set, presentaba las características de cada una de las alarmas o incidencias, que se suscitaran en un determinado período. Dichas características, son recogidas por las diferentes variables que integran el dataset, en formato de columnas, y las filas representan cada una de las alarmas o incidencias registradas.

El contenido de las mencionadas bases de datos, será explicado con mayor detalle en el capítulo 2 de este TFM, en la sección *Minería de Datos*, 2.1. Allí además, se incluye una tabla, que evidencia el contenido de cada una de las variables.

Se contaba con varios dataset, de diferentes torres, denominados: Torre España, Santiago y Ares. Dichos nombres, hacían alusión a la zona de la red a la que pertenecían los datos.

Las bases poseían una gran cantidad de registros, pero no contaban con lo esencial, una variable respuesta, que figurara como tal y permitiera predecir la **Causa Raíz**. Tampoco se contaba con una variable, que dejara explícita la relación entre unos elementos y otros.

En dichas bases, se encontraban separados los datos generados por incidencias, de los generados por alarmas; y no evidenciaban tener ninguna variable que permitiera vincularlas. Este dato era esencial, porque las incidencias reportan incidentes desde el punto de vista del servicio afectado, esos ya son problemas (que pueden requerir o no intervención). En cambio, las alarmas, reportan alertas en algún elemento o servicio, al cual le sucede “algo”. Ese “algo”, no necesariamente expone un problema. Esto lo evidenciaba que si el estado de alarma era normal, solo estaría indicando que a pesar de que sucede “algo”, la red sigue en estado normal.

Inicialmente, se trabajó sobre estas bases de datos reales, donde existían además, datos faltantes, denominados NA, quienes habitualmente provocan muchos problemas en las funciones de R. Esto condujo a realizar un análisis más exhaustivo de aquellos datos, para realizar una correcta reparación o imputación de los mismos. Dichos datos, en general, se debían a una mala recolección por parte del sistema de información, encargado de monitorizar las redes de telecomunicación. En otros casos, provenían de errores humanos en el procesamiento de los datos. Errores generados cuando se pasaron los datos de una base a otra, para generar un resumen de ellos, por ejemplo. Los NA se presentaban, habitualmente, en aquellos datos que estaban duplicados. Esto provocó que al depurar las bases, que en principio parecían grandes, estas terminaban aportando muy poca información. En dichos casos, lo que se hizo fue analizar a que variables pertenecían dichos NA, y con que elementos se relacionaban para poder resolverlos.

Los nombres de las variables en estudio hacen referencia a datos técnicos, volviendo más compleja su comprensión. Este hecho, motivó la solicitud de una codificación que explicara las variables. Cabe destacar que, dichas variables, no son numéricas sino categóricas, y contienen además, una gran cantidad de categorías en su interior.

En cuanto a la estructura de los elementos presentes en la red, esta era incompleta, con poco volumen, y su profundidad de interconexión muy baja. Esto hacía el trabajo muy sencillo para los árboles de decisión, porque tenían poca cantidad de datos a clasificar en pocas categorías, pero poco aplicable sobre datos futuros. La falta de datos, mencionada anteriormente, se relaciona con el requerimiento de grandes volúmenes de memoria, para almacenar dicha información. Esto es un inconveniente a la hora de tener acceso a una base de datos grande, que cuente con la profundidad requerida, para que los modelos de Machine Learning puedan aprender esa topología del dato que nos interesa. Estos modelos, son conocidos por su capacidad para modelar grandes cantidades de datos, y por tanto requieren del llamado Big Data, para poder trabajar. Esta estructura, que describe la topología, es esencial para poder rastrear la posible **Causa Raíz** de las incidencias.

En una primera etapa del estudio, se analizó sobre dichos datos, cuáles eran las causas del 80 % de las incidencias, utilizando herramientas como el Gráfico Pareto. Se realizó un análisis exploratorio, para generar información relevante para la toma de decisiones en ese sentido. A la empresa, le interesaba saber que elementos estaban presentando mayor cantidad de incidencias, y que características tenían. Datos muy útiles a la hora de actuar como consejero para sus clientes.

En cuanto a las alarmas, es importante destacar, que no solo se cuenta con aquellas que han reportado inconvenientes. Para poder determinar si una alarma, o conjunto de ellas, es un problema a resolver, deben confluir varias características, que al momento del estudio estaban pendientes de definirse. Pero existen algunos indicadores a tener en cuenta, como una variable llamada “AlarmState”, que refleja el estado de la alarma. Dicha variable, oscila entre los valores Normal, Warning, Minor, Major o Critical. Estos valores se refieren a cuál es el alcance de esa alarma, no solo al estado del elemento en si. Además, hay otra variable, denominada “Type”, que indica si esa alarma es nueva, o si fue escalando o bajando de nivel, y de que nivel proviene. Los niveles son los estados de alarma, citados anteriormente.

Los nombres de las variables, cambian según el sistema que se utilice para recabar la información, y el cliente que haya brindado la base de datos para su análisis. Dato, no menor, a tener en cuenta para la creación del algoritmo, y principalmente, del modelo. Si los nombres de las variables cambian, el modelo ya no puede trabajar con ellas, y por tanto, debe re-calcularse.

Estos inconvenientes resultaron en la creación de una base de datos sintética, que apoyándose en la base real, presentara una mayor cantidad y profundidad de datos. El objetivo es generar y probar sobre ella distintos modelos. Estos modelos, ayudarán a comprender la relación subyacente entre elementos y alarmas. A partir de allí, se vincularán dichas alarmas con las incidencias, para realizar predicciones sobre el OwnerName, al cual se le atribuye dicha incidencia. Entendiendo por OwnerName, el elemento padre del alarmado.

1.2. Estructura y forma de la red que subyace en nuestros datos

Tal como se comentó en la sección anterior, en el entorno IT se les llama incidencias y alarmas, a cada uno de los eventos que conforman los datos que se están analizando. Así, el sistema de monitoreo va reportando diferentes estados de los elementos que componen la red. Dichas alarmas, pueden presentar un estado normal, warning, critical, o mayor, según los clasifique el mismo sistema de recolección de datos. De aquellos que no exhiben un estado Normal, algunos requieren intervención para su resolución, y otros se resuelven solos.

Lo correcto en este tipo de entornos, es analizar las alarmas, para tratar de identificar su origen, y sus principales características. Estas características son descritas, para cada uno de los registros, por las variables que conforman el data set en estudio.

Con los datos referidos, se construye la topología del dato o elemento. Este es el mapa, donde constan las relaciones de dependencia, entre los diferentes componentes de una red de telecomunicaciones. Allí se puede ver, que hay algunos elementos que dependen directamente de otros. Esto hace que si falla el "padre", entonces es probable que sus "hijos" fallen, y por eso vemos alarmas generadas por ellos. Aunque también existen otros casos, en donde el problema está solamente en el "hijo", o en la comunicación entre unos y otros, etc. Estas relaciones de "padre e hijo", tampoco son directas. Existen elementos que funcionan de intermediarios entre unos y otros, hecho que incorpora mayor cantidad de niveles, y por consiguiente mayor profundidad a la red, haciendo aún mas complejo el estudio en este contexto. Desde la empresa representan la estructura, presente en la base de datos en estudio, como una red integrada por elementos de tipo box, elementos de tipo group, y transmisores. Dicha representación, se puede interpretar de la siguiente manera:

- Hay varios elementos de tipo box (del inglés, caja), sobre los que se ofrecen servicios. Esto significa que se tienen elementos de tipo box, que poseen un nombre y un id para identificarlos, por ende puede aparecer una alarma sobre ellos. Se pueden asimilar con el concepto de una caja, o una plaqueta de computadora, encargada de brindar un servicio determinad. Pero, además, en ella se encuentran insertos elementos, que hacen posible su tarea.

- De los elementos tipo box cuelgan elementos de tipo group (del inglés, grupo), por tipo de transmisor. Son grupos de transmisores que se identifican bajo un nombre e id específico, según su tipo. Los diferentes tipos de elementos group, trabajan en conjunto para lograr la tarea requerida por el elemento de tipo box, quien los contiene. Los elementos group pueden ser OwnerName (elemento padre), de más de un elemento box.

- De los elementos de tipo group cuelgan transmisores. Los transmisores también poseen un nombre e id único que los identifica. Aquí ya se hace referencia a los elementos en sí. Estos son los encargados de realizar tareas puntuales. Dichos elementos trabajan, en conexión con otros elementos de su mismo grupo y de otros grupos, para satisfacer los requerimientos del elemento de tipo box que los contiene, y así poder brindar el servicio.

Esta división colabora con la visualización del origen de las alarmas. Existen fallas de los elementos tipo box, que no se deben a fallas de los grupos, o de los elementos que los componen. Si los elementos de tipo box accionan la alarma, dicha alarma puede indicar que el problema está en dicho elemento box, en sus elementos componentes, o en elementos no monitorizados. Si los transmisores y grupos que componen al elemento box no presentan alarma, o todos presentan una alarma general, eso podría indicar que, efectivamente, el problema está en el elemento de tipo box, o en alguno de los elementos no monitorizados. Un elemento no monitorizado causal de falla, puede ser es el clima.

El hecho de tener alarmas que se reportan en elementos de tipo group, estaría indicando que el problema ya no es localizado en un elemento transmisor, sino que hay un conjunto de ellos afectado. Habrá entonces, varios servicios afectados, varios box que presenten alarmas, y también, sendos elementos transmisores que componen el grupo, que presentarán alarmas.

Finalmente, aquellas fallas que accionan alarmas en los elementos de tipo transmisor, permiten rastrear su alcance hasta los elementos transmisores que sean OwnerName de los alarmados. Esto se analiza en función del tipo de falla que presenten dichos elementos. Aunque también en este caso, aparecerán alarmas en los elementos de tipo box.

El modelo entonces, debe indicar cuál es el elemento padre del elemento alarmado. El cual, para facilitar el estudio, se encuentra expuesto en la columna OwnerName.

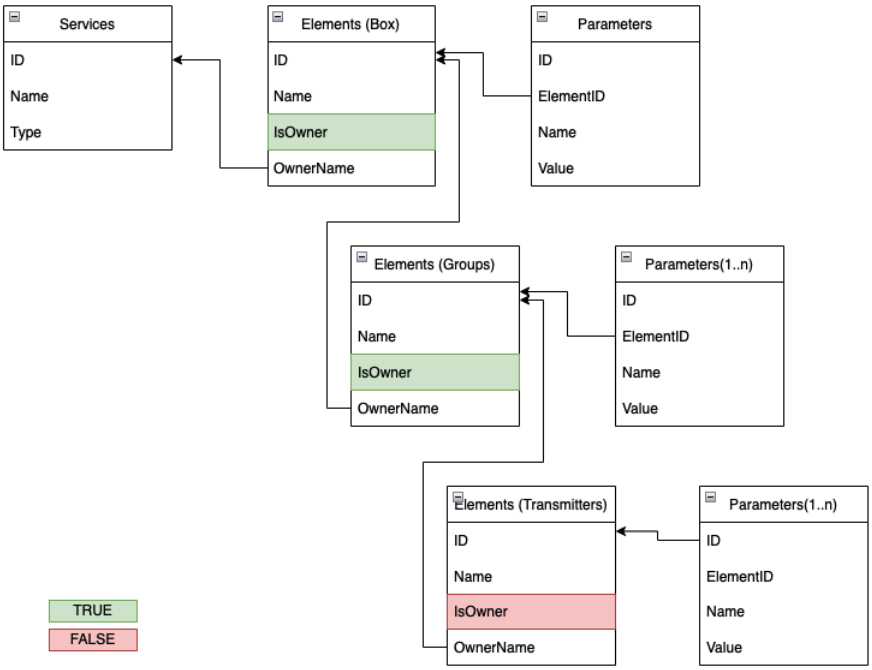

Figura 1.1: Estructura simplificada de una Red de Telecomunicaciones (Proporcionado por la empresa)

Posteriormente a este trabajo se creará un algoritmo, que asocie las alarmas con cada incidencia, y así obtener una cierta cantidad de padres probables para ella. Esto permitirá exhibir cual es el padre **Causa Raíz** más probable, para esa incidencia. Dicho algoritmo, también informará de otros posibles padres que puedan ser considerados **Causa Raíz**, y sus probabilidades de ocurrencia.

Como se mencionó anteriormente, los datos estudiados son de tipo categórico, nominal, no ordinal, y no numéricos, (es texto). Tanto la variable que se utiliza como respuesta, como las variables predictoras utilizadas en el modelo, son de este tipo. Todas las variables contienen una serie de subcategorías, también llamadas clases o niveles. La cantidad de niveles o categorías, presentes en cada variable, oscila entre 1 y 837. En la base sintética, tal cual fue creada, se cuenta con 28 variables. Una variable respuesta, y 27 variables que pueden usarse como explicativas. Se cuenta con 12 mil datos o registros, aproximadamente en dicha base. Luego de la depuración de registros duplicados, tratamiento de NA, y filtrado, dicha base alcanza los 10mil registros, y queda conformada con 26 posibles variables explicativas.

Como se puede ver el tema a abordar es complejo, no solo por ser una red de elementos interconectados, sino porque además, no se cuenta con la información completa de las incidencias: no se tiene una forma de relacionarlas con las alarmas, ni se conoce como se resuelven las incidencias, una vez reportada y detectada su causa. En este momento, se cuenta con una base de información de alarmas, donde constan las relaciones entre los elementos. A partir de las alarmas presentes en el data-set, se puede inferir que la **Causa Raíz** de dicha alarma, se encuentra en el elemento padre del elemento alarmado, el OwnerName.

Como se mencionó anteriormente, se busca comprender correctamente esas relaciones, para luego poder predecir en caso de aparecer una nueva incidencia, cuál es su **Causa Raíz** más probable. Además, se pretende ver cuáles de esas incidencias efectivamente se han transformado en problemas. Por tal motivo, se procede a analizar, en primera instancia, si el pasar de un estado normal a otro de los enumerados, como puede ser critical, major,o warning; nos indica que se ha convertido en un problema. O si, por el contrario, este hecho no contribuye a que la incidencia se transforme en un problema a resolver. Dado que, se cuenta con muchas incidencias que se terminan resolviendo por si solas, o mediante un reinicio. Estas tareas de reinicio pueden haber sido programadas previamente, o pueden ejecutarse a distancia en el momento que se crea necesario, evitando así la presencia física del técnico en el área del problema. Otras, incluso, solo representan un aviso de que, por un error de un elemento, se va a conmutar automáticamente al elemento de reserva. Para algunos servicios existen este tipo de sistemas, en donde si uno de los transmisores falla se posee uno de reserva, al cual se conmuta automáticamente. Por tanto se verá una alarma de la falla del transmisor, por cada uno de los servicios en los cuales este servía de apoyo. Luego de dicha alarma, aparecerá otra, que notifica que se enciende el elemento encargado de realizar la conmutación. A continuación, habrá otra que indica el paso al elemento de reserva. Luego de esa, aparecerían los elementos dependientes de estos dos, con sendas alarmas que indican su regreso a estado normal, y allí terminará dicho conflicto.

Antes de realizar el estudio de estas relaciones, se realizó una primera instancia sobre los datos reales originales, con Herramientas de análisis de **Causa Raíz**. Algunas de las técnicas utilizadas fueron, el análisis de Pareto, tablas de contingencia, entre otras.

Esta primera instancia permitió ver, que la base de datos real,con la que se contaba, no era la adecuada para realizar este tipo de estudio. Hecho que, una vez extraída la mayor información posible de dicha base, suscitó la generación de la ya mencionada base sintética, para tener mejor representadas las interrelaciones entre los elementos. Desde la empresa, se aseguraron de que dicha base contara con una estructura similar a la original, con la intención de que esta investigación fuera aplicable sobre datos futuros. Esta primera instancia, realizada sobre los datos originales, está descrita a modo de resumen, en la primera sección del apartado denominado *Minería de Datos*, 2.1, de este trabajo de fin de máster. Para una versión más detallada del mismo, dirigirse a los anexos del presente documento, apartado *Análisis exploratorio de los datos originales*, A.

Durante la segunda instancia de este estudio, se han generado diferentes modelos matemáticos de clasificación. Estos modelos de clasificación, como se verá más adelante en la sección *Modelando la estructura subyacente*, 1.4, serán utilizados para clasificar adecuadamente las alarmas, en relación con cada una de las categorías que componen la variable respuesta que nos interesa predecir, denominada OwnerName. Todo esto, apoyándose en la información que nos brindan las variables explicativas presentes en el data-set. Dichas variables, describen las características que presenta el elemento alarmado. Los modelos de clasificación generados, que serán explicados en detalle más adelante en las siguientes secciones, fueron los siguientes:

- Se ha construido un árbol de clasificación, que emplea el método de inferencia condicional, para definir que variables incorporar en cada uno de los cortes del árbol.

- Se ha construido un modelo de bagging, que en vez de realizar un solo árbol realiza muchos, y "promedia" los resultados de todos ellos. Trabaja por validación cruzada, y por tanto incorpora de esa forma, aleatoriedad en las observaciones.

- Se ha construido un modelo de bosques aleatorios, que no solo incorpora aleatoriedad en las observaciones, sino también en las variables involucradas. Este hecho se logra, al ir eligiendo la variable a incluir en cada corte del árbol, a partir de un subgrupo de dichas variables, seleccionado aleatoriamente.

- Se ha construido un modelo de clasificación que utiliza redes neuronales, para analizar las relaciones entre las variables. Este modelo realiza un proceso recursivo de aprendizaje, sobre los datos que le han sido proporcionados.

Para lograr los modelos, que serán presentados en las secciones siguientes, se fraccionaron los datos en dos muestras, una para entrenamiento, y otra para test. La muestra de entrenamiento es la que se ingresa al modelo para que este aprenda con esos datos, y genere la estructura de clasificación para los mismos. Con la muestra de test, se prueba el funcionamiento del modelo, para evaluar su precisión al predecir la respuesta. La muestra de test, contiene datos que el modelo desconoce, de esta forma se busca obtener índices de precisión más reales, por que la situación se asemeja al momento en que el modelo deba trabajar con datos nuevos. Los datos elegidos para constituir las muestras, fueron los mismos para todos los modelos calculados, para poder comparar su rendimiento en condiciones similares. De esa forma, los índices de efectividad de los modelos, no se verían afectados por la aleatoriedad en la selección de las muestras.

Es importante destacar, que se ha tenido que realizar una gran cantidad y variedad de modelos previos. Esa tarea permitió analizar cual era la técnica más apropiada, y dentro de ella, cual el modelo más adecuado para dichos datos. Esto se debió, a la gran cantidad de posibles variables predictoras que tenía el dataset, a que existía información solapada entre las distintas variables, y a que no se contaba con demasiada información sobre los datos, que nos pudiera indicar que variables predictoras podrían ser las más idóneas para el modelado. Pese a esta dificultad, se han alcanzado modelos con buenos indices. Dichos índices, reflejan la exactitud del modelo, al predecir una nueva observación. En ellos, se utilizan solo 4 o 5 variables predictoras, para el caso de los modelos manuales. En los modelos creados a partir del algoritmo automático, se trabaja con hasta 7 variables predictoras. Además, no solo se buscó cual era la mejor combinación de variables para el modelo, también se invirtió tiempo en refinar los parámetros, para cada uno de los modelos elegidos.

Estas herramientas, y otras que fueron de utilidad para la elaboración del presente documento, se presentan y detallan a continuación en el siguiente apartado denominado *Herramientas para el análisis de RCA en un entorno IT* 1.3 . Allí se resaltan aquellos aspectos teóricos, que serán de utilidad para una correcta comprensión de los pasos dados. Dichos pasos, serán explicados en detalle más adelante, en el apartado *Investigación Paso a Paso* 2.

1.3. Herramientas para el análisis de RCA en un entorno IT

Al hablar de Análisis de **Causa Raíz** (del inglés Root Cause Analysis - RCA), se hace referencia a buscar cual o cuales son las causas que están originando los problemas que queremos resolver, o los síntomas que estamos viendo. Dichas causas son de interés, porque se busca modificar o mejorar alguna situación, que está siendo objeto de estudio en ese momento. Por tal motivo, el desarrollo de este tipo de análisis, es muy importante para el crecimiento de una organización. Se la denomina **Causa Raíz**, porque en caso de que el estudio conduzca, como es habitual, a mas de una causa asignable, interesa aquella que es el verdadero origen. Se busca actuar sobre dicha causa, para mejorar la situación analizada, y que dicha mejora sea consistente perdurando en el tiempo. Por ello, es vital realizar cada uno de los pasos de forma consciente y firme, para alcanzar el resultado buscado. Para mayor información al respecto, puede consultarse *Ovalles Acosta, et al.(2017)* [11].

En un entorno IT, se trata de ir rastreando a partir de las incidencias, que son los síntomas del problema, cuáles son las posibles causas, que estarían generando estos efectos en el sistema. Para ubicar la causa, hay que unir las incidencias con las alarmas. Esto se puede hacer vinculándolas por momento en que aparecen, por la zona, por el servicio afectado, etcétera. En base a la información de las alarmas, se rastrea utilizando la topología, (el "mapa" de la red), cuáles son los elementos que están reportando dichas alarmas. Una vez ubicados, se debe encontrar el OwnerName de cada uno de ellos. Este último paso, se realiza porque puede que la **Causa Raíz**, sea un corte en la comunicación con ellos, y en ese caso, no presentarían alarmas, pero estarían involucrados en la **Causa Raíz** del problema.

Por todo lo expuesto, es vital contar con alguna herramienta que facilite el acceso a la topología del dato, y así poder acceder a las posibles causas el problema. Este primer paso de acceso a las posibles causas, en otros entornos, se consigue mediante herramientas como:

- El camino de los 5 porque: en sencillas palabras sería, se pregunta como mínimo 5 veces el porque de una incidencia, hasta hallar su posible raíz.
- El brainstorming (lluvia de ideas): aquí se van proponiendo diferentes causas, que pueden estar vinculadas al síntoma o efecto que se quiere resolver o mejorar. En este caso, lo más importante es la no censura, para dar paso a la creatividad, y así alcanzar las respuestas buscadas.
- El gráfico Ishikawa, unido a las dos técnicas anteriores: este gráfico, también llamado gráfico de espina de pescado, por su apariencia. Posee en la cabeza del "pescado" el incidente a analizar, y la "espina" son las posibles causas, clasificadas por grandes temas. Resulta muy útil para asociar múltiples causas a una sola incidencia, problema o síntoma.
- El gráfico Pareto: se basa en la regla del 80-20. Esta regla indica que el 20 % de las causas probables, son responsables del 80 % de los problemas analizados. Dicha técnica busca, entonces, en función de la cantidad de problemas y sus causas, cuál es ese 20 % de las causas, responsables del 80 % de los problemas.

Estas son algunas de las técnicas más conocidas. Cuando este análisis se apoya en una buena base de datos, donde consten los registros de cuales han sido las causas asociadas en el pasado, estas técnicas contribuyen a realizar un efectivo estudio de RCA, tendiente a encontrar las verdaderas causas o **Causa Raíz** de las incidencias en cuestión. Con dicho análisis, se busca resolver los problemas "de raíz" para que no se vuelvan a presentar, o prever dichos sucesos, para evitar males mayores en el futuro. Los datos históricos, pueden ser estudiados mediante modelos matemáticos de clasificación, que permitan vincular las causas que han sido asignadas en el pasado, con los efectos que se están observando y se quieren mejorar. Existen otro tipo de modelos, llamados de regresión, pero en nuestro caso no son aplicables por el tipo de variable que se está manejando.

Las herramientas, descritas anteriormente, también son utilizadas dentro de la metodología 6 Sigma, que busca mejorar la calidad de los procesos. Dichas herramientas, se encuentran dentro de la etapa Definir, incluida en el ciclo denominado DMAIC (Del ingles DEFINE=Definir, MESSURE=Medir, ANALIZE= Analizar, IMPROVE=Mejorar y CONTROL=Controlar). El objetivo de la fase definir, es reunir a todas las partes interesadas, juntar sus conocimientos e ideas para diseñar el proceso en cuestión, establecer un objetivo común, y definir cómo contribuye cada parte, (o el papel que cada parte tiene), en la solución del problema. Para mayor información, puede consultarse *Ishikawa (1976)*[7].

Este tipo de herramientas de RCA, se usan muy a menudo en los entornos más variados, como por ejemplo en la medicina. Dichos entornos son complejos porque cuentan con muchos agentes involucrados, encargados de realizar muchas tareas, que van incluso encadenadas entre sí. La ejecución exitosa de esas tareas, depende además, de una respuesta rápida, y de contar con los elementos adecuados en el momento preciso. Dado que estas técnicas de análisis de **Causa Raíz**, permiten abordar los problemas de una manera clara e irlos desmenuzando. Tanto en cuanto a tareas que los componen, como a colaboradores responsables de llevarlas adelante. Es importante destacar, que estos pasos no se realizan con la idea de buscar culpables, sino de ahondar lo más posible en el conocimiento del problema que se quiere resolver, y su entorno, para alcanzar una comprensión completa del mismo. Etapa fundamental, para dar con las soluciones óptimas, y evitar la reaparición de los problemas. Puede consultarse *Ruiz-López, et al.(2005)*[13].

También, se las suele combinar con modelos de aprendizaje automático o machine learning. En empresas de manufactura de productos, por ejemplo, se usan para controlar los fallos, y analizar los desvíos en la calidad de los productos. Esto se debe a la cantidad de información presente en dichos entornos. Dado que, en entornos automatizados, la información es brindada por las propias maquinas encargadas del proceso. Puede consultarse *Lokrantz, et al.(2018)*[10].

Este último hecho, es uno de los factores, por los cuales se plantea la necesidad de trabajar en la nube con dichos modelos para poder garantizar su ejecución durante todo el tiempo requerido. También se contribuye, de este modo, a facilitar el manejo de la información. Estos modelos trabajan con grandes cantidades de información interrelacionada, y el manejo en la nube hace más sencilla, no solo su ejecución, sino el acceso a la información, y la interrelación entre los responsables del proceso. De esta forma todos pueden colaborar en el análisis, sin duplicar información ni sobrecargar sus ordenadores, por ejemplo. En relación a este tema puede consultarse *Josefsson (2017)* [8].

El factor determinante en este tipo de entornos IT, es su complejidad. Esta se ve reflejada en parte por los estudios en el campo de la medicina, en relación a la cantidad de elementos, sumados a la cantidad de personas interrelacionadas, y la velocidad de respuesta requerida. Así como también, en los estudios que versan sobre empresas de manufactura, en cuanto al nivel de automatización, y la cantidad de elementos relacionados, que por estar automatizados, exigen una velocidad alta de respuesta, para evitar pérdidas mayores. En el ámbito IT estudiado, el trabajo se hace más difícil, debido a la complejidad de la red, la velocidad con la que se deben responder a estos incidentes y resolverlos, y la cantidad de incidencias que se reportan por día. Se requiere, entonces, de herramientas que faciliten la comprensión de la topología del dato. Ellas permitirán analizar, cuáles son las componentes que están fallando con mayor asiduidad, para realizar tareas tendientes a resolver el problema desde su raíz, para que no vuelva a aparecer.

Actualmente se trabaja haciendo un “camino de hormiga”, siguiendo cada incidencia para ver en donde termina, o en donde comenzó. Esta tarea se basa en los conocimientos y experiencia del personal responsable de resolverlas. Estos encargados, realizan tareas de “parche” para salvar la situación. Estas acciones, muchas veces, resultan ser adecuadas para que no vuelvan a aparecer los problemas, pero otras veces no son suficientes. Es por eso, que en estos casos, puede ser de utilidad generar modelos matemáticos, que permitan clasificar y organizar los elementos de la red. Establecer de manera más clara esas relaciones, permitirá comprender con mayor velocidad la estructura, y dar con el o los elementos que pueden ser los responsables de la incidencia, de manera mas rápida y efectiva.

Otra complicación, presente en el caso de estudio, es que no se cuenta con registros de la resolución de las incidencias. Por ende, tampoco se conoce cuál fue su causa en el pasado. Esa sería la segunda etapa de este análisis. Consistiría en generar un registro, de las acciones tomadas para resolver esos incidentes, y sus posibles causas. Esto sería un punto a abordar, en un trabajo futuro. Ese trabajo debería recabar dichos datos, vincularlos con este primer estudio, y tener así, un análisis más completo de **Causa Raíz**.

Enlazada con ese estudio futuro, se encuentra la última fase del análisis de **Causa Raíz**, la retroalimentación. Allí se realiza un control de esas acciones tomadas, para saber si sus efectos fueron los deseados o no. Se analiza: si se ha logrado resolver el problema, si se lo ha mejorado, se lo ha dejado igual, o en el peor de los casos, ha empeorado. A partir de allí, se fijan las nuevas acciones, tendientes a mejorar la situación, buscando explorar otras opciones posibles para resolver el problema. Una vez resuelto, se dejará explícito en un manual de procedimiento, para que si en el futuro se vuelve a presentar el problema, poder resolver en base a estas experiencias, de manera mas precisa.

Todos estos puntos expuestos, son los que justifican este trabajo de fin de máster, que en este caso se enfocará en el paso 1 del análisis de **Causa Raíz**. Se busca establecer las relaciones entre los distintos componentes de la red, para que cada vez que se presente una alarma, con determinadas características en las variables predictoras, se pueda hallar el OwnerName de dicha alarma. Además, esto contribuye para que la empresa pueda aconsejar algunas acciones, tendientes a mejorar esa situación, en base a su experiencia en el tema. Estas acciones podrían ser, realizar un mantenimiento preventivo, o un cambio preventivo de determinados elementos de la red, etcétera.

1.4. Modelando la estructura subyacente

Como ya se mencionó anteriormente, la estructura de nuestro entorno IT es compleja. Esto motiva el uso de herramientas de modelado matemático, para establecer la relación de la variable respuesta con las distintas variables predictoras presentes en nuestros data-set.

Cuando se hace referencia a un modelo matemático aditivo, para el caso de variables categóricas, este es algo diferente al que se conoce con variables numéricas, aunque se puede explicar de forma similar. La idea principal, es ir incorporando una a una las variables que se consideran más importantes, para la predicción de la variable respuesta de nuestro modelo. Se asume, de esta manera, que los efectos de cada una de ellas se van sumando entre si, por eso se habla de un modelo aditivo.

Para el caso de variables categóricas, se va analizando cada categoría de la respuesta, para señalar que variables predictoras y con que categoría se presentan en cada caso. Esto es, porque se entiende que la conjunción de dichas categorías, da como resultado una respuesta determinada. Durante la construcción del modelo se analiza, si en la variable predictora R se presenta tal categoría **A** $\boldsymbol{y}$ en la variable predictora S se exhibe la categoría **B**, entonces, el valor más probable que se obtendrá en la respuesta será **V**, porque es aquel que más proporción de datos posee, cuando se presentaron esos valores en la muestra de entrenamiento. Ese $\boldsymbol{y}$ funciona como sumatorio de las categorías de las variables predictoras. En el modelo, cuando ingresa un nuevo dato, se tiene entonces, a todas las categorías que pueden presentar las variables predictoras, donde si su valor está presente en el nuevo dato entonces esa categoría se multiplica por 1, y sino por 0 para anular su efecto. Así nacen las llamadas variables dummy, que representan el parámetro que acompaña a cada una de las categorías, de cada una de las variables predictoras.

Una vez construido el modelo, si es por árbol de decisión por ejemplo, se tendrán unos valores asociados a cada una de las categorías de las variables predictoras. Estos valores, representan la proporción de datos de entrenamiento que poseía dicha categoría, al presentarse en conjunto con otras. De esta forma el modelo estimará, en función de un nuevo dato, que valor tomará la variable respuesta, en base a cual fue la clase mayoritaria en la muestra de entrenamiento, que presentaba las mismas características. Las características a las que se hace referencia, son las recogidas por las variables predictoras, a través de sus categorías. Como se puede ver, se trata de porcentajes, en base a proporciones de datos con las mismas características, presentes en la muestra de entrenamiento. A partir de dichos datos, se estima que el modelo se equivocará por cada categoría de la respuesta, en una proporción determinada, en función de que variables predictoras lo lleven a ese nodo. Esta proporción, son aquellos datos que a pesar de presentar dichas categorías en las variables predictoras, la respuesta era otra. Este es el llamado porcentaje de error del modelo. Con ellos se obtiene el Accuracy, al promediar todos los porcentajes de error que se tienen en cada categoría. Por otra parte, el Kappa, tiene en cuenta el peso de cada categoría, y algunos temas adicionales como la dificultad del modelo, por ejemplo. Para el caso de bosques aleatorios o bagging, que es un caso particular de bosques aleatorios, en ambos, en vez de un solo árbol se calculan muchos de ellos, y se "promedian" sus resultados. Estos temas, serán explicados en mayor profundidad más adelante, dentro de este mismo apartado. Y en cuanto a bibliografía, puede consultarse *Fernández Casal R., et al. (2021)*[5].

El modelo, calculará estos índices realizando validación cruzada, pero con los datos de entrenamiento. Entonces, para que estos índices sean mas fiables, se le pide al modelo que prediga los valores de la respuesta, pero sobre una muestra de test a la cual no ha tenido acceso anteriormente. El resultado de la predicción se compara con los valores que en realidad exhibía la variable respuesta, en la muestra de test, para dichos registros. De esa forma, al calcular estas medidas de rendimiento, se estará simulando el funcionamiento del modelo, sobre datos nuevos. Para evitar que estos índices den valores sospechosamente altos, lo cual estaría indicando que existen variables correladas, es que se debe prestar especial atención en la selección de variables a incluir en el modelo.

En el caso de modelos que trabajan con redes neuronales, el cálculo interno para la obtención del modelo ya es más complejo. Sin embargo, la selección de variables a incluir, es un paso requerido para este tipo de modelos también. Como se verá más adelante, las redes neuronales son modelos robustos, aunque sensibles a las escalas de los predictores. Pero, principalmente, son modelos muy susceptibles a la presencia de variables predictoras correladas. Estos modelos son muy complejos, y por tanto, se suele decir que están hiper-parametrizados (muchos parámetros que acompañan a cada categoría de las variables predictoras). Este hecho, acentúa las correlaciones entre variables predictoras, en caso de que existan, y vuelve esencial la correcta selección de variables a incluir en el modelo.

La idea es que los modelos permitan obtener una clasificación de los componentes incluidos en la red, permitiendo acceder a una especie de "mapa" de las relaciones. Dicho "mapa", contendría las probabilidades asociadas de que determinado elemento sea el responsable de una incidencia, al llegar a él a través de los hijos señalados allí.

Cabe recordar, que se busca asociar la incidencia x (y sus numerosas apariciones), con las alarmas de servicios que estén vinculadas a esta, (sea en base a una ventana temporal, ubicación, servicios vinculados a la misma, etc.). Para luego con ello, hallar su relación con las alarmas de elementos que correspondan. Con esa información, y en base a las vinculaciones existentes entre los elementos, se buscará saber cual o cuales elementos están más atrás en la cadena de relaciones. Esto es, por que se busca donde confluyen los elementos alarmados para dicha incidencia, por eso se dice que se está buscando al "elemento padre" de los alarmados. Siguiendo ese camino, obtendremos el primer acercamiento a la **Causa Raíz** de dicha incidencia.

En primera instancia se intentó realizar un modelo con el método de árboles de decisión, también llamados árboles de clasificación. Se buscaba mostrar la estructura subyacente en los datos, asociando probabilidades de ocurrencia a los distintas categorías en las que se encontraba dividida la variable respuesta, en función de las variables predictoras. Esto se lograría, a través de evaluar que categorías de las variables predictoras aparecían. Sobre cada registro se analiza, por ejemplo, si sale categoría x de la variable id, entonces con cierta probabilidad de ocurrencia la categoría que tendrá la variable respuesta es z. Si además de la variable id, se encuentra presente la variable name, y de esta sale la categoría h, entonces con tanta probabilidad puede ser z, pero con tanta otra puede ser g, y así sucesivamente.

La herramienta rpart, presente como paquete en R, permite realizar este tipo de modelos, y ver un gráfico de relaciones entre las variables predictoras, (variables de entrada o variables independientes), y la variable respuesta, (o variable dependiente). La variable respuesta, se entiende que es dependiente de las variables predictoras, por que cambia en función de los valores, o categorías en este caso, que adoptan dichas variables. A este gráfico, se lo asemeja con un árbol, y de de allí deriva el nombre de la técnica. Cada variable predictora, representa un corte del árbol, donde se bifurcan sus ramas en función de los valores que toma dicha variable. Para generar las bifurcaciones, se utiliza la proporción de datos que presentan una determinada característica, sea para desembocar en otro corte de otra variable, generando mas bifurcaciones, o en una "hoja" del árbol. La hoja del árbol, muestra con que probabilidad los datos adoptan determinado valor en la respuesta, en función de que características de las variables de entrada poseen. Esto es más fácil de ver con un esquema alusivo como el siguiente, en donde se busca clasificar la calidad de un vino, medida por la variable respuesta sabor(del inglés taste), en bueno (del ingles good), o malo, (del ingles bad), según si obtiene determinados parámetros de las variables de entrada, o predictoras.

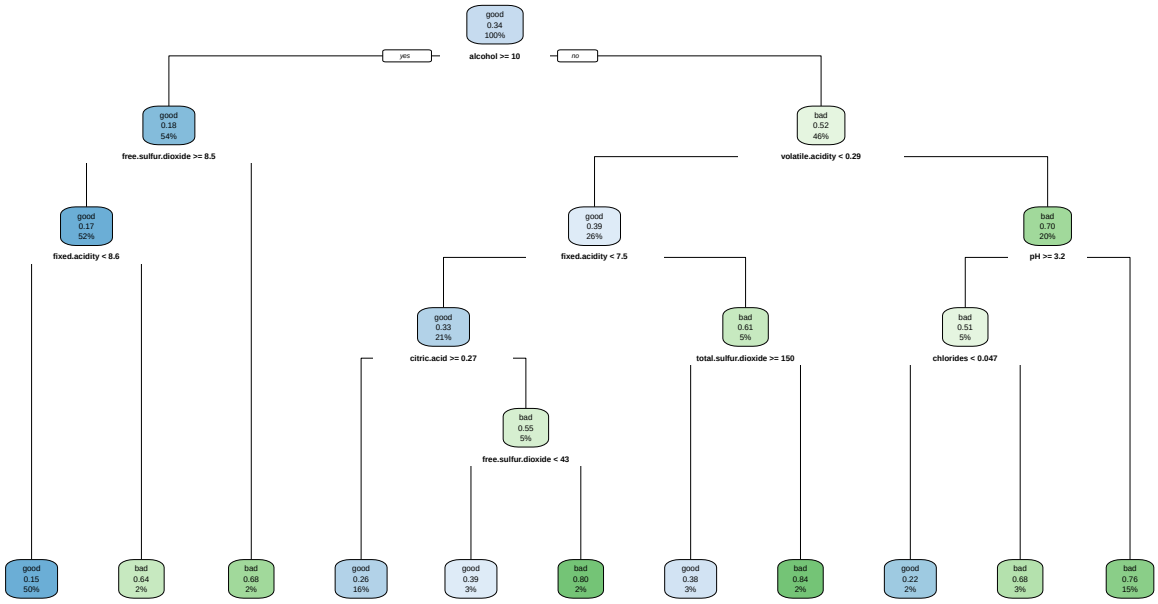

Figura 1.2: Representación del árbol de clasificación del sabor del vino

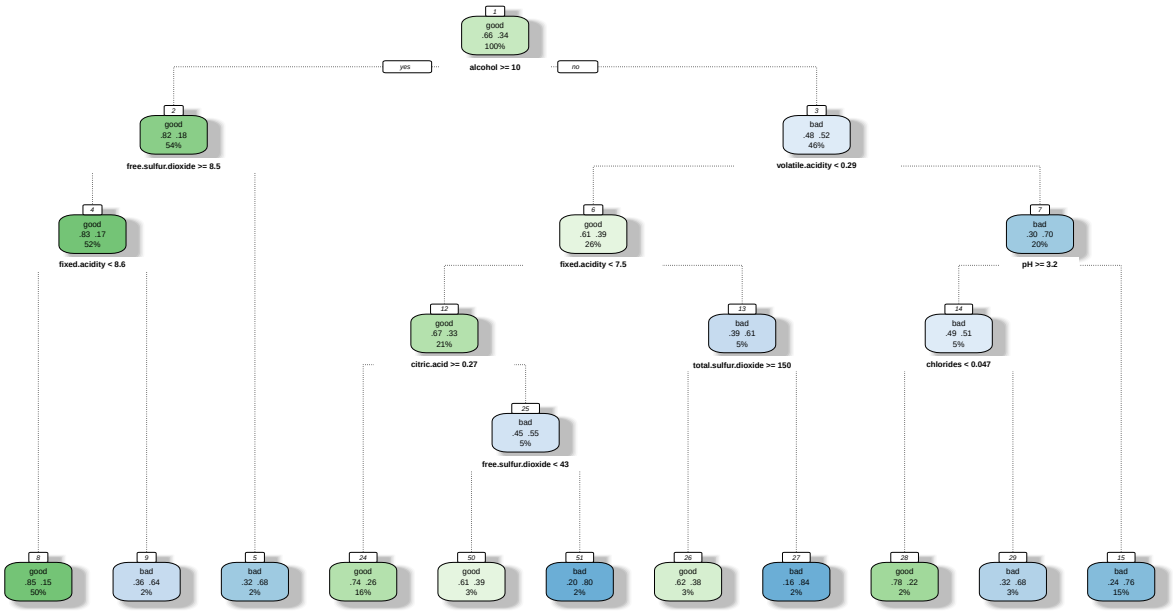

Figura 1.3: Representación del mismo árbol de clasificación anterior, con información adicional
Ejemplo extraído del libro Fernández Casal R.et al. (2021)[5]

El gráfico se lee de la siguiente forma: para la primera hoja que está abajo a la izquierda, (en color celeste en el primer grafo, o en color verde en el segundo), sería: el sabor será clasificado como bueno en un 15 % de los casos, cuando el valor de alcohol esté igual o por encima de 10, el dióxido de sulfuro libre sea mayor o igual a 8.5, y la acidez esté por debajo de 8.6 puntos. Al mirar el segundo grafo, se confirma que por consiguiente, será clasificado como malo un 85 % de las veces aproximadamente, de cumplir con estas mismas características. En la hoja analizada, encontramos al 50 % de la muestra de entrenamiento, es decir el 50 % de la muestra cumple con estas características de alcohol, acidez y sulfuro, y de ellos, solo el 15 % ha resultado tener buen sabor el vino. Dicho dato señala entonces, que en la muestra de entrenamiento, hubo menos vinos buenos, que malos. Esto queda evidenciado en el origen del árbol o raíz, donde figura que la clase positiva es bueno, (cosa que se ha configurado a propósito previamente, para que salga en función de ella el árbol), y que ésta representa el 34 % de la muestra, por lo tanto el 66 % posee la categoría malo. A los efectos de entender el modelo creado, se puede decir que, de presentarse estas características, clasificará la el sabor del vino como malo, por ser la categoría mayoritaria en dicho nodo, y por ende según la muestra de entrenamiento, acertará un 85 % de las veces, y se equivocará en el 15 % de los casos.

Se puede llamar a la salida que entrega el modelo, para verlo más claramente. En ella, se va explicando paso a paso, cuantos elementos de la muestra cumplen con la condición que le hemos marcado como positiva, y cual es entonces, el porcentaje asignado, en consecuencia, a cada categoría. Es importante destacar que, se encuentra expresado de esta forma el árbol, porque definimos que nos interesaba saber que pasaba con la clase bueno. Dado que, en realidad, la herramienta tomaría como clase positiva la mayoritaria, y el árbol se leería al revés, aunque nos indicaría los mismos porcentajes. Para mayor detalle, puede consultarse *Fernández Casal R., et al. (2021)*[5]

Se deseaba generar este modelo con dicha herramienta, por considerarlo de suma utilidad, para poder clasificar los elementos que intervenían en la red en base a las variables de entrada. Este modelo, además, informaría con cuanta probabilidad aparecería cada una de las categorías de la respuesta, en relación a las características particulares que presentara el nuevo dato a predecir. Dichas características, contendrían el tipo de incidencia, y los datos específicos de sus alarmas.

El problema era, que se no lograba avanzar con el modelado de los datos, dada la complejidad de los mismos y la cantidad de registros, el programa se quedaba congelado. No solo, no evolucionaba, sino que además provocaba el colapso del programa.

Por dicho motivo, se decidió virar hacia la técnica de modelado por bosques aleatorios, presente en el paquete de R denominado randomForest, (del inglés bosques aleatorios), puede consultarse *Breiman L. y Cutler R., (2022)*[1]. Se usará para el cálculo de los modelos su función principal, denominada igual que el paquete. Esta técnica, mejora el modelo que se obtendría con un único árbol de clasificación, ya que se construye mediante la combinación de sendos árboles de clasificación, lo cual permite incorporar mayor variabilidad al modelo. La cantidad por defecto de árboles, que utiliza en su función principal, es 500, y dicho valor se controla con el parámetro ntree.

Con esta función, se puede realizar tanto bagging, como bosques aleatorios, según sea el caso de que parámetros se elijan para generar el modelo. En este caso, el parámetro mtry es quien define cuantas variables entran en cada corte de cada árbol, porque como se vio en el ejemplo anterior, para ir construyendo el árbol se van haciendo cortes binarios, y en cada uno de ellos se debe elegir que variable se utiliza en dicho corte. Este parámetro nace, en respuesta al tema puntual de la correlación entre variables predictoras. Si un árbol resulta ser muy adecuado para describir la relación entre los predictores y la respuesta, o si unas variables explican mejor que otras los datos, (que significaría que son mas fuertes que otras), esto hace que los árboles se parezcan entre si, y por consiguiente haya una alta correlación entre ellos, lo que se transmite en una correlación entre los predictores. Esto se suele dar principalmente, en los primeros cortes de cada árbol. El hecho de calcular con dichos árboles, el promedio, (para modelos de regresión), o el mayoritario, (para modelos de clasificación), con poca variabilidad, hace que el modelo tenga una reducción en su varianza mucho menor, que si las variables son incorreladas. En palabras mas sencillas sería, la idea de reducir la varianza, es para que el modelo sea mas flexible. La varianza, es la medida que representa, entre que extremos se mueve nuestra variable respuesta. Esta medida, se ve afectada por ejemplo, por la variabilidad generada cuando se decide que

datos incorporar en la muestra de entrenamiento, y cuales en la muestra de test. Esta selección, se hace para que los estudios tengan mayor variabilidad, y para poder probar el modelo sobre datos distintos a los presentes en la muestra de entrenamiento, y así simular como trabajaría dicho modelo con los nuevos datos. En este punto, se elige un porcentaje de datos para entrenamiento, y otro para test del modelo, "al azar", pero utilizando un valor previo a la selección que se le llama semilla, el cual al fijarlo nos permite luego volver a reproducir esos resultados obtenidos. Pero, se dice "al azar", porque la intención es que el modelo aprenda todas las respuestas que están presentes en los datos. Esto, debido a que la variable respuesta es categórica, y no numérica, exige aún más al modelo, en cuanto a que tiene que saber, para poder clasificar los datos, e intentar predecir el valor de nuevas observaciones. De ser variables numéricas, el estudio sería un poco mas sencillo en este sentido, porque permitiría analizar la tendencia de la respuesta, según los valores de los predictores. Cabe recordar, que los predictores en el caso de estudio, también son categóricos, añadiendo así, aún más complejidad al modelo. Además, este hecho es especialmente relevante a la hora de dividir los datos, en las muestras de entrenamiento y test, no solo por lo descrito anteriormente en relación a la variabilidad, sino también para evitar la aparición de categorías vacías, que provocan un error en la función, utilizada para modelar, y no aportan información para el entrenamiento del modelo. Por este motivo, se debe garantizar, que todas las categorías presentes en cada una de las variables involucradas, estén debidamente representadas. Para lograrlo, se realiza la selección de la muestra de entrenamiento y test, garantizando que en ambas se encuentran incluidos sendos representantes de cada categoría de la variable respuesta, y de cada una de las categorías contenidas en las variables predictoras. Dicha selección debe hacerse, respetando las proporciones presentes, de cada categoría, en los datos originales. Para el caso en estudio se estableció como semilla el valor 1, y se tomó el 80 % de los datos para entrenamiento, y el 20 % restante para test.

Todos estos pasos se realizan, con la intención de generar una reducción de la varianza, que se traduzca en una mayor capacidad del modelo para adaptarse a una variedad más amplia de datos. Si se le enseña más cantidad de datos diversos, el modelo sabrá como responder ante diferentes combinaciones de estos, lo que hace que cuando los datos cambien, o no se cuente con determinados datos de alguna variable predictora, el modelo pueda igualmente, seguir trabajando con los demás datos presentes, sin resentirse tanto su fiabilidad. Esto, se diferencia de si solo cuenta con unos pocos datos, o poco diversos, (esto es cuando se dice que hay poca variabilidad en los datos), porque en este caso, le será muy difícil predecir, con un nivel de error bajo. Incluso, podría simplemente no poder predecir el valor de la respuesta que se espera, por que por ejemplo, dicha categoría no se encontraba presente en la muestra de entrenamiento del modelo.

Entonces, volviendo al tema de los parámetros, siendo el bagging un caso particular de bosques aleatorios, se puede destacar que en la función randomForest, existe el parámetro mtry que permite calcular ambos tipos de modelos. En efecto, si dicho parámetro, es igual a la cantidad de predictores que le hemos ingresado, entonces se estará realizando bagging, y sino será bosques aleatorios.

Estos métodos, además de incorporar aleatoriedad en cuanto a la selección de las muestras, también incorporan aleatoriedad en las variables que se utilizan para realizar cada corte de cada árbol. Esta variabilidad extra, ayuda a mejorar en cuanto a precisión, pero a la vez provoca una pérdida de interpretabilidad, en comparación con el modelo de árbol de clasificación. Dada su naturaleza y construcción, no permite obtener un grafo tipo árbol, que ilustre y ayude, a comprender la estructura de los datos, y la clasificación realizada.

En este TFM, el modelo manual final, que mejores resultados dio, quedó construido con la técnica de bagging, dicho modelo se verá más adelante en la sección *Modelos óptimos con selección manual de variables* 2.2.1. Se realiza Bagging, cuando el número de predictores seleccionados al azar, coincide con el número de variables predictoras introducidas al modelo, (número seleccionado a través del parámetro mtry). Este tipo de modelos se construyen calculando una gran cantidad de árboles, basados en un re-muestreo de los datos de entrenamiento, para ir cambiando que variables van a entrar en el primer corte del árbol, lo cual, habitualmente, provoca una modificación en los cortes siguientes, también. Con dichos árboles, se calcula cual es la decisión mayoritaria, y esta será la información que contendrá el modelo para predecir. No se puede calcular un promedio de las predicciones, (como es el caso en los árboles de regresión, que contienen variables numéricas), porque en este caso, se trata de árboles de clasificación, pero la esencia es la misma.

Este método, también permite calcular, una estimación de la precisión de las predicciones, mediante el cálculo del error OOB. *"...Un dato que no es utilizado para construir un árbol se denomina un dato out-of-bag (OOB). De este modo, para cada observación se pueden utilizar los árboles para los que esa observación es out-of-bag (aproximadamente una tercera parte de los árboles construidos) para generar una única predicción para ella. Repitiendo el proceso para todas las observaciones se obtiene una medida del error..."Fernández Casal R.et al. (2021)*[5]. En la práctica, estos OOB se pueden ver a través de un gráfico, que muestra la convergencia a un determinado valor de error. Dicho valor, depende del número de árboles a considerar, (configurado en el parámetro ntree, comentado anteriormente). Este valor, si es lo suficientemente grande, permite al error OOB estabilizarse en torno a una cifra puntual. Se tendrá entonces, un valor de error OOB, para cada una de las categorías en las que está dividida nuestra variable respuesta. Al promediar dichos valores, se obtiene el error OOB global del modelo. Se busca minimizar este valor, pues de esa forma se incrementa la precisión del modelo, es por esto que a lo largo de este TFM, se han ido aplicando diferentes herramientas para lograrlo, como la selección de variables, o el refinamiento de parámetros del modelo, por ejemplo. Para mayor información, puede consultarse *Fernández-Casal, et al. (2023)*[4]

Todos los modelos, fueron generados sobre muestras de entrenamiento, y validados sobre muestras test. Para compararlos, se utilizaron los índices de precisión obtenidos, durante la validación, sobre la muestra de test, como el Accuracy o el Kappa, en vez del OOB global. Esto fue debido a que, al refinar los parámetros del modelo, se obtenían mejores resultados si se buscaba mejorar el Kappa, que si se buscaba minimizar el OOB. Esto se debe a que, pese a que el OOB es una estimación del error basado en validación cruzada, esta validación cruzada se realiza sobre la misma muestra de entrenamiento, y es más probable, que el modelo tenga un mejor desempeño sobre los datos con los que fue entrenado, que sobre datos nuevos. Debido a esto, se busca calcular las predicciones sobre una muestra de test, que el modelo desconoce. En base a dichas predicciones, se calculan los índices Accuracy y Kappa para evaluar el rendimiento del modelo. Cabe destacar, que nos centramos en el Kappa, porque se trataba de muestras des-balanceadas. Esto implica, que no se cuenta con las mismas cantidades o porcentajes, de cada categoría, sino que existen categorías que son extremadamente grandes, respecto de otras. Este hecho vuelve al índice Accuracy poco fiable, por representar un promedio de la precisión obtenida en cada categoría. El Kappa, es entonces el elegido como medida más acertada, por tener en cuenta este tema, y además, otros aspectos del modelo para calcularse.

En casos des-balanceados, lo recomendable es evaluar sensibilidad y especificidad de cada una de las categorías, para ver que modelo es mejor, en las categorías que más relevantes resultan para el estudio en cuestión. Pero en este caso, la generación y posterior evaluación de los modelos se vuelve una tarea bastante más intrincada de lo habitual. En el caso de estudio, el número de categorías, presentes en la variable respuesta, es muy elevado. Más aún, la empresa indica que no hay categorías más importantes que otras. Además, también se tienen gran cantidad de categorías presentes en las variables explicativas. Volviendo de suma importancia, el contar con herramientas que faciliten la búsqueda del mejor modelo, y colaboren con reducir el número de variables utilizadas. Así, se podrá realizar una búsqueda más eficiente, permitiendo contar más tiempo y recursos, para analizar en mayor profundidad el modelo obtenido, y mejorar sus resultados.

Por otra parte, la aleatoriedad sujeta a la selección de las muestras, también afecta los resultados. Esto hecho, se intenta solventar, al realizar pruebas de aleatoriedad sobre los mejores modelos obtenidos. Este proceso será expuesto más adelante, dentro de la sección *Pruebas de aleatoriedad a los modelos obtenidos* 2.3.1.

El modelo alcanzado en esta instancia, obtenía muy buenos resultados debido a:

- la búsqueda que se había realizado, para encontrar la mejor combinación de variables predictoras a incluir;
- al método utilizado, ya que al ser un método que incluye aleatoriedad en las muestras, y en las variables predictoras que incorpora en cada corte, se estima que sus resultados son mejores que los proporcionados por un único árbol de clasificación;
- a que había realizado un refinamiento de sus parámetros para mejorar su rendimiento. Estos parámetros eran:
  - el parámetro denominado mtry, mencionado anteriormente;
  - el parámetro nodesize: que es cuantas observaciones tiene que haber como mínimo, en un nodo terminal u hoja, de cada árbol. Cada nodo terminal u hoja, es el encargado de clasificar las distintas categorías en las que se puede dividir la variable respuesta. Con estas categorías, se trazará una especie de camino, donde se evalúa cada una de las variables predictoras, analizando que valores tomaron cuando apareció dicha respuesta. Se analiza cuantas observaciones han aparecido con una determinada respuesta, y que otras variables, y con que valores, estaban presentes en dicho dato. Por ejemplo, si el nodesize es 5, se estará indicando que en cada nodo terminal, como mínimo, tiene que haber 5 observaciones, que presentan determinada respuesta, y cumplen con las características descritas de las distintas variables predictoras analizadas. Estas características estarán descritas en las ramas del árbol, que desembocan en dicha hoja, o nodo. Si este número es pequeño, implicará que la estructura del árbol será mas completa y compleja, dando como resultado árboles más grandes, y profundos. Este hecho demandará, más recursos computacionales y tiempo para su construcción. Cuando se hace referencia a un número pequeño, (por ejemplo 1,2,5), se debe tener en cuenta que todo depende de las características propias de los datos a analizar. Si se cuenta con una base compuesta por 5 mil datos, en donde las categorías a analizar de las variables tienen alrededor de 1000 datos cada una, se debería incrementar dicho número. Esto debido a que, tener nodos terminales con tan pocas observaciones, no sería útil, pues se podría caer en casos particulares que no aportaran al estudio futuro. En cambio, si los datos son pocos, unos 500 en total, y cada categoría de la respuesta cuenta con 100 datos para clasificar, según los valores de las variables predictoras, en este caso podríamos configurarlo en 1, 2, o hasta 5, dependiendo de cuantas observaciones se alcanzan en los nodos finales, u hojas. Dichos valores serían considerados pequeños, pero no demasiado pequeños, como en el caso anterior. En otras palabras, se podría decir que se llega a los nodos terminales, a través de ir considerando, cuando clasificamos una nueva observación, si tenemos o no determinada variable. De estar presente, se analiza que valor ha adoptado. En función de esto, se tendrá que la variable respuesta adoptará x valor, con cierta probabilidad de ocurrencia, y sino será un valor y, con la restante probabilidad de ocurrencia. Esta probabilidad de ocurrencia, estará dada por la proporción de datos, que cumplen con dichas características en la muestra de entrenamiento. Aunque, el modelo al predecir, solo entregará aquella respuesta mayoritaria, que presentaba las mismas características en las variables predictoras, que el nuevo dato introducido para su predicción.

- el ntree: que es el número de árboles a construir, por defecto en la función viene en 500. La intención, es que una vez observado el gráfico de OOB, se pueda elegir este parámetro. Esto se hace, analizando el momento en que el error se estabiliza en torno a un valor determinado. Es común configurar al inicio, un valor alto, para analizar a partir de que cantidad de árboles el valor se estabiliza, y realizar el ajuste. Así, el modelo final será más eficiente, en cuanto a recursos computacionales, porque mientras más árboles se generan, más recursos se requieren.

Los ajustes finos, mencionados más arriba, requieren del factor humano para analizar los valores de salida del modelo, y así detectar problemas con el mismo. Un problema que puede presentarse, es la dependencia entre los árboles construidos. Esto se hace evidente al observar, que al hacer el corte de cada árbol, la función elige, casi siempre, de primer corte la misma variable. Este hecho, muchas veces viene acompañado de un valor de Accuracy, y/o Kappa, sospechosamente alto. Otro problema, que puede requerir atención, se presenta cuando las muestras son des-balanceadas. En ese caso, el Accuracy ya no sería un índice acorde para evaluar la precisión del modelo, y habría que evaluar otras alternativas; como usar el Kappa, analizar la sensibilidad y especificidad, usar tablas de frecuencia, entre otras. Todo esto se hace más difícil de ver en nuestro campo, dada la cantidad de sub categorías en las que esta fragmentada nuestra variable respuesta, hecho que dificulta poder estudiar el tema a través de tablas de frecuencia, o con análisis de sensibilidad y especificidad, para cada categoría, por ejemplo.

Una vez sorteados todos estos inconvenientes, se logra generar un modelo, que era capaz de calcular con una probabilidad bastante alta de exactitud cual era el elemento padre de esa alarma, y por tanto era un primer acercamiento al RCA, que se pretendía.

Es en este punto, que la empresa solicita que se generen otros modelos, que utilicen técnicas distintas al randomForest. La intención es poner a competir los modelos, para evaluar su desempeño, y quedarse con aquel que mejor se adapte a los requerimientos de la empresa. Como se mencionó anteriormente, al utlizar randomForest, se pierde interpretabilidad, debido a su construcción. Como ya se mencionó para el caso de clasificación, se calculan muchos árboles, y se construye el modelo en base a las categorías mayoritarias. En este segundo modelo entonces, se buscaría ganar interpretabilidad, mediante la construcción de un modelo de árbol de clasificación, que mostrara de forma mas expresa la relación entre las variables. Por tanto, se volvió a realizar el análisis precedente, en cuanto a selección de variables predictoras, pero esta vez empleando el método de inferencia condicional para construir los modelos, utilizado en la herramienta de R llamada party. Puede consultarse *Hothorn T., et al. (2006)* [6]. Este modelo será expuesto con mayor detalle en la sección *Modelos óptimos con selección manual de variables* 2.2.1. La herramienta party, para elegir que variable introducir en cada división del árbol, se basa en el método de inferencia condicional. Establece como hipótesis nula, la independencia de la respuesta respecto de las variables explicativas, en términos generales. En ese punto genera un contraste de hipótesis, y cuando se rechaza la hipótesis nula de independencia, el algoritmo elige aquella variable explicativa con asociación más fuerte a la respuesta, en base a su p-valor, para realizar un corte binario en el árbol. Este proceso lo ejecuta de forma recursiva, hasta que en el contraste de hipótesis, no se rechace la hipótesis nula.

Esto es distinto, de lo que hace la herramienta rpart, que era la que se intentó utilizar en un principio, y no se pudo.Como se comentó anteriormente, dicha herramienta, también construye un único árbol, pero basándose en el índice de importancia de las variables predictoras, para la construcción del modelo. Dicho índice, es calculado por la misma función, y se llama índice de Gini. Este actúa como medida alternativa al RSS, que es una forma de medir el error, utilizado en árboles de regresión.

También es diferente, de la tarea que realiza la herramienta randomForest. Dicha herramienta, explicada anteriormente, calcula el modelo de clasificación por decisión mayoritaria, en base a los árboles que construye. Esto lo hace, a partir de ir generando sub-muestras aleatorias, y construye un árbol con cada una de ellas, definiendo, que variable entra en el primer corte del árbol, cual en el siguiente, y así sucesivamente. Esto, utilizando para cada corte, un grupo de variables elegidas aleatoriamente, cuyo tamaño del grupo se ajusta al número de predictores seleccionado en el parámetro mtry. Dicho parámetro, por defecto para clasificación, utiliza el entero más bajo, resultante de calcularle la raíz al número de predictores incorporados. Luego dentro de ese grupo, utilizará el comentado índice de Gini, para elegir que variable empleará, para realizar el corte binario. El éxito de esta técnica radica en lograr, que estos árboles elijan de forma equitativa, a todas las variables presentes en el modelo, como primera variable de corte. De esta forma, se reduciría la varianza, logrando estabilizar el error OOB en torno a un valor pequeño, lo cual significaría, que se tiene un modelo más preciso y versátil.

Para el caso del árbol de clasificación, generado con la herramienta party, algunos parámetros relevantes, que se pueden considerar para su refinamiento son:

- el minsplit: que es el número mínimo de observaciones en un nodo intermedio, dicho valor es requerido para realizar la partición del mismo. Este parámetro viene configurado por defecto en 20 observaciones.
- El minbucket: que es el equivalente al nodesize de randomForest, (número mínimo de observaciones en un nodo terminal), su configuración por defecto es 7 observaciones.
- El mincriterion: que es el valor que usa para hacer el test de significación, y así evaluar la incorporación de la variable al modelo. Este valor por defecto viene en 0.95, este equivale a $1-$p-valor, con lo cual el p-valor que usa por defecto, es 0.05.

Una vez generado el modelo con esta herramienta, surge la inquietud de realizar un modelado utilizando la técnica de redes neuronales . Esta técnica ha tenido un gran auge en estos últimos tiempos, enmarcada dentro de lo que se suele llamar Aprendizaje Automático, (del inglés, Machine Learning). Suelen ser adecuadas para trabajar con grandes bases de datos, en donde las relaciones entre las variables explicativas y la respuesta, sean complejas. Desde el entorno estadístico, se suele hacer referencia a las técnicas de Aprendizaje Automático, como técnicas de Aprendizaje Estadístico. Dado que, justamente eso es lo que está haciendo el modelo, aprendiendo del análisis estadístico que realiza, a partir de los datos que le fueron brindados. Teniendo en cuenta el tipo de datos con los que se contaba, y el volumen de los mismos, era una buena oportunidad para evaluar si este tipo de modelo, era superior en desempeño a los ya construidos.

Es importante resaltar que, la interpretabilidad en cuanto a la estructura de relaciones que construiría el modelo, se perdería casi por completo, dada la gran complejidad con la cual se maneja este tipo de herramientas. Se trata de un tipo de modelo que hace transformaciones a los datos, para a través de ellas encontrar como relacionar las variables de entrada con la respuesta. Combina las variables predictoras presentes en la capa de entrada, con sus elementos en las capas ocultas, allí es donde se encuentran las transformaciones realizadas, y luego vuelve a transformar los resultados obtenidos, para adaptarlos al formato de la variable respuesta, y así producir la salida del modelo, la predicción. La cantidad de capas cambia, según cuantas transformaciones le realice el modelo a los datos. Por ejemplo, en una red básica se realizan dos transformaciones, y por tanto dicha red contendrá tres capas, la de entrada que contiene las variables originales, una oculta, y la de salida, donde se encuentran las predicciones del modelo final. Como puede verse, al construirla, se genera una gran cantidad de parámetros, lo cual le permite, a la red, abordar problemas mas complejos, pero también la vuelve más difícil de interpretar. Unido a esto, requiere configurar una importante cantidad de hiperparámetros, convirtiéndola en más demandantes en cuanto a cantidad de datos requeridos para aprender, y a los recursos computacionales que utiliza para calcularse. Debido a esto, se requiere mayor experimentación, tiempo y recursos computacionales para su armado.

Para construir la red, que se verá en detalle más adelante en la sección *Modelos óptimos con selección manual de variables* 2.2.1, se utilizó el paquete NNet, incluído en la librería de caret, para

R. Dicho paquete, permite obtener la red por validación cruzada, a partir de una rejilla contemplada dentro de los valores aconsejables, para una red básica con los datos de estudio. Así una vez construida, se puede buscar mejorarla, en caso de ser necesario. Puede consultarse *Kuhn., et al. (2023)* [9] para mayor información sobre el meta-paquete caret. Puede consultarse *Ripley, Venables. (2023)* [12] para mayor información sobre el paquete NNet, que es el paquete que llama caret como método a aplicar, para generar la red.

A continuación se detallan, los parámetros, paquetes y funciones, que resultaron más relevantes, para la construcción del modelo, con dicha herramienta, estos son:

- train: es la función que llamamos del meta-paquete caret, para construir el modelo.
- El parámetro method: se establece dentro la función train, para configurar el método de cálculo del modelo. Para generar una red neuronal con el paquete mencionado anteriormente, le asignamos el valor "NNet"
- El parámetro Size: define el número de nodos en la capa oculta, como se comentó brevemente, es donde se hallan las transformaciones realizadas por la red.
- El parámetro Decay: representa la tasa de aprendizaje de la red, su valor por defecto es 0, aunque este valor suele configurarse entre 0 y 0.1.
- El parámetro tuneGrid: presente en la función train, de caret, permite incorporar una rejilla con todas las combinaciones posibles, de los dos parámetros referenciados anteriormente, para encontrar que combinación de parámetros funciona mejor para el modelo NNet en este caso.
- El parámetro Maxit: representa el número máximo de iteraciones, (se establece como criterio de parada junto con el decay), su valor por defecto es 100. Se debe tener en cuenta, que el calculo de la red óptima se logra a partir de una combinación de muchas redes, en donde se van tomando diferentes submuestras, de la muestra de entrenamiento, para que el modelo aprenda con ellas. Dichos grupos, llamados batch, son de tamaños fijos, grupos de 32 datos, 64 datos, etc. Cada vez que el algoritmo completa el procesado de todos los datos, ahí se cuenta una iteración, cada una de estas iteraciones se las llama epochs, y son las que se limitan con el parámetro Maxit. De esta forma, se busca mitigar la inestabilidad del modelo, al promediar las predicciones obtenidas. Esta inestabilidad, es debida a que la estimación de los parámetros, se realiza resolviendo un problema, no convexo, de optimización matemática. Esto significa, que debe resolverse por aproximación, dado que en la práctica, suele ser imposible encontrar una única solución óptima. Por eso se resuelve mediante un algoritmo heurístico de descenso de gradientes, el cual logrará converger a un óptimo local, pero difícilmente hallará un óptimo global. Para mayor información puede consultarse *Fernández Casal R.et al. (2021)* [5].
- El parámetro MaxNWts: limita el número máximo de pesos que usará la red. No hay un límite intrínseco en el código, pero incrementarlo probablemente conduciría a modelos que son muy lentos, y que representan una pérdida de tiempo, según indica el propio documento del paquete NNet. Para mayor información puede consultarse *Ripley, Venables. (2023)* [12].
- El parámetro preproc: permite realizar un pre-procesado a los datos, viene inserto en la función train, de caret, que estamos usando. La red es muy sensible a las escalas de los datos. Este parámetro permite re-escalar los valores de las variables entre 0 y 1, configurándolo a "range" en caso de tener variables numéricas, en el caso de estudio no sería requerido, en principio, por ser todas variables de tipo factor.

- El parámetro lineout: permite seleccionar la función de activación en los nodos finales, por defecto viene en FALSE. Con su configuración por defecto, la función NNet utilizará la función logística para el caso de variables binarias, (respuesta con solo dos categorías dentro, como podría ser bueno, malo, por ejemplo), o una función llamada softmax, preparada para trabajar con variables de tipo factor con múltiples niveles. En este punto no se realizó ningún cambio, por considerarlo acorde al caso en estudio. Vale recordar, que la variable respuesta analizada tiene más de dos categorías. Para mayor información puede consultarse *Chollet, F., y Allaire, J. J. (2018)* [2]. Es importante destacar, que durante el modelado con redes neuronales, en lugar de construir un único modelo construyen tantos como categorías se tengan en la variable respuesta. Por ello, se requiere de una función de activación adecuada en los nodos finales, la cual, en palabras sencillas, contribuye a re-transformar la salida, para entregar la predicción en los términos de la respuesta.

- El parámetro trControl: allí se establecen los parámetros que controlan el entrenamiento del modelo. Aquí se presentan algunos de los parámetros que incluye, utilizados para el desarrollo del citado modelo de redes neuronales:

  - el parámetro method: permite definir el método de remuestreo para evaluar los hiperparámetros, en este caso se usa "cv", que es validación cruzada, (del ingles cross validation).
  - El parámetro number: señala el número de sub-muestras,(del inglés kfold), en que se dividirá la muestra de entrenamiento, para hacer la validación cruzada por grupos. Esta es una forma de hacer validación cruzada, ya que hay otras alternativas. Una de ellas, puede ser "LOOCV" ,(del ingles leave one out cross validation), en donde se van comparando todos los datos contra uno, realizando el proceso de validación cruzada de manera recursiva, cambiando en cada iteración el dato que se deja fuera para predecir. Pero esta última forma descrita, es computacional-mente más exigente que "kfold".

  Además, el parámetro trControl, permite configurar el método de selección del óptimo, y las medidas de precisión deseadas, con solo ingresarlas con el nombre de parámetro, al que se refieren. Para mayor información, puede consultarse *Kuhn., et al. (2023)* [9], *Fernández Casal R.et al. (2021)* [5].

Con las herramientas descritas, se logra generar un modelo de red neuronal, de tres capas. En base a una extensa búsqueda de las variables e hiper-parámetros adecuados, y a experimentar diferentes opciones de ejecución del algoritmo, para optimizar su rendimiento. Dicho modelo, lograba una pequeña mejora, en comparación con los resultados obtenidos por los modelos anteriores, en cuanto a precisión. Por lo expuesto anteriormente, se puede ver que esta red no era profunda, pero se decidió incorporarla como opción, ya que no siempre una red más profunda implica una mejora. Cabe destacar que esta red, demandó el uso de muchos recursos, tanto computacionales, como de tiempo para su ejecución, para apenas comenzar a calcularse. Si se le ingresaban todas las variables predictoras juntas, con las que se había estado trabajando en los modelos anteriores, la función colapsaba. Esto era, debido a la cantidad de recursos computacionales que requería para su cálculo. Cabe recordar que la base de datos, con la que se estaba trabajando, era grande pero no inmensa. Se contaba con 10 mil registros y 20 variables, aproximadamente, luego de su depuración.

Como demostración de lo anterior, se puede comentar, que la generación del modelo demoró unas 15 horas aproximadamente. Dicho modelo, utilizaba las mismas variables que el modelo de bagging, calculado con la herramienta randomForest, el cual era apenas inferior a la red, en cuanto a índice Kappa. Por tanto, la mayor inversión computacional y de experimentación requeridos por la red, no eran justificables en cuanto a mejora en la precisión. Para alcanzar una supuesta mejora, se hubieran requerido muchas más horas de ejecución, y más recursos computacionales. Ya que en varias ocasiones, la función colapsaba por falta de memoria de trabajo y almacenamiento, hecho que requería que se reiniciara la herramienta, y se perdieran todos los avances realizados hasta ese momento. En esta etapa, se estaba trabajando en un servidor de la empresa, con unos 5gb de memoria ram. Esto provocó, en varias oportunidades, que dichos recursos no fueran suficientes, provocando un reajuste de los

parámetros de los modelos que se intentaban calcular, para su correcta ejecución. Este problema, se presentó con todos los modelos calculados, y fue el que más tiempo consumió para su resolución. Las comparaciones mencionadas anteriormente, provienen de un estudio comparativo realizado sobre los modelos. En dicho estudio, hecho sobre la muestra de test, los modelos calculados obtenían valores de efectividad en torno al 80 %, (representado en el valor de Kappa), lo cual se podrá ver más adelante en la tabla comparativa de los modelos generados, presentada en el capítulo 2, denominado *Investigación Paso a Paso*, en la sección de *Cálculo de los modelos* 2.2.

Una vez generados estos tres modelos, desde la empresa se plantea el requerimiento de hacerlos competir entre ellos, pero aplicándolos en bases de datos diferentes a las utilizadas para entrenamiento. Esto, en principio, no era posible, sin la reconfiguración manual, de cada uno de los modelos. Dado que, cada modelo se genera en base a unos datos de entrenamiento, y está preparado para aplicarse sobre nuevos datos, siempre y cuando, la estructura de los datos nuevos sea la misma que la aprendida por el modelo. Si el modelo, se aplica sobre datos nuevos, pero estos contienen por ejemplo, categorías distintas a las aprendidas, u otros nombres de variables, eso generaría un desempeño inferior, o hasta incluso malo de los modelos, llegando al extremo de que no pudieran predecir. Debido a que, no estarían operando sobre el tipo de datos que fueron entrenados, lo cual obligaría a generar un modelo nuevo, con cada una de las herramientas utilizadas, cada vez que las bases de datos se cambiaran. Por consiguiente, se tendría que realizar el estudio nuevamente, con cada nueva base de datos, y esto claramente, no era conveniente para la empresa.

Este requerimiento, motivó la generación de un algoritmo, que de manera automática, seleccione las variables a incluir en el modelo, eligiendo aquellas que produjeran un mejor desempeño del mismo. Con este paso, se ahorrarían muchos días de trabajo. Esta primera parte del TFM, que será descrita en el capítulo 2, denominado *Investigación Paso a Paso*, había requerido entre 2 y 3 semanas. La mayor parte del tiempo, se había dedicado a la selección de variables. Porque con las variables correctas, luego se podía generar el modelo, evaluarlo e incluso refinar sus parámetros, para obtener los mejores resultados. A pesar del tiempo invertido, solo se había podido probar con 6 o 7 variables, que se estimaba que eran las más relevantes para el modelado, y que además no contenían información solapada con otras. Esto, llevó a programar un algoritmo, por cada una de las técnicas de modelado que se habían utilizado.

Este conjunto de algoritmos, es uno de los hallazgos fundamentales de esta investigación. A partir de aquí, se irá viendo paso a paso, cual fue la técnica desarrollada, para realizar la ***Búsqueda de Causa Raíz de un problema en un entorno IT****, y así se podrá entender mejor el cómo, y porque de su creación. Además se señalará, cuales son sus principales ventajas, y requerimientos.*

Este aporte, vino a resolver el inconveniente que se planteaba en la empresa, ante la necesidad de generar el modelado de distintas bases de datos. El mismo, permitiría sentar las bases para entender la estructura de los datos, poder predecir la variable respuesta en base a las siguientes alertas, y colaborar con trabajos futuros del análisis de **Causa Raíz**, de un problema en un entorno IT.

Más aún, se tiene la intención de crear un paquete de R, que los contenga. Para colaborar con aquellos trabajos, en donde se requiera generar una selección de variables, para un primer modelo aditivo. Siempre que, se trabaje sobre una base de datos categóricos, nominales, no ordinales, y que se requiera un modelo matemático de clasificación, con alguna de estas tres herramientas informáticas. Esto le permitirá al investigador, dedicar más tiempo y esfuerzo, al refinamiento de los hiperparámetros, del modelo en cuestión.

Capítulo 2

Investigación paso a paso

2.1. Minería de Datos

Como ya se comentó anteriormente, una empresa de servicios de telecomunicación se encuentra especialmente interesada en realizar un análisis de **Causa Raíz**, para problemas en un entorno de telecomunicaciones, (conocido como entorno IT). La idea principal es encontrar un algoritmo, que de manera automática, logre inferir las posibles causas, para aquellos problemas cotidianos, con los que se enfrenta el sector. Conocer dichas causas, permitirá analizar las soluciones posibles. Con dicha información, se podrán evaluar los resultados de las decisiones tomadas en este sentido, para ratificarlas o rectificarlas. Provocando así, la retroalimentación del proceso de análisis de **Causa Raíz**. Para poder realizarlo, primeramente se debe realizar una técnica de minería de datos, y así extraer la mayor información posible de los mismos. Luego de esto, se pasará a la generación de un modelo matemático, perteneciente al ámbito del aprendizaje automático, o aprendizaje estadístico. Este modelo, aprenderá a identificar la relación entre las variables explicativas y la respuesta, en base a la información histórica proporcionada. En función de lo aprendido, estimará entonces cual es la causa, o causas más probables de las incidencias que se requieren resolver. Finalmente, otro algoritmo, aprenderá la vinculación de las causas, con las soluciones adoptadas en cada caso. Este modelo, brindará cuales son las soluciones, que se utilizan más frecuentemente, para resolver dicho inconveniente estudiado. Tal como se explicó anteriormente, en la sección *Herramientas para el análisis de RCA en un entorno IT* 1.3, este es un enfoque válido para entornos en que se cuenta con gran cantidad de datos, y con complejas interrelaciones que los vinculan. En este sentido, se puede ver que los datos estudiados, presentan características tales como:

- Una compleja interrelación, entre los elementos que componen la red de telecomunicación.
- La dificultad para determinar cuando una incidencia se transforma en un problema a resolver.
- La necesidad de hallar un método, que permita asociar las incidencias con las alarmas.
- La inexistencia de parámetros claros, que indiquen la necesidad de actuar, y en que dirección. Se depende del conocimiento y experiencia de los agentes que intervienen en los procesos, quienes en tiempo real, toman las decisiones pertinentes.

Con este estudio se busca, entonces, incrementar la velocidad de respuesta, y capitalizar el conocimiento de los agentes intervinientes. Por eso, el uso de modelos matemáticos, puede ser de gran ayuda para comprender el entorno de trabajo, y colaborar así con el análisis de RCA que se está realizando.

Cabe recordar, que por tratarse de variables categóricas nominales no ordinales, los modelos de clasificación no puede suponer tendencias más allá de los límites aprendidos. Esta tarea es posible, cuando se trabaja con modelos de regresión, donde las variables intervinientes son de tipo numéricas. Aunque no es muy recomendable extrapolar mucho más allá de ciertos límites lógicos, los resultados obtenidos. Si se piensa en entornos IT, allí el cambio y la actualización es constante. Lo cual vuelve a los modelos obsoletos con mayor velocidad. Por eso, lo primordial es optimizar la capacidad de adaptación de los modelos.

Una vez determinado el problema a resolver, se realiza una investigación sobre los avances en relación a RCA, tanto en este campo como en otros, para poder sentar las bases para el TFM. En base a lo estudiado, y dado que no se han encontrado estudios aplicados directamente en este tipo de entornos, se decide, comenzar con un análisis de causas probables, con la técnica de análisis de Pareto. Se busca dar con las causas históricas que representan ese 20 %, responsable del 80 % de las incidencias observadas. Desde la empresa expresaban, que les interesaba saber si la prioridad y el impacto, tenían alguna relación directa, con el momento en que una incidencia se volvía un problema a resolver. Por una cuestión de análisis lógico a primera vista, estimaban que si una incidencia se convertía en problema, era porque su impacto era grande, o porque su prioridad era alta, por citar algunos ejemplos. Se analizaron entonces, las categorías de las variables, teniendo en cuenta la prioridad y el impacto de las mismas. Se detecta en este paso, que la mayor cantidad de incidencias se encuentran en la franja de prioridad baja, y de impacto menor o localizado. Se continúa la búsqueda, aplicando la herramienta de Pareto, para poder ubicar con que prioridad e impacto se corresponde ese 20 % de las causas, relacionadas con el 80 % de las incidencias. Finalmente, se logra identificar que la prioridad y el impacto asociado, a mas del 80 % de las incidencias, se distribuye entre las primeras 6 categorías. Había 16 categorías, debidas a las distintas combinaciones que pueden presentarse en los datos. El análisis de Pareto, distribuía dichas categorías de la siguiente manera: Un 25 % de las incidencias, se categorizan en prioridad baja, y con impacto menor o localizado. Luego están aquellas de prioridad media, y de impacto moderado, alcanzando así un porcentaje acumulado del 44 %. A continuación, aparecen las incidencias catalogadas con prioridad critica, y de impacto extenso. Seguidas de aquellas con prioridad alta, e impacto moderado. Mas allá, aparecen las de prioridad crítica e impacto significativo, y por último las de prioridad alta, e impacto significativo. Todas las categorías citadas previamente, representan en conjunto el 84.55 % de las incidencias analizadas, aproximadamente. Lo expuesto anteriormente, resalta la importancia de realizar un buen análisis de RCA, permitiendo concentrar el esfuerzo en unas pocas categorías, para encontrar la **Causa Raíz** asociada a ellas. Así, esto nos permitiría atender y resolver más del 80 % de las incidencias, de manera más eficiente. Durante este análisis, y debido a la detección de incongruencias relacionadas con el registro de los datos, se realizó un doble estudio. Se detectó que figuraban registros con causa desconocida, que luego fueron cancelados. Estos en su mayoría, al estudiarlos en profundidad, se relacionaban con un evento llamado corte netcool. Entonces, primero se los tuvo en cuenta como una categoría aparte. Y luego se les asignó valor NA, para que en el análisis de Pareto no figuraran, tal como sería si estos eventos no sucedieran. Al comparar el análisis de Pareto, donde se tenían en cuenta los incidentes desconocidos y cancelados, como una categoría aparte; con el que se los establecía como NA, se puede ver que los totales cambian, incluyendo entonces, algunas de las causas que antes no se presentaban, además de asignarle mayor relevancia a otras causas, ya presentes. Razón por la cual, es muy importante tener bien diferenciados los códigos que se usen, identificando que registros de este tipo fueron problemas a resolver, y cuales solo errores en la toma de datos. También es importante resaltar, que las 4 primeras causas, al no tener en cuenta las desconocidas y canceladas en el análisis, se presentan como las responsables de mas del 60 %, de los incidentes registrados. Esto es relevante dado que, de poder prever este tipo de fallas, tanto en equipos propios como en equipos de la red de transporte, además de las referidas al cableado o elementos auxiliares, se estarían evitando cerca del 44 % de las alertas antes de que sucedan. Sería una buena elección entonces, estudiar como evitar este tipo de registros, (los de tipo desconocidos y cancelados), para que no vuelvan a producirse, y así ahorrar todo ese esfuerzo y recursos.

Por otra parte, al realizar una lectura más profunda de la información, se puede resaltar, que a pesar de existir alrededor de 300 alertas identificadas, con nombres muy variados y diversos, preocupa ver que hay 211 asignadas a una categoría llamada other, que no aporta datos sobre la alerta en si. Este hecho, dificulta aún más el rastreo de la **Causa Raíz**, y la vinculación de dicha alerta con las incidencia. Los aspectos más relevantes, del mencionado análisis de Pareto, se encuentran detallados en el Apéndice A. De dicho estudio además se desprende, que existen diferencias en cuanto a los nombres de causa, dados a las distintas incidencias. Esto, para hacer un análisis mas exhaustivo, debería de tratarse y depurarse, unificando aquellos que son iguales, escritos con cambios en la puntuación por ejemplo, para tener una mejor visión global del tema. Un ejemplo claro es que, en un principio figuran incidencias cuya causa se nombra como desconocida, luego aparece otra categoría que simplemente figura un guion medio como causa, (la cual también entonces, sería desconocida), etc.

Los citados inconvenientes junto con otros, estimularon la decisión de la empresa de crear una base de datos sintética, basada en los datos originales. Este nuevo conjunto de datos, permitirá generar un modelo matemático, que explique la relación entre las incidencias y sus causas. Estas, en primera medida, se ubican en él o los elementos que han sido alarmados, y que se relacionan de alguna forma con dicha incidencia. Por tanto, si se encuentra la **Causa Raíz** de las alarmas en los elementos que las generan, al vincularlas con las incidencias, se tendrá la **Causa Raíz** de estas últimas. Uno de los inconvenientes relacionados con estas bases, era la falta de volumen y diversidad de datos, una vez hecha la limpieza y filtrado de los mismos.

Cabe recordar, que los datos utilizados son complejos, por ello a continuación, se presenta la explicación brindada por la empresa, acerca del contenido de cada columna, en la base de datos de alarmas.

Campo	Información representada
DataMinerID	Identificador del agente DataMiner donde se creó el elemento. En un enfoque práctico, corresponde a la instancia de la plataforma DataMiner donde se generó la alarma. En el caso de uso actual, los centros Ares y Santiago-Pedroso tienen un DataMinerID y Torrespaña otro diferente.
HostingAgentID	Identificador del agente DataMiner donde se almacena actualmente el elemento. En general, para los casos de uso actuales, es equivalente al DataMinerID.
ID	Identificador de la alarma.
RootAlarmID	Identificador de la alarma raíz. La alarma raíz es la alarma origen o inicial, de forma que las sucesivas evoluciones de dicha alarma tendrán el mismo RootAlarmID.
ElementID	Identificador del elemento alarmado
ElementName	Nombre del elemento alarmado
IsElementMasked	Booleano que determina si el elemento está "enmascarado" (se ignoran notificaciones asociadas durante cierto tiempo)
ParameterID	Identificador del parámetro (código numérico único)
ParameterName	Nombre del parámetro alarmado, según el tipo de alarma: - Alarma de equipo: Componente dentro del equipo que genera la alarma - Alarma de servicio: El nombre se establece a "Service State"
TableIndex	En caso de alarmas de equipos, identificador de la tabla asociada al valor del parámetro alarmado, en caso de existir.
DisplayValue	Valor a mostrar, según el tipo de alarma: - Alarma de equipo: Valor del parámetro que genera la alarma. - Alarma de servicio: Grado de afectación ('Normal', 'Risk', 'Anomaly', 'Degradation', 'Outage')
AlarmState	Severidad de la alarma ('Normal', 'Warning', 'Minor', 'Major', 'Critical')
Type	Tipo de alarma: - *New Alarm*: Nuevo evento de alarma (antes de este evento, el parámetro asociado no estaba en estado de alarma). - *Dropped from .../ Escalated from ...*: Alarma cuya severidad ha decaído/aumentado desde un nivel superior/inferior. - *Flipped*: Alarma cuya severidad ha pasado de "low" a "high" o viceversa. - *Service impact changed*: Alarma cuyo número de servicios afectados ha cambiado. - *View impact changed*: Alarma cuyas vistas afectadas han sido modificadas. - *Acknowledged/Unresolved*: Alarma que ha sido asignada/desasignada a un usuario de DataMiner. - *Mask/Unmask*: Alarma que ha sido enmascarada/desenmascarada por un usuario de DataMiner. - *Comment added*: Alarma a la que se ha añadido un comentario. - *Name changed*: Alarma causada por un parámetro cuyo nombre ha cambiado.

IsAggregation	Booleano que determina si la alarma es generada como agregado o agrupación de otras alarmas. Para el caso de uso actual, siempre será falso.
IsMasked	Booleano que identifica si la alarma está "enmascarada" (se ignoran notificaciones asociadas durante cierto tiempo)
Services	Nombres de los servicios que están afectados por la ocurrencia de la alarma actual
TimeOfArrival	Instante de tiempo en el que se genera la alarma, en formato GMT
TimeOfArrivalUTC	Instante de tiempo en el que se genera la alarma, en formato milisegundos desde *epoch*, UTC
RootTime	Instante de tiempo en el que se generó la alarma raíz asociada, en formato GMT
RootTimeUTC	Instante de tiempo en el que se generó la alarma raíz asociada, en formato milisegundos desde *epoch*, UTC
IsTrending	Booleano que indica si la tendencia media del parámetro está siendo monitorizada
IsOwner	Booleano que indica si, a nivel de operador, la alarma ha sido asignada.
OwnerName	Nombre del operador que se ha asignado la alarma, en caso de existir.
LastChangeUTC	PENDIENTE
IsCleared	Booleano que indica si la alarma está solventada

Figura 2.1: Explicación del contenido de las columnas en los Datos Originales de las alarmas

Sumado a esto, hay que tener en cuenta la información que contiene el data-set de incidencias. A cada una de las incidencias, las tenemos que vincular con las alarmas que se hayan presentado, sea por ventana temporal, por servicios afectados, o por ambos. Las alarmas se encuentran divididas en alarmas de servicios y de elementos. En el data-set de alarmas de servicios, tenemos los servicios alarmados que podríamos relacionar con las incidencias, sea por ubicación, ventana temporal, o alguna otra variable que vincule ambos data-set.

Por lo expuesto anteriormente, resulta relevante conocer los nombres de las columnas del data-set de incidencias, presentados a continuación.

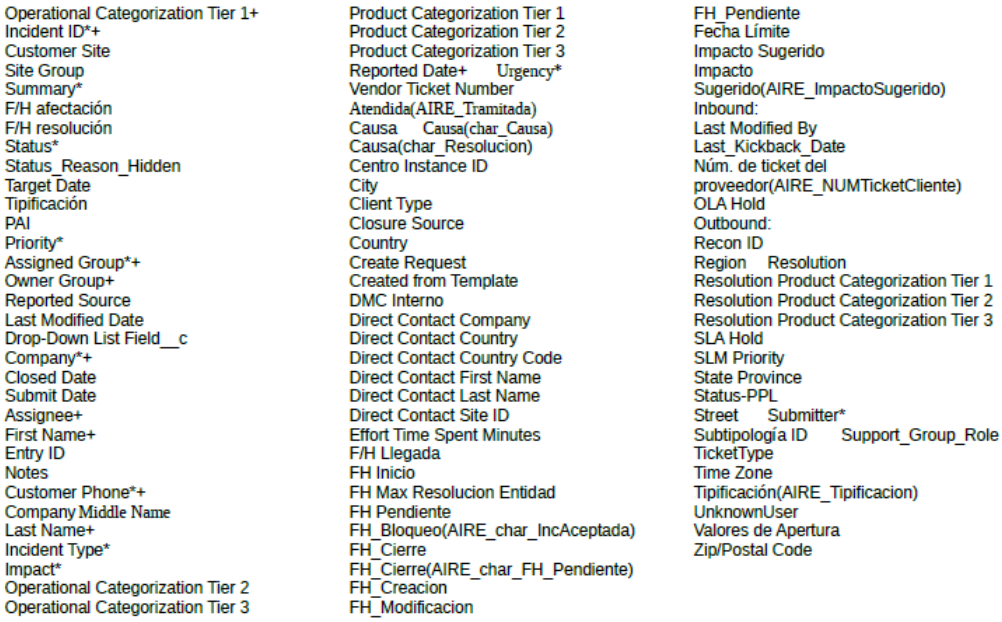

Figura 2.2: Nombres de las columnas de nuestro data-set de incidencias

Para mayor información acerca del contenido y complejidad de los datos, pueden encontrarse en los anexos, en la sección *Pantallazo del data-set de incidencias* A.2, unas imágenes de la cabecera de los data-set.

Aquí se puede ver, que existe mucha información que no sería de utilidad para los modelos, para vincular las incidencias con sus causas. Pero si había otras variables interesantes, como las de f/h de afectacion y f/h de resolución, para construir una primera ventana temporal. También se podría buscar, las alarmas que se correspondan con el mismo centro de la incidencia, por ejemplo, (contemplado en Customer Site). Además la información contenida en Summary, podría dar una idea de que habría pasado y con que equipo, aunque no en todos los casos. Por tanto, dicha variable no se podría usar como respuesta directa desde las incidencias, hecho que obliga a concentrar el modelado sobre las alarmas, para luego vincularlas con las incidencias, en base a estas y otras variables a definir. Debido a esto, y a la reciente advertencia que se había recibido, en cuanto a que las incidencias serían cambiadas por otras en otro formato de data-set, en un futuro cercano, es que a partir de aquí, se centrarán los esfuerzos en realizar un modelo que pueda clasificar, y posteriormente predecir, la variable OwnerName.

Se estima que en un siguiente paso, se usarían las alarmas de servicios. Ya que estas servirían para vincular las incidencias, con las alarmas de elementos. Cabe recordar, que en las alarmas de elementos se encuentran los componentes alarmados de la red que podrían estar presentando inconvenientes, según indica la incidencia relacionada. De ellos, interesa saber cual es el elemento padre donde confluyen esa serie de alarmas, para así poder detectar el elemento **Causa Raíz** de la incidencia, en principio. Se dice en principio, porque la **Causa Raíz**, podría estar en una conexión, y no en un elemento puntual, o ser una mezcla de varios elementos padre interrelacionados, por ejemplo. Por tanto en este primer estudio, se abordará la realización de un modelo matemático, que permita poner de manifiesto la red que subyace entre los elementos alarmados.

Como se dijo anteriormente, los datos reales no eran de la profundidad adecuada para el estudio, y además luego de la depuración de los mismos, se descubre que había bastante información solapada, y poca variabilidad. Hecho que se descubre, al realizar algunos estudios para evaluar interrelaciones entre las variables. También, se analizó la posibilidad de generar un análisis de series de tiempo, pero los datos no estaban tomados equidistantes, y aunque se los separara por períodos, se estaría perdiendo información detallada, al resumir los datos en franjas temporales. Más aún, como se mencionó anteriormente, los datos son categóricos, con lo cual una tendencia en las apariciones de los eventos, se podría ver solamente, a partir de analizar si una categoría aparece enlazada después, o antes que otra. Aunque, de encontrar dicha tendencia, se estaría dando una idea errada de la verdadera magnitud, y encadenamiento de los sucesos. Esto, debido a la poca variabilidad de los datos, a que los períodos en que se debería de fraccionar el estudio serían demasiado grandes, o demasiado pequeños, y al resumen realizado para poder agruparlos. Esto provocaría, que el estudio no sea aplicable a datos futuros. Lamentablemente la base original de alarmas, sumando las de servicios y elementos en la zona nombrada como ARES SANTIAGO por ejemplo, quedaba reducida a unos 60 registros. Este era un valor muy pequeño, para generar un algoritmo que aprenda de dichos datos, con la idea de poder trabajar luego sobre una supuesta base mas completa y compleja. Además, la estructura subyacente que los unía no era profunda, por tanto no hubieran permitido al modelo alcanzar buenos resultados a partir de ellos, y no se hubiera podido testar correctamente tampoco. Esto, en aras de modelar esa estructura de interrelaciones tan compleja, sobre la cual la empresa quería trabajar en un futuro cercano. Dicho conjunto de datos, contendría una gran cantidad de elementos, que brindan diferentes servicios, al enlazarse unos con otros. Por tal motivo se genera una base sintética, más completa y compleja, (con una interrelación más profunda), para poder analizar mejor su estructura. Para construirla, se tomó como punto de partida los data-set reales originales, por tanto los nombres de las columnas y el contenido al que se refieren es el mismo, aunque con la salvedad de que ahora la columna IsOwner indica si el elemento es padre de otro o no, y en OwnerName se tiene quien es el elemento padre del elemento alarmado, que en ese caso habría sido el responsable de dicha alarma. Esto se hizo, para incorporar la estructura de red subyacente directamente en el data-set, facilitando de este modo el manejo de los datos, más adelante esta búsqueda se realizaría a partir de un data-set adicional, que contendría una lista con todos los elementos de la red, sus datos de ubicación, sus relaciones entre ellos en cuanto a padres e hijos, y un listado de causas vinculadas con cada una de las alarmas que presenten.

Para preparar dicha base para el modelado, se hace un filtrado de los datos, que permita, por ejemplo, asignar una palabra clave a los NA, (se le asigna el término noinfo), para que estos registros no causen inconvenientes a la hora de ingresarlos al modelo. Además, dicha asignación sirve, para no perderlos de vista en caso de que su proporción resultara relevante en el total de datos. En la base real, había muchos datos de este tipo, y es por eso que se volvieron una cuestión importante a considerar en el análisis. También se pueden destacar, algunas salvedades en cuanto a los datos, y a su ingreso dentro de los modelos. Es conveniente transformar una variable de tipo carácter en tipo factor, antes de ingresarla para el cálculo del modelo, ya que de esta forma el modelo demorará menos en calcularse, por no tener que realizar esta tarea internamente. Al convertirla a factor, lo que se logra es poder tener ya dividida la variable en las categorías que la conforman. Cuando se trata de modelos de clasificación, a realizar con randomForest, si las variables ya están como tipo factor, esta puede contener, solo, entre 2 y 53 niveles, (cantidad de categorías). Si posee un nivel solo, significa que para todos los datos el valor de dicha variable es el mismo, y por tanto no aporta información adicional al modelado. Si es superior a 53, el calculo del modelo falla, entonces aquellas que tengan un número superior a 53 niveles, es conveniente dejarlas como variables de tipo carácter, y no de tipo factor, para que la herramienta calcule ella misma los cortes en las categorías. Algo similar ocurre para los modelos con redes neuronales, pero en este caso no pueden manejar variables con más de 200 niveles, y por ende se recomienda utilizar la misma metódica que antes, para que el cálculo del modelo se pueda realizar sin inconvenientes. Una vez logrado el preparado de los datos, se procede a la búsqueda de un modelo, que permita encontrar las relaciones existentes entre las variables independientes y la respuesta elegida. Como se mencionó anteriormente, la variable respuesta elegida es OwnerName, que

señala el elemento padre, o elemento al que pertenece el elemento alarmado.

2.2. Cálculo de los modelos

2.2.1. Modelos óptimos con selección manual de variables

Inicialmente, se intentó realizar un modelo con un único árbol de clasificación, dada la naturaleza de los datos, pero la herramienta utilizada para crear dicho árbol, (rpart a través de caret), falló repetidas veces, y no lograba generar un modelo, el programa se quedaba congelado. Por ende, luego de probar otras alternativas en cuanto a variaciones en los datos, como por ejemplo:

- convertir todas las variables a factor, antes de ingresarlas para su modelado. Como se explicó anteriormente, esto ayuda al algoritmo de cálculo, porque evita que tenga que realizar la propia función, dicha tarea internamente.
- Apoyar el modelo en variables dummy, transformando cada categoría, de la respuesta y de las explicativas, en dicotómica, antes de ingresar los datos para su modelado. Esto evitaría que el algoritmo tenga que realizar dicha tarea internamente.
- Acotar la cantidad de datos, pero asegurando que todas las categorías estuvieran presentes, aunque con menos repeticiones.
- Utilizar el árbol de clasificación, pero llamando a la herramienta original de forma directa (rpart), en vez de llamarlo a través de caret.

Ninguna de estas idea funcionó como se requería, y por tanto se decidió utilizar la herramienta randomForest, para R, pero por fuera de caret. Dicha herramienta trabaja con el método de bosques aleatorios, lo cual como se ha comentado anteriormente, ayuda a mejorar el modelo al incorporar variabilidad no solo en los datos, sino también en las variables de corte. Se trabajó por fuera de caret, en este caso, porque al llamar a la función a través de caret tampoco funcionaba, tal como le ocurría a rpart. De esta manera, se logró generar un primer modelo, utilizando la mencionada herramienta. Para este primer modelo, se dejaron los parámetros de la función en su estado por defecto. En este caso, se estaban utilizando todas las variables independientes como variables predictoras, pero esto no estaba dando buenos resultados. Al evaluar el modelo con las muestras de test, se observaban índices de precisión sospechosamente altos, en torno a 1. Esto ocurre cuando las variables independientes están correladas, es decir que vuelcan información solapada al modelo. También, puede deberse a cuando se está utilizando variables numéricas poco correctas, porque son variables que no se repiten, o por que su evolución, (incremento o disminución), no condiciona a la variable respuesta que se está analizando. Un ejemplo es el número de orden, ya que en ese caso el modelo utiliza dicha variable como un predictor válido, que colabora a independizar el resultado obtenido, al igual que ocurre con las fechas precisas, (dia,mes,año,hora,minuto,segundo). Tanto el orden, en nuestro caso indicado por un id único de evento, y la fecha precisa, no solo no se volverán a repetir, sino que su evolución no condiciona a la variable respuesta que buscamos, que es la **Causa Raíz**, señalada a través del elemento padre de aquel, que produce la alarma, y que está involucrado en la incidencia. Por eso, luego de analizarlo, se eliminaron aquellas variables como las que indicaban el momento exacto en que sucedió la alarma. Porque estas no iban a ser variables que pudieran volver a repetir sus valores, (si se tomaban como carácter), y tampoco si se las tomaba como numéricas, porque en las alarmas, este número siempre iba a ir en aumento, y solo serviría si, por ejemplo, nos hubiera indicado que existía algún patrón repetitivo de su aparición. Esta última suposición había sido sugerida desde la empresa, pero fue descartada luego de su análisis. Se estudió, mediante la generación de una variable alternativa, que midiera en intervalos de tiempo equidistantes, la periodicidad de los diferentes eventos. Ya que la recolección de estos datos, no había sido realizada en períodos equidistantes, y este es uno de los requerimientos para analizar una serie de tiempo. Así, se buscó evaluar si existía algún tipo de patrón repetitivo, o serie temporal, buscando si efectivamente aparecían alarmas del mismo tipo, o alguna cadena de alarmas cada cierto período de tiempo. Pero no era el caso en estos datos, el patrón expuesto no era tal, no exponía una aparición repetitiva, ni ninguna relación entre el tiempo transcurrido y las alarmas que se presentaban. Dada

la mencionada característica de las variables de entrada, que son de tipo categóricas, podría haber ocurrido que mostrara una especie de sucesión recursiva de ciertos tipo de alarmas, o de incidencias incluso, (que son un conjunto de alarmas). Pero en este caso, lo que se veía, era una serie de puntos que no exhibía formato alguno que indicara que luego de una alarma venía otro de tal otro tipo, o que un tipo de incidencia iba luego de otra, o algo similar. En los anexos, en la sección *Análisis y Modelos óptimos a mano* B, figura cual ha sido la codificación utilizada, tanto para la variable respuesta, como para las variables explicativas del modelo. Además, en la sección *Análisis de la variable respuesta en relación con el tiempo* B.1, aparecen los grafos que, se estima, son la evidencia de la ausencia de relación, entre la variable respuesta y el tiempo.

Como se comentó anteriormente, se contaba solo con las alarmas, y no con como se relacionaban estas con las incidencias, entonces se decidió avanzar en la búsqueda de un modelo, que pudiera señalar al elemento padre de aquel alarmado, como posible **Causa Raíz** de la alarma. Así, una vez que se tenga la relación de las alarmas con las incidencias, se procedería a asignar a dicha incidencia sus posibles causas. El modelo señalaría, el o los elementos padre, donde confluyen los elementos alarmados vinculados a dicha incidencia. Esta sería, una primera aproximación al RCA, dado que en la base de incidencias no se cuenta con dicha información histórica, y de esta forma se la estaría creando, para poder usarla en el análisis de **Causa Raíz**. Pero para eso, es importante poder elegir correctamente las variables de entrada, ya que en los datos históricos no siempre está a la vista la relación de padre e hijo que se está buscando. Por este motivo, se continuó analizando que variables incorporar y cuales no, para lograr un modelo mas sencillo. Puesto que, el inicial lograba buenos resultados, pero no los esperados, y tenía una complejidad enorme debida a la cantidad de variables involucradas, sumado a que probablemente estarían existiendo correlaciones entre variables que empeoraban el panorama. En los anexos, en la sección denominada *Modelo a mano randomForest* B.2, se puede encontrar el detalle del modelo inicial con todas las variables, y del modelo alcanzado, luego de la selección a mano, de las variables de entrada.

Como se comentaba anteriormente, se realizó la selección de variables, a mano, para arribar a un modelo menos complejo, que tenga buen desempeño, y que no contenga información repetida en las variables de entrada. Esto se logró mediante una búsqueda paso a paso, donde se fue combinando una a una las variables del modelo, y una vez combinadas de a 2, se pasó a combinarlas de a tres, y así sucesivamente. Se trataba de mantener en la combinación siguiente, aquellas variables que mejor desempeño habían obtenido hasta el momento. Es decir, aquellas que daban un índice Kappa alto, (al evaluar el modelo en la muestra de test). Se usó este índice, y no el Accuracy, porque en este caso se trata de una variable respuesta con muchas categorías, y donde además la muestra, (tanto de entrenamiento, como de test), está desbalanceada. Esto significa, que no se tienen cantidades similares de cada categoría, sino que unas pesan mucho y otras muy poco en la muestra. A su vez, también las variables explicativas contienen muchas categorías, que presentan el mismo tipo de característica, esto contribuye a que sea más difícil controlar cada una de las categorías de la respuesta, por separado. Por eso, se eligió el Kappa como medida representativa. Además, al evaluarlo sobre la muestra de test, se está utilizando un índice que incorpora aleatoriedad, porque estoy evaluando en la muestra test, y no mediante el OOB, que saca el propio modelo, cuando se calcula sobre la muestra de entrenamiento por validación cruzada. Este último punto se advirtió cuando, en un principio, se estuvo comparando modelos, buscando el de menor OOB, (error en la muestra de entrenamiento). Al validar el modelo sobre la muestra de test, sus índices de asertividad eran peores que los de otros modelos, que alcanzaban mejores resultados en las mismas condiciones, a pesar de tener un OOB superior. A continuación, se expone una tabla que muestra algunos ejemplos de lo comentado.

Variables del Modelo	Accuracy	Kappa	OOB
Todas las variables	0.8178711	0.7683046	0.04345173
Todas menos IsMasked	0.8257	0.7784	0.04589284
Todas menos IsMasked e IsAggregation	0.8213	0.7727	0.04442817
Todas menos IsMasked e IsAggregation e IsElementMasked	0.8242	0.7764	0.04332967
Con 19 variables	0.8394	0.7957	0.07945807
ElementName, ElementId y TableIndex	0.8989	0.8722	0.1009398
ElementName, ElementId, IsOwner y TableIndex	0.9018555	0.8759034	0.09752228

Cuadro 2.1: Tabla de Accuracy, Kappa y OOB de algunos modelos a mano con randomForest

En el cuadro precedente, se puede ver el Kappa en comparación con el OOB. Allí se expone, que el modelo final respecto de sus predecesores, es uno de los que peor OOB posee. Sin embargo, en cuanto a Kappa, dicho modelo es superior a sus predecesores. Aunque la distancia entre estos índices no sea grande, el modelo también se considera mejor por el hecho de poseer muchas menos variables, con lo cual también se gana en sencillez. Al cabo de un par de semanas de búsqueda, se logró arribar al modelo con 4 variables explicativas, que figura en la última línea de la tabla precedente. Para ello, se probaron, una a una, todas las variables que se consideraban más importantes, se ensayó sobre sus posibles combinaciones, y se realizó el refinado de los hiperparámetros del modelo. Además, se disminuyó la cantidad de árboles a 100, porque ya en dicho número, se veía estable el índice OOB, para cada una de las categorías de la respuesta. El mtry óptimo para dicha búsqueda resultó en el valor 4, por tanto se trata de un modelo de bagging, por ser coincidente con la cantidad de variables predictoras presentes en el modelo. El nodesize óptimo fue de 10. A continuación se presentan las variables que incorpora este primer modelo, hecho con la herramienta randomForest, y los índices de dicho modelo, al validarlo sobre la muestra de test:

- OwnerName, que es la variable respuesta
- ElementName
- ElementID
- IsOwner
- TableIndex

Índices de desempeño del modelo calculados sobre los datos de test.

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
0.9018555	0.8759034	0.8881452	0.9144025	0.3437500	0.0000000	NaN

Cuadro 2.2: Índices mejor modelo a mano randomForest

Luego en los anexos en la sección *Modelo a mano randomForest* B.2, se encuentran algunos datos adicionales sobre el modelo como el grafo OOB, y la importancia de las variables utilizadas.

Como se comentó anteriormente, una vez presentado este modelo, desde la empresa solicitan que se realicen otros modelos alternativos, con otras técnicas de modelado, para poder contar con varias opciones, y ver cual se desempeñaba mejor. Entonces, utilizando los conocimientos adquiridos con el primer modelo, se logró generar un segundo modelo, que obtiene resultados, tan buenos, como el primero. Esta vez, se utilizó el paquete de R llamado party, que realiza un árbol de clasificación, utilizando contrastes de hipótesis para su construcción. Dicha herramienta, analiza el nivel de significación, de la incorporación de cada variable, como variable de corte del árbol. Su metodología de trabajo puede encontrarse descripta en mayor profundidad, en la sección Modelando la estructura subyacente 1.4. Como el gráfico del árbol party sale muy confuso, se expone además, las reglas del árbol para que se pueda ver la clasificación realizada claramente. Ambos detalles pueden encontrarse en los anexos, en la sección denominada *Modelo a mano Party* B.3. A continuación, se exponen los indices y variables del modelo party, que resultaron ser los mismos que los del modelo randomForest. El hecho de que los índices del modelo party, que cuenta con un solo árbol de clasificación, sean iguales que los de randomForest, podría estar indicando que a pesar de hacer bagging con randomForest, la variable para el primer corte de cada árbol haya sido la misma en la mayoría de los casos, y que por tanto los árboles hayan tenido muy poca variabilidad entre ellos, coincidiendo así con el modelo de un solo árbol, que se construyó con party.

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
0.9018555	0.8759034	0.8881452	0.9144025	0.3437500	0.0000000	NaN

Cuadro 2.3: Índices mejor modelo a mano party

Al presentar este segundo modelo, desde la empresa solicitan un tercer modelo, pero que esta vez se basara en la técnica de redes neuronales. Este tipo de modelos son bastante complejos, dada su manera de trabajar. En breves palabras, estos modelos incorporan diferentes transformaciones a las variables de entrada, para poder encontrar la relación entre estas y la variable respuesta, permitiendo predecir, a esta última, con la mayor exactitud posible. Por este motivo, es que requieren una gran cantidad de datos para trabajar, y por eso son conocidos como exitosos para el análisis de big data, en donde, además, las relaciones entre las variables sean complejas. Para este último modelo, se eligió como punto de partida, las mismas variables que se habían usado anteriormente como punto de partida, tanto para el modelo generado con randomForest, como para el modelo con party. Dichas variables, habían demostrado ser útiles en ambos casos, y además el hecho de que randomForest permitiera ordenarlas por nivel de importancia, era un dato útil adicional, para orientar la búsqueda de las posibles combinaciones. Esto se hizo como alternativa de eficiencia, para evitar tomar todas las variables que traía el data-set. Con este paso, se lograba acotar un poco la búsqueda, dado que el primer modelo de redes demoró una tarde en entregar un resultado. Este hecho evidenciaba, la gran exigencia computacional de este tipo de modelos. Es importante destacar, que cada ejecución del algoritmo para modelar una red neuronal, con un conjunto nuevo de variables, demoraba entre 7hs y 24hs en generar el modelo, dependiendo de cuantas variables se querían incorporar, y cuan complejas eran. Este procedimiento era bastante conflictivo, dado que, para cualquiera de las tres herramientas utilizadas, mientras mayor sea el número de variables incorporadas, y/o exista un mayor número de categorías, en las que están divididos los predictores, se incrementará el tiempo de computación, y la exigencia computacional para el cálculo del modelo. Y realizar esta selección de variables, era vital para conseguir un modelo matemático eficiente. Curiosamente el resultado final alcanzado, no difería demasiado del obtenido con el modelo de bosques aleatorios, que además, era un modelo mucho mas sencillo, y rápido de calcular.

A continuación se expone una comparación de los índices de asertividad de los modelos generados con las 3 herramientas. Empezando con caret (NNet), el primer conjunto de índices que aparece es con la variable dist, y el segundo es sin ella, el tercer conjunto de índices expuestos, son los pertenecientes al modelo calculado con el paquete party, y el cuarto conjunto de índices, son los que pertenecen al modelo calculado con el paquete randomForest. Para cada una de las tres herramientas, se ha realizado la selección de variables a mano.

```
Comparación de los índices de asertividad de los modelos a mano, generados con las 3 herramientas descritas.

Modelo NNet con variable dist
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue  McnemarPValue
     0.9151786      0.8925309      0.9021559      0.9269782      0.3263889      0.0000000            NaN
Modelo NNet sin variable dist
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue  McnemarPValue
     0.9171627      0.8951300      0.9042667      0.9288294      0.32787700      0.0000000            NaN
Modelo con party
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue  McnemarPValue
     0.9018555      0.8759034      0.8881452      0.9144025      0.3437500      0.0000000            NaN
Modelo con randomForest
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue  McnemarPValue
     0.9018555      0.8759034      0.8881452      0.9144025      0.3437500      0.0000000            NaN
```

Hasta aquí, el modelo que mejor funciona es el de redes neuronales, sin la variable dist. Se puede ver, que su índice Kappa es superior en 0.02 puntos a los modelos de randomForest y party. A continuación, se detallan las variables que incorpora el mejor modelo de red neuronal:

- OwnerName, que es la variable respuesta
- ElementName
- ElementID
- IsOwner
- TableIndex
- ParameterName
- Services

Cabe aclarar que la variable dist, mencionada anteriormente, recogía la distancia transcurrida en segundos, entre alarmas de la misma categoría en la variable respuesta, (mismo OwnerName). Así, se buscaba aprovechar la relación temporal entre los eventos de una misma categoría, dentro de la respuesta, para poder mejorar el desempeño del modelo. Pero luego de realizar algunas pruebas, se logró advertir, que efectivamente no aportaba mayor información. Además, los índices del modelo mejoraban, en una mínima cantidad, al sacarla. Más aún, al tener una variable menos, se ganaba simplicidad. Por este motivo, el modelo a mano de redes, que finalmente se presentó, no la contenía, y sus índices eran los siguientes.

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
0.9171627	0.8951300	0.9042667	0.9288294	0.32787700	0.0000000	NaN

Cuadro 2.4: Índices mejor modelo a mano NNet

Los detalles de esta red final, se encuentran en el anexo denominado *Modelo a mano NNet* B.4. A modo de resumen de sus principales características, se puede mencionar que, el número de hiperparámetros o pesos es 597. La cantidad de nodos en la capa de entrada es 275, que se corresponde

con la cantidad de categorías incluidas en el modelo, a través de las variables de entrada. Hay 2 nodos en la capa oculta, 15 nodos en la capa de salida, y la tasa de aprendizaje óptima, (decay), es de 0.001. Además, por tratarse de variables categóricas, el modelo utiliza la función softmax, como función de activación en los nodos finales. Para comprender mejor, que función cumple cada una de estas características en el modelo, remitirse a la sección *Modelando la estructura subyacente* 1.4.

Como se comentó en las secciones anteriores, una vez presentados estos tres modelos, desde la empresa, indican que se pretendía ponerlos a competir, sobre una base de datos completamente nueva, para ver cual de ellos, obtenía los mejores resultados. La intención era utilizar el modelo elegido, para brindar servicio a diferentes clientes. Este hecho, como se mencionó anteriormente, no era posible, sin una etapa de ajuste previa. Ya que un modelo, es aplicable sobre datos nuevos, siempre que estos no difieran en cuanto a estructura y contenido, de los que se han usado para entrenar el modelo. Teniendo en cuenta las diferentes bases de datos que se habían analizado, y la intención mencionada, se presumía que estas condiciones no se cumplirían. Por tanto sería necesario, que la empresa realizara esta búsqueda manual, de la mejor combinación de variables para construir el modelo, cada vez que cambiara la estructura de los datos. O que al menos, se tuviera que volver a entrenar el modelo con los datos nuevos, si es que había variaciones en cuanto a las categorías involucradas. Este último punto, no se podría evitar, ya que es requerido en caso de que existan nuevas categorías en las variables. Pero se tenía la intención, de acelerar el trabajo de búsqueda de la mejor combinación de variables, para generar un modelo con las técnicas utilizadas.

Así, surge la necesidad de crear una herramienta, que permitiera hacer toda esta búsqueda más rápido, y de manera mas eficiente. Dicha herramienta, se presentará a continuación, en los siguientes apartados de este TFM.

2.2.2. Algoritmo automático para la búsqueda del mejor modelo aditivo

Luego de una exhaustiva investigación, se logró desarrollar un algoritmo, que a través de la combinatoria, permitiera ir probando todas las combinaciones posibles de las variables de entrada que se le incorporaran. Calculara el modelo para cada una de ellas, lo testara con la muestra de test, y volcara los resultados de precisión y Kappa en un data frame. Se utilizó ambos índices, por que si se trata de muestras des-balanceadas, como fue el caso en estudio, el Kappa es un índice más fiable en cuanto a rendimiento del modelo, cosa que fue explicada anteriormente. Una vez obtenido dicho dataframe, con todos los índices para cada combinación de variables, se busca dentro de él cual es la combinación de variables que mejor Accuracy y Kappa ha obtenido. Se testea si es la misma fila para ambos índices, y se la extre en una variable. En caso de que sean filas diferentes, se extrae la de mejor Kappa. Con ello, se obtiene los nombres de las variables explicativas a considerar en el modelo, y se realiza un primer modelo básico con ellas.

Como paso posterior, para mejorar el modelo obtenido, se utiliza un algoritmo para refinar los hiperparámetros del modelo. Dicho algoritmo, fue creado durante las clases de la materia Aprendizaje Estadístico, y modificado para este propósito. Para mayor detalle, puede consultarse *Fernández Casal R., et al. (2021)* [5]. Allí se busca cuál es la combinación mas acertada de los hiperparámetros, para obtener el mejor modelo posible con ellos. A continuación, se comparan los resultados obtenidos en ambas instancias, y se elige el modelo que brinde mejores resultados. En casos donde las categorías no son tantas, se puede realizar un análisis con mayor detalle en cuanto a sensibilidad y especificidad, en aras de obtener el mejor modelo posible, según las necesidades del caso. Además, hay que recordar, que se debe evaluar el modelo, en cuanto al cumplimiento de las hipótesis estructurales que le sean aplicables, según el caso.

El algoritmo desarrollado para la búsqueda automática del mejor modelo, a través de buscar la mejor combinación de variables, fue creado tres veces:

- para obtener el mejor modelo de bosques aleatorios, con la herramienta randomForest.
- Para generar el mejor modelo de árbol de decisión, utilizando contrastes de hipótesis, en base al nivel de significación, utilizando la herramienta party.
- Para calcular el mejor modelo de redes neuronales, utilizando la herramienta NNet, a través de caret.

Se desarrolló de este modo, por que al cambiar el método de modelado a utilizar, y por consiguiente el paquete empleado para su cálculo, se deben modificar algunos pasos internos del algoritmo para su correcto funcionamiento. Además, existen algunas restricciones en cuanto a la preparación de los datos, que difieren según la técnica elegida. El modelo entregado por el algoritmo automático, se almacena en un objeto, tal como si se hubiera generado manualmente. Esto permite trabajar sobre él, todo lo que se considere necesario, con los métodos ya conocidos.

Explicación en forma de esquema de como funciona el algoritmo:

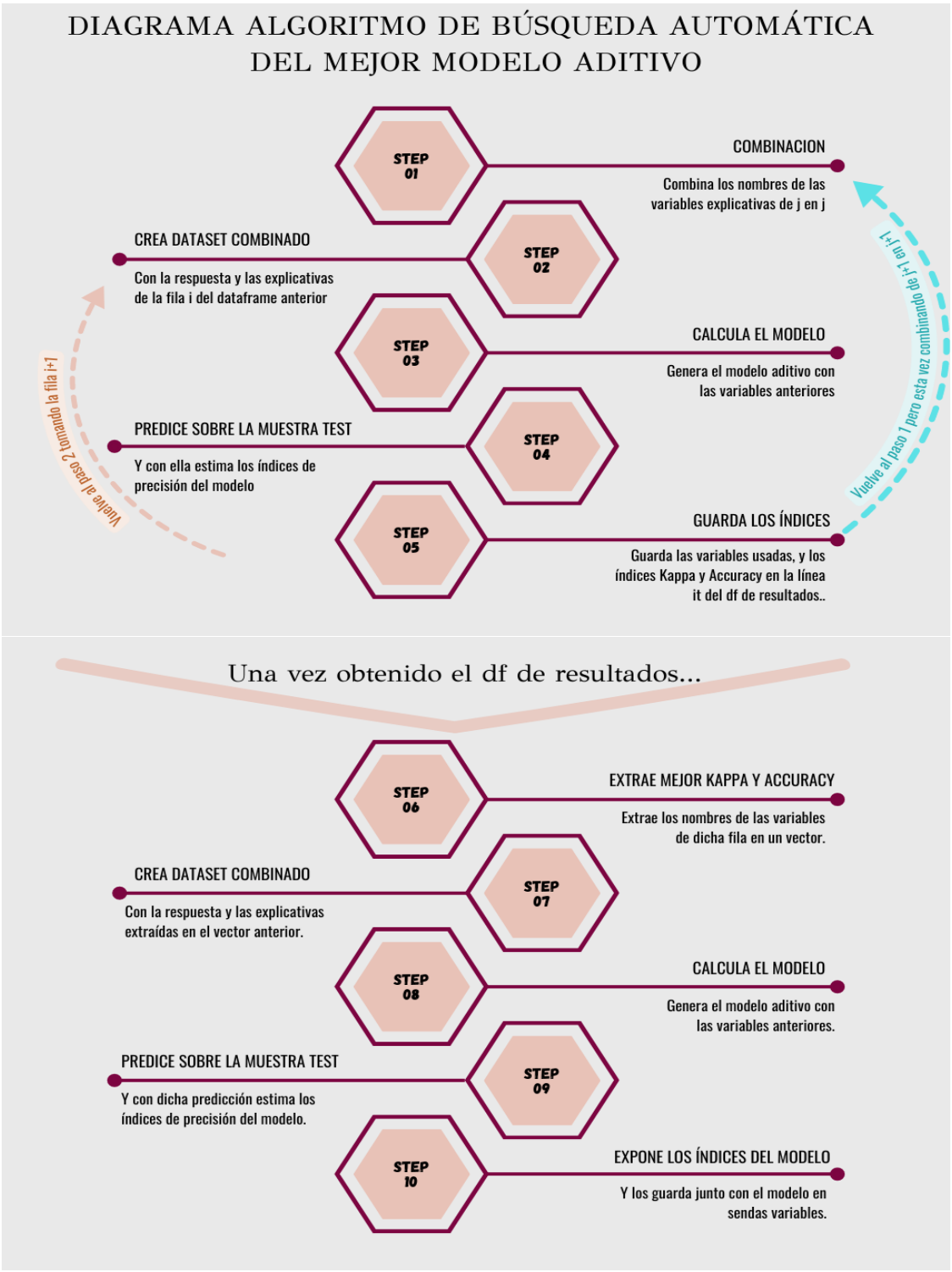

Figura 2.3: Diagrama del Algoritmo de Búsqueda del mejor modelo aditivo básico para randomForest - Party - NNet (Producción Propia)

Tal como se comentó anteriormente en la sección *Modelando la estructura subyacente* 1.4, para la realización de este TFM se ha trabajado en un servidor que pertenece a la empresa, con un espacio de memoria especialmente dedicado. Este hecho, permite dejar el algoritmo ejecutándose durante varios días, sin inconveniente. Dicho recurso no es pequeño, pero a la vez si se le exigen modelos muy complejos, colapsa por falta de espacio. Dicho espacio, lo requiere para almacenar los datos previos que va generando al calcularse. Este fue uno de los inconvenientes, a tener en cuenta, a la hora de construir estos algoritmos. Por salvar dicho inconveniente, en el caso de randomForest por ejemplo, se le baja la cantidad de árboles que realiza, para modelar con los distintos conjuntos de variables seleccionadas. Además, se le dejan los demás parámetros del modelo, con sus valores por defecto, salvo que se detecte que se están generando inconvenientes. En ese caso, se refinará alguno de ellos. Una alternativa en caso de inconvenientes, es recortar la profundidad de los árboles. También es importante destacar, que en el caso de redes neuronales, se trabajó con un modelo de una sola capa, es decir que se trata de una red básica. Al analizar los resultados obtenidos con este modelo, y dado que:

- para calcular este tipo de modelo se requirió, en este caso, que el algoritmo trabaje todo un día completo.
- El hecho de generar un modelo más complejo, implicaría mayores exigencias computacionales y de tiempo.
- Se habían probado algunos cambios en los hiperparámetros del modelo, pero por falta de memoria no se lograban almacenar los resultados, perdiendo los avances realizados. Esto fue, debido a que el algoritmo fallaba por la cantidad de pesos, y terminaba en un error donde los valores de los índices del modelo quedaban en cero. En otros casos, se quedaba congelado el programa, y al reiniciarlo se perdía todo el avance realizado en el cálculo del modelo.
- Si se quisiera incrementar la complejidad del modelo, este hecho representaría una mayor inversión para la empresa.

Por todo lo comentado anteriormente, se decidió no incrementar el número de capas del modelo. Se analizó, que para el caso en estudio, no representaría una mejora significativa, que justificase incurrir en lo antedicho.

Parámetros a considerar en los modelos

En este segmento, se describirá que parámetros se mantienen y cuales se modifican, para cada tipo de modelo, en las distintas instancias del algoritmo generado. Además, se acompañarán con una breve referencia, que justifica la decisión tomada en cada caso. En principio, se describirán los pasos realizados para preparar los datos, para su incorporación al algoritmo de búsqueda automática.

Como se comentó anteriormente, como primer paso se realiza una depuración de los datos: se asignan los NA a una categoría especial, llamada "noinfo", se eliminan los registros duplicados, se le quita la cabecera doble y la columna con el número de orden.

En un segundo paso, se convierten las variables a tipo factor. Para generar un modelo con randomForest, las categorías en las que puede estar dividida la variable de tipo factor, son como mínimo 1 y como máximo 53. Como se comentó brevemente en la sección *Minería de Datos* 2.1, si la variable contiene solo una categoría, conviene excluirla del modelo, porque estaría indicando que todos los datos presentes contienen esa variable con ese único nivel, y por tanto no aporta información relevante, para poder individualizar la salida que se pretende predecir en base a ella. En el otro extremo se tiene que, si la variable contiene más de 53 categorías, es conveniente dejarla en tipo carácter para que el modelo se encargue de generarle los cortes que crea convenientes, y no falle en la ejecución. Este límite superior, para el caso de redes neuronales es de 200 categorías.Como será visto en detalle más adelante en esta misma sección, en el caso de estudio, inicialmente, se mantuvo el límite de 53, para el cálculo de todos los modelos. Con estos datos, se genera el modelo base, del cual se escogen las variables explicativas, en función del nivel de importancia de las mismas. A continuación, se incorporaban dichos datos a cada uno de los algoritmos de búsqueda, para generar cada uno de los modelos. Pero luego de algunos inconvenientes con el algoritmo de party y el de red neuronal, se decantó por adaptar los datos. Este hecho se encuentra detallado, al final de esta misma sección, en el apartado *salvedad para el modelo de redes neuronales* 2.2.2.

En este paso, una vez realizada la depuración de los datos, y convertidas las variables a factor, se calculan las muestras de entrenamiento y test, fijando una semilla arbitraria para garantizar la reproducibilidad del estudio, es este caso se fija el número 1. Se divide la muestra en un 80 % para entrenamiento (del ingles train) y un 20 % para test, esto en el algoritmo original se hacía directo, pero generaba inconvenientes en relación a los niveles presentes en las variables de tipo factor. Esto se debía a que, no siempre eran los mismos niveles en la muestra de entrenamiento, que en la muestra de test. El error se refería, a que no se puede calcular un modelo sobre clases vacías, ni tampoco el modelo puede predecir, si alguna de las variables presentes en su interior, posee una cantidad diferente de niveles de los presentes en los datos a predecir. Por eso, en los sucesivos usos, esta parte del algoritmo se mejoró. Se buscó tener en cuenta las categorías de la variable respuesta, para que todas se hallen reflejadas en la proporción que les corresponda, según su peso, tanto en la muestra train, como en la muestra test. Estos pasos se realizan con la herramienta dplyr, mediante el comando group_by, y de esta forma se agrupa en cuanto a los niveles de la variable respuesta. Además, cuando las variables explicativas del modelo son de tipo factor, va representando cada categoría de las explicativas, dentro de la respuesta. La opción para corroborar si los nuevos datos tienen el mismo formato, en cuanto niveles, que los usados en el entrenamiento, es aplicarle a los nuevos datos los niveles que presenta el modelo en el apartado xlevels. De esta forma se sabrá, si los niveles que contienen los datos son coincidentes. Si son distintos, se tendrá que volver a entrenar el modelo, con estos nuevos datos, para que contemple los nuevos niveles que se encuentran presentes en la información actual. Así podrá clasificarla correctamente, y predecir mejor los futuros datos, evitando que se genere un error en la ejecución del modelo, relacionada con este hecho.

Este paso es la fase previa a correr el algoritmo de búsqueda automática de la mejor combinación de variables, para cada modelo. Aquí se realiza un modelo randomForest, con aquellas variables explicativas incluidas en el data-set, que se consideren útiles luego de la depuración. De la salida de dicho modelo, se extrae la importancia de las variables, con la herramienta importance (del paquete randomForest). Las primeras 10 variables que aparezcan allí, serán las elegidas para filtrar el data-set de entrenamiento, y dejarlo preparado para ejecutar el algoritmo de búsqueda que sigue. Los datos

vinculados a este primer modelo pueden verse en los anexos, en la sección *Modelo general base, para randomForest con algoritmo automático* B.5. Este primer modelo, es el que se ha utilizado como guía y punto de partida para ejecutar el algoritmo de búsqueda automática, para encontrar el mejor modelo con el paquete randomForest. Para el modelo con Party, y para el modelo con NNet, se ha usado otro modelo base, que aparecerá explicado más adelante, en los párrafos que describen la *salvedad para el modelo de redes neuronales* 2.2.2.

Todos estos pasos previos, se realizaron para los tres tipos de modelos. Con ellos, se buscaba unificar las variables elegidas como punto de partida, y permitir la comparación de los distintos tipos de modelos generados, mediante sus índices de desempeño, calculados sobre la muestra de test.

Salvedad para el Modelo de Redes Neuronales

En este punto, dada la cantidad de pesos con la que trabaja el modelo, y la gran cantidad de categorías presentes en las variables, se ha tomado una acción para evitar los fallos que se estaban presentando. Dichos errores contenían la leyenda "Too many weights NNet" para la variable "DisplayValue". En dicha variable, se evaluó lo siguiente: La salida "fácil", hubiera sido intentar aumentar el parámetro MaxNWts. Este parámetro limita cuantos pesos usa la red, pero lo ideal es analizar la variable, y ver qué información relevante está entregando. Al incrementarlo, como se comentó anteriormente en la sección *Modelando la estructura subyacente* 1.4, conlleva que el modelo se vuelva muy complejo, lento y más difícil de manejar. En este caso, la variable contiene parámetros numéricos y parámetros de texto, que por ejemplo presentan el texto: "OK", "Responding", "Not Responding", etc. Por este motivo, se decidió separar la variable "DisplayValue" en dos variables. Una recoge los valores numéricos "DisplayValue_num", y la otra los valores de texto "DisplayValue_txt". Como los valores numéricos eran en kHz, entonces todas las entradas que exhiben valores en "DisplayValue_num", tendrán dicha sigla en "DisplayValue_txt". Para completar "DisplayValue_num", en aquellos lugares donde no hay valor numérico, se le asigna valor cero, y de esta forma todas las que tengan valor 0, podrán ser clasificadas en función del valor texto que figure en "DisplayValue_txt", si es que lo poseen. Por que, en principio se estima que, en esos casos va a utilizar como variable para clasificar la variable que contiene el texto explicativo. De esta manera se estaría introduciendo, en principio, dos variables correladas al modelo, para el caso en que se tiene texto en la variable "DisplayValue_txt", y valor 0 en la variable "DisplayValue_num". Pero existen otros casos, donde no hay datos en ninguna de las dos, y casos donde existe texto distinto de kHz, y a la vez un valor distinto de 0 para "DisplayValue_num". Es para poder clasificar estos datos, que en principio se trabaja con ellas. Ya que, el hecho de estar correladas, provocaría una sobreestimación del modelo. Pero se realiza, con la idea de que al evaluar la salida del algoritmo, se pueda elegir aquella combinación que mejor índice tenga, y que a la vez posea las variables que no estén correladas, o cuya correlación sea muy baja, de forma que influya lo menos posible en el modelo. Además, después de algunas pruebas adicionales, se decide establecer la variable creada recientemente "DisplayValue_txt", a tipo factor. Esto, para evitar un error recurrente en cuanto a la selección de las muestras de entrenamiento y test. Dicho error se provocaba, porque el algoritmo de generación del modelo, encontraba diferentes niveles en la muestra de test, de aquellos con los que se había entrenado en la muestra train. Este hecho le impedía continuar con la ejecución, y calcular los índices de asertividad del modelo sobre la muestra de test. Al realizar un modelo de randomForest general con esta variable dividida en dos, se comprobó que las clasifica, a ambas, dentro de las 10 más significativas, al igual que también colocaba a la original dentro de dicho conjunto, por ende esta maniobra, estaría funcionando como se esperaba. Con ella, se busca volver más manejable a la variable tratada, en cuanto a los hiper parámetros, y a sus pesos. Ya que, en este caso, se estaba hablando de unas 837 categorías, a las cuales se le aplicaban diferentes transformaciones, y pesos, para generar los hiperparámetros, que acompañarían al modelo, volviendo a la Red neuronal generada, virtualmente inmanejable.

A continuación, se logró comprobar que existe otra variable, que para el modelo de redes está generando errores. Se trata de "Services", dado que contiene 171 niveles, y esto es mucho más que 53, (que es lo máximo admitido por randomForest). Por ende, esta variable se ingresaba en tipo texto en el modelo general, para la selección de variables, pero este hecho le estaba trayendo inconvenientes a la generación posterior de la red. Al tomar la variable de esa forma, y luego cotejar en la muestra de test, hallaba valores nuevos y no podía calcular los índices. Por tal motivo, se decide convertir dicha variable a factor, luego de la selección de las 10 variables más relevantes con randomForest. En este caso, entonces, se vuelve a generar las muestras de entrenamiento y test, pero esta vez con la variable "Services", como de tipo factor. Esto garantiza, que al tomar las muestras con la herramienta dplyr, que nos permite agrupar los datos por los factores de la respuesta, todos los factores de la variable explicativa "Services" serán tenidos en cuenta. Así, se evitará el problema explicado anteriormente, al intentar realizar el testeo de la red, sobre la muestra test. Esta misma estructura de los datos, y el modelo base realizado con ella, se mantuvo como punto de partida para el algoritmo que utiliza el

paquete Party, dado que demostró ser el menos problemático para dicha herramienta, también. Para ver el modelo Base en mayor profundidad, dirigirse a la sección *Modelo general base, para NNet y Party, con algoritmo automático* B.6.

Modelo con randomForest

Para el modelo de randomForest, que se calcula dentro del algoritmo creado para generar un modelo con este paquete, solo se modificó el parámetro ntree. Tal como se explicó anteriormente, en la sección *Modelando la estructura subyacente* 1.4, este parámetro hace alusión al número de árboles generados por el modelo. En este caso, se tuvo que reducir drásticamente el número de árboles, dada la cantidad de datos, y las fallas de memoria que se presentaron. Aquí se usaron 200 árboles, para realizar cada iteración. Se procedió a dejar todos los demás parámetros como estaban, pues de esta forma el algoritmo lograba buenos resultados. En caso de tener una mayor cantidad de datos, para obtener resultados rápidos, se logró comprobar un buen funcionamiento en otros valores para dicho parámetro, llegando a un mínimo de 80 árboles, sin generar fallas de memoria. En las primeras pruebas, la memoria ram era de 6 gb, y no alcanzaba para este tipo de modelos. Al incrementar la cantidad de árboles, aumentaba significativamente el tiempo computacional de unos minutos, a horas. Cabe destacar, que para las sucesivas ejecuciones, la empresa accedió a brindar mayores recursos computacionales, como ser memoria ram, y memoria de almacenamiento. Estos recursos, son vitales para obtener los modelos previos, ya que en este caso el modelo final fallaba, indicando que le faltaba espacio para colocar un vector de tamaño "xxx", (expresado así porque las fallas fueron muchas, y con distintos valores). Este hecho señalaba que, para poder almacenar los datos de construcción y generar con ellos el modelo, se requería de más de 6 gb de ram, y de un espacio mayor para su almacenamiento. Las últimas pruebas presentadas en este TFM, se realizaron con una ram de 16 gb, y con 80gb para almacenamiento de datos, en un servidor de la empresa. Esto permitió el guardado, que automáticamente realiza R, cuando está trabajando de este modo, (en servidor), para poder continuar con la tarea aunque se apague el ordenador.

Una vez que finaliza el bucle del algoritmo, donde se generan los 1023 modelos posibles en base a combinar las variables, se extrae el modelo de mejor Kappa. Dicho valor se encuentra en el dataframe que recoge los índices, calculados sobre la muestra de test, de cada modelo. Se debe tener en cuenta, que dicho índice sea distinto de 1, porque en caso contrario, estaría indicando que hay variables correladas. Con dichas variables, se arma el nuevo data-set para ingresar al modelo y calcular sus índices, esta vez con su valor por defecto que es ntree=500.

Luego se utiliza otro algoritmo para "refinar" los hiperparámetros. Allí se buscan los valores óptimos, de los hiperparámetros del modelo, para esa combinación de variables. Esto se hace, con un algoritmo aprendido en clase de aprendizaje estadístico. En él, los parámetros óptimos que se buscan son, por ejemplo para el caso de randomForest, mtry y nodesize. Para ejecutarlo, se debe generar una rejilla con los valores que se creen más convenientes, para los datos en estudio. En este caso, los valores utilizados fueron los siguientes :

- Para mtry, en casos de clasificación como el estudiado, lo más común es usar, el entero más próximo por debajo de la raíz cuadrada de la cantidad de variables predictoras. Por tanto, para la rejilla, se tuvieron en cuenta 3 valores: el mencionado valor, ese valor dividido 2, y ese valor multiplicado por 2.

- Para nodesize, en casos de clasificación, el valor por defecto es 1. En este caso, se crea una rejilla que contiene los siguientes valores: 1, 3, 5 y 10. Dichos valores se ciñen a los datos adecuadamente.

A continuación, se genera un bucle que recorre, una a una, todas las posibles combinaciones de estos dos parámetros, y calcula los modelos pertinentes, extrayendo luego en una variable el valor de OOB, obtenido en cada modelo. En este caso, como no se puede trabajar con caret porque ha fallado en repetidas ocasiones, se utiliza la función de randomForest, llamándola directamente de dicho paquete.

Una vez que finaliza dicha ejecución, se extrae de ese vector el mejor valor, (el valor más bajo), y se almacenan los parámetros relacionados con este. Con ellos, se genera el modelo otra vez, para comparar sus resultados, con los obtenidos anteriormente, sobre la muestra de test. En este paso se respeta el número de ntree anterior, y en todo caso se eleva en ambos, para lograr una mejor estabilidad en los resultados. Dicho valor, se ha elevado sin comprometer la consecución del modelo, hasta por ejemplo 1000 o 2000 árboles, siempre teniendo en cuenta que mientras más árboles se construyan, más

grande será el modelo, y más requerimientos computacionales tendrá. Se pretende encontrar el balance entre el rendimiento, y el índice de error del modelo. De dicha comparación se elegirá el mejor modelo, obtenido en esta instancia, con esta técnica. A este modelo, luego de validarlo, se le practicarán pruebas de aleatoriedad para evaluar la estabilidad de sus resultados, y se lo comparará con las otras técnicas de cálculo. Así, se podrá optar por aquel modelo que mejor desempeño tenga. Aunque, también se ha de tener en cuenta, la eficiencia computacional de cada modelo, y su complejidad. Es importante tener en cuenta, que dicho modelo debe colaborar con el estudio de RCA, en este caso. Además de cumplir con una serie de requerimientos específicos, planteados desde la empresa.

A continuación, se presentan los índices del modelo que mejor se adapta a los requerimientos del caso, encontrado con el algoritmo automático desarrollado en este TFM. Sus variables explicativas fueron: ElementName, ElementID, ParameterID, TableIndex, DisplayValue, IsCleared y Type. Se expone en comparación con los resultados del modelo obtenido mediante la búsqueda manual, que se había realizado previamente, y se encuentra expuesto en la sección *Modelos óptimos con selección manual de variables* 2.2.1, cuyas variables explicativas eran: ElementName, ElementID, IsOwner y TableIndex.

	Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue	Variables Explicativas
Algoritmo automático	0.9106445	0.8875885	0.8974507	0.9226485	0.3315430	0.0000000	NaN	7
Búsqueda Manual	0.9018555	0.8759034	0.8881452	0.9144025	0.3437500	0.0000000	NaN	4

Cuadro 2.5: Comparación Índices mejores modelos con algoritmo automático y con selección de variables a mano, para randomForest

En este punto, se puede evaluar si el incremento en los índices de desempeño del modelo, sobre la muestra de test, justifica su mayor complejidad, para el caso puntual en estudio.

Cabe destacar que al refinar los hiperparámetros del modelo, se obtenían resultados inferiores en cuanto a desempeño, aunque mejores en OOB. Este hecho puede verse a continuación, en donde el modelo llamado "Básico", es el modelo tal como sale del algoritmo de selección de variables, y el refinado, es luego de aplicarle los hiperparámetros que dan lugar al modelo con mejor OOB. Ambos modelos usaban ntree=500, y la semilla=1. Ademas cuentan con las mismas 7 variables explicativas, ya que se toma como modelo de partida para refinar parámetros el modelo "Básico", generado por el algoritmo automático, en la etapa correspondiente. En el mismo cuadro, figura también el Modelo de Base, que es el generado con las 21 variables explicativas. Este modelo, se usó de punto de partida para la selección de las variables a considerar, en el algoritmo de búsqueda. Los mencionados modelos, al probarlos sobre la muestra de test, exponían los siguientes índices:

	Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	mtry	nodesize	OOB
Modelo Refinado	0.8999023	0.8744281	0.8860818	0.9125656	0.3437500	5	5	0.07359043
Modelo Básico	0.9106445	0.8875885	0.8974507	0.9226485	0.3315430	2	1	0.08713693
Modelo de Base	0.8417969	0.7995656	0.8252605	0.8573482	0.3315430	4	1	0.07212595

Cuadro 2.6: Comparación Índices mejores modelos con algoritmo automático básico y refinado, en paralelo con el de base que incluía todas las variables, para randomForest

El modelo "Básico" de randomForest, calculado con el algoritmo automático, se encuentra construido con mtry=2, y nodesize=1. Por tanto, este es un modelo de bosques aleatorios, a diferencia del modelo obtenido manualmente que era de bagging. Como se explicó anteriormente en la sección *Modelando la estructura subyacente* 1.4, se trata de un modelo de bosques aleatorios, por incorporar aleatoriedad en la elección de las variables a introducir en cada corte. Como se puede ver, este modelo posee un valor de mtry=2, y dicho valor es menor al número de variables explicativas a considerar, se cuenta con 7 variables explicativas en este modelo.

Modelo de árbol de clasificación con party

Para generar el modelo de árbol de clasificación, calculado con la herramienta party dentro del algoritmo creado, se utilizan los parámetros por defecto, para la búsqueda de la mejor combinación de variables. En este sentido, no se hicieron demasiadas pruebas adicionales, ya que la empresa estaba más interesada en el desarrollo de los otros dos tipos de modelos. Aunque cabe destacar, que el modelo generado con party mediante el algoritmo automático, mostró un resultado muy competitivo en las comparaciones.

A continuación, se presentan los índices del modelo que mejor se adapta a los requerimientos en este caso. Dicho modelo fue encontrado con el algoritmo automático. Sus variables explicativas son: DisplayValue_num, DisplayValue_txt, ElementName, IsOwner, Services y Type. Este modelo, se exhibe en comparación con los resultados del mejor modelo obtenido con la búsqueda manual, que se había realizado previamente con el paquete party, y presentado en la sección *Modelos óptimos con selección manual de variables* 2.2.1. Las variables explicativas de dicho modelo, eran las mismas que las del modelo randomForest a mano: ElementName, ElementID, IsOwnery TableIndex, y sus índices los mismos también que los del modelo randomForest de búsqueda manual.

	Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue	Variables Explicativas
Algoritmo automático	0.9243164	0.9046712	0.9119991	0.9354018	0.3315430	0.0000000	NaN	6
Búsqueda Manual	0.9018555	0.8759034	0.8881452	0.9144025	0.3437500	0.0000000	NaN	4

Cuadro 2.7: Comparación Índices mejores modelos con algoritmo automático y con selección de variables a mano, para Party

En este caso se usaron los mismos parámetros que vienen por defecto en la función. Estos son minsplit=20, minbucket=7, y mincriterion=0.95.

Posteriormente en el paso de refinamiento de hiperparámetros, se debe tener en cuenta que por tratarse de un solo árbol lo que se tendría que hacer, si se trabajara con rpart por ejemplo, sería construir un árbol lo más detallado posible para luego podarlo. Esto, en el caso de la herramienta party no es requerido, por ser distinta la forma de elección de las variables para cada corte. En el documento donde se explica dicho paquete para R, se puede leer lo siguiente: *..."This tree doesn't require a pruning procedure because an internal stop criterion based on formal statistical tests prevents the procedure from overfitting the data. (Traducción: Este árbol posee un criterio de detensión interno basado en test de estadística formal, que previene al procedimiento de caer en sobreajustes.)..." Everitt, Hothorn (2022)*[3]. Por ende, se podría probar con bajar el minbucket a 1, lo cual daría un árbol más detallado, pero a la vez más complejo. Dado que la empresa está más interesada en otro tipo de opciones, no se realizarán más pruebas sobre el modelo obtenido, y luego de exponer los índices del mencionado modelo, se continuará con el desarrollo del algoritmo, pero para utilizarlo con Redes Neuronales.

Se deja expuesto a continuación, los resultados obtenidos hasta aquí, para dicho modelo. Se puede ver que los índices obtenidos, son superiores a los del modelo generado con randomForest, mediante el algoritmo automático, y también son superiores a los obtenidos con la propia herramienta del paquete party, pero utilizando la selección manual de variables.

	Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue	Variables Explicativas
Algoritmo automático randomForest	0.9106445	0.8875885	0.8974507	0.9226485	0.3315430	0.0000000	NaN	7
Algoritmo automático Party	0.9243164	0.9046712	0.9119991	0.9354018	0.3315430	0.0000000	NaN	6

Cuadro 2.8: Comparación entre el modelo de bosques aleatorios realizado con *randomForest*, y el modelo de árbol de clasificación en base al p-valor realizado con *party*

Modelo de redes neuronales con NNet a través del meta-paquete caret

Para el modelo realizado con redes neuronales, se llama a la herramienta NNet, pero en este caso, a través de caret. Esto permite, aprovechar la posibilidad de seleccionar los hiperparámetros óptimos, a través de una rejilla de valores, y a la vez, pedirle a la función que realice el cálculo del modelo mediante validación cruzada. A continuación se detallan la función, y parámetros utilizados:

- Como se mencionó anteriormente en el capítulo 1, la función utilizada de caret se llama train. Dentro de ella, para establecer el método de cálculo del modelo, se usa el parámetro *method*, al cual se le asigna el valor "NNet".

- El parámetro *size* = 2*1:5, por tanto en este caso los valores serán 2, 4, 6, 8 y 10. Dichos valores, representan cuantos nodos tendrá la capa oculta, como se explicó anteriormente en la sección 1.4, una red básica realiza 2 transformaciones, y por tanto cuenta con tres capas,(entrada, oculta y salida).

- El parámetro *decay* = c(0, 0.001, 0.01). Este es el parámetro que representa la tasa de aprendizaje de la red, y como el valor por defecto es cero, se utilizan además estos otros dos valores. Estos actúan como criterio de parada.

- Estos dos últimos parámetros mencionados, son los que conformarán la rejilla denominada *tuneGrid*. Dicha rejilla presenta todas las combinaciones posibles entre ambos. La rejilla se ingresa en el parámetro con el mismo nombre.

- El parámetro *maxit* =100, como se mencionó anteriormente, representa el número máximo de iteraciones, (que se establece como criterio de parada, junto con el decay).

- El parámetro *preproc*='range', de este modo no falla el algoritmo. Esto es debido a que se vuelve más estable, al configurar un rango donde se quiere que se pre-procesen los datos. Aunque la mayoría de los datos son de tipo factor, con muchas categorías dentro, cabe recordar que se cuenta además, con una variable numérica llamada DisplayValue_num.

- El parámetro *lineout*= TRUE, pero no se realiza ningún cambio adicional, porque en este caso sirve la función de activación en los nodos finales por defecto, que es softmax, por ser datos con más de dos categorías en la respuesta. Esto significa que para adecuar la salida del modelo, por ser la variable respuesta de tipo factor, con muchas categorías dentro, usará su valor por defecto, que es softmax, para la activación de salida. En el caso de estudio, no se requeriría otro tipo de función de activación, para adaptar las predicciones a los distintos tipos de respuestas. Para mayor información puede consultarse *Fernández Casal R., et al. (2021)* [5].

- El parámetro *trace*=FALSE, para que la función no vaya mostrando el proceso previo por pantalla, y así mejorar el rendimiento computacional.

- El parámetro *trControl= trainControl (method="cv", number=10)*, en este punto se define que realice el remuestreo, para la evaluación de los hiperparámetros listados en la rejilla, por el método de validación cruzada, estableciendo el número de grupos a seleccionar igual a 10.

Una vez ejecutada esta sección del algoritmo, se obtiene un dataframe con los indices asociados a cada modelo de redes neuronales, calculados realizando las 1023 combinaciones de variables, y validando cada modelo sobre la muestra de test. De allí, se elige el modelo de mayor indice, y se filtra el dataframe de entrenamiento con dichas variables para poder entrenar al modelo nuevamente. Dicho modelo se construye con la misma rejilla de hiperparámetros, presentes en tuneGrid, elevando el número máximo de iteraciones a 200, y dejando los demás parámetros en los valores ya comentados. Así se obtiene el mejor modelo de redes neuronales, con sus hiperparámetros óptimos. A continuación, se lo valida sobre la muestra de test, para comparar sus índices con los de los modelos presentados anteriormente.

He aquí los índices del modelo que mejor se adapta a los requerimientos en este caso. Dicho modelo, fue encontrado con el algoritmo automático. Sus variables explicativas son OwnerName, DisplayValue_num, ElementID, IsOwner, ParameterID, y Services. En el cuadro siguiente, se lo puede ver frente a los resultados obtenidos por el modelo alcanzado con la búsqueda manual. Aquel modelo, realizado y comentado anteriormente para la red neuronal, contaba con las siguientes variables explicativas: ElementName, ElementID, IsOwner, TableIndex, ParameterName y Services.

	Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue	Variables Explicativas
Algoritmo automático	0.9326172	0.9151866	0.9208836	0.9430928	0.331543	0.0000000	NaN	5
Búsqueda Manual	0.9171627	0.8951300	0.9042667	0.9288294	0.32787700	0.0000000	NaN	6

Cuadro 2.9: Comparación Índices mejores modelos con algoritmo automático y con selección de variables a mano, para la red neuronal calculada con NNet

El modelo de la red neuronal, obtenido gracias al algoritmo automático, ha resultado con un número más elevado de hiperparámetros o pesos que su predecesor manual, en este caso se trata de 983. Aunque resultó menor la cantidad de nodos en la capa de entrada, que fue de 226. Dicho valor, se corresponde con la cantidad de categorías incluidas en el modelo, a través de las variables explicativas. Contiene además, 4 nodos en la capa oculta, lo cual se traduce en dos nodos ocultos mas que su predecesor. Para finalizar, cuenta con 15 nodos en la capa de salida, que en este caso es el mismo valor que tenía la red buscada a mano. Este valor, se corresponde con la cantidad de categorías que incluye la variable respuesta. La tasa de aprendizaje óptima, (decay), es la misma que en su predecesor, 0.001. Además la salida del modelo indica que, como se mencionó anteriormente, utiliza la función softmax para la activación en los nodos finales, del mismo modo que su antecesor manual.

En la siguiente ecuación, se presenta como se relacionan, dentro del modelo, los diferentes valores de los parámetros mencionados.

4*	(226	+15)	+4	+15	=983
nodos	categorías	categorías	nodos	categorías	pesos
ocultos	predictores	respuesta	ocultos	respuesta	

2.3. Desempeño del algoritmo

A continuación, se puede ver una comparación del rendimiento del algoritmo al generar: el modelo de bosques aleatorios generado con *randomForest*, el modelo de árbol de clasificación realizado en base a inferencia condicional, generado con *party*, y el modelo de redes neuronales generado con *NNet*.

	randomForest	Party	NNet
Tiempo de Búsqueda de las variables con el algoritmo creado	1.541677 hours	9.42874 días	7.528777 días
Tiempo de Generación del modelo	user 18.100, system 0.300 y elapsed 18.397	user 2757.755 system 4.282 y elapsed 2761.179	user 1167.399 elapsed 1167.114
Tiempo de Predicción sobre muestra test	user y elapsed 0.064	user y elapsed 0.021	user 0.210 elapsed 0.211
Kappa	0.8875885	0.9046712	0.9151866
Accuracy	0.9106445	0.9243164	0.9326172
Variables explicativas del modelo	ElementName, ElementID, TableIndex, DisplayValue, Type, IsCleared, ParameterID	ElementName, IsOwner, DisplayValue_num, Services DisplayValue_txt y Type	DisplayValue_num, ElementID IsOwner, ParameterID Services
Cantidad de Variables	7 + 1	6 + 1	5 + 1

Cuadro 2.10: Cuadro comparativo del rendimiento del algoritmo para las distintas herramientas utilizadas, y principales características de los modelos alcanzados.

En el cuadro precedente, se puede ver que los índices alcanzados por el modelo randomForest son altamente competitivos en comparación con los del modelo Party e incluso con la red neuronal, obtenida con NNet para el caso en estudio. Además, el incremento obtenido con NNet, o con Party, no sería justificación suficiente para optar por alguna de estas herramientas, en esta ocasión. Al tener en cuenta la eficiencia computacional del algoritmo para la red, este demora 7 días y medio en calcular los 1023 modelos, utilizando las diferentes combinaciones de variables. Aunque su modelo final, es el que mejores índices obtiene sobre la muestra de test. El mismo ítem, evaluado para Party, expone que consigue índices inferiores a los de la red, aunque superiores a los de randomForest, pero su exigencia computacional es superior a todos ellos, al requerir 9 días y medio aproximadamente para calcular los 1023 modelos, y obtener el modelo óptimo. Esto en comparación con randomForest, que el algoritmo obtiene los 1023 modelos, y luego selecciona el óptimo en poco más de hora y media, y cuyos índices son apenas inferiores a los de NNet. Obviamente cada caso es diferente y requiere un análisis exhaustivo, pero para el estudio realizado en esta empresa, dicho dato no es menor. Dado que, se debe contar con un mínimo de 8 días, para volver a calcular la mejor combinación de variables, en caso de cambios en el data-set, de optar por la red, que es la que mejores índices expone. En comparación con las 2 horas, que demora el algoritmo que utiliza la función randomForest, para realizar la misma tarea.

También en cuanto a tiempos de generación, la construcción del modelo de redes neuronales demora bastante más que la construcción de su competidor con randomForest,(1167 segundos- 19 minutos-aproximadamente, contra alrededor de 18 segundos). Esta tarea, es necesaria en caso de querer reentrenar el modelo, por la incorporación de nuevos datos. Lo mismo ocurre con los tiempos de predicción de nuevos datos, donde la red demora bastante más que el modelo de bosques aleatorios, (unos 0.211

segundos contra 0.064). Este hecho no es relevante si se trata de pocos datos, pero si pasa a un primer plano, si los data-set a predecir poseen una gran cantidad de registros. Para el caso de Party el algoritmo demora mucho más, llegando a doblar los tiempos de generación del modelo de redes neurales, (una vez seleccionadas las variables a incluir). Aunque cabe destacar, que de los tres modelos, es el más rápido en obtener las predicciones, sobre la muestra de test.

Otro aspecto a considerar es que, el modelo randomForest, a pesar de tener dos variables más que el modelo de redes, es más sencillo de explicar el efecto de los predictores sobre la respuesta, que en lo que respecta al modelo de redes neuronales. Aunque en este mismo aspecto, el modelo calculado con Party es el más sencillo de los 3, para explicar dicho efecto. Por otra parte, si se habla de sencillez, pero en referencia a la cantidad de predictores, el modelo generado con Party posee una variable menos que randomForest, y una más que la red de NNet, con lo cual el modelo de red neuronal sería el más sencillo, en ese sentido.

2.3.1. Pruebas de aleatoriedad a los modelos obtenidos

Se han realizado pruebas de aleatoriedad, en cuanto a la elección de las muestras de entrenamiento y test. Se simuló un reentrenamiento de los modelos con nuevos datos, con la intención de que las muestras de test sean siempre datos que el modelo no hubiera utilizado para entrenarse. Cada modelo ha manteniendo las variables seleccionadas, con el algoritmo automático, y re-calculado los hiper-parámetros óptimos, en caso de ser necesario. Este proceso se realizó cambiando la semilla, que se coloca antes de hacer la selección de las muestras. Se eligió como primera semilla el 1, que fue la utilizada hasta el momento, y a continuación, 20 números aleatorios entre 1 y 9999. Las semillas resultaron, entonces, en los siguientes valores: 1, 6590, 4626, 7896, 1384, 6132, 5955, 3604, 4692, 7042, 7591, 3495, 9271, 1392, 9211, 518, 551, 7905, 1831, 9064 y 6577. Los modelos fueron calculados en el orden que se exponen las semillas. Este dato es relevante, para poder comprender el gráfico expuesto a continuación. En dicho gráfico, se compara la evolución del Kappa, para los tres modelos, con estas 21 semillas. De esta forma, la línea de cada modelo parte del valor Kappa con la semilla 1. Esta semilla, fue utilizada para generar los modelos expuestos previamente, en la sección anterior. Se puede ver en azul las redes neuronales calculadas con NNet, en naranja los modelos de bosques aleatorios con randomForest, y en gris los árboles de clasificación con Party.

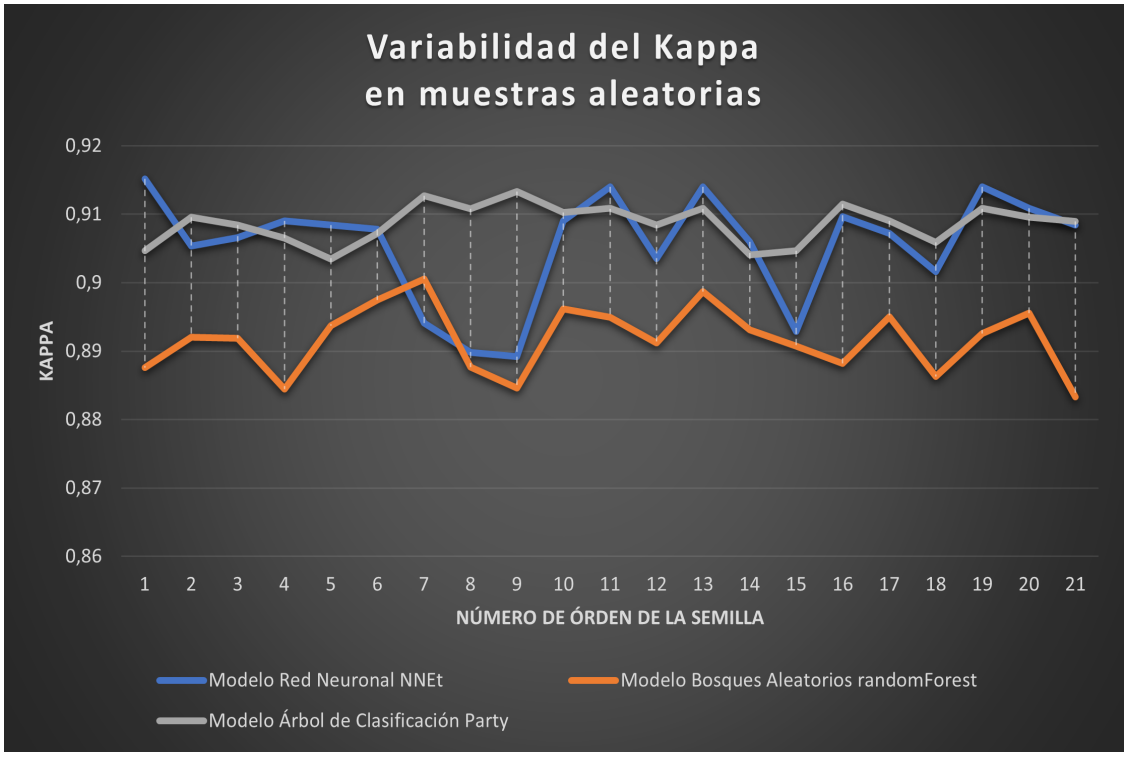

Figura 2.4: Gráfico de variabilidad del Kappa, para los 3 modelos, al realizar pruebas de aleatoriedad en la selección de la muestra train y test (Producción Propia)

Modelo	Kappa Promedio	Accuracy Promedio	Menor AccuracyLower	Mayor AccuracyUpper	Menor Kappa	Mayor Kappa
randomForest	0.8916959	0.9138997	0.8938279	0.9322227	0.8833019	0.9005386
Party	0.9086555	0.9274554	0.9109566	0.9417388	0.9034679	0.913296
NNet	0.9055576	0.9249907	0.8990049	0.9430928	0.8891911	0.9151866

Cuadro 2.11: Rango de variabilidad de los índices presentados, frente a cambios en la semilla, para la selección de la muestra de entrenamiento y test

Mediante la comparación presentada en el cuadro precedente, se puede apreciar que el Kappa para los modelos con randomForest, tienen una variabilidad en un rango de 0.0172367. En relación a esto, se podría decir que los modelos presentados con esta herramienta son bastante estables, frente a la variabilidad en las muestras. Para los modelos realizados con la herramienta Party, se alcanza una variabilidad de 0.0098281, por tanto se puede ver, que son aún más estables que randomForest, un 57 % más estable que randomForest en este caso. Los modelos de Party, cuentan con una variable menos, pero aún así el desempeño de ambas herramientas es muy bueno en este caso. Pero, si se tiene en cuenta la eficiencia computacional, es probable que el elegido fuera el modelo de randomForest, ya que el algoritmo de búsqueda demora poco menos de dos horas, contra los 9 días del algoritmo de Party. Incluso ya contando con las variables, luego de ejecutar el algoritmo de búsqueda, el modelo de Party tanto para su generación, como para realizar la predicción, demora mucho más que el de randomForest, e incluso más que el de NNet. En el caso de NNet se tiene un rango, algo superior, de 0.0259955 para el Kappa. Este valor estaría indicando, que dicho modelo se ve un poco más afectado ante la variabilidad en las muestras, que el modelo de randomForest, y más aún que el de Party. Esto podría deberse a varios factores, como ser: que tiene separada una de las variables explicativas, por los requerimientos específicos de construcción del modelo. Aunque este hecho, no parece afectarle al modelo generado con Party. En este caso dicho modelo cuenta con las dos partes de la variable, a diferencia de la red, que se apoya solo en el formato contínuo de la misma. Otro factor puede ser, que posee una variable explicativa menos que Party, y dos menos que randomForest. Por este motivo, se lo podría considerar más inestable, por contar con menos variables para poder clasificar los datos correctamente. Aunque en vista de los resultados obtenidos, es claro que el desempeño de los tres modelos es muy bueno, dentro del entorno controlado en el cual se los ha probado.

Capítulo 3

Conclusión

Los algoritmos desarrollados en este TFM, permiten realizar el modelado de distintas bases de datos, con diferentes técnicas, en menor tiempo, y con menor esfuerzo. De dichos modelos, se elegirá el que mejor se adapte a los datos. Esto permite sentar las bases, para entender la estructura de esos datos. Con dicha estructura, se busca predecir la variable respuesta analizada, denominada OwnerName, en base a las alertas. Al unir dicha información, a la que brindada por las incidencias, se estarán generando datos necesarios, para que, en el futuro, se puedan realizar los pasos siguientes del análisis de **Causa Raíz**, de un problema en un entorno IT. Con los mencionados algoritmos, se ahorra mucho tiempo y esfuerzo en la primera fase de generación del modelo. Se trata de la fase de selección de las variables, para generar con ellas, el mejor modelo aditivo posible, con cada una de las técnicas elegidas. Como se comentó en el transcurso de este TFM, cada uno de los algoritmos, genera 1023 modelos, al realizar todas las combinaciones posibles, de las 10 variables predictoras que se le ingresaron. A cada modelo generado, se lo evalua sobre la muestra de test, y se extrae en un dataframe los índices Kappa y Accuracy de dicho modelo. De este data frame, se elige el modelo que mejores índices posee, y se vuelve a generar dicho modelo, para almacenarlo en un objeto. Además, del mencionado data-frame se pueden ir explorando los demás modelos obtenidos, en función de lo que se crea más conveniente en cada caso. Algunos de los aspectos a considerar en los modelos pueden ser: correlación entre las variables, análisis de sensibilidad y especificidad, curvas ROC, entre otros. Pero lo más interesante es que, de esta forma, ya se cuenta con información sobre los índices de los modelos, lo cual permite ordenarlos, y orientar la búsqueda, para realizarla de manera más eficiente. Así posteriormente, si los datos y el modelo lo permiten, realizar un refinamiento de parámetros para obtener un modelo, mas preciso. Cabe recordar, que la intención es que dicho modelo, permita saber cuál es el elemento padre del alarmado, quien probablemente estaría relacionado con la **Causa Raíz**, que estaba originando el problema analizado.

En el caso de estudio, se trabajó sobre modelos aditivos. Esto se hizo de este modo, porque dados los índices que se obtenían, y los tiempos con los que la empresa contaba, no se consideró necesario realizar otro tipo de búsqueda más compleja, en esta primera etapa al menos. Cuando se hace alusión a una búsqueda más compleja, se refiere a por ejemplo, analizar la interacción entre las variables predictoras, hecho que como se mencionó anteriormente, no se consideró necesario analizar, en este caso.

Se ha podido comprobar, que los algoritmos generados, han sido de vital importancia para acortar los períodos de búsqueda del mejor modelo. Este hecho se pudo verificar durante los meses posteriores a la finalización de este TFM, dado que durante estos meses, la empresa tuvo acceso a nuevos datos, que incluían información relevante para analizar la **Causa Raíz** de los mismos. De realizarse sobre ellos una búsqueda y selección de variables a mano, se hubiera demorado mucho más en la obtención del modelo óptimo.

En pasos posteriores, habría que incorporar a los datos que se utilizan para el modelado en un entorno IT, un listado completo que deje de manifiesto la estructura de relaciones entre los elementos, cuales son las causas que pueden afectarla, y que alarmas generan. El referido listado, debería estar

unido al data-set de entrenamiento del modelo, para que efectivamente indique, quien sería el elemento **Causa Raíz**, de cada incidencia. Con esta información, y como la incidencia estaría vinculada a varias alarmas, se le asignaría peso a cada elemento padre, en función de cuantas veces dicho padre aparece como responsable de las alarmas de elementos relacionadas con esa incidencia. Con dicha información, en otro paso posterior, se buscaría analizar los datos históricos, que señalen la causa de aquella incidencia en el pasado, para cotejar los resultados obtenidos. Finalmente se analizaría, cuales fueron las acciones que se tomaron para corregir dicha incidencia, con la intención de incorporarlas en el análisis de **Causa Raíz**, este último paso, quedaría enmarcado dentro de la etapa de retroalimentación.

Además, sería de especial importancia generar otros modelos, que trabajen sobre las causas asignables que son elaboradas por los técnicos, una vez que se han resuelto las incidencias. Con esto se podría colaborar con ellos en este sentido, brindándoles la información que han considerado relevante en el pasado, en el formato que ellos mismos manejan.

Es muy importante, trabajar sobre la vinculación entre las incidencias y las alarmas, para que se perfeccione lo más posible. Una opción es utilizar una ventana temporal, que se construya en base a los tiempos de resolución en que se dirimen las alarmas, (calculando la media, o mediana si hay datos atípicos en este sentido). Además, se deberían vincular en base a la ubicación, sumado a alguna otra variable que se crea conveniente, y mediante la cual, las bases de datos compartan información, como es el caso de los servicios afectados, por ejemplo.

Luego desde el punto de vista de los algoritmos generados y de su utilidad, se puede comentar que estos fueron creados sobre datos categóricos nominales, no ordinales, pero de forma independiente de los nombres de las variables presentes en el data-set. Esto quiere decir, que se han creado variables alternativas sobre las cuales se apoya el algoritmo. Esto es relevante porque, durante la investigación previa, se detectó que, la mayoría de herramientas no se centran tanto en datos de este tipo, y por tanto son un tipo de datos para los cuales se cuenta con menos recursos para su estudio. Dado que estos tipos de datos, no permiten realizar sobre ellos las técnicas más clásicas y comentadas en el ambiente estadístico, y por ende se suelen ver como algo mas complejos de abordar.

También es importante destacar, que estos algoritmos se pueden modificar fácilmente para abordar, no solo este tipo de variables, sino otras como las de tipo numérico, por ejemplo.

Se considera que en investigaciones futuras, desde el punto de vista de los algoritmos creados, se podrían abordar los siguientes aspectos:

- Investigar la posibilidad de una mejora, en cuanto a eficiencia computacional, y generar algún paquete para R que los contenga.
- Trasladar dichos algoritmos a otros lenguajes de programación, como por ejemplo python, para que puedan ser llamados desde otros entornos de trabajo. Este paso es de especial relevancia para el sector privado, dada la popularidad de dicho lenguaje.
- Volcar los algoritmos desarrollados en una librería para python.

Con estos pasos, se lograría colaborar con aquellos trabajos en que se requiera generar una selección de variables, para un primer modelo aditivo. En donde además, se pretenda utilizar alguna de las tres herramientas informáticas empleadas en este TFM, para el cálculo de modelos matemáticos. Dichas herramientas son randomForest, Party y NNet a través de caret. Esto le permitirá al investigador, dedicar más tiempo y esfuerzo, a otras tareas de análisis. Como probar si el efecto de las variables en el modelo va más allá de lo aditivo, es decir si el modelo es más complejo en cuanto a como se relaciona dicha variable con el efecto visto en la respuesta. Así como también, para realizar el ajuste fino de los hiperparámetros del modelo en cuestión. Ya que para este tipo de modelos, estos algoritmos no existían, y antes de su creación, la búsqueda y selección de variables relevantes para el modelo, se hacía a mano.

Índice de figuras

Índice de cuadros

Apéndice A

Análisis exploratorio de los datos originales

A.1. Rastreando la Causa Raíz a través de la herramienta de Pareto

Aquí se expone como están distribuidas las distintas categorías en los datos, en cuanto a la variable prioridad. Dicha variable, está definida por los términos “Critical”, “High”, “Medium, “Low”(Critico, alto, medio y bajo).

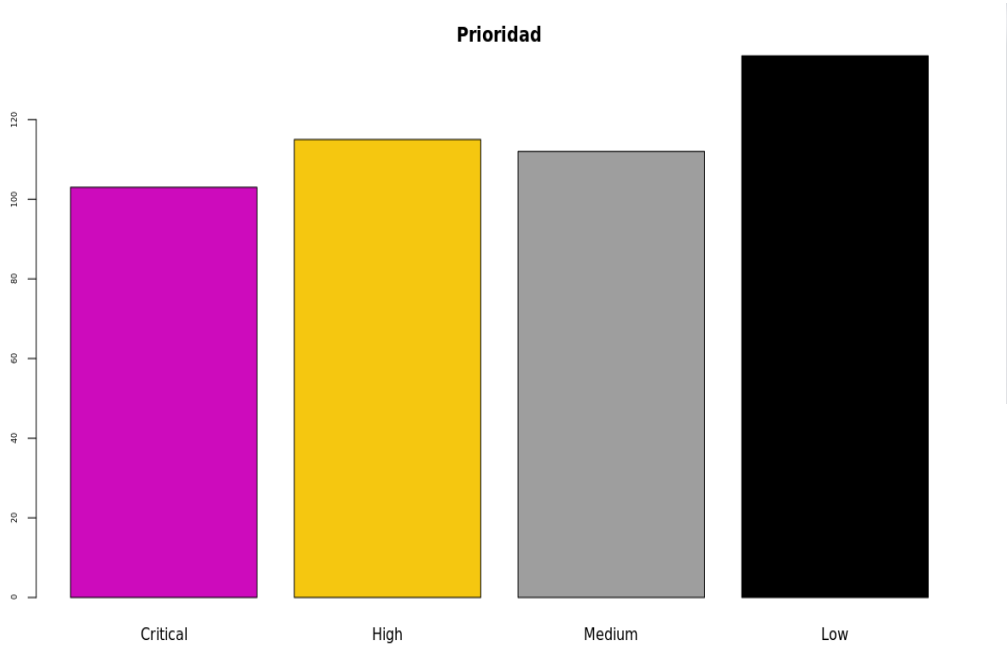

Figura A.1: Gráfico de distribución de la prioridad en los datos

A continuación se presenta la variable impacto, la cual se clasifica en “1-Extensive/Widespread”, “2-Significant/Large”, “3-Moderate/Limited”, “4-Minor/Localized”(1-Extenso/generalizado, 2-significativo/grande, 3-moderado/limitado y 4-menor/localizado).

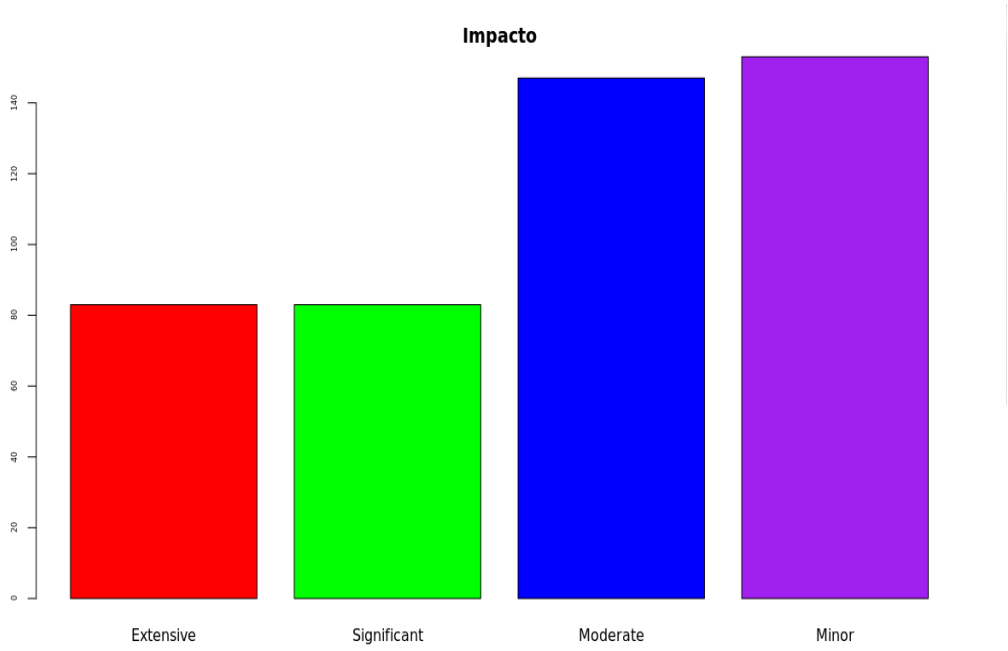

Figura A.2: Gráfico de distribución del impacto en los datos

A continuación, se colocan en conjunto para analizar su relación.

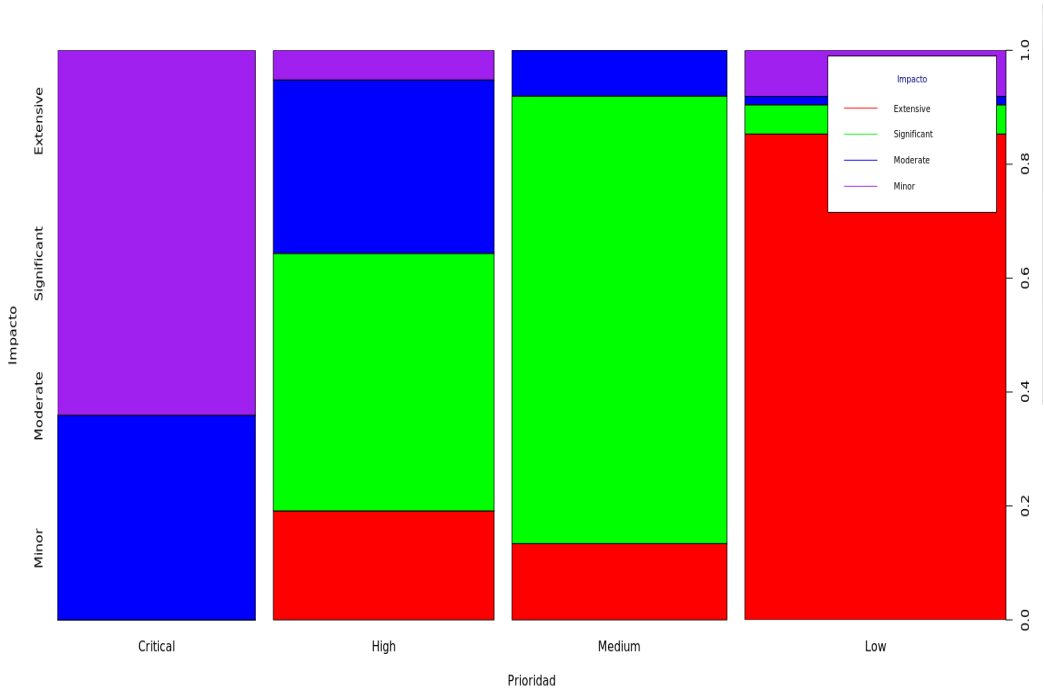

Figura A.3: Gráfico de relación entre la prioridad y el impacto en los datos

En estos gráficos, se puede apreciar que la mayor cantidad de incidencias se encuentran en la franja de prioridad baja y de impacto menor o localizado. Para analizar cual es la prioridad y el impacto, del 80 % de las incidencias, se utilizará un Gráfico de Pareto.

A.1.1. Gráfico Pareto vinculando la prioridad y el impacto, con las incidencias registradas

Aquí se analiza, que vinculo tienen las distintas categorías del impacto y la prioridad con las incidencias, que pueden luego convertirse en problemas. Se busca determinar la proporción de dichas categorías, en relación a las incidencias analizadas.

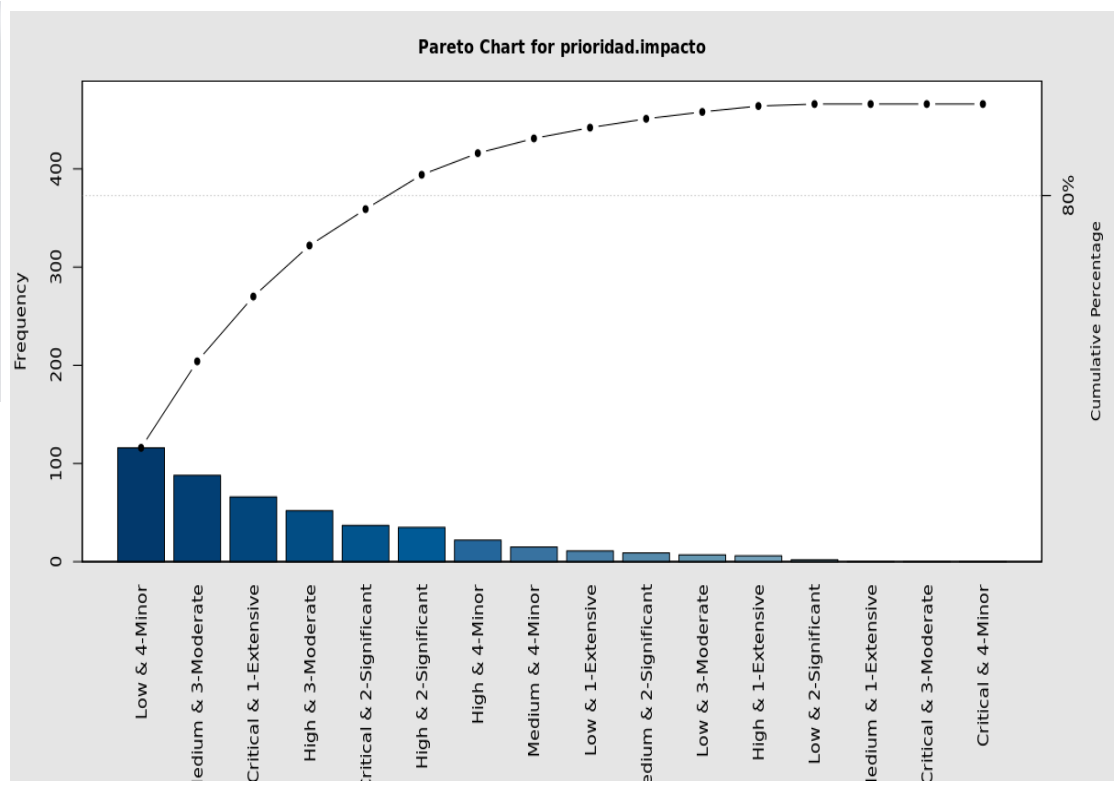

Figura A.4: Gráfico Pareto de prioridad e impacto que se dan mayoritariamente en los datos

El gráfico ordena las categorías de izquierda a derecha, empezando por aquellas de mayor peso en los datos y completando con las siguientes en orden decreciente. Además señala con una línea de puntos horizontal, donde está el corte del 80 % de los datos, para poder identificar mejor cuantas, cuales y que peso tienen, las distintas categorías mayoritarias responsables de dichos porcentajes.

En la tabla siguiente, se detallan los porcentajes individuales de cada categoría. Allí se puede ver, además, una columna que recoge los acumulados. De su análisis se desprende, que la prioridad y el impacto asociado a mas del 80 % de las incidencias, se distribuye entre las primeras 6 categorías, de las 16 presentes. El total de categorías surge de las distintas combinaciones que pueden presentarse en los datos, y estas son:

```
Pareto chart analysis for prioridad.impacto
                          Percentage Cum.Percent.
  Low & 4-Minor            24.892704    24.892704
  Medium & 3-Moderate      18.884120    43.776824
  Critical & 1-Extensive   14.163090    57.939914
  High & 3-Moderate        11.158798    69.098712
  Critical & 2-Significant  7.939914    77.038627
  High & 2-Significant      7.510730    84.549356
```

A partir de este análisis, se estima que existe un porcentaje cercano al 25 % de las incidencias, que se categorizan en prioridad baja y con impacto menor o localizado. Si se lograra que dichas incidencias no volvieran a aparecer, esto ayudaría a ahorrar un 25 % de los recursos e infraestructura. Más aún, si a esto se le suman las incidencias con prioridad media e impacto moderado, ya se estaría alcanzando cerca de un 44 % del total. Este hecho generaría una mejora, que no solo sería en mayor disponibilidad de los recursos humanos y técnicos, sino que también podría influir positivamente en el ambiente de trabajo, y más aún en generar una mejor imagen de la empresa. Por ejemplo, si se tiene mayor disponibilidad de técnicos, esto les permitiría atender más rápido y mejor otros conflictos, y/o encargarse de tareas de mantenimiento preventivo en mayor medida, que las actividades de tipo correctivo. También se considera que generaría un ambiente de trabajo más tranquilo, al situar a los técnicos en una actitud mas proactiva, en lugar de reactiva. Ya que, según se ha comprendido de la explicación brindada por la empresa, a día de hoy hay personas encargadas de vigilar estas incidencias y vincularlas con las alarmas, dilucidando en base a sus conocimientos y experiencia, que actitud tomar en cada caso. Por otra parte se considera, que de haber personal de atención al cliente, encargado de recoger los avisos, y/o reclamos, por inconvenientes con el servicio, ellos podrían estar avocados en mayor medida, más a tareas de fidelización de clientes, que de recolección de quejas. Pero aún más significativo sería, si se tiene en cuenta el adelanto en relación a la imagen corporativa para los clientes. Esto generaría una mejora a nivel cualitativo, que sería medible a través de la satisfacción del cliente. De esta manera, se lograría que los clientes actúen como voceros de la empresa operadora, pregonando que en ella se preocupan por perfeccionarse y resolver,de raíz, aquellos problemas que se presenten, para que no los vuelvan a afectar.

Luego de estas dos primeras categorías, aparecen aquellas incidencias catalogadas como criticas y extensas, luego las altas y moderadas, mas allá las criticas y significativas, y por último las altas y significativas. Con ellas se completa el 84.55 % de incidencias analizadas, aproximadamente. Poniendo de manifiesto la importancia, de que a través de un buen análisis de RCA, se pueden concentrar los esfuerzos en unas pocas categorías, y encontrar la **Causa Raíz** asociada a ellas. Esto se vuelve muy relevante, por ser las categorías presentes en más del 80 % de las incidencias.

A.1.2. Análisis de Causas vinculadas a las alarmas o incidencias

A continuación, se realiza el análisis de las causas asignadas a las incidencias que se fueron presentando. La intención es rastrear la **Causa Raíz**, responsable del 80 % de las incidencias, (efectos), observados, en los datos analizados. En el siguiente gráfico, se presentan con un código representativo de cada una de las causas, y luego, en el cuadro que figura a continuación, se las puede ver con su nombre completo, para comprender de que se trata cada una.

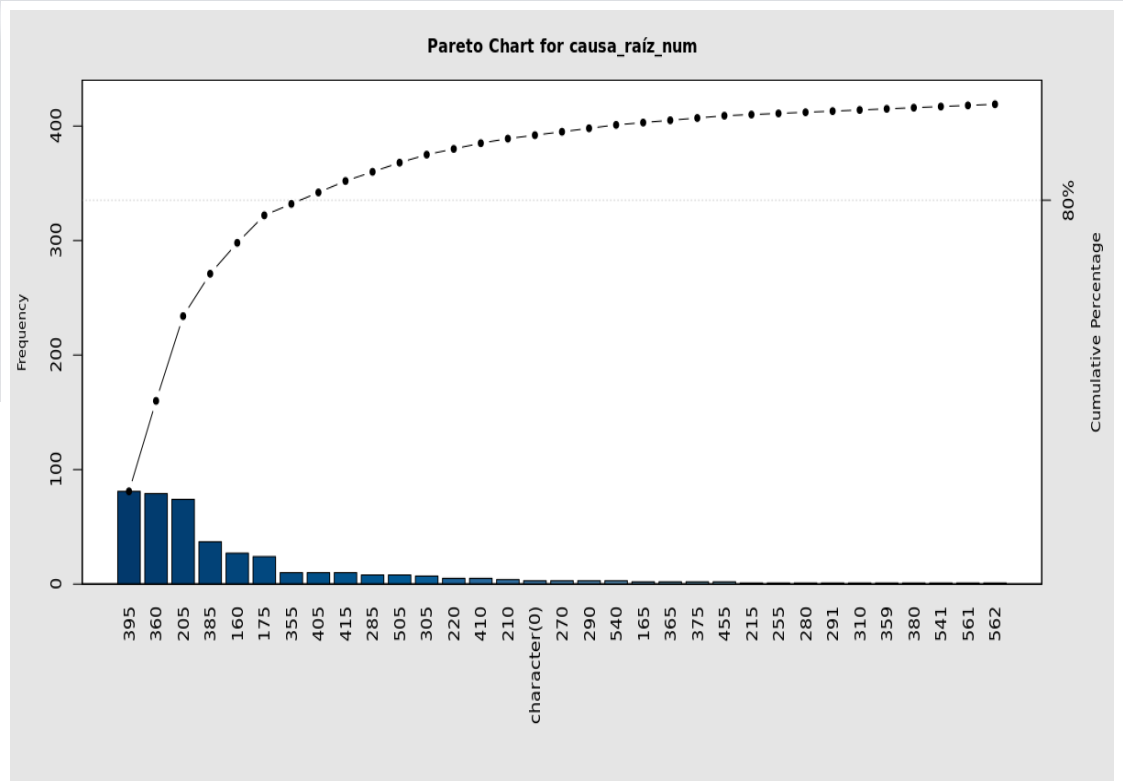

Figura A.5: Gráfico de las causas asignadas con los porcentajes que ocupa cada categoría

Aquí, se observan las causas asignadas al 80 % de las incidencias, ordenadas por orden de imputación, en relación a la cantidad de incidencias que generaron.

```
Pareto chart analysis for causa_raíz
                                                          Percentage Cum.Percent.
  395. Del propio equipo que presta el servicio final      19.331742    19.331742
  360. De un equipo de red de transporte                   18.854415    38.186158
  205. Programado, autorizado por el cliente               17.661098    55.847255
  385. De cableado o elementos auxiliares                   8.830549    64.677804
  160. Verificado tras aviso del propio cliente             6.443914    71.121718
  175. Verificado tras falsa alarma en supervisión/ telemedida  5.727924    76.849642
  355. Del receptor satélite                                2.386635    79.236277
  405. Causas meteorológicas o fading                       2.386635    81.622912
```

Detectando y salvando inconvenientes en la toma de datos

Debido a diferencias encontradas en el criterio de etiquetar a los datos con causa desconocida, se unificaron dichos campos bajo la denominación: "desconocidos", ya presente en la base de datos. Los campos mencionados figuraban como no asignados, o con espacio en blanco, (dicho dato se traduce en NA al importar la base de datos), o con un guion medio. Se genera, además, una categoría aparte para los cancelados, a fin de analizar si esto aporta mayor información al análisis. Dado que, aquellos campos identificados como NA, no son tomados en cuenta para el gráfico de Pareto. Una vez realizadas estas tareas de depuración, se realiza un análisis de causas con Pareto. Dichas tareas de depuración, incluían entre otras, asignar a la categoría "desconocidos" los NA, unificar los códigos denominados "455.desconocida", y crear la categoría de desconocidos.cancelados, (00. desc.cancelados). El análisis de Pareto se realiza teniendo en cuenta: los desconocidos por un lado, y los desconocidos que han sido cancelados por otro. Esto se traduce, en dividir en dos categorías separadas a los desconocidos, según si la incidencia ha sido cancelada o no.

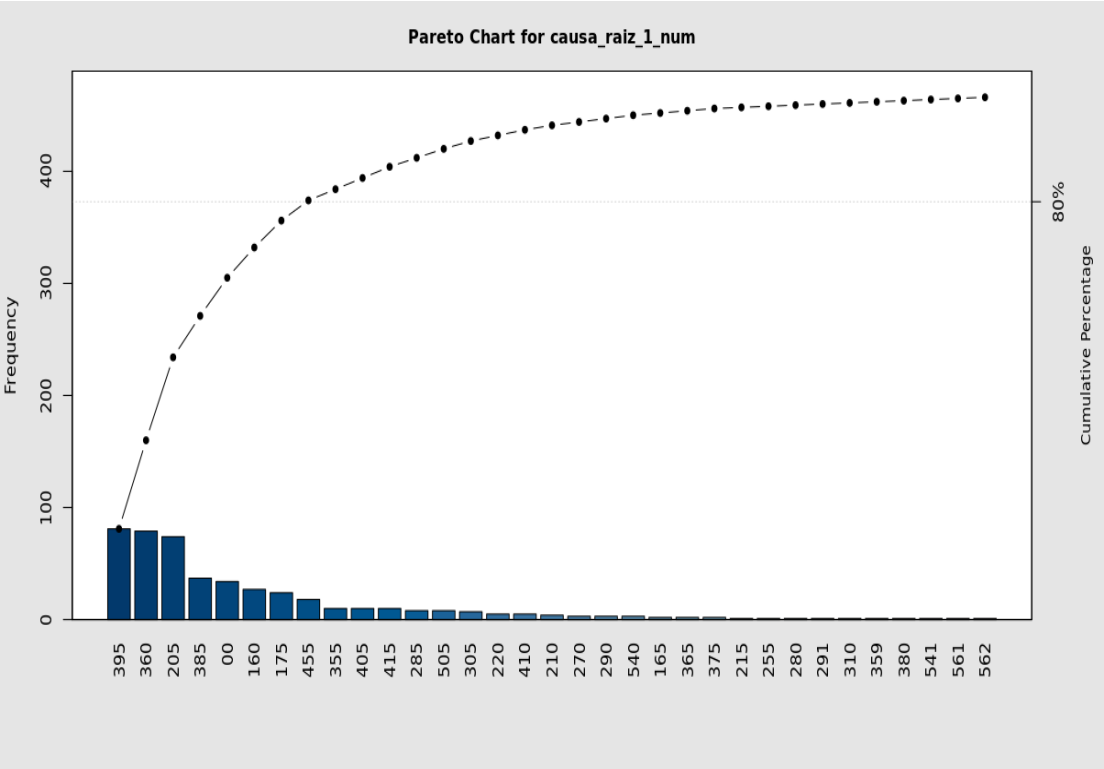

Figura A.6: Gráfico Pareto de las causas asignadas al 80 % de las incidencias con los porcentajes que abarca cada una, teniendo en cuenta las desconocidas y canceladas como una categoría independiente

```
Pareto chart analysis for causa_raiz_1
                                                          Percentage Cum.Percent.
  395. Del propio equipo que presta el servicio final       17.381974    17.381974
  360. De un equipo de red de transporte                    16.952790    34.334764
  205. Programado, autorizado por el cliente                15.879828    50.214592
  385. De cableado o elementos auxiliares                    7.939914    58.154506
  00. desc.cancelados                                        7.296137    65.450644
  160. Verificado tras aviso del propio cliente              5.793991    71.244635
  175. Verificado tras falsa alarma en supervisión/ telemedida  5.150215    76.394850
  455. Desconocida                                           3.862661    80.257511
```

A continuación, se realizó un análisis con los "desc.cancelados" estudiados en profundidad, porque se estima que aquellos que tienen causa desconocida, y que fueron cancelados, en realidad son una categoría que debería ser estudiada a parte, para analizar su **Causa Raíz**. Hasta aquí, se definió el vector, que contiene los datos de los incidentes que fueron asignados a causa desconocida, y cancelados posteriormente. En ellos se analizó con Pareto, a que sub-causas estaban vinculados.

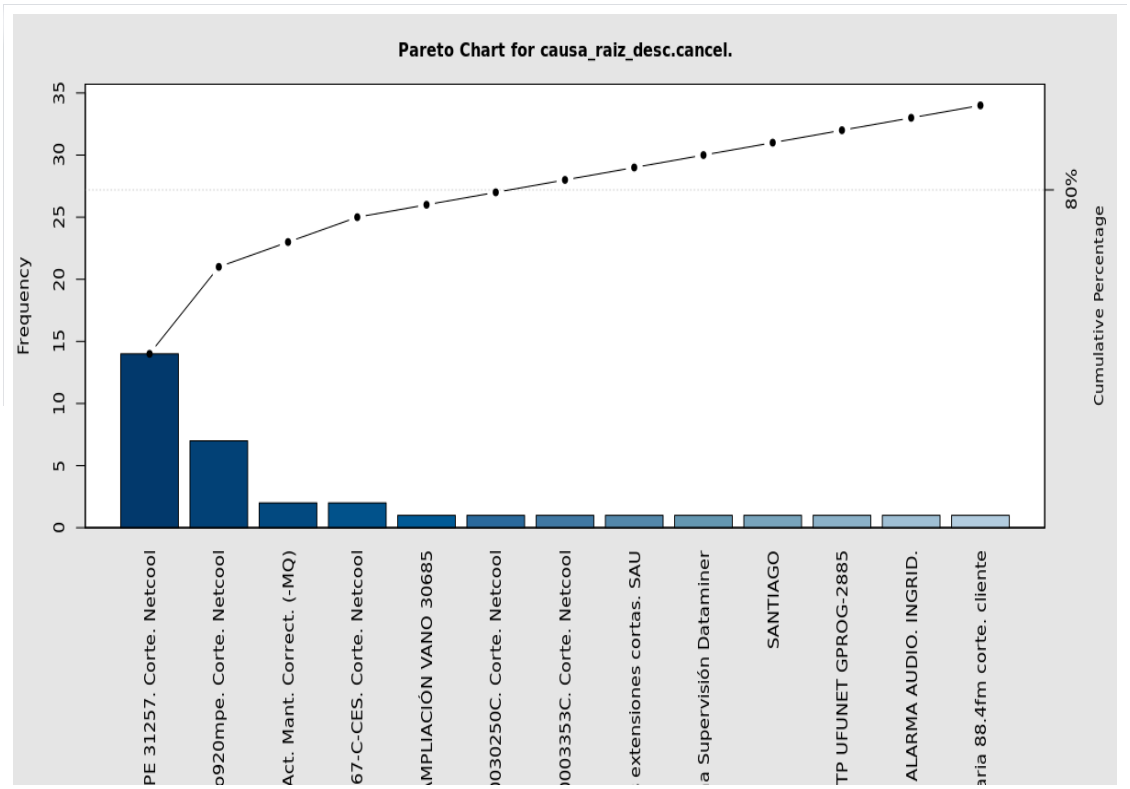

Figura A.7: Gráfico Pareto de las causas asignadas al 80 % de las incidencias, registradas como desconocidas y canceladas, con los porcentajes que abarca cada una

```
Pareto chart analysis for causa_raiz_desc.cancel.
                                                                                Percentage Cum.Percent.
  MPLS pon920lalin - aco9001mpe PPE 31257. Corte. Netcool                        41.176471    41.176471
  MPLS aco7604mpe - aco920mpe. Corte. Netcool                                    20.588235    61.764706
  Act. Mant. Correct. (-MQ)                                                       5.882353    67.647059
  MPLS aco7604mpe - viz7604arch ALQ LYNTIA ABERT-25167-C-CES. Corte. Netcool      5.882353    73.529412
  AMPLIACIÓN VANO 30685                                                           2.941176    76.470588
  MPLS aco9001nlys2 - mad9906aval2 ALQ LYNTIA ESESRET10030250C. Corte. Netcool    2.941176    79.411765
  MPLS our920carb - aco920mpe2 ALQ_LYNTIA ESESRET10003353C. Corte. Netcool        2.941176    82.352941
```

Se puede ver, que salvo los incidentes relacionados con "Mant. Correct.", el resto están vinculados en su mayoría, de una u otra forma, a Corte.Netcool. Sería interesante analizar cual es la **Causa Raíz** de estos cortes, para poder trabajar sobre ella, o en todo caso, encontrar una manera de evitar que estos eventos, queden ocupando espacio y recursos, en las bases de datos.

Para simular que dichas incidencias no estuvieran, se asignan los desconocidos y cancelados como NA, y se realiza el análisis de Pareto nuevamente. La intención es ver, como afectan dichas incidencias, al estudio de **Causas Raíz**. A continuación se vuelve a presentar el primer pareto, en comparación con el segundo.

```
Pareto chart analysis for causa_raiz_1
                                                      Percentage Cum.Percent.
  395. Del propio equipo que presta el servicio final   17.381974    17.381974
  360. De un equipo de red de transporte                16.952790    34.334764
  205. Programado, autorizado por el cliente            15.879828    50.214592
  385. De cableado o elementos auxiliares                7.939914    58.154506
  00. desc.cancelados                                    7.296137    65.450644
  160. Verificado tras aviso del propio cliente          5.793991    71.244635
  175. Verificado tras falsa alarma en supervisión/ telemedida   5.150215    76.394850
  455. Desconocida                                       3.862661    80.257511

Pareto chart analysis for causa_raiz_2
                                                      Percentage Cum.Percent.
  395. Del propio equipo que presta el servicio final   18.750000    18.750000
  360. De un equipo de red de transporte                18.287037    37.037037
  205. Programado, autorizado por el cliente            17.129630    54.166667
  385. De cableado o elementos auxiliares                8.564815    62.731481
  160. Verificado tras aviso del propio cliente          6.250000    68.981481
  175. Verificado tras falsa alarma en supervisión/ telemedida   5.555556    74.537037
  455. Desconocida                                       4.166667    78.703704
  355. Del receptor satélite                             2.314815    81.018519
```

Al comparar el Pareto donde se tenían en cuenta los incidentes desconocidos y cancelados, y se les registraba en una categoría aparte, (pareto causa_raíz_1), con el que se estableció a dicha combinación como NA, (pareto causa_raíz_2), se puede ver que cambian los totales. De esta forma, se incluyen algunas de las causas que antes no figuraban como asignables, a ese 80 % de los incidentes, y se incrementa la relevancia de varias categorías. Razón por la cual, es muy importante tener bien diferenciados los códigos que se usen, para poder individualizar aquellas incidencias que fueron problemas a resolver, respecto de las que fueron errores en la toma de datos. Cabe destacar, que las 4 primeras causas, al no tener en cuenta las desconocidas y canceladas en el análisis, aparecen como responsables de mas del 60 % de los incidentes registrados. Esto es importante, dado que de poder prever este tipo de fallas, se estaría evitando cerca del 44 % de las incidencias antes de que sucedan. Esto es aplicable a las fallas, tanto en equipos propios, como en equipos de la red de transporte, además de las referidas al cableado, o elementos auxiliares. Por tanto, sería una buena elección estudiar como evitar este tipo de registros, para que no vuelvan a producirse, y entonces ahorrar todo ese esfuerzo, y recursos.

Por otra parte, al realizar una lectura de los tipos de incidentes, asociados a las causas, responsables de aproximadamente el 80 % de las incidencias, se aprecia que existiendo alrededor de 300 alertas identificadas, con nombres muy variados y diversos, lo preocupante es que hay 211 asignadas a una categoría llamada other, que no aporta datos sobre la incidencia en si. De este análisis se desprende que, existen diferencias, en cuanto a los nombres dados a las distintas incidencias, lo cual para hacer un análisis mas exhaustivo, debería tratarse y depurarse. Se debería unificar aquellos que son el mismo, escritos con cambios en la puntuación por ejemplo, para tener una mejor visión global del problema. Un ejemplo claro, comentado anteriormente, es que, en principio, figuran incidencias cuya causa aparece como desconocida, pero además aparece otra categoría que la causa es un guión medio, etc.

A.2. Pantallazo del data-set de incidencias

Operational (	Incident ID*+	Customer Sit	Site Group	Summary*	F/H afectació	F/H resoluci	Status*	Status_Reas	Target Date	Tipificación	PAI	Priority*	Assigned Gr	Owner Grou	Reported Sou
Sin afectación	INC00010084	SANTIAGO	A CORUÑA	CLIMATE SY	29/04/2022 11	29/04/2022 15	Resolved	Resolved incident		SANTIAGO		Medium	NOC CNX ES	NOC CNX ES	Notice by NM
Sin afectación	INC00010084	SANTIAGO	A CORUÑA	MPLS aco760	18/04/2022 17	18/04/2022 19	Cancelled	Cancelled		SANTIAGO		High	Network Servi	Network Servi	Notice by NM
Sin afectación	INC00010084	SANTIAGO	A CORUÑA	MPLS aco760	18/04/2022 17	18/04/2022 18	Cancelled	Cancelled		SANTIAGO		High	Network Servi	Network Servi	Notice by NM
Sin afectación	INC00010084	SANTIAGO	A CORUÑA	MPLS aco760	18/04/2022 17	18/04/2022 18	Cancelled	Cancelled		SANTIAGO		High	Network Servi	Network Servi	Notice by NM
Sin afectación	INC00010084	SANTIAGO	A CORUÑA	MPLS aco760	18/04/2022 14	18/04/2022 15	Cancelled	Cancelled		SANTIAGO		High	Network Servi	Network Servi	Notice by NM
Sin afectación	INC00010084	SANTIAGO	A CORUÑA	MPLS aco760	18/04/2022 13	18/04/2022 13	Cancelled	Cancelled		SANTIAGO		High	Network Servi	Network Servi	Notice by NM
Con afectació	INC00010084	SANTIAGO	A CORUÑA	SANTIAGO (/	18/04/2022 9:	18/04/2022 17	Resolved	Resolved inci	18/04/2022 15	SANTIAGO	F	High	Broadcast Se	Front Office T	Customer/pro
Con afectació	INC00010083	SANTIAGO	A CORUÑA	CABINET DIS	25/03/2022 4:	25/03/2022 4:	Resolved	Resolved incident		SANTIAGO	C	High	NOC CNX ES	NOC CNX ES	Change
Con afectació	INC00010083	SANTIAGO	A CORUÑA	INSIDE TEMI	25/03/2022 4:	25/03/2022 5:	Resolved	Resolved incident		SANTIAGO	C	Low	NOC CNX ES	NOC CNX ES	Change
Con afectació	INC00010083	SANTIAGO	A CORUÑA	CABINET AC	25/03/2022 4:	25/03/2022 5:	Resolved	Resolved incident		SANTIAGO	C	High	NOC CNX ES	NOC CNX ES	Change
Con afectació	INC00010083	SANTIAGO	A CORUÑA	INSIDE TEMI	25/03/2022 4:	25/03/2022 5:	Resolved	Resolved incident		SANTIAGO	C	Medium	NOC CNX ES	NOC CNX ES	Change
Con afectació	INC00010083	SANTIAGO	A CORUÑA	CABINET DIS	25/03/2022 4:	25/03/2022 4:	Resolved	Resolved incident		SANTIAGO	C	Medium	NOC CNX ES	NOC CNX ES	Change
Con afectació	INC00010083	SANTIAGO	A CORUÑA	CABINET DIS	25/03/2022 4:	25/03/2022 5:	Resolved	Resolved incident		SANTIAGO	C	High	NOC CNX ES	NOC CNX ES	Change
Con afectació	INC00010083	SANTIAGO	A CORUÑA	PUBLIC GRIE	25/03/2022 3:	25/03/2022 5:	Resolved	Resolved incident		SANTIAGO	C	Critical	NOC CNX ES	NOC CNX ES	Change
Con afectació	INC00010083	SANTIAGO	A CORUÑA	Mejora de sup	25/03/2022 2:	25/03/2022 5:	Resolved	Resolved inci	25/03/2022 5:	SANTIAGO	C	High	NOC CNX ES	NOC CNX ES	Change
Con afectació	INC00010083	SANTIAGO	A CORUÑA	INSIDE TEMI	25/03/2022 1:	25/03/2022 3:	Resolved	Resolved incident		SANTIAGO	C	Critical	NOC CNX ES	NOC CNX ES	Notice by NM
Con afectació	INC00010082	SANTIAGO	A CORUÑA	SANTIAGO (/	16/03/2022 9:	16/03/2022 9:	Resolved	Resolved incident		SANTIAGO	F	High	Front Office T	Front Office T	Customer/pro
Con afectació	INC00010082	SANTIAGO	A CORUÑA	aco920mpe3	14/03/2022 13	14/03/2022 14	Resolved	Resolved incident		SANTIAGO	A	Critical	Network Servi	NOC XOC Té	Cellnex Perso
Sin afectación	INC00010082	SANTIAGO	A CORUÑA	SANTIAGO (	08/03/2022 15	08/03/2022 15	Closed	Automated Resolution Repo		SANTIAGO		Low	Broadcast Se	Broadcast Se	Cellnex Perso
Con afectació	INC00010082	SANTIAGO	A CORUÑA	MPLS aco920	07/03/2022 15	07/03/2022 15	Closed	Automated Resolution Repo		SANTIAGO	A	Critical	Network Servi	Network Servi	Notice by NM
Sin afectación	INC00010082	SANTIAGO	A CORUÑA	SANTIAGO (I	04/03/2022 23	05/03/2022 1:	Closed	Automated Resolution Repo		SANTIAGO		Low	Broadcast Se	Broadcast Se	Supervision
Con afectació	INC00010082	SANTIAGO	A CORUÑA	MPLS aco900	04/03/2022 11	05/03/2022 1:	Closed	Automated Resolution Repo		SANTIAGO	NO	High	Network Servi	NOC XOC Té	Supervision
Con afectació	INC00010082	SANTIAGO	A CORUÑA	SANTIAGO. (	03/03/2022 11	03/03/2022 17	Closed	Automated Resolution Repo		SANTIAGO	C	High	NOC CNX ES	NOC CNX ES	Notice by Ser
Con afectació	INC00010082	SANTIAGO	A CORUÑA	SANTIAGO. (	03/03/2022 16	03/03/2022 16	Closed	Automated Resolution Repo		SANTIAGO	C	Low	NOC CNX ES	NOC CNX ES	Notice by Ser

Figura A.8: Pantallazo del data-set de incidencias

Reported Sou	Last Modifie	Drop-Down L	Company*+	Closed Date	Submit Date	Assignee+	First Name+	Entry ID	Notes	Customer Ph	Company	Middle Name	Last Name+	Incident Type	Impact*	Operational (	Operational (	Product Cate	Product Cate	Product Cate
Notice by NM	29/04/2022 15	TIS	NOC CNX ES	29/04/2022 11:26:56		es cnx	INC000000683	Urgencia Ong	###	NOC CNX ES		es cnx	User Service	4-Minor/Locali	TIS	Temperature				
Notice by NM	19/04/2022 3:	TRANSPORT	NOC Cellnex	18/04/2022 15	18/04/2022 18:36:54	cellnex	INC000000683	Urgencia Ong	###	NOC Cellnex		NOC	User Service	3-Moderate/	Limited					
Notice by NM	18/04/2022 16	TRANSPORT	NOC Cellnex	18/04/2022 16	18/04/2022 18:04:42	cellnex	INC000000683	Urgencia Ong	###	NOC Cellnex		NOC	User Service	3-Moderate/	Limited					
Notice by NM	18/04/2022 16	TRANSPORT	NOC Cellnex	18/04/2022 16	18/04/2022 17:48:40	cellnex	INC000000683	Urgencia Ong	###	NOC Cellnex		NOC	User Service	3-Moderate/	Limited					
Notice by NM	18/04/2022 16	TRANSPORT	NOC Cellnex	18/04/2022 16	18/04/2022 14:48:32	cellnex	INC000000683	Urgencia Ong	###	NOC Cellnex		NOC	User Service	3-Moderate/	Limited					
Notice by NM	18/04/2022 13	TRANSPORT	NOC Cellnex	18/04/2022 13	18/04/2022 13:36:07	cellnex	INC000000683	Urgencia Ong	###	NOC Cellnex		NOC	User Service	3-Moderate/	Limited					
Customer/pro	26/04/2022 3:	DIFUSIÓN	NOC Cellnex	18/04/2022 10	18/04/2022 10:23:11	cellnex	INC000000684	16	###	NOC Cellnex		NOC	User Service	4-Minor/Locali	BROADCAST	Difusión				
Change	28/03/2022 14	TIS	NOC CNX ES	25/03/2022	25/03/2022 4:36:54	es cnx	INC000000683	Urgencia Ong	###	NOC CNX ES		es cnx	User Service	3-Moderate/	LI TIS	AC Energy				
Change	25/03/2022 7:	TIS	NOC CNX ES	25/03/2022	25/03/2022 4:31:52	es cnx	INC000000683	Urgencia Ong	###	NOC CNX ES		es cnx	User Service	4-Minor/Locali	TIS	Temperature				
Change	25/03/2022 7:	TIS	NOC CNX ES	25/03/2022	25/03/2022 4:23:34	es cnx	INC000000683	Urgencia Ong	###	NOC CNX ES		es cnx	User Service	3-Moderate/	LI TIS	AC Energy				
Change	25/03/2022 7:	TIS	NOC CNX ES	25/03/2022	25/03/2022 4:20:05	es cnx	INC000000683	Urgencia Ong	###	NOC CNX ES		es cnx	User Service	3-Moderate/	LI TIS	Temperature				
Change	25/03/2022 7:	TIS	NOC CNX ES	25/03/2022	25/03/2022 4:14:27	es cnx	INC000000683	Urgencia Ong	###	NOC CNX ES		es cnx	User Service	3-Moderate/	LI TIS	AC Energy				
Change	25/03/2022 7:	TIS	NOC CNX ES	25/03/2022	25/03/2022 4:07:19	es cnx	INC000000683	Urgencia Ong	###	NOC CNX ES		es cnx	User Service	3-Moderate/	LI TIS	AC Energy				
Change	25/03/2022 7:	TIS	NOC CNX ES	25/03/2022	25/03/2022 3:58:37	es cnx	INC000000683	Urgencia Ong	###	NOC CNX ES		es cnx	User Service	1-Extensive/	Widespread					
Change	25/03/2022 7:	TIS	NOC CNX ES	25/03/2022	25/03/2022 2:06:29	Cristina	INC000000683	MasMovil	###	NOC CNX ES		Tarabana	User Service	3-Moderate/	Limited		Servicio			
Notice by NM	30/03/2022 4:	TIS	NOC CNX ES	25/03/2022	25/03/2022 1:25:33	es cnx	INC000000683	Urgencia Ong	###	NOC CNX ES		es cnx	User Service	2-Significant/	TIS	Temperature				
Customer/pro	17/03/2022 8:	DIFUSIÓN	NOC Cellnex	16/03/2022	16/03/2022 12:28:48	cellnex	INC000000679	231	###	NOC Cellnex		NOC	User Service	4-Minor/Locali	BROADCAST	Difusión				
Cellnex Perso	04/04/2022 9:	TRANSPORT	NOC Cellnex	14/03/2022	14/03/2022 14:04:05	cellnex	INC000000678	536	###	NOC Cellnex		NOC	User Service	1-Extensive/	Widespread					
Cellnex Perso	10/05/2022 2:	DIFUSIÓN	NOC Cellnex	10/05/2022 2:	08/03/2022 15:32:52	cellnex	INC000000671	SANTIAGO (	###	NOC Cellnex		NOC	User Service	4-Minor/Locali	BROADCAST	Difusión				
Notice by NM	10/05/2022 2:	TRANSPORT	NOC Cellnex	10/05/2022 2:	07/03/2022 15:57:27	cellnex	INC000000671	Urgencia Ong	###	NOC Cellnex		NOC	User Service	1-Extensive/	Widespread					
Supervision	07/05/2022 2:	DIFUSIÓN	NOC Cellnex	07/05/2022 2:	05/03/2022 1:36:15	cellnex	INC000000675	667	###	NOC Cellnex		NOC	User Service	4-Minor/Locali	BROADCAST	Difusión				
Supervision	07/05/2022 2:	TRANSPORT	NOC Cellnex	07/05/2022 2:	04/03/2022 11:47:52	NOC Cellnex	INC000000675	665	###	NOC Cellnex		NOC Cellnex	User Service	3-Moderate/	Limited					
Notice by Ser	10/05/2022 2:	TIS	NOC CNX ES	10/05/2022 2:	03/03/2022 1(Robert	Bobor es cnx	INC000000671	FrH reportada	###	NOC CNX ES		es cnx	User Service	4-Minor/Locali	TIS	Temperature				
Notice by Ser	10/05/2022 2:	TIS	NOC CNX ES	10/05/2022 2:	03/03/2022 16:31:37	es cnx	INC000000671	FrH reportada	###	NOC CNX ES		es cnx	User Service	4-Minor/Locali	TIS	Temperature				

Reported Dat	Urgency*	Vendor Ticke	Atendida(AIR	Causa	Causa(char. I	Causa(char. I	Centro Instal	City	Client Type	Closure Sou	Country	Create Reque	Created from	DMC Interno	Direct Conta
29/04/2022 11	3-Medium	No		155. De aberti	540. CLIMA1	045. Tras con	OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No		
18/04/2022 16	1-Critical	No					OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No		
18/04/2022 16	1-Critical	No					OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No		
18/04/2022 17	1-Critical	No					OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No		
18/04/2022 14	1-Critical	No					OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No		
18/04/2022 13	1-Critical	No					OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No		
18/04/2022 10	1-Critical	No		155. De aberti	305. Fallo de	060. Tras rec	OI-7F93797F7D14DC984C		Office-Based Employee			No	No	4320	
25/03/2022 4:	1-Critical	No		155. De aberti	285. Por fallo	105. Reposici	OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No	240	
25/03/2022 4:	4-Low	No		155. De aberti	205. Program	005. Tras rep	OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No	7200	
25/03/2022 4:	1-Critical	No		155. De aberti	205. Program	030. Tras rep	OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No	240	
25/03/2022 4:	3-Medium	No		155. De aberti	205. Program	030. Tras rep	OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No	5760	
25/03/2022 4:	3-Medium	No		155. De aberti	285. Por fallo	105. Reposici	OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No	4320	
25/03/2022 4:	2-High	No		155. De aberti	205. Program	030. Tras rep	OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No	240	
25/03/2022 3:	1-Critical	No		155. De aberti	205. Program	030. Tras rep	OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No	240	
25/03/2022 2:	1-Critical	No		155. De aberti	205. Program	030. Tras rep	OI-7F93797F	SANTIAGO D	Contractor		ESPAÑA	No	No	240	NOC CNX ES
25/03/2022 1:	1-Critical	No		155. De aberti	540. CLIMA1	005. Tras rep	OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No	240	
16/03/2022 12	1-Critical	No		155. De aberti	175. Verificad	105. Reposici	OI-7F93797F7D14DC984C		Office-Based Employee			No	No	4320	
14/03/2022 14	1-Critical	No		155. De aberti	360. De un eq	010. Tras rein	OI-7F93797F	SANTIAGO D	Office-Based Employee		ESPAÑA	No	No	43	
08/03/2022 16	4-Low	No		155. De aberti	395. Del propi	005. Tras rep	OI-7F93797F7D14DC984C		Office-Based System			No	No		
07/03/2022 16	1-Critical	No		155. De aberti	360. De un eq	105. Reposici	OI-7F93797F	SANTIAGO D	Office-Based System		ESPAÑA	No	No	43	
05/03/2022 1:	4-Low	No		155. De aberti	395. Del propi	055. Tras rein	OI-7F93797F7D14DC984C		Office-Based System			No	No		
04/03/2022 11	1-Critical	GDINE-65508	No	156. De prove	359. De fibra	030. Tras rep	OI-7F93797F7D14DC984C		Office-Based System			No	No	260	
03/03/2022 16	1-Critical	Yes		155. De aberti	540. CLIMA1	010. Tras rein	OI-7F93797F	-	Office-Based System		ESPAÑA	No	No	240	
03/03/2022 16	4-Low	Yes		255. Sin caus	160. Verificad	065. Tras con	OI-7F93797F	-	Office-Based System		ESPAÑA	No	No	240	

Direct Contac	Direct Conta	Direct Contac	Direct Contac	Direct Contac	Effort Time S	F/H Llegada	FH Inicio	FH Max Reso	FH Pendiente	FH Bloqueo	FH Cierre	FH Cierre(Al	FH Creacion	FH Modifica	FH Pendient	Fecha Limite	Impacto Sug	Impacto Sug
					0		29/04/2022 11:25:53		29/04/2022 15:54:32		29/04/2022 15:54:46	29/04/2022 15	29/04/2022 11	29/04/2022 11	29/04/2022 11:26:51			
					0		18/04/2022 17:45:26				18/04/2022 19:20:38	19/04/2022 3:	18/04/2022 16	18/04/2022 18	18/04/2022 18:36:41			
					0		18/04/2022 17:45:26				18/04/2022 18:25:12	18/04/2022 16	18/04/2022 16	18/04/2022 18	18/04/2022 18:04:35			
					0		18/04/2022 17:45:26				18/04/2022 18:03:35	18/04/2022 16	18/04/2022 17	18/04/2022 17	18/04/2022 17:48:33			
					0		18/04/2022 14:46:25				18/04/2022 15:00:27	18/04/2022 15	18/04/2022 14	18/04/2022 14	18/04/2022 14:48:26			
					0		18/04/2022 13:33:38				18/04/2022 13:40:41	18/04/2022 13	18/04/2022 13	18/04/2022 13	18/04/2022 13:36:00			
					0		18/04/2022 9:50:00	18/04/2022 17:35:00			18/04/2022 17:35:00	26/04/2022 3:	18/04/2022 10	21/04/2022 9:	4-Minor/Locali	4-Minor/Locali		
					0		25/03/2022 4:36:49	25/03/2022 4:36:50			25/03/2022 4:36:50	28/03/2022 9:	25/03/2022 4:	25/03/2022 8:	3-Moderate/Li	3-Moderate/Li		
					0		25/03/2022 4:31:37	25/03/2022 5:35:21			25/03/2022 5:36:04	25/03/2022 5:	25/03/2022 4:	30/03/2022 5:	4-Minor/Locali	4-Minor/Locali		
					0		25/03/2022 4:23:18	25/03/2022 5:18:48			25/03/2022 5:19:22	25/03/2022 5:	25/03/2022 4:	25/03/2022 8:	3-Moderate/Li	3-Moderate/Li		
					0		25/03/2022 4:20:00	25/03/2022 5:17:38			25/03/2022 5:18:09	25/03/2022 5:	25/03/2022 4:	29/03/2022 5:	4-Minor/Locali	4-Minor/Locali		
					0		25/03/2022 4:14:17	25/03/2022 4:14:18			25/03/2022 4:14:18	28/03/2022 9:	25/03/2022 4:	28/03/2022 5:	14:17	3-Moderate/Li		
					0		25/03/2022 4:07:12	25/03/2022 5:16:19			25/03/2022 5:16:51	25/03/2022 5:	25/03/2022 4:	25/03/2022 8:	3-Moderate/Li	3-Moderate/Li		
					0		25/03/2022 3:58:31	25/03/2022 5:15:03			25/03/2022 5:15:32	25/03/2022 5:	25/03/2022 3:	25/03/2022 7:	1-Extensive/V	1-Extensive/V		
					0		25/03/2022 2:03:13	25/03/2022 5:10:13			25/03/2022 5:10:52	25/03/2022 5:	25/03/2022 2:	25/03/2022 6:	3-Moderate/Li	3-Moderate/Li		
					0	25/03/2022 1:	25/03/2022 1:26:27	25/03/2022 3:38:34			25/03/2022 3:38:34	25/03/2022 6:	25/03/2022 1:	25/03/2022 5:	2-Significant/	2-Significant/		
					0		16/03/2022 9:59:00	16/03/2022 9:59:00			16/03/2022 9:59:00	16/03/2022 14	16/03/2022 12	19/03/2022 9:	4-Minor/Locali	4-Minor/Locali		
					0		14/03/2022 13:49:30	14/03/2022 14:17:03			14/03/2022 14:18:24	04/04/2022 0(	14/03/2022 14	14/03/2022 14	1-Extensive/V	1-Extensive/V		
					0		08/03/2022 15:32:05	08/03/2022 15:34:31			08/03/2022 15:34:31	08/03/2022 15	08/03/2022 15	08/03/2022 15	15:32:44			
					0		07/03/2022 15:52:09	07/03/2022 15:53:12			07/03/2022 15:53:12	07/03/2022 16	07/03/2022 15	07/03/2022 15	1-Extensive/V	1-Extensive/V		
					0		04/03/2022 23:30:10	05/03/2022 1:49:07			05/03/2022 1:49:07	05/03/2022 01	05/03/2022 01	05/03/2022 01	01:36:07			
					0		04/03/2022 11:00:00	05/03/2022 1:41:13			05/03/2022 1:41:13	05/03/2022 4:	04/03/2022 11	04/03/2022 11	1E-3-Moderate/Li	3-Moderate/Li		
					0	03/03/2022 17	03/03/2022 11	03/03/2022 16	03/03/2022 17:03:10		03/03/2022 17	03/03/2022 17	07/03/2022 15	03/03/2022 16	03/03/2022 2(	4-Minor/Locali	4-Minor/Locali	
					0		03/03/2022 16:31:28	03/03/2022 16	ACEPTADA		03/03/2022 16:34:05	08/03/2022 16	03/03/2022 16	03/03/2022 16	03/03/2022 20:31:28		4-Minor/Locali	

Inbound:	Last Modifiec	Last_Kickbac	Núm. de tick	OLA Hold	Outbound:	Recon ID	Region	Resolution	Resolution P	Resolution P	SLA Hold	SLM Priority	State Provin	Status-PPL	Street	Submitter*	Subtipología	Support_Gro	TicketType	Time Zone	Tipificación	UnknownUse	Valores de A	ZipPostal Cc
5	IGANEA		No	No	0	00150780	GALICIA	No está instalado completamente el sistema y sigue func		No	No	Medium	A CORUÑA		Monte do Pe	Incident	2026 Nivel 1			N/A	False			
5	Integraciones		No	No	0	00150780	GALICIA	Se tiral puerto el servicio esta de baja		No	No	High	A CORUÑA		Monte do Pe	Integraciones	Nivel 1		Incident	N/A	False			
4	xfiguera		No	No	0	00150780	GALICIA	Circuito dado de baja		No	No	High	A CORUÑA		Monte do Pe	Integraciones	Nivel 1		Incident	N/A	False			
2	xfiguera		No	No	0	00150780	GALICIA	Circuito dado de baja		No	No	High	A CORUÑA		Monte do Pe	Integraciones	Nivel 1		Incident	N/A	False			
4	mblascod		No	No	0	00150780	GALICIA	CETH-2568 de baja		No	No	High	A CORUÑA		Monte do Pe	Integraciones	Nivel 1		Incident	N/A	False			
2	Integraciones		No	No	0	00150780	GALICIA	circuito de baja		No	No	High	A CORUÑA		Monte do Pe	Integraciones	Nivel 1		Incident	N/A	False			
13	abiazque		No	No	2	00150780	GALICIA	Bloqueo en la pasarela del canal de comunicación con el		No	No	High		Enabled		eabiazlv	1800 Nivel 1		Incident	N/A				
5	AR_ESCALA	25/03/2022 4:38:19	No	No	1	00150780	GALICIA	Power supply interrupted due to faulty AC devices. Spon		No	No	Low	A CORUÑA		Monte do Pe	Integraciones	2026 Nivel 1		Incident	N/A	False			
5	AR_ESCALA		No	No	1	00150780	GALICIA	se ha finalizado bien el CR2000000067815		No	No	High	A CORUÑA		Monte do Pe	Integraciones	2026 Nivel 1		Incident	N/A	False			
6	AR_ESCALA		No	No	1	00150780	GALICIA	finalizado		No	No	High	A CORUÑA		Monte do Pe	Integraciones	2026 Nivel 1		Incident	N/A	False			
6	AR_ESCALA		No	No	1	00150780	GALICIA	finalizado		No	No	Medium	A CORUÑA		Monte do Pe	Integraciones	2026 Nivel 1		Incident	N/A	False			
5	AR_ESCALA		No	No	1	00150780	GALICIA	Power supply interrupted due to faulty AC devices. Spon		No	No	Medium	A CORUÑA		Monte do Pe	Integraciones	2026 Nivel 1		Incident	N/A	False			
6	AR_ESCALA		No	No	1	00150780	GALICIA	finalizado		No	No	High	A CORUÑA		Monte do Pe	Integraciones	2026 Nivel 1		Incident	N/A	False			
5	AR_ESCALA		No	No	2	00150780	GALICIA	finalizado		No	No	Critical	A CORUÑA		Monte do Pe	Integraciones	2026 Nivel 1		Incident	N/A	False			
3	AR_ESCALA		No	No	2	00150780	GALICIA	Nos llama el t Servicio	Servicio de re	Cxubicación	No	High	A CORUÑA	Enabled	Monte do Pe	Integraciones	2026 Nivel 1		Incident	N/A	False			
14	AR_ESCALA		No	No	4	00150780	GALICIA	Air conditioner failure. Repair or replacement of the equi			No	Critical	A CORUÑA		Monte do Pe	CTARABAN	2026 Nivel 1		Incident	N/A	False	15698		
4	AR_ESCALA		No	No	1	00150780	GALICIA	Verificada emisión correcta. WC se ve recuperada 9:59			No	High		Enabled	Monte do Pe	Integraciones	1800 Nivel 1		Incident	N/A	False			
4	AR_ESCALA		No	No	0	00150780	GALICIA	Reinicio urgente y subida de version del ac-d02Omega3 po			No	Critical	A CORUÑA	Enabled	Monte do Pe	JRODRIG1	1300 Nivel 1		Incident	N/A		15898		
2	AR_ESCALA		No	No	0	00150780	GALICIA	Se sustituye fuente averiada			No	Low		Enabled		lgoncalv	2026 Nivel 1		Incident	N/A	False			
2	AR_ESCALA		No	No	0	00150780	GALICIA	Caída tunel			No	Critical		Enabled	Monte do Pe	Integraciones	1300 Nivel 1		Incident	N/A				
3	AR_ESCALA		No	No	0	00150780	GALICIA	Se normaliza remotamente el Tx principal sobre antena			No	Low		Enabled		rmartine	1000 Nivel 1		Incident	N/A				
8	AR_ESCALA		No	No	0	00150780	GALICIA	Corte de Fibras en la red de Lyritta			No	High		Enabled		dloreito	2005 Nivel 1		Incident	N/A				
15	AR_ESCALA		INC000068804	No	2	00150780	GALICIA	Se llama al responsable de la U.O.y nos indican que el tr			No	High				FRANYXF10	1000 Nivel 1		Incident	Service Affec	False		ACEPTADA	
5	AR_ESCALA		INC000068804	No	1	00150780	GALICIA	After complaint from the customer, no issues detected			No	Low				FRANYXF10	1500 Nivel 1		Incident	Service Affec	False		ACEPTADA	

Apéndice B

Análisis y Modelos óptimos a mano

Para poder comprender mejor los resultados de los gráficos expuestos en esta sección, se han codificado las variables del data-set que contenía los datos previos a la creación del conjunto de datos sintéticos. Con los primeros, se realizó el análisis de causa raíz, a partir de gráficos y de la herramienta de Pareto. Con los segundos, se apoyó la realización de los algoritmos para el cálculo de modelos aditivos, referenciado en el cuerpo del documento. En la mencionada codificación se ha asignado, de manera arbitraria, números del 1 en adelante a cada categoría, para cada una de las variables explicativas y para la respuesta. Luego de ellos, se los establece como tipo carácter, para que los modelos comprendieran de que se sigue tratando de variables categóricas, no numéricas ni ordinales.

En primer lugar se presenta una tabla con las categorías de la variable respuesta OwnerName. En español se podría traducir como Nombre del dueño, o como se le ha llamado a lo largo de este TFM, Componente Padre.

[1] 09bdc651461f25b65e47ee89adc56f99
[2] 2e3ffce303618b8feba24ccb715dcdca
[3] 5b51b199bebdc111d995e0821ca9cd83
[4] 5cde2aa811d16198dd1921fbab4fba6a
[5] 6e79db69d2527cbc796e0748e1f902f4
[6] 763ae14091aaa5e830edbb098aae73b5
[7] aa42197b97746ff2ccf265c398e99053
[8] d99c51f199953f710e9ce8f6f8015daa
[9] e3b4bfa08634b6659130b3ebaa2c0bdc
[10] e5dc75a344103af785484b74becc0c22
[11] fba95413ea8c28348cd9df92add4f111
[12] Service
[13] System

Cuadro B.1: Categorías de la variable respuesta llamada OwnerName y que identifica al Componente Padre

A continuación, se exponen las variables explicativas IsOwner, ElementName, TableIndex y ElementID, en ese orden.

[1] TORRESPAÑA-CONTIN_MPE3 False

[3] True

Cuadro B.2: Categorías de la variable IsOwner y que identifica si el elemento alarmado es padre de otro o no

[1] mad_tesy	madGsertelrcstesy01	madGsertelrcstesy02rx
[4] madRohdeccutesy01	madRohdeccutesy02	madRohdeccutesy03
[7] madRohdeccutesy04	madRohdethrtesy01	madRohdethrtesy02
[10] madRohdethrtesy03	madRohdethrtesy04	madRohdethrtesy05
[13] madSimetint16	tGroup_Type_A	tGroup_Type_B
[16] Transmitter A1	Transmitter A2	Transmitter A3
[19] Transmitter A4	Transmitter B	

Cuadro B.3: Categorías de la variable ElementName. Identifica el nombre del elemento alarmado

[1]1	1.1
[3]1000.1	2
[5]2.1	3
[7] 3.1	4
[9]4.1	]FM/TRANSMITTER/TX INPUT ALARM 01/Service OM
[11]FM/TRANSMITTER/TX INPUT ALARM 04/Service R2	FM/TRANSMITTER/TX REFLECTED POWER 01/Service KISS
[13]FM/UCA/TX WARNING 01/Service OM	FM/UCA/TX WARNING 02/Service ROCK
[15]FM/UCA/TX WARNING 03/Service CAD100	FM/UCA/TX WARNING 04/Service R2
[17]noinfo	RF1 - Ch22 - MPE5
[19]RF1 - Ch25 - MPE3	]RF1 - Ch26 - MPE4
[21]RF1 - Ch32 - MPE1	RF1 - Ch33 - RGE MADRID
[23]RF1 - Ch34 - MPE2	RF1 - Ch36 - 4K_Pruebas
[25]RF1 - Ch36(plp#0) - 4K_Pruebas	RF1 - Ch38 - TDT-CAM
[27]RF1 - Ch41 - RGE2	Summary.Ok
[29]Summary.Warning	Transmitter A1.Exciter Summary.Ok
[31]Transmitter A1.Exciter Summary.Warning	Transmitter A2
[33]Transmitter A2_Af Left	Transmitter A2_Af Right
[35]Transmitter A3	Transmitter A3_Af Left
[37]Transmitter A3_Af Right	Transmitter A4.Exciter Summary.Ok
[39]Transmitter A4.Exciter Summary.Warning	

Cuadro B.4: Categorías de la variable TableIndex. Identifica algunas de las características propias del elemento alarmado

[1] 09bdc651461f25b65e47ee89adc56f99	0b0ee8b71cee482463da696285a4def0
[3] 10508	10509
[5] 10521	10535
[7] 12906	12917
[9] 24a29a235c0678859695b10896513b3d	274895fd4fa237a59a5a2f2aab135e5e
[11] 2e3ffce303618b8feba24ccb715dcdca	33f2481179ad598a2b87005cec4b6830
[13] 3e2500c59ab4ea1d914ac27cec5ae528	45dfe530d26895e48996b0a7e7fcd442
[15] 4d877a995bfda26baf8b6ab8fd0547e3	51200d29d1fc15f5a71c1dab4bb54f7c
[17] 59727fa212140da7999b9e659b3196aa	5b51b199bebdc111d995e0821ca9cd83
[19] 5cde2aa811d16198dd1921fbab4fba6a	5dc2596e6bd8d9a511a4af7e6d1fef81
[21] 6e28943943dbed3c7f82fc05f269947a	6e79db69d2527cbc796e0748e1f902f4
[23] 763ae14091aaa5e830edbb098aae73b5	867c51ec0949a1a888b8ddd7ccd77ca8
[25] 8fa2a95ee83cd1633cfd64f78e856bd3	9db6faeef387dc789777227a8bed4d52
[27] 9dd16e049becf4d5087c90a83fea403b	aa42197b97746ff2ccf265c398e99053
[29] b53b8bcccc2850c4f7bc651343e63dc0	b84a25d15d9d44f58640a485c1387df8
[31] be590ce170386babcf557b982eee50b4	d99c51f199953f710e9ce8f6f8015daa
[33] e3b4bfa08634b6659130b3ebaa2c0bdc	e5dc75a344103af785484b74becc0c22
[35] e82a7c661e834d935311ae02bdb4314c	ebd64e2bf193fc8c658af2b91952ce8d
[37] f05da679342107f92111ad9d65959cd3	f194ca6b8132aaa26558b309fa044e01
[39] f2682c2ed6193e284aef8107b2e62a4e	fae034d20a4c0184e93c2c5594a934c4
[41] fb45343c4635769a36c5072e04b6b978	fba95413ea8c28348cd9df92add4f111

Cuadro B.5: Categorías de la variable ElementID. Identifica el código con el cual se reconoce a dicho elemento alarmado.

B.1. Análisis de la variable respuesta en relación con el tiempo

En el primer grafo, se puede ver a la variable OwnerName, (Componente Padre), en el eje x, y a la variable Time Of Arrival, en el eje y. Aquí, se busca analizar, si durante el día los eventos relacionados con un mismo componente padre, se sucedían de algún modo en especial, si existía algún patrón diario. Se realiza de esta forma, porque los momentos de los eventos no estaban equidistantes. Por tanto, se analiza si existe relación entre el horario del día en que aparece dicha categoría de la respuesta, y la respuesta exhibida. Además, este gráfico permite apreciar, si existen patrones entre las categorías respecto del momento del día.

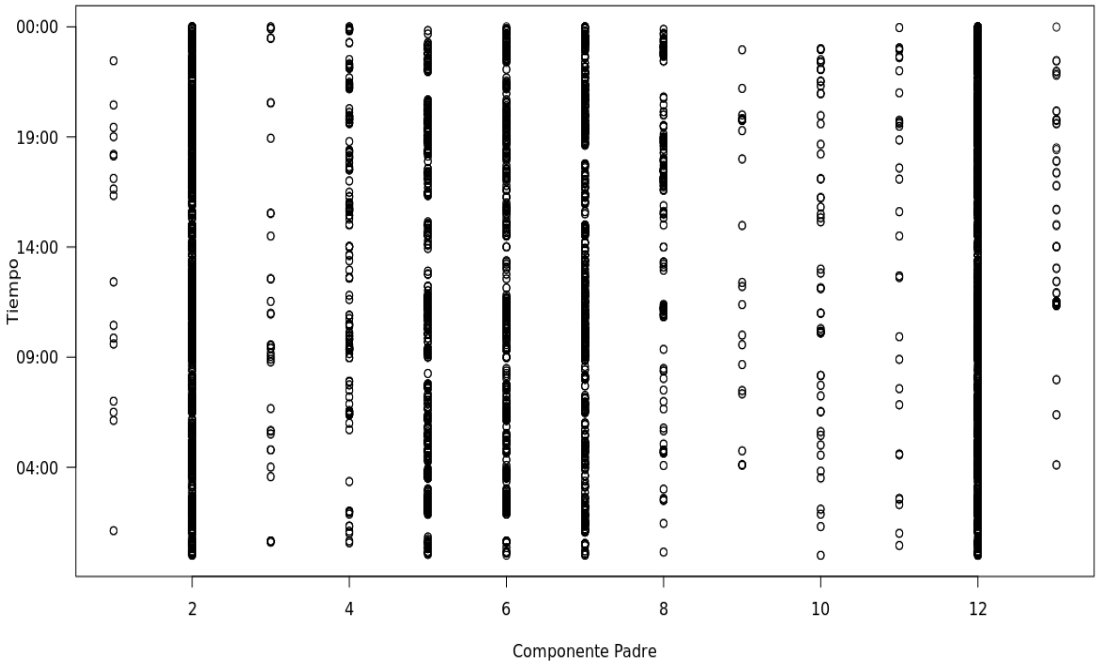

Figura B.1: Gráfico de la relación entre la variable respuesta OwnerName, y el tiempo.

Dado que, este primer grafo aparenta exhibir relación en algunos casos, se realiza otro. Esta vez, vinculando la respuesta con el número de orden que posee, para intentar divisar patrones entre las categorías, en cuanto al orden de aparición.

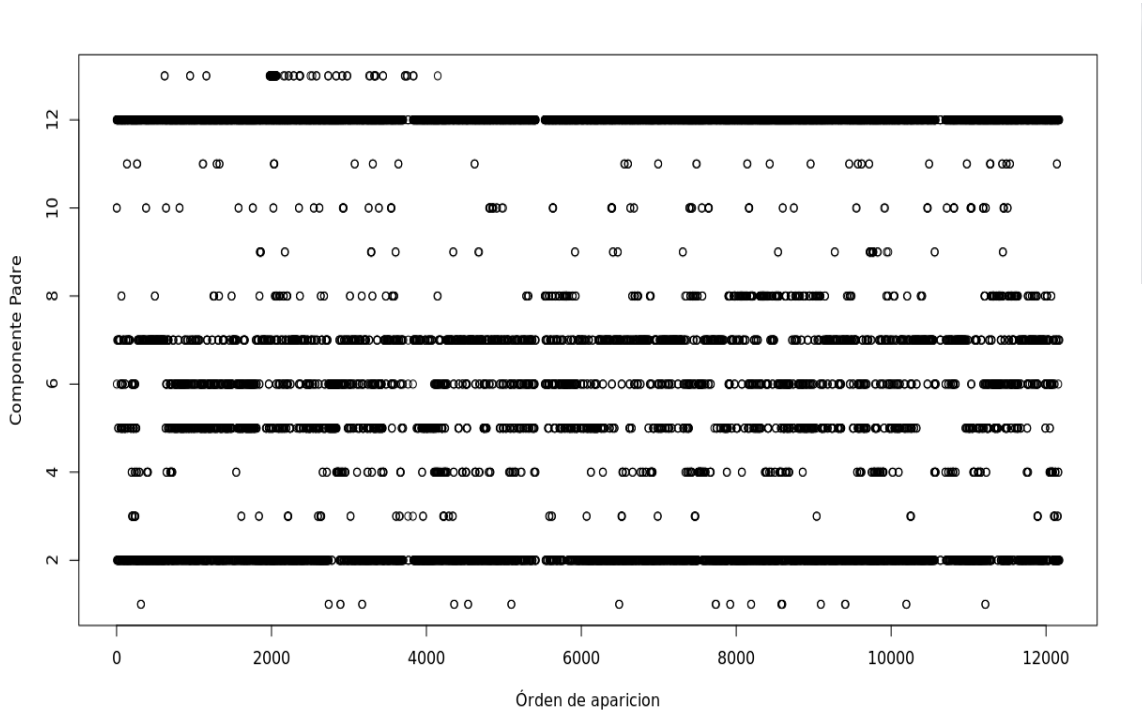

Figura B.2: Gráfico de la relación entre la variable respuesta OwnerName, en sus distintas categorías y el orden de aparición de las mismas.

En este segundo grafo, presentado sobre estas líneas, se exhibe una aparente relación entre algunas de las categorías, pero dentro de sí mismas, y no tanto respecto de otras. Con la intención de aclarar el panorama al respecto, se realiza un tercer grafo, que exhibe las categorías de la respuesta, pero en relación a la distancia en segundos existente entre, el momento en que una cierta categoría se presenta, y su reaparición. Con ello, se busca ver si realmente hay categorías, que se presentan de manera regular.

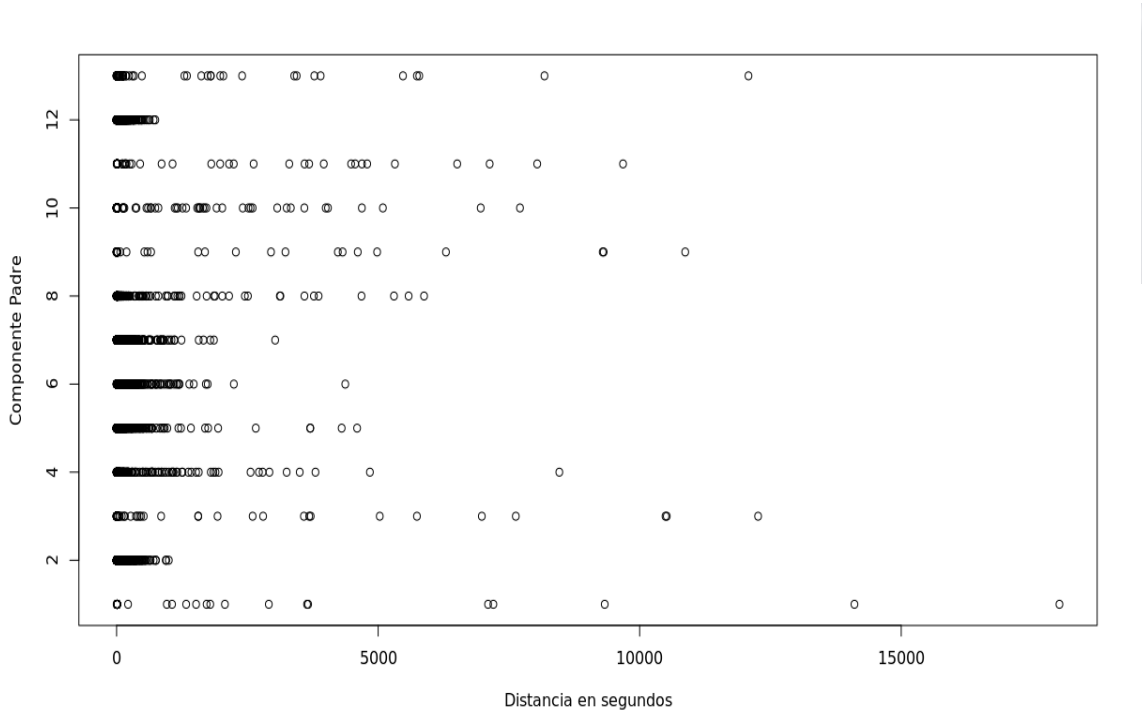

Figura B.3: Gráfico de la relación entre la variable respuesta OwnerName, en sus distintas categorías, y la cantidad de segundos que pasa entre apariciones de la misma categoría de dicha variable.

Este tercer grafo nos señala, que la mayoría de los sucesos de la misma categoría aparecen con poca distancia en segundos entre si, y que luego se van espaciando, según los datos. Esto es comprensible, ya que si salta una alarma de un elemento hijo, en la columna respuesta aparecerá su OwnerName, y es muy probable que a continuación, se presenten alarmas en todos los hijos de dicho OwnerName, si el inconveniente está en él, y no en el elemento hijo alarmado.

Por último, se realiza un grafo buscando cuales son aquellos OwnerName que más se repiten, y he aquí el grafo resultante. En él, cada vez que un valor de OwnerName aparece una única vez en ese instante, se representa con un punto, cuando aparece dos, es una raya que equivale a dos pétalos, si aparece 4 veces, serán 4 pétalos, (dos rayas cruzadas), y así sucesivamente. Así, sabremos cuantas veces en un mismo instante de tiempo, se reportan diferentes incidentes, que tienen como causa origen el mismo padre o RCA en este TFM, (el mismo OwnerName), y para representar el tiempo, se usa el tiempo en UTC.

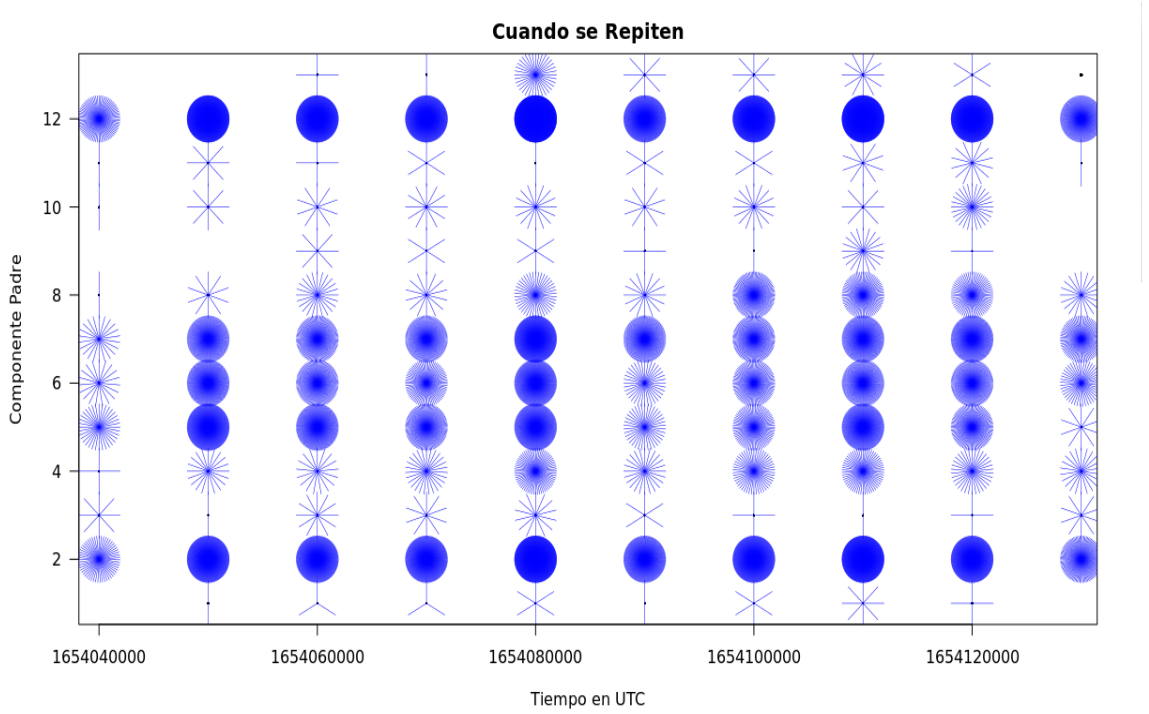

Figura B.4: Gráfico de repeticiones de la misma categoría de dicha variable.

Este gráfico, evidencia que hay unas categorías que se repiten más que otras. Además de que, en su mayoría, se repiten en los mismos momentos del día. Es por esto, que se utiliza la variable dist, (que contabiliza la distancia en segundos), para intentar mejorar los modelos. Luego de probar dicha variable, en las tres técnicas, se logra llegar a la conclusión de que solo aporta mayor complejidad al modelo, y no representa una mejora en cuanto a rendimiento del modelo, razón por la cual se elimina del modelo final elegido. Esto puede verse, en los resultados comparativos de las redes neuronales, con el resto de modelos, en donde, pese a ser una técnica capaz de encontrar relaciones complejas, entre gran cantidad de datos, para ella tampoco resulta relevante esa variable. Los mencionados resultados se encuentran en el cuadro que recoge los índices de los modelos a mano, en la sección de correspondiente 2.2.1.

B.2. Modelo a mano randomForest

A continuación, se presentan algunas de las combinaciones de variables probadas, con sus índices asociados. Luego de realizar un modelo utilizando randomForest, que incluya todas las variables explicativas que se consideren apropiadas, se analizan sus índices y el orden de importancia de las variables que el modelo brinda. Con ello, se realiza una selección paso a paso, de la cual se presentan algunos de los resultados obtenidos, al testear cada modelo sobre la muestra test. El modelo que incorpora 26 variables, 25 explicativas, y Owner name, expuso los siguientes índices, al probarlo sobre la muestra de test.

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
0.8178711	0.7683046	0.8004607	0.8343664	0.3437500	0.0000000	NaN

El gráfico de importancia de las variables, del citado modelo, ordenadas por el índice de Gini en forma decreciente, expone lo siguiente:

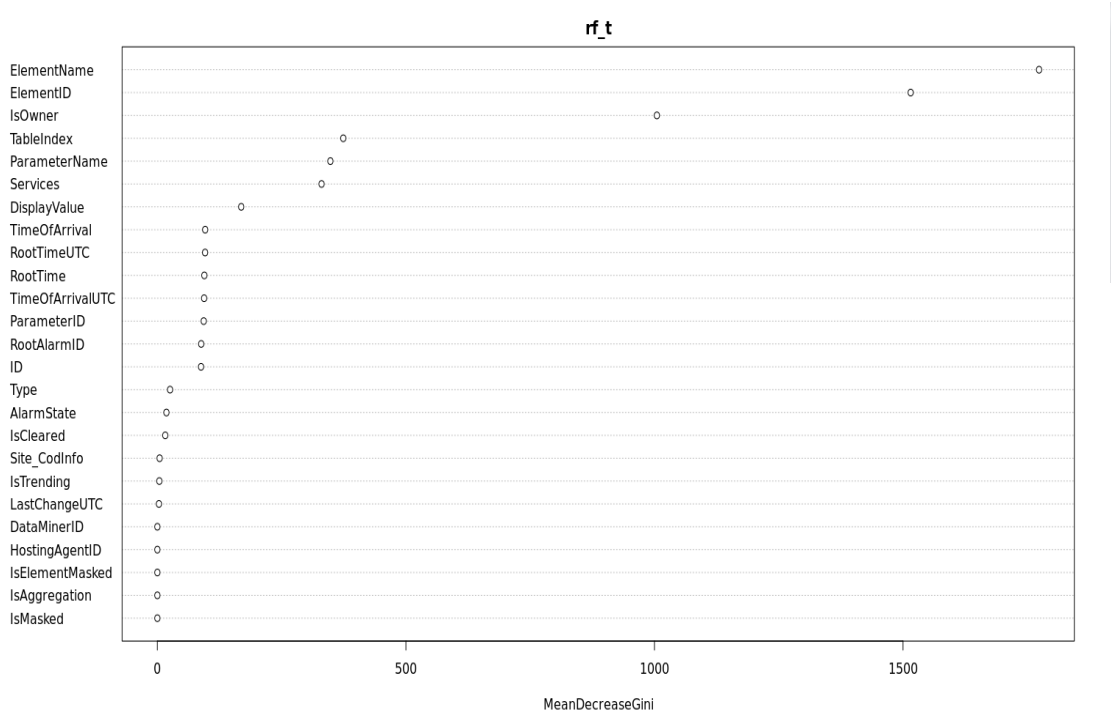

Figura B.5: Gráfico de importancia de las variables en el modelo de randomForest que incluye todas las variables explicativas de nuestro data-set.

Se comienza, entonces, con la selección de variables. Uno de los modelos probados se presenta a continuación. En él, se saca IsMasked, ya que, según el modelo total, es quien menos importancia tiene, y se añaden todas las demás variables del anterior modelo.

Overall Statistics
Accuracy : 0.8257
95 % CI : (0.8085, 0.8419)
No Information Rate : 0.3438
P-Value [Acc> NIR] : < 2.2e-16
Kappa : 0.7784

Como mejora, se continúa eliminando variables. En el caso siguiente, se quita IsAggregation, porque según el grafico de importancia de las variables, es la segunda menos importante, entonces se saca la anterior mas esta.

Overall Statistics
Accuracy : 0.8213
95 % CI : (0.804, 0.8377)
No Information Rate : 0.3438
P-Value [Acc > NIR] : <2.2e-16
Kappa : 0.7727

Empeora un poco, pero se prosigue eliminando variables del modelo. Ahora se elimina a IsElement-Masked, aquí se sacan las dos anteriores, mas esta.

Overall Statistics
Accuracy : 0.8242
95 % CI : (0.807, 0.8405)
No Information Rate : 0.3438
P-Value [Acc > NIR] : < 2.2e-16
Kappa : 0.7764

Vuelve a mejorar el índice Kappa, luego de buscar un poco más, se decide hacer una búsqueda a la inversa. A la vez, también se genera un modelo que cuenta con 19 variables, porque de las 28 que figuraban en el data-set original, se deben restar: la primera columna que era el número de orden del registro y la última que estaba vacía, luego de eso, también se sacan las columnas cuyo nombre contenga la palabra Time, o la palabra UTC, además también se quitan las columnas de ID y RootAlarmID, y es así que se llega a las 19 variables, 18 explicativas y la respuesta. Estos son sus índices:

Overall Statistics
Accuracy : 0.8394
95 % CI : (0.8227, 0.855)
No Information Rate : 0.3438
P-Value [Acc > NIR] : < 2.2e-16
Kappa : 0.7957

Como se puede ver vuelve a mejorar, pero la cantidad de variables es elevada aún, por tanto se continúa buscando un mejor modelo. Luego de varias pruebas más, en uno y otro sentido, se alcanza un modelo de 3 variables, que tiene mayor presición que el último presentado. Sus variables eran ElementName, ElementId y TableIndex, este modelo era superior en dichos índices al anterior. De esta forma, se iba en uno y otro sentido, incorporando y sacando variables, hasta dar con el modelo que mejor desempeño en la muestra de test mostrara, y cuya complejidad se justifique en dicho valor.

Los índices del modelo recientemente mencionado eran:

Overall Statistics
Accuracy : 0.8989
95 % CI : (0.8851, 0.9116)
No Information Rate : 0.3438
P-Value [Acc > NIR] : < 2.2e-16
Kappa : 0.8722

Luego de muchas pruebas se arriba al modelo final, presentado en el cuerpo de este documento.

Aquí se presenta el gráfico de la evolución del error, (obtenido directamente del modelo, y que es proveniente de la evaluación que este hace con la muestra de entrenamiento). Figura el OOB, en línea negra continua, y las diferentes categorías de la respuesta, en distintos colores y tipos de línea. Se puede apreciar que estos índices aparentan estabilizarse en torno a 50 o 60 árboles. Es por eso, que se recortan la cantidad de árboles a 100. Este hecho es poco convencional, ya que en general, se requiere de muchos más árboles para lograr dicho efecto. Este índice expone, la variabilidad del error en cada categoría. Cabe tener en cuenta que, el OOB es un promedio de todas las categorías, y que además, está medido sobre la muestra de entrenamiento, es por eso que exhibe dichos resultados.

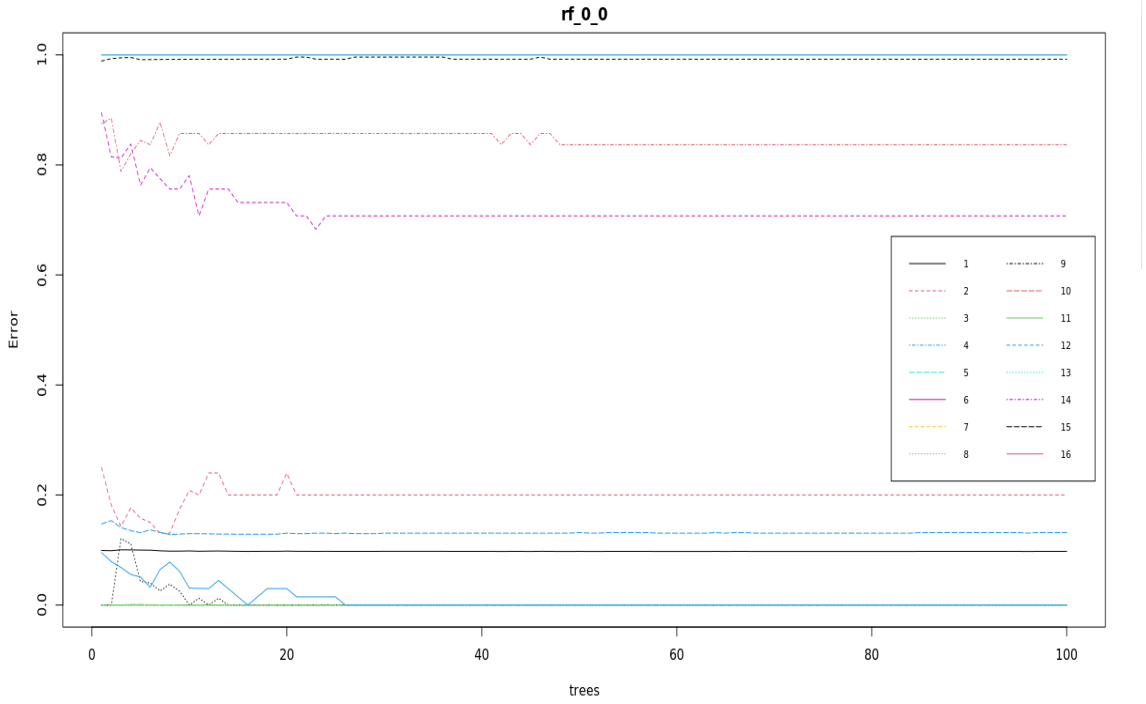

Figura B.6: Gráfico de la evolución del error OOB a medida que aumentan la cantidad de árboles, calculado para el mejor modelo de randomForest obtenido a mano

Para la mejor comprensión del gráfico anterior, se citan los nombres de las categorías, con las cuales se corresponde cada número en la leyenda del gráfico anterior. Se trata del índice OOB en primer lugar, que es la línea negra llena, y luego cada una de las categorías en las que está dividida la variable respuesta.

[1] OOB	[9] 763ae14091aaa5e830edbb098aae73b5
[2] 09bdc651461f25b65e47ee89adc56f99	[10]aa42197b97746ff2ccf265c398e99053
[3] 2e3ffce303618b8feba24ccb715dcdca	[11] d99c51f199953f710e9ce8f6f8015daa
[4]45dfe530d26895e48996b0a7e7fcd442	[12]e3b4bfa08634b6659130b3ebaa2c0bdc
[5]4d877a995bfda26baf8b6ab8fd0547e3	[13] e5dc75a344103af785484b74becc0c22
[6]5b51b199bebdc111d995e0821ca9cd83	[14]fba95413ea8c28348cd9df92add4f111
[7] 5cde2aa811d16198dd1921fbab4fba6a	[15] Service
[8]6e79db69d2527cbc796e0748e1f902f4	[16]System

Del análisis de estos resultados, se puede intuir que existen variables que están correladas, es decir, volcando información que se superpone dentro del modelo. Esto sucedería en algunas categorías, y por el contrario en otras falta información, para poder clasificar correctamente el dato. Es por eso que podríamos intuir, que el índice OOB en algunas categorías sale tan alto.

Luego, si se analiza la importancia de las variables predictoras incluidas en el modelo, se puede ver que desde el punto de vista del indice de Gini, la variable que mas importante es ElementName. Dicha variable, posee un índice que cuadriplica al de la siguiente variable, en orden de importancia, que es TableIndex. Luego de ellas, le siguen con aproximadamente la mitad de dicho índice la variable ElementID, y finalmente IsOwner, con un valor cercano a la mitad de este último.

```
            MeanDecreaseGini
ElementName        3787.5300
ElementID           463.6566
IsOwner             332.4765
TableIndex          871.7848
```

Para el modelo, entonces, la variable que más información estaría aportando sobre la respuesta, es ElementName. Esto es comprensible desde el punto de vista técnico, porque si se sabe el nombre del elemento, se puede identificar con bastante precisión, para poder saber cual es el elemento padre de este. Pero este hecho, aparentemente, no es tan inequívoco como se podría pensar, ya que requiere de TableIndex, y de las otras 3 variables para completar la información, y alcanzar el rendimiento expuesto, en los índices presentados en el cuerpo del presente documento.

Esta fue la mejor combinación de variables hallada, luego de realizar sendas combinaciones de las numerosas variables presentes en el data-set sintético de forma manual. Como se puede apreciar, por ser un data-set tan grande, tal tarea demanda muchísimo tiempo, esfuerzo y recursos computacionales.

B.3. Modelo a mano Party

Este modelo de árbol de clasificación, obtenido con la herramienta party, se genera a partir de una selección de variables hecha a mano. En un modelo previo a la generación del algoritmo de selección de variables, de modo automático.

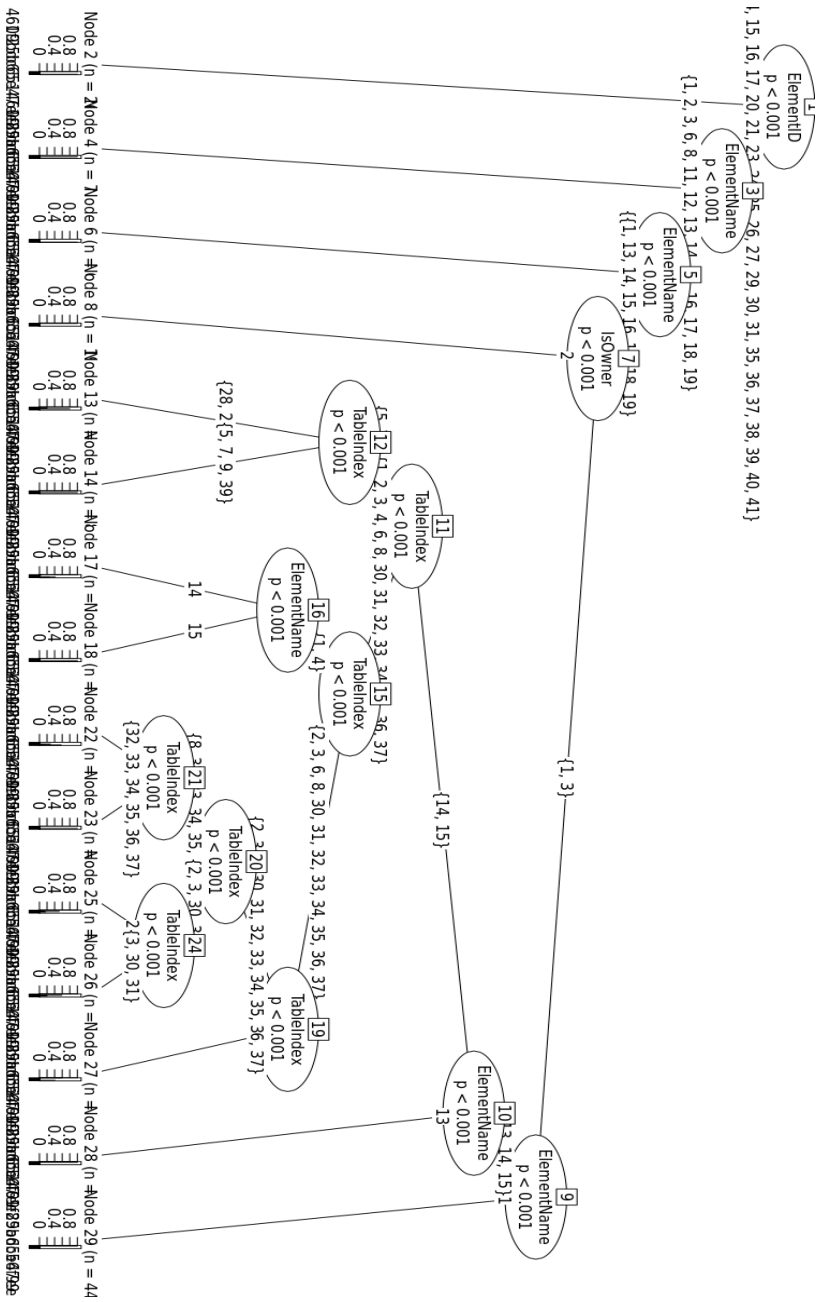

Figura B.7: Gráfico del mejor modelo de árbol, obtenido con party, realizando a mano la selección de variables

Como el gráfico es bastante confuso, es por esto que se presentan las reglas del árbol, con la intención de que se comprenda mejor su estructura.

```
 Conditional inference tree with 15 terminal nodes

Response:  OwnerName
Inputs:  ElementName, ElementID, IsOwner, TableIndex
Number of observations:  8193

1) ElementID == {1, 12, 18, 19, 22, 28, 32, 33, 34, 42}; criterion = 90.51, statistic = 90.51
  2)*  weights = 2693
1) ElementID == {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 20, 21, 23, 24, 25, 26, 27, 29, 30, 31, 35, 36, 37,
38, 39, 40, 41}
  3) ElementName == {20}; criterion = 74.155, statistic = 74.155
    4)*  weights = 758
  3) ElementName == {1, 2, 3, 6, 8, 11, 12, 13, 14, 15, 16, 17, 18, 19}
    5) ElementName == {2, 3, 6, 8, 11, 12}; criterion = 68.855, statistic = 68.855
      6)*  weights = 67
    5) ElementName == {1, 13, 14, 15, 16, 17, 18, 19}
      7) IsOwner == {2}; criterion = 68.367, statistic = 68.367
        8)*  weights = 1936
      7) IsOwner == {1, 3}
        9) ElementName == {13, 14, 15}; criterion = 52.326, statistic = 52.326
          10) ElementName == {14, 15}; criterion = 51.904, statistic = 51.904
            11) TableIndex == {5, 7, 9, 28, 29, 38, 39}; criterion = 37.472, statistic = 37.472
              12) TableIndex == {28, 29, 38}; criterion = 11.923, statistic = 11.923
                13)*  weights = 10
              12) TableIndex == {5, 7, 9, 39}
                14)*  weights = 851
            11) TableIndex == {1, 2, 3, 4, 6, 8, 30, 31, 32, 33, 34, 35, 36, 37}
              15) TableIndex == {1, 4}; criterion = 28.834, statistic = 28.834
                16) ElementName == {14}; criterion = 6.316, statistic = 6.316
                  17)*  weights = 101
                16) ElementName == {15}
                  18)*  weights = 40
              15) TableIndex == {2, 3, 6, 8, 30, 31, 32, 33, 34, 35, 36, 37}
                19) TableIndex == {2, 3, 8, 30, 31, 32, 33, 34, 35, 36, 37}; criterion = 31.039, statistic = 31.039
                  20) TableIndex == {8, 32, 33, 34, 35, 36, 37}; criterion = 30.35, statistic = 30.35
                    21) TableIndex == {8}; criterion = 4.5, statistic = 4.5
                      22)*  weights = 14
                    21) TableIndex == {32, 33, 34, 35, 36, 37}
                      23)*  weights = 14
                  20) TableIndex == {2, 3, 30, 31}
                    24) TableIndex == {2}; criterion = 10.407, statistic = 10.407
                      25)*  weights = 881
                    24) TableIndex == {3, 30, 31}
                      26)*  weights = 710
                19) TableIndex == {6}
                  27)*  weights = 34
          10) ElementName == {13}
            28)*  weights = 40
        9) ElementName == {1}
          29)*  weights = 44
```

B.4. Modelo a mano NNet

Este modelo de redes neuronales, obtenido con la herramienta NNet a través de caret, se genera a partir de una selección de variables hecha a mano.Es un modelo obtenido con anterioridad a la generación del algoritmo de selección de variables, de modo automático.

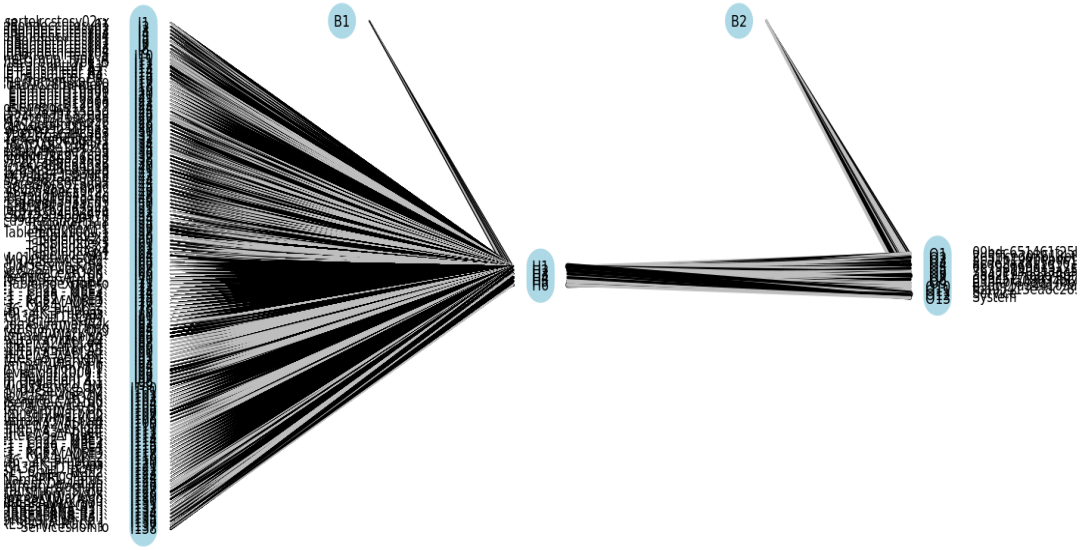

Figura B.8: Gráfico del mejor modelo de red neuronal obtenido con NNet a través de Caret realizando a mano la selección de variables

Como se puede ver, las líneas y los nombres de las variables se confunden muchísimo, para una comprensión general es recomendable tener en cuenta que mientras más gruesa es la línea más peso tiene, dicha categoría de esa variable, en el modelo. Además, si la línea es negra el peso es positivo, y si es gris el peso es negativo. A primera vista, dicho gráfico, ya señala la complejidad del modelo. Cabe aclarar, que se trata de una red que se considera básica, desde el punto de vista de la bibliografía estudiada.

A continuación, para entender mejor el gráfico anterior, se le solicita el summary del modelo final a la función NNet, alcanzado en esta fase. Allí aparecen los pesos asociados a cada transformación, sobre cada una de las categorías de cada variable, además constan el número de hiperparámetros o pesos, que en este caso se trata de 597. También informa que la cantidad de nodos en la capa de entrada es de 275, (que se corresponde con la cantidad de categorías incluidas en el modelo, a través de las variables explicativas). Allí consta además, que hay 2 nodos en la capa oculta, y que se tienen 15 nodos en la capa de salida. Sobre el final de la salida figura que, la tasa de aprendizaje óptima, (decay), es de 0.001, para el modelo final. Y finalmente indica que, por tratarse de variables categóricas, utiliza la función softmax, como función de activación en los nodos finales. Si se le solicitara a la función el modelo final elegido, esta volvería a indicar los hiperparámetros, nodos y capas. Agregando los nombres de las categorías incluidas en el modelo, no se cita aquí dicha información, debido a su gran extensión. Estos datos, más allá del hecho de brindar mayor información de la red, sirven para observar la complejidad de una red neuronal básica. Dicha red cuenta, en este caso, con 7 variables explicativas, donde tanto estas

como la respuesta son de tipo categóricas, con una gran cantidad de categorías dentro. Por ello, es que se requiere de una especial atención en la selección de variables, para no complicar innecesariamente un modelo, que ya de por sí, es complejo. Por este motivo, es que se decide sacar la variable dist, mencionada anteriormente, quien era la que encargada de contar la distancia, en segundos, entre los modelos.

Dada la complejidad vista, en algunos ámbitos se suele conocer a las redes neuronales, como modelos de caja negra. Porque es extremadamente difícil poder explicar la relación entre las variables de entrada, y la respuesta. A diferencia de la claridad que brinda, por ejemplo, un modelo de árbol de clasificación, o incluso un modelo que incorpora bosques aleatorios, que son muchos árboles combinados.

B.5. Modelo general base, para randomForest con algoritmo automático

Aquí, primero se presentan los datos del modelo randomForest con todas las variables explicativas, sin las que recogen fechas y horas de ocurrencia. Este, sirvió de base para seleccionar las 10 variables mas relevantes según el índice de Gini, entregado por dicho modelo. El señalado índice es quien ordena las variables, por nivel de importancia. Dicho modelo, cuenta con 21 variables explicativas, y sus nombres son: DataMinerID, HostingAgentID, ID, RootAlarmID, ElementID, ElementName, IsElementMasked, ParameterID, ParameterName, TableIndex, DisplayValue, AlarmState, Type, IsAggregation,IsMasked, Services, IsTrending, IsOwner, LastChangeUTC, IsCleared y Site_CodInfo.

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
0.8417969	0.7995656	0.8252605	0.8573482	0.3315430	0.0000000	NaN

Cuadro B.6: Índices del modelo base con todas las variables

En este caso se puede ver, que los índices alcanzados por el modelo, a pesar de contar con todas las variables explicativas, no llegan a ser tan altos, como los alcanzados en el modelo mediante la búsqueda manual. Tampoco son superiores, a los alcanzados luego de ejecutar el algoritmo de búsqueda automática, y realizar el modelo con menos variables. Dichos indicadores, se encuentran presentes en el cuadro 2.5, para el caso de randomForest.

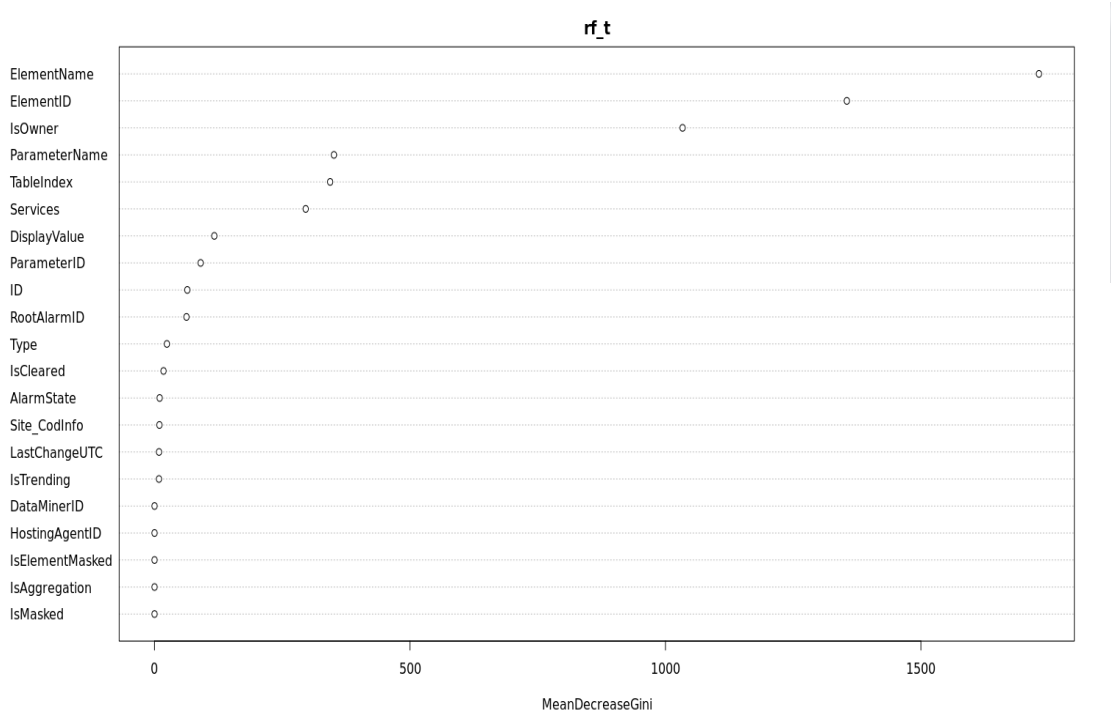

Figura B.9: Gráfico de importancia de las variables en el modelo de randomForest general, para extraer las 10 más importantes, y ejecutar el algoritmo automático de búsqueda.

Si no se tiene en cuenta ni el ID, (que lo calcula la función dplyr, para realizar las muestras), ni el RootAlarmID, (que actúa como un número de las incidencias, y que solo aparece dos veces si dicha incidencia tienen alguna modificación), entonces las 10 variables que quedan para incluir en el algoritmo automático son: ElementName, ElementID, IsOwner, ParameterName, TableIndex, Services, DisplayValue, ParameterID, IsCleared y Type.

A continuación, se puede ver el gráfico de la convergencia del error, en el modelo general, para cada categoría, y para el OOB en su conjunto.

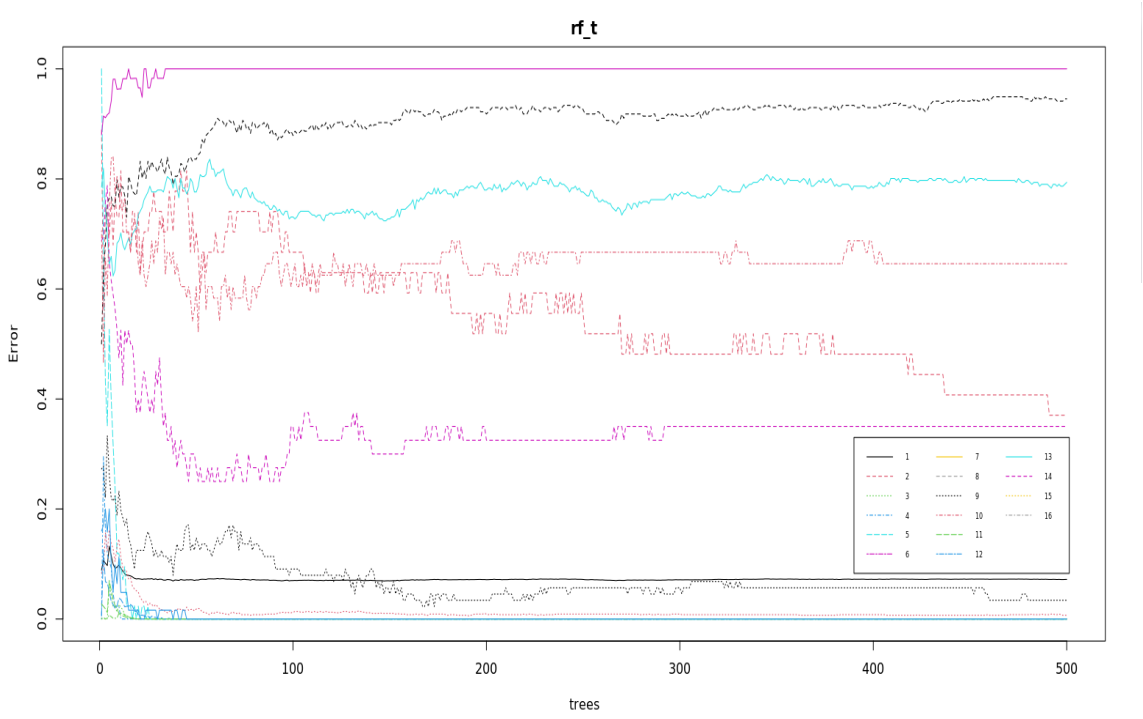

Figura B.10: Gráfico del error OOB para el modelo con todas las variables, previo a ejecutar el algoritmo automático de búsqueda.

En el gráfico anterior, se puede ver que el error OOB, de cada categoría, no esta muy estable, con 500 árboles para un modelo con 21 variables. No se realizó con 2000 árboles o más, dado que colapsaba la herramienta de cálculo por la cantidad de variables, y sus categorías. De hecho su OOB es inferior al del modelo siguiente, pero su rendimiento no es superior, teniendo incluso 3 veces la cantidad de variables, que posee el modelo entregado por el algoritmo automático.

Para la mejor comprensión del gráfico anterior, se citan los nombres de las categorías con las cuales se corresponde cada número, en la leyenda del gráfico anterior. Se trata del índice OOB en primer lugar, que es la línea negra llena, y luego cada una de las categorías en las que está dividida la variable respuesta, (siempre hablando en referencia al data-set sintético).

[1] OOB	[9] 763ae14091aaa5e830edbb098aae73b5
[2] 09bdc651461f25b65e47ee89adc56f99	[10]aa42197b97746ff2ccf265c398e99053
[3] 2e3ffce303618b8feba24ccb715dcdca	[11] d99c51f199953f710e9ce8f6f8015daa
[4]45dfe530d26895e48996b0a7e7fcd442	[12]e3b4bfa08634b6659130b3ebaa2c0bdc
[5]4d877a995bfda26baf8b6ab8fd0547e3	[13] e5dc75a344103af785484b74becc0c22
[6]5b51b199bebdc111d995e0821ca9cd83	[14]fba95413ea8c28348cd9df92add4f111
[7] 5cde2aa811d16198dd1921fbab4fba6a	[15] Service
[8]6e79db69d2527cbc796e0748e1f902f4	[16]System

Finalmente, al comparar con el gráfico que se alcanza, mediante el modelo buscado con el algoritmo automático, se puede ver que el error OOB, de cada categoría, está más estable en el presente modelo, aunque es algo más alto. Este hecho, se puede observar, también, en el cuadro 2.6.

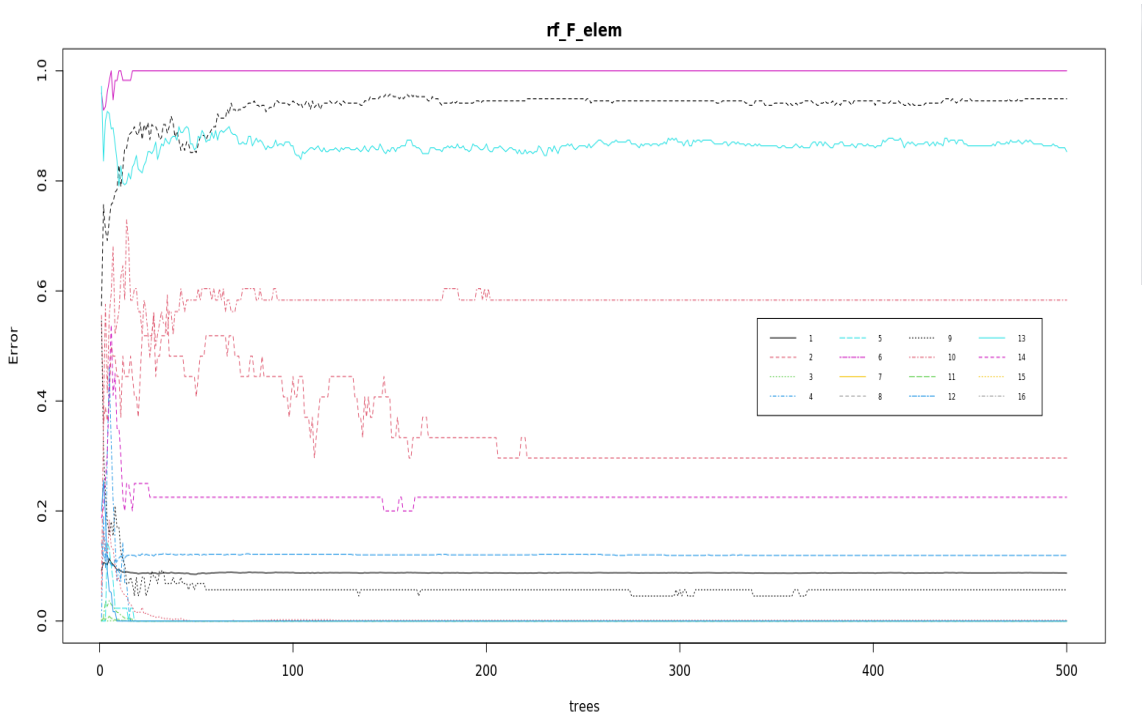

Figura B.11: Gráfico del error OOB, para el modelo alcanzado con el algoritmo automático de búsqueda.

B.6. Modelo general base, para NNet y Party, con algoritmo automático

En esta sección, se presentan los datos del modelo randomForest con todas las variables explicativas, sin las que recogen fechas y horas de ocurrencia, y con la variable DisplayValue, dividida en dos. Este, sirvió de base para seleccionar las 10 variables mas relevantes, según el índice de Gini, entregado por dicho modelo. Como se mencionó anteriormente, el señalado índice es quien ordena las variables, por nivel de importancia. Dicho modelo, cuenta con 22 variables explicativas, y sus nombres son: DataMinerID, HostingAgentID, ID, RootAlarmID, ElementID, ElementName, IsElementMasked, ParameterID, ParameterName, TableIndex, DisplayValue_num, DisplayValue_txt, AlarmState, Type, IsAggregation, IsMasked, Services, IsTrending, IsOwner, LastChangeUTC, IsCleared y Site_CodInfo.

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
0.8496094	0.8095394	0.8333849	0.8648259	0.3315430	0.0000000	NaN

Cuadro B.7: Índices del modelo base con todas las variables

En este caso, los índices alcanzados por el modelo, son superiores a los del modelo base para randomForest, presentado anteriormente. Además, a pesar de contar con todas las variables explicativas, dichos índices, no llegan a ser tan altos, como los alcanzados en el modelo mediante la búsqueda manual. Tampoco son superiores, a los alcanzados luego de ejecutar el algoritmo de búsqueda automática, y realizar el modelo con menos variables. Dichos indicadores, se encuentran presentes en el cuadro 2.7, para el caso de Party, y en el cuadro 2.9, para el caso de NNet.

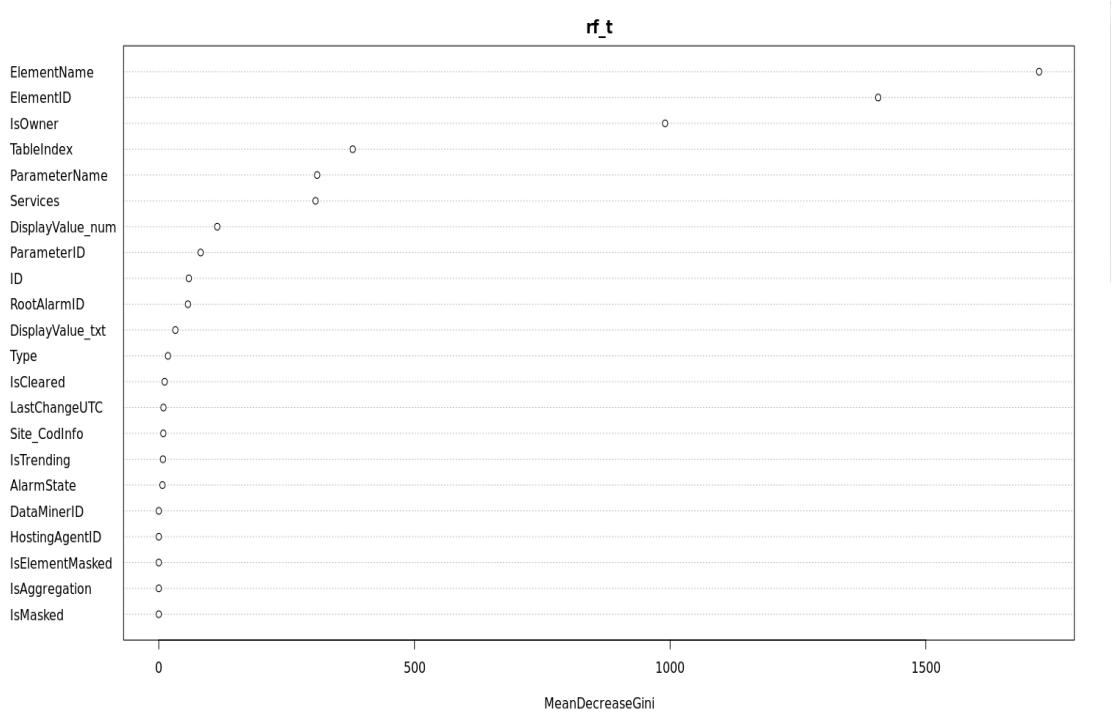

Figura B.12: Gráfico de importancia de las variables, en el modelo de randomForest general, para extraer las 10 más importantes, y ejecutar el algoritmo automático de búsqueda con ellas.

Si no se tiene en cuenta ni el ID, (que lo calcula la función dplyr, para realizar las muestras), ni el RootAlarmID, (que actúa como un número de las incidencias, y que solo aparece dos veces si dicha incidencia tienen alguna modificación), entonces las 10 variables que quedan para incluir en el algoritmo automático son: ElementName, ElementID, IsOwner, ParameterName, TableIndex, Services, DisplayValue_num, ParameterID, DisplayValue_txt y Type.

A continuación, se puede ver el gráfico de la convergencia del error, en el modelo general, para cada categoría, y para el OOB en su conjunto.

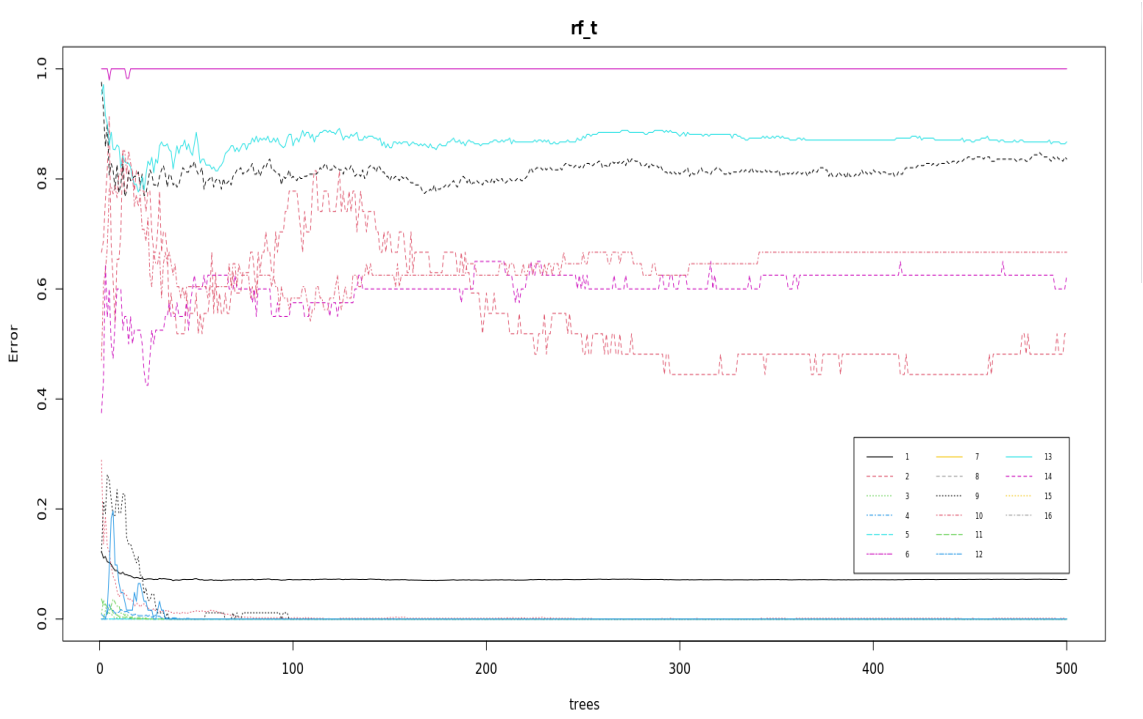

Figura B.13: Gráfico del error OOB para el modelo con todas las variables previo a ejecutar el algoritmo automático de búsqueda.

En el gráfico anterior, se puede ver que el error OOB, de cada categoría, no esta muy estable, con 500 árboles para un modelo con 22 variables. Además, su rendimiento no es superior, teniendo incluso mas de 3 veces, la cantidad de variables que poseen los modelos entregados por Party y NNet, con el algoritmo automático.

Los nombres de las categorías con las cuales se corresponde cada número, en la leyenda del gráfico anterior, son los mismos que los presentados en la sección anterior, ya que la variable respuesta no se ha modificado. Se trata del índice OOB en primer lugar, que es la línea negra llena, y luego cada una de las categorías en las que está dividida la variable respuesta, (siempre hablando en referencia al data-set sintético).

Apéndice C

Datos sobre el ordenador utilizado

La empresa ha informado que se trabajó en un servidor de su propiedad. Este contaba con una CPU de 4 núcleos, 16gb. de memoria RAM y disco rígido de 80gb.

Bibliografía

[1] [Material Online] Breiman L., Cutler A., (2022) randomForest: Breiman and Cutler's Random Forests for Classification and Regression. R package version 4.7-1.1 , Basado en Breinman (2001). Accedido 25 de abril de 2023.

https://cran.r-project.org/web/packages/randomForest/randomForest.pdf.

[2] [Libro] Chollet, F., y Allaire, J. J. (2018). Deep Learning with R. Manning Publications.

[3] [Material Online]Everitt,B., Hothorn, T. (2023). Package ĺSAUR2; A Handbook of Statistical Analyses Using R (2nd Edition), Capítulo 9 "Recursive Partitioning: Predicting Body Fat and Glaucoma Diagnosis". Accedido 28 de abril de 2023.

https://cran.r-project.org/web/packages/HSAUR2/vignettes/Ch_recursive_partitioning.pdfl.

[4] [Libro Online]Fernández-Casal, R., Cao, R., Costa, J. (2023). Técnicas de Simulación y Remuestreo. Capítulo 3, "Análisis de resultados de simulación" Accedido 27 de abril de 2023.

https://rubenfcasal.github.io/simbook/index.html.

[5] [Libro Online] Fernández Casal R.,Costa Bouzas J. , Oviedo de la Fuente M., (2021) Aprendizaje Estadístico. Capítulo 2 "Árboles", Capítulo 3, "Bagging y Boosting" y Capítulo 8, "Redes neuronales" . Accedido 25 de abril de 2023.

https://rubenfcasal.github.io/aprendizaje_estadistico/.html.

[6] [Material online] Hothorn T., Hornik K., Zeileis A. (2023) party: A Laboratory for Recursive Partytioning. R package version 1.3-13. Accedido 25 de abril de 2023.

https://cran.r-project.org/web/packages/party/party.pdf.

[7] [Libro] Ishikawa K., (1976) Guide to quality control

https://openlibrary.org/books/OL4595409M/Guide_to_quality_control.

[8] [Tesis] Josefsson, (2017). Root-cause analysis through machine learning in the cloud (Dissertation).

http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-340428

[9] [Material online] Kuhn, et al. (2023) caret: Classification and Regression Training R package version 6.0-94 Accedido 25 de abril de 2023.

https://cran.r-project.org/web/packages/caret/caret.pdf.

[10] [Artículo] Lokrantz, Gustavsson, Jirstrand,(2018). Root cause analysis of failures and quality deviations in manufacturing using machine learning. Procedia CIRP, Vol. 72, 1057-1062.

https://doi.org/10.1016/j.procir.2018.03.229

https://www.sciencedirect.com/science/article/pii/S2212827118303895

[11] [Artículo] Ovalles Acosta, Gisbert Soler y Pérez Molina, (2017). Herramientas para el análisis de **Causa Raíz** (ACR). 3C Empresa: investigación y pensamiento crítico, Edición Especial, 1-9. DOI:

http://dx.doi.org/10.17993/3cemp.2017.especial.1-9

[12] [Material online] Ripley, Venables. (2023) NNet: Feed-Forward Neural Networks and Multinomial Log-Linear Models. R package version 7.3-18 Accedido 25 de abril de 2023.

https://cran.r-project.org/web/packages/NNet/NNet.pdf.

[13] [Artículo] Ruiz-López, González Rodríguez-Salinas, Alcalde-Escribano, (2005). Análisis de causas raíz. Una herramienta útil para la prevención de errores. Revista de Calidad Asistencial, Vol. 20, Nro. 2, 71-78.

https://www.elsevier.es/es-revista-revista-calidad-asistencial-256-articulo-analisis-causas-raiz-u