Reporte final del reto:
Análisis de tendencias y predicción de fallas en sistemas de bases de datos relacionales

Diana Guadalupe García Aguirre[1 [A01276380]]

Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Guadalajara, Av. Gral Ramón Corona 2514, Zapopan, MX

Abstract. Una plataforma de sistemas de bases de datos relacionales contiene, en sus capas física y lógica, un gran número de componentes que permiten que almacene información de manera confiable y robusta, siendo todos ellos importantes para la persistencia de los datos y el rendimiento, escalabilidad, mantenimiento y disponibilidad de sus contenedores. En tiempos modernos, las compañías dedicadas a ofrecer soluciones empresariales de almacenamiento de datos físico y en la nube -un servicio conocido como *Cloud Storage*-, han dedicado esfuerzos titánicos para implementar herramientas que permitan simplificar el mantenimiento de sus sistemas y asegurar la detección de errores a sus clientes. El uso de algoritmos de aprendizaje profundo supone, por ende, una mejora que puede coadyuvar de forma potencial en el cometido antes descrito y reducir la cantidad de recursos necesarios y previsibles errores humanos. En el presente documento, se abordará el análisis e implementación de un algoritmo de esta índole, que se basa en los datos arrojados por una herramienta de chequeo de salud para este tipo de sistemas de almacenamiento, propiedad de Oracle®, y que ayudará a predecir el momento en el que dichos sistemas se encuentren en riesgo de incurrir en un error fatal.

Keywords: Plataforma de bases de datos relacionales · ORAchk® · Exadata · *Relational Database Management System (RDBMS)· Cloud Storage· Oracle Engineered Systems· Oracle Grid Infrastructure* · detección de errores automatizada· tendencias· predicción de errores críticos

1 Antecedentes e introducción

El buen funcionamiento de un sistema de software depende, de manera usual, en la forma en la que se ha configurado . Así, cualquier error de configuración, por mínimo que parezca, puede conseguir que los sistemas presenten comportamientos indeseados, con errores importantes y difíciles de diagnosticar y reparar. [1] [2] Esta misma premisa podría extrapolarse a los sistemas de hardware, que fungen un rol vital en la supervivencia del software y la existencia del almacenamiento de datos. En relación con esto último, en particular, si se habla de sistemas que

almacenan información muy importante y la distribuyen a numerosos clientes, en niveles de implementación corporativos, se tiene la necesidad imperiosa de contar con marcos de diagnóstico de errores fiables, que permitan mantener un nivel de desempeño óptimo de manera sostenida.

En los inicios de la década del 2010, Oracle contaba ya con una herramienta para auditar las configuraciones establecidas en sus sistemas, en categorías tales como: parámetros del *kernel* del sistema operativo, paquetes de software importantes para *Oracle Real Application Clusters* -en adelante RAC-, parámetros de funcionamiento de *Oracle Grid Infrastructure* y *Oracle Automatic Storage Management* -en adelante ASM-, así como configuraciones de bases de datos; llamada RACcheck® [3]. Brindaba un escaneo no intrusivo de todas estas áreas, que se ejecutaba por medio de un *daemon* integrado, lo que permitía ejecuciones silenciosas, programadas y sin intervención humana necesaria para ello. Creaba una serie de logs que le informaban al cliente sobre el estado actual de los servidores de bases de datos de Oracle, -por lo que se recomendaba correrlo especialmente al tener una instancia de base de datos cargada y ejecutándose-, y un reporte `.HTML` detallado que contenía beneficios, impactos, riesgos, acciones por tomar e información de reparación de varios aspectos del sistema analizado, según se requiriera.[4]

Asimismo, durante la década pasada, la ola del *Cloud Computing* comenzó a alzarse más y más, obligando a los proveedores de servicios e infraestructura computacional a tomar un segundo aire para replantearse sus modelos de negocio. Oracle, líder de la industria de las bases de datos desde hace más de treinta años, no se quedó atrás y, además de redefinir su oferta de servicios para ponerla a disposición de la nube, evolucionó su producto estrella de detección de errores de bases de datos automatizado, dando paso a lo que hoy por hoy se conoce como ORAchk®/EXAchk® y dejando atrás al antepasado RACcheck®. Este nuevo *framework* integraba las utilidades de RACcheck® con nuevas funcionalidades, que van perfeccionándose gracias a los clientes y a los desarrolladores, que van implementando mejoras de manera continua y conjunta.

Se describe en el manual *Oracle Engineered Systems for High Availability* [5] el funcionamiento de ORAchk® y EXAchk®, que luego de este párrafo se referenciarán únicamente como ORAchk®. Ambas herramientas son útiles para obtener chequeos de salud y pruebas de buenas prácticas en entornos Oracle de una forma comprensible para el usuario. Se pueden englobar a la postre sólo como ORAchk®, como ya se ha mencionado, pues su funcionamiento es equivalente en todo excepto en lo que concierne a la plataforma de Exadata [6], -que engloba un amplio servicio de bases de datos de Oracle en la nube-, como los chequeos a las celdas de almacenamiento, por ejemplo. En este documento, debido a la naturaleza del proyecto, resulta más útil mencionar a ORAchk®, por ser la versión que se encarga exclusivamente de productos de Oracle, pues su contraparte es capaz de analizar sistemas que no pertenecen al *Oracle Database Appliance* y el objetivo es poder ampliar el banco de conocimiento de esta compañía sobre su propia línea de negocio principal.

Conviene mencionar que ORAchk® ya viene preinstalado como parte del kit

de herramientas de *Oracle Grid Infrastructure*. Sin embargo, puede descargarse también desde servidores de Oracle de manera directa, algo recomendado para asegurar siempre que se cuenta con la versión más reciente del programa [7]. Incluso, convendrá siempre conectar cualquier plataforma que ejecute la herramienta a la red para que pueda mantenerse actualizada. ORAchk® es un programa no intrusivo, igual que su antepasado. Corre de manera silenciosa y ágil, en segundo plano, gracias a su *daemon*, configurado por el usuario, y es capaz de almacenar los resultados obtenidos por el tiempo que el usuario determine y de enviar notificaciones y alertas de funcionamiento del sistema vía correo electrónico [5] [8] [9] [10].
Al ser una herramienta en constante evolución, se ha planteado la posibilidad de mejorarla por medio de modelos de aprendizaje de máquina. En este caso, se ha sugerido, mediante un levantamiento de requerimientos exhaustivo, que se describe en diversos lugares de este documento, según sea pertinente, que una de las mejoras más prometedoras consistiría en poder anticiparse a un estado de falla de los sistemas que ORAchk vigila. Es decir, en contar con un modelo, implementado dentro de la misma suite de software, capaz de predecir un estado de error basándose en el contexto proveído por los datos obtenidos a partir de los escaneos realizados. El camino proyectado para ello no es corto ni falto de dificultades, pero resulta, cuando menos, prometedor. En el presente documento, se relatará el proceso y los hallazgos obtenidos en la primera iteración en la búsqueda de dicho modelo, donde se proponen dos modelos estadísticos y uno de aprendizaje profundo que buscan predecir un estado de falla a partir de un conjunto de variables.
A manera de descargo de responsabilidad, vale la pena mencionar que este reporte incluye únicamente los hallazgos, metodologías y aprendizajes de once semanas de trabajo. El proyecto, cuyos alcances en cuestión de tiempo podrían verse incrementados cuantiosamente en el futuro en vista del alcance y recursos disponibles hasta el momento de redacción de este escrito, consideró también los entrenamientos previos en materia de inteligencia artificial y conocimientos técnicos internos de la empresa involucrada. [11].

2 Metodología

La metodología CRISP-DM, cuyas fases se ilustran en la figura 1, fue acuñada en 1996 por expertos provenientes de tres organizaciones que, en aquel momento, eran punta de lanza del mercado en ciernes de la minería de datos, como una manera de estandarizar y regular las prácticas que gobernarían esta industria, aún inmadura y con un potencial tan enorme que resultaba casi atemorizante. Se trataba de NCR, SPSS y Daimler Chrysler. [12]
De acuerdo con lo expuesto por Haya (2021), la estandarización no buscaba únicamente regular este tipo de prácticas, sino también dar control a quienes se dedican a este ramo sobre sus resultados. Se trata de una metodología capaz de iterar sobre el propio proceso que se sigue, de tal forma que se pueda resolver el

problema planteado de manera eficaz. En palabras del Espinoza-Zúñiga (2020), IBM es la empresa que más promueve el uso de esta metodología, si bien se trata en nuestros tiempos de un estándar de facto en el desarrollo de soluciones de minería de datos, por lo que no solamente IBM lo usa, sino también cientos de mineros de datos en otras industrias y corporaciones y, cabe resaltar, es el más usado en la academia.
CRISP-DM son las siglas en inglés de *CRoss Industry Standard Process for Data Mining* y describen el proceso que idealmente debería seguirse para la Ciencia de Datos. Consiste en seis etapas:

- Entendimiento del negocio, o *business understanding*, etapa crucial en la que se entiende lo más posible la problemática y se conocen los requisitos, restricciones y beneficios de ejecutar el proyecto. De igual forma resulta útil para marcar un punto de partida, un marcador de inicio, que permita contrastar el estado del negocio previo y posterior a la minería de datos al llegar a la quinta etapa.
- Entendimiento de los datos, o *data understanding*, etapa en la que se identifican claramente los formatos, contenidos y roles de los datos con los que se cuenta, de forma que sea posible manipularlos posteriormente con soltura y eficacia.
- Preparación de los datos, o *data preparation*, etapa que, por regla general, suele consumir más tiempo, dado que permite seleccionar aquellos datos que sean útiles para el posterior modelado y eliminar aquellos que no aporten realmente -o cuya aportación sea virtualmente nula- en ese sentido.
- Modelado, o *modeling*, etapa en la que se procesa lo aprendido en la etapa previa para generar modelos estadísticos y algorítmicos mediante técnicas apropiadas de acuerdo con el problema a resolver. También implica el cambio de parámetros iterativo hasta encontrar un modelo viable y óptimo.
- Evaluación, o *evaluation*: En la penúltima etapa, por medio de métricas concretas, se evalúa la calidad del modelo previamente planteado, ya sean medidas estadísticas o un análisis con los clientes al aplicar la solución propuesta. De ser necesario, se iterará a partir del paso previo conveniente, pudiendo ser el caso de tener que comenzar desde cero.
- Despliegue, o *deployment*, etapa final en la que se documenta el proceso para recabar todo el conocimiento adquirido en un corpus concreto y se monitorea la implantación de la solución, en aras de detectar áreas de oportunidad.

Durante once semanas se ha trabajado siguiendo los pasos marcados por esta metodología. Cabe hacer mención del hecho de que el proyecto principal del que se desprende este reporte se encuentra actualmente en las fases de entendimiento y preparación de datos, por lo que sólo se alcanzará el estatus de evaluación y, en el presente documento, se plasmará el proceso en una suerte de despliegue previo.

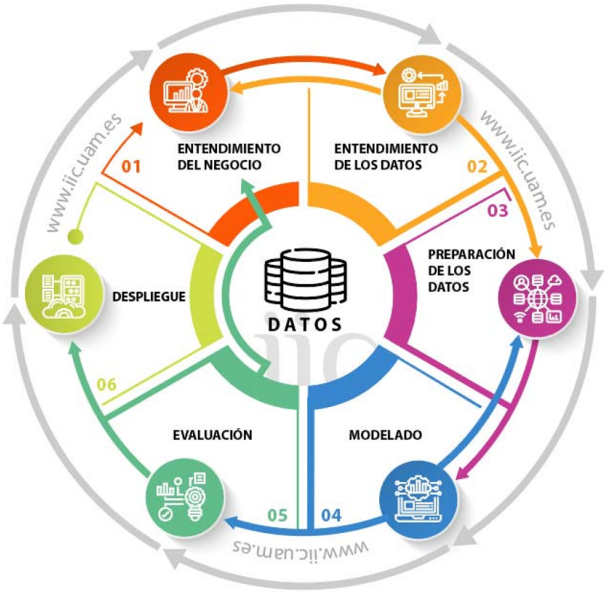


Fig. 1. Esquema del ciclo CRISP-DM estándar. Tomado del sitio web del Instituto de Ingeniería del Conocimiento de la Universidad Autónoma de Madrid, en https://www.iic.uam.es/innovacion/metodologia-crisp-dm-ciencia-de-datos/

3 Análisis exploratorio de los datos

Con el objetivo de realizar cualquier tipo de análisis de datos, se requiere, por supuesto, describirlos, conocerlos y visualizarlos en primera instancia. En este punto del documento, cabe destacar algunos puntos importantes respecto a su manejo y disponibilidad:

- El conjunto de datos que se utilizará en este documento para evidenciar la estructura con la que se cuenta, así como el usado para entrenar el modelo aquí descrito, han sido previamente sanitizados y anonimizados para proteger los intereses de todas las partes involucradas, así como de los entes propietarios de dichos datos, de acuerdo con las legislaciones aplicables en nuestro país y el estado de Jalisco y lo que consta en las políticas de la compañía que los provee, misma que previamente recolectó los datos gracias a un acuerdo de confidencialidad y anonimato absolutos con sus antiguos propietarios. Todos los datos recabados se usan e ilustran en este documento con fines meramente académicos y se han censurado de manera adicional para poder mostrarlos aquí.
- El acceso a los datos se logra a través de un canal especializado y confidencial, del que sólo los interesados tienen conocimiento, y que cumple con las normativas expresadas en el acuerdo entre la compañía propietaria de

los conjuntos de datos y demás entes involucrados en el desarrollo de este proyecto.
- Los datos recabados hasta este momento, debido a los costes de sanitización, son limitados. Se usarán para realizar un análisis exploratorio que se pueda mostrar en este documento y, posteriormente, los modelos propuestos y construidos se probarán de manera interna en la compañía que los provee. Aquí se incluirán únicamente los resultados obtenidos en dichas pruebas, pero no se compartirá, bajo ningún concepto, el conjunto de datos de prueba, al no encontrarse debidamente sanitizado ni anonimizado.
- Dado lo anterior, no se compartirán datos de ninguna índole mediante repositorios de GitHub ni con personas ajenas a la realización del proyecto, siendo la única demostración del trabajo realizado el presente documento. Se anexa al final un documento que prueba la participación de la autora en el proyecto.

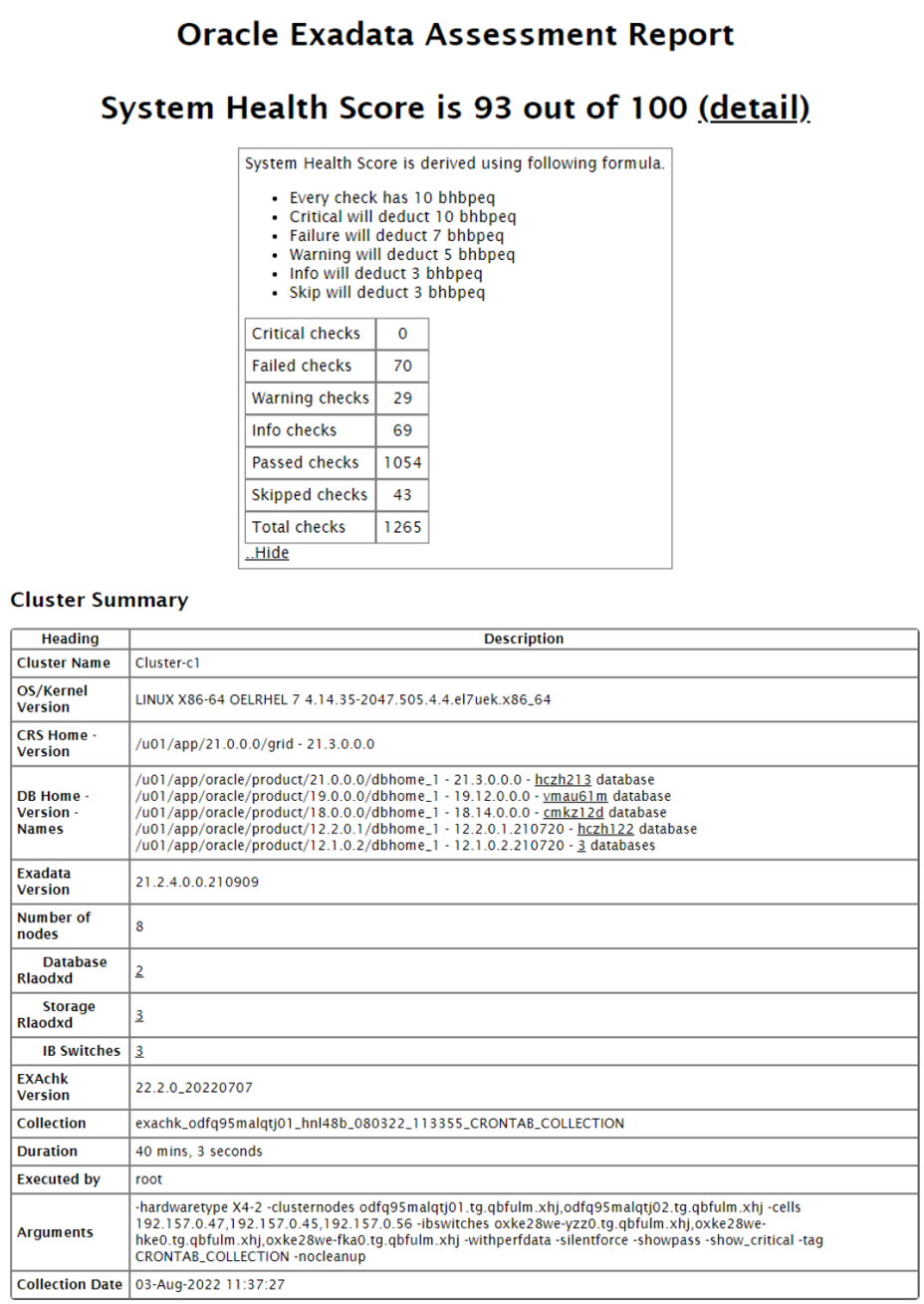
Dicho lo anterior, resulta preciso señalar la manera en la que ORAchk® trabaja, de manera general. La herramienta en cuestión realiza chequeos de diversas índoles en el sistema de almacenamiento, a niveles lógico y físico. Posteriormente, despliega los resultados obtenidos por medio de múltiples archivos en formato `.JSON`, `.HTML` y `.OUT`. A pesar de que existe una amplia variedad de archivos por analizar, por cuestiones de tiempo y alcance, para el presente análisis, se usará sólo uno de los archivos de salida de tipo `.JSON`, que contiene un condensado de resultados de las pruebas realizadas e informa del estado de cada uno de los chequeos. Para el entrenamiento, contamos con treinta reportes de salida, de los que se extraerá el archivo `.JSON` que se ha mencionado previamente y se concatenarán en un único DataFrame. A partir de este archivo, se buscará encontrar tendencias concretas que permitan realizar predicciones de estados futuros. El resto de los archivos de salida no se usará, además, por cuestiones relacionadas con la sanitización de datos, pues no todos los archivos pueden limpiarse y anonimizarse correctamente y quedarían fuera de un posible análisis externo a Oracle. Si se carga el archivo en una variable de tipo DataFrame, usando la librería Pandas de Python, se obtendrá una tabla con 25 columnas. El conjunto de datos sanitizados cuenta con 34,484 registros de chequeos.
A continuación, se lista una descripción de las columnas del DataFrame obtenido:

1. `modelVersion`: Describe la versión del software que realiza las pruebas que se está ejecutando en cada chequeo.
2. `modelVersion`: Describe la configuración física del rack donde reside el sistema sobre el que se ejecuta la prueba.
3. `engineeredSystems`: Describe el tipo de contenedor físico donde reside el sistema sobre el que se ejecuta la prueba.
4. `RackIdentifier`: Contiene el identificador del rack donde reside el sistema sobre el que se ejecuta la prueba.
5. `OSVersion`: Describe la versión del sistema operativo del sistema.
6. `DBVersion`: Describe la versión de la base de datos sobre la que se ejecuta la prueba -sólo se presenta en chequeos de la base de datos, lo que excluye otros tipos de chequeos-.

7. `ExadataVersion`: Describe la versión del contenedor físico donde reside el sistema sobre el que se ejecuta la prueba.
8. `exachkExecTimestamp`: Contiene el *timestamp* de inicio de ejecución de cada chequeo, incluyendo fecha, hora en formato hh:mm:ss y zona horaria en formato de tres letras.
9. `exachkID`: Contiene el identificador único del chequeo realizado
10. `exachkType`: Describe el componente sobre el que se ejecuta el chequeo -por ejemplo, sistema operativo, SQL, etc.
11. `exachkName`: Contiene el nombre del chequeo realizado.
12. `exachkMessage`: Contiene una descripción corta del chequeo realizado.
13. `exachkAlertType`: Contiene el estado de falla del chequeo en cuestión, pudiendo ser `FAIL`, `WARNING`, `INFO` o `CRITICAL`, en función del chequeo *per se*.
14. `exachkTargetType`: Describe el componente sobre el que se ejecuta el chequeo, en términos de los componentes del sistema -por ejemplo, host, storage cell, switch, `ASM`, etc.
15. `exachkActualValue`: Expresa el valor real del chequeo.
16. `exachkStatus`: Expresa el estado final del chequeo, pudiendo ser `PASS` o un fallo del chequeo. Si esto último ocurre, el chequeo podrá encontrarse en diversos estados, en función de cada uno, como se mencionó en el punto quince de esta lista.
17. `exachkStatusCode`: Expresa el estado final del chequeo de manera numérica.
18. `exachkReturnCode`: Expresa el valor numérico de retorno del chequeo.
19. `exachkMsgDetail`: Contiene mensajes de información adicional de cada chequeo.
20. `exachkOutfilePath`: Contiene la ruta en la que se almacenó el archivo `.OUT` del chequeo, que contiene los logs obtenidos por el chequeo. Nótese que puede estar en blanco.
21. `NodeName`: Contiene el nombre del nodo sobre el que se ejecutó el chequeo.
22. `exachkTypeInHtml`: Contiene el componente sobre el que se corrió el chequeo.
23. `PreviousStatus`: Expresa el valor obtenido en la corrida previa de ese mismo chequeo (nota: este valor únicamente se obtendrá a partir de corridas programadas mediante el *daemon* del sistema operativo sobre el que se ejecuta el chequeo).
24. `DBName`: Contiene el nombre de la base de datos sobre la que se ejecutó el chequeo (nótese que este campo sólo aplica para los chequeos propios de base de datos).
25. `InstanceName`: Contiene el nombre de la instancia sobre la que se ejecutó el chequeo.

Adicionalmente, conviene echar un vistazo al reporte que entrega la herramienta en formato `.HTML`, que también describe de forma clara, especialmente dirigida a los clientes, algunos puntos importantes:

A partir de la descripción anterior, de conversaciones sostenidas con los desarrolladores de la herramienta y del análisis de lo que cada columna representa, se obtienen los siguientes hallazgos y tareas:



Oracle Exadata Assessment Report

System Health Score is 93 out of 100 (detail)

System Health Score is derived using following formula.

- Every check has 10 bhbpeq
- Critical will deduct 10 bhbpeq
- Failure will deduct 7 bhbpeq
- Warning will deduct 5 bhbpeq
- Info will deduct 3 bhbpeq
- Skip will deduct 3 bhbpeq

Critical checks	0
Failed checks	70
Warning checks	29
Info checks	69
Passed checks	1054
Skipped checks	43
Total checks	1265

..Hide

Cluster Summary

Heading	Description
Cluster Name	Cluster-c1
OS/Kernel Version	LINUX X86-64 OELRHEL 7 4.14.35-2047.505.4.4.el7uek.x86_64
CRS Home - Version	/u01/app/21.0.0.0/grid - 21.3.0.0.0
DB Home - Version - Names	/u01/app/oracle/product/21.0.0.0/dbhome_1 - 21.3.0.0.0 - hczh213 database /u01/app/oracle/product/19.0.0.0/dbhome_1 - 19.12.0.0.0 - vmau61m database /u01/app/oracle/product/18.0.0.0/dbhome_1 - 18.14.0.0.0 - cmkz12d database /u01/app/oracle/product/12.2.0.1/dbhome_1 - 12.2.0.1.210720 - hczh122 database /u01/app/oracle/product/12.1.0.2/dbhome_1 - 12.1.0.2.210720 - 3 databases
Exadata Version	21.2.4.0.0.210909
Number of nodes	8
Database Rlaodxd	2
Storage Rlaodxd	3
IB Switches	3
EXAchk Version	22.2.0_20220707
Collection	exachk_odfq95malqtj01_hnl48b_080322_113355_CRONTAB_COLLECTION
Duration	40 mins, 3 seconds
Executed by	root
Arguments	-hardwaretype X4-2 -clusternodes odfq95malqtj01.tg.qbfulm.xhj,odfq95malqtj02.tg.qbfulm.xhj -cells 192.157.0.47,192.157.0.45,192.157.0.56 -ibswitches oxke28we-yzz0.tg.qbfulm.xhj,oxke28we-hke0.tg.qbfulm.xhj,oxke28we-fka0.tg.qbfulm.xhj -withperfdata -silentforce -showpass -show_critical -tag CRONTAB_COLLECTION -nocleanup
Collection Date	03-Aug-2022 11:37:27

Fig. 2. Reporte de resultados de la herramienta ORAchk en formato HTML. Obtenida de los reportes compartidos de manera confidencial.

– La columna de `exachkID` resulta de vital importancia para filtrar algunos registros que no aportarán nada al análisis. Al observar que la columna `exachkAlertType` debería contener únicamente los potenciales estatus de error de cada chequeo y que en nuestro conjunto de datos existen algunos registros con valores `PASS` para esa categoría, se averiguó que existen chequeos que no deberán ser tomados en cuenta por ser genéricos y estar ligados a la instalación de la herramienta y no propiamente a la salud y funcionamiento de la plataforma de base de datos. Estos chequeos tienen un ID que comienza con el prefijo `GCHECKID`, por lo que para el análisis deberán desecharse del conjunto de datos. Se realizará el filtrado por medio de expresiones regulares, quedando con 33,092 registros útiles.
– Se realizaron gráficos de comparación entre pares de variables para observar la incidencia de errores en relación con las variables que, de acuerdo con los desarrolladores de la herramienta, cuentan con un mayor impacto de desempeño en la variable de salida identificada, en este caso, el `exachkStatus`, que indica si el chequeo fue exitoso o falló: Es importante mencionar que cada

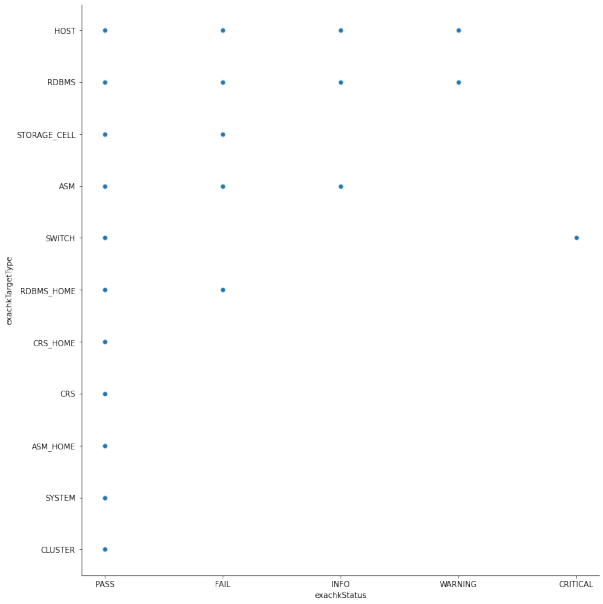


Fig. 3. Relación entre exachkTargetType y exachkStatus. El TargetType describe el componente de la plataforma Exadata sobre el que se ejecuta la comprobación de estado.

aspecto auditado funciona de manera similar a como lo hace un interruptor: cada *check* solamente puede tener dos estados: el de `PASS` y el de `FAIL`. No obstante, la falla puede ser de cinco tipos diferentes: `CRITICAL`, `FAIL`,

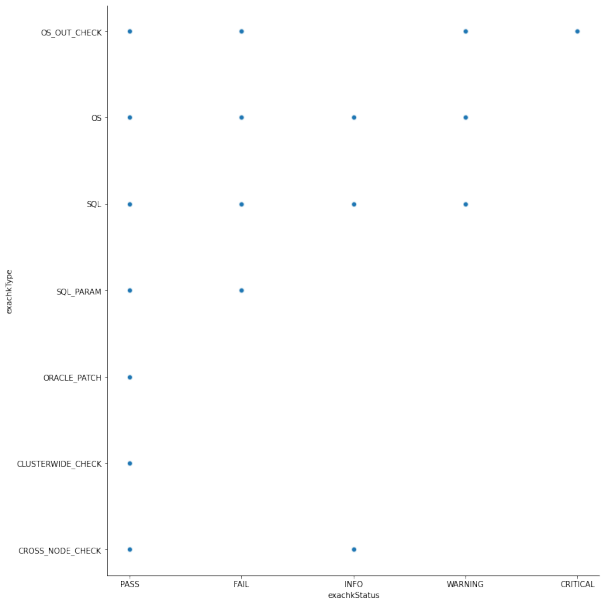


Fig. 4. Relación entre exachkType y exachkStatus. El EXAchk Type describe el tipo de comprobación de estado ejecutada a nivel de software.

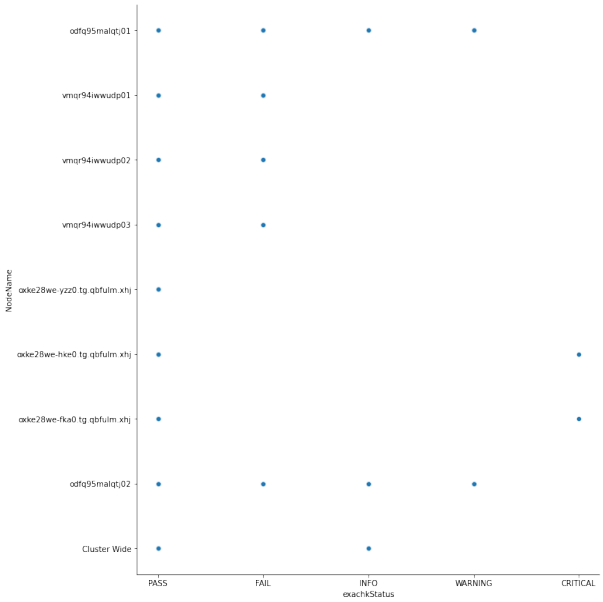


Fig. 5. Relación entre NodeName y exachkStatus. El Node Name describe el nodo sobre el que se ejecutó el chequeo.

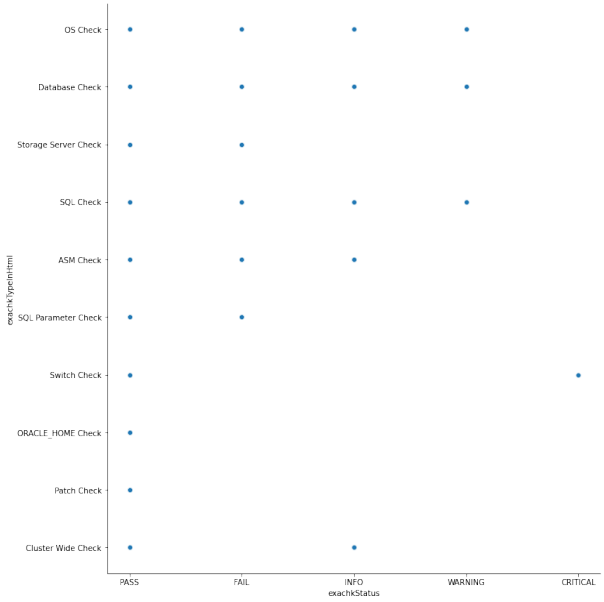


Fig. 6. Relación entre exachkTypeInHTML y exachkStatus. El EXAchk Type in HTML describe el tipo de comprobación de estado ejecutada en general.

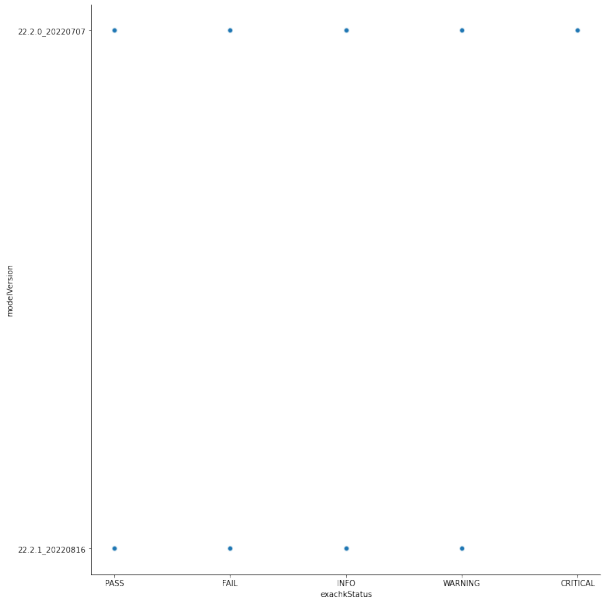


Fig. 7. Relación entre modelVersion y exachkStatus. El Model Version describe la versión de ORAchk ejecutada.

`WARNING`, `INFO` y `SKIP`, que dependerán de cada *check* en particular y cuya ponderación se observa en el reporte generado por la herramienta en `.HTML`:
A partir de los gráficos previos, se puede inferir lo siguiente: El análisis de

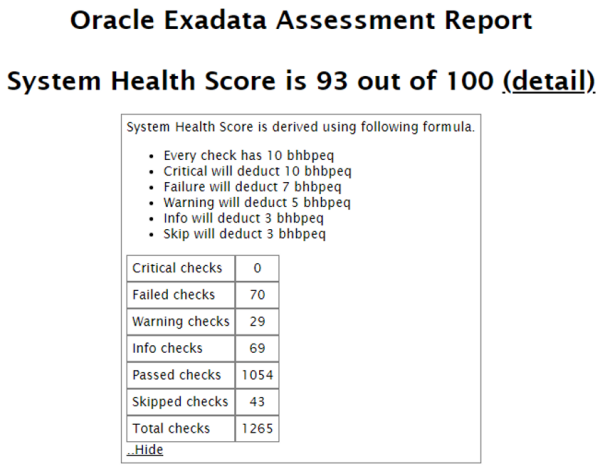


Fig. 8. Deducciones en la puntuación de salud del sistema con base en el tipo de falla generada.

`PASS` y `FAIL` de cada *check* deberá tomarse con dos perspectivas posibles. En la primera, se puede tratar a la variable de salida como una variable de Bernoulli, en la que sólo se cuenta con dos estados posibles. De hacer esto, se pueden considerar todas las fallas como un cero y todos los `PASS` como un uno y realizar modelos en consecuencia. En la segunda perspectiva, se toma en consideración que cada falla pondera de manera distinta en el resultado, con lo que claramente valdría la pena centrar la atención en aquellas fallas que descuentan más puntos, como los `CRITICAL`, que son, por supuesto, los menos deseados en los resultados de ejecución. En esta segunda perspectiva las figuras antes mostradas resultan de utilidad, pues es evidente que hay aspectos específicos en los que se observan *check* con resultado `CRITICAL`. Por ejemplo, en la figura 2 se puede ver que los *check* con esta salida son únicamente aquellos que validan el estado del switch. Esto, de acuerdo con los desarrolladores de la herramienta, es más bien producto de observar la arquitectura de un sistema de Exadata.

En el diagrama, se puede observar que el switch es responsable de administrar la comunicación entre el resto de los componentes del Exadata e, inclusive, de la comunicación entre varios racks. Por ende, puede suponerse que una falla ahí será mucho más importante que una falla en los servidores de almacenamiento o de base de datos, si bien las fallas en estos son también negativas. Un `FAIL` implica una deducción de cinco puntos, por lo que será

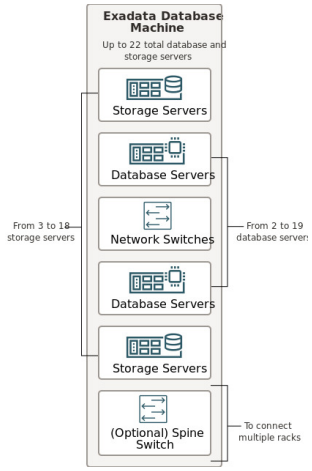


Fig. 9. Arquitectura de un Engineered Systems Exadata. Tomada de Oracle Docs, en https://docs.oracle.com/en/engineered-systems/exadata-database-machine/edbid/ [13]

el siguiente en importancia. La mitad de los componentes: `HOST`, `RDBMS`, `ASM`, `STORAGE_CELL` y `RDBMS_HOME`, tienen fallas de tipo `FAIL`. De ahí existen errores `WARNING` e `INFO`, que deducen cinco y tres puntos, respectivamente. Los conjuntos de datos de entrenamiento y pruebas están libres de incidencias de `SKIP`. No obstante, resulta interesante el hecho de que en `CRS`, `CRS_HOME`, `ASM_HOME`, `SYSTEM` y `CLUSTER` no hay incidencia de errores, los *check* pasan la prueba en su totalidad. Resultados similares se observan con componentes y nombres y versiones de nodos: los fallos críticos son exclusivos de ciertos tipos, hay otros que siempre pasan las pruebas y hay algunos otros que tienen una mescolanza de tipos de fallos y aciertos.

Cualquiera de las perspectivas, tanto la dicotómica como la que contempla una salida múltiple, son dignas de estudio y, con un alcance de proyecto mayor, *a posteriori*, podrán implementarse para observar contrastes de desempeño en los modelos obtenidos. Además, la interpretación de los datos ofrece un panorama de conocimiento mayor de la estructura arquitectónica del sistema analizado y permite hacer hipótesis pequeñas de correlación entre variables, que habrán de comprobarse. Para los propósitos de este documento, se optará primordialmente por tratar a todos los fallos como uno solo, de modo que cada muestra aleatoria -o *check*-, sea en cierta manera similar a un experimento de Bernoulli, si bien se realizarán modelos de contraste para observar si hay mejoras de desempeño.

Es importante remarcar una premisa mencionada en la introducción de este documento de la cual sería útil extraer información, pero para la cual no se cuenta con los datos suficientes: ORAchk® implementa un *daemon* que permite ejecuciones silenciosas y no intrusivas, que el usuario configura man-

ualmente por medio de un *cronjob*. En un escenario realista, lo ideal sería configurar este programa para que se ejecute con cierta periodicidad sobre el producto de Oracle en cuestión. El usuario entonces deja al sistema a cargo de su propia detección de errores, pues el programa implementa también funcionalidades como: envío automatizado de notificaciones y alertas vía correo electrónico y tiempo límite de ejecución, que asegura que el programa no se quedará colgado si encuentra una excepción, sino que se saltará la ejecución del *check*, -dando como resultado un `SKIP`-, lo que, como ya se ha mencionado, restará puntos al estado de salud en general y permitirá la solución de los errores que se encuentren o provoquen un salto de *check*. No obstante, ORAchk® también puede ejecutarse sin el *daemon*, lo que genera resultados que no cuentan con registros de estados previos y tampoco poseen una periodicidad exacta de ejecución. Los registros con los que se cuenta para entrenar y hacer exploración de datos, que, como ya se ha mencionado, están anonimizados y sanitizados y, cabe añadir, son fruto de ejecuciones realizadas sin el *daemon* y en intervalos irregulares, por lo que un análisis sobre tiempo de ejecución y periodicidad es imposible. Asimismo, luego de analizar los datos disponibles, si bien los *timestamps* se encuentran presentes en las ejecuciones, no permiten conocer el tiempo que le toma correr a los *check* con exactitud. Debido a lo anterior, si bien resultaría interesante analizar el fenómeno basándonos en el tiempo para modelarlo de manera estocástica, se carece de los datos necesarios para hacerlo de esa forma y, por ello, se propondrán en su lugar modelos deterministas, como Redes Neuronales Convolucionales.

Tal como se mencionó previamente, se ha de considerar como un factor relevante lo mencionado por los desarrolladores respecto a las variables con mayor influencia en el desempeño general de un sistema de bases de datos. Conocer ese dato y sopesar el contenido del DataFrame en su totalidad permite el discernimiento de aquellas variables que pueden resultar útiles y aquellas que no. En ese sentido, se considera pertinente filtrar el DataFrame para enfocarnos, al menos en esta primera etapa, en analizar la correlación existente entre las variables regresoras más importantes para la compañía y la variable de salida (`exachkStatus`)

Por otro lado, resulta necesario compartir también en este reporte otro de los enfoques con el que se ha intentado analizar tendencias y patrones en los datos. Se trata de un modelo de análisis estadístico automatizado que se ejecuta por medio de una interfaz de línea de comandos, -en adelante `CLI`-. Desde la línea de comandos, se pueden descargar, computar y generar gráficas de contraste con un banco de reportes disponible únicamente de manera interna y reservada -motivo por el cual no se compartirá ni el código implementado ni los datos analizados-, pero cuyos resultados son dignos de mención en el análisis exploratorio de datos de este documento. Bien, ¿qué pasaría si se intentara visualizar el porcentaje de éxito en la ejecución de una comprobación basándonos en el desempeño de varios sistemas individuales? Aquí se muestran algunos ejemplos de las visualizaciones obtenidas

por medio de la herramienta antes descrita:
La figura 10 muestra cuatro de las gráficas que se han generado por medio

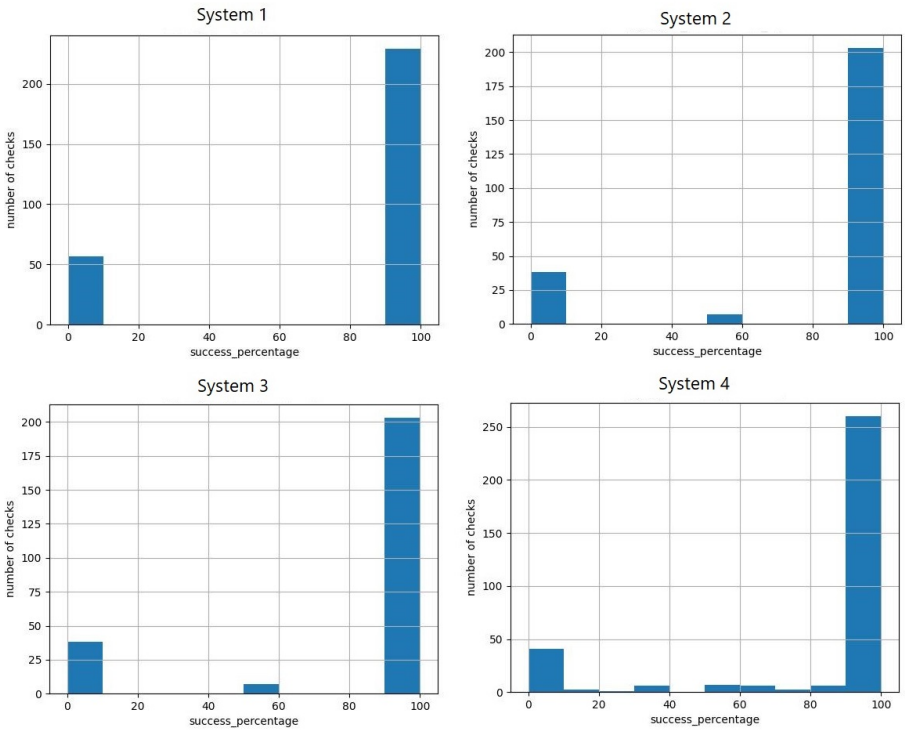


Fig. 10. Visualización del porcentaje de éxito en las ejecuciones de cuatro sistemas de bases de datos. Los nombres de los sistemas están anonimizados. Gráficos de elaboración interna.

de la CLI. En el eje de las x, se encuentra el porcentaje de éxito de los *checks* expresado como porcentaje, pero basado en el estatus: por ejemplo, un `PASS` tendría un porcentaje de éxito equivalente a 100, mientras que uno de `CRITICAL` tendría uno de 0. En el eje de las y se encuentra la cantidad de *checks* encontrados en cada sistema por estatus. A partir de su análisis se hallaron cosas interesantes. Por cuestiones de espacio, no se mostrarán en este documento todas las gráficas generadas, pero todas ellas siguen estos mismos cuatro patrones: existen ejecuciones en los que los *checks* se distribuyen en dos columnas únicamente, con porcentajes de éxito de cien y cero, sin puntos medios, otras comparten el mismo patrón, pero tienen una o más barras intermedias que denotan a cierta cantidad de *checks* cuyo desempeño ronda el 20 y el 80 por ciento (es decir, se trata de *checks* con estatus

de `WARNING`, `INFO` o `FAIL`). La tendencia se inclina hacia este último ejemplo.

4 Implementación de modelos *benchmark* de análisis estadístico: Regresión lineal multivariada

Toda vez que se han analizado los datos disponibles, se mostrará en este apartado del documento la manera en la que se ha seleccionado, implementado y evaluado un modelo *benchmark* de análisis estadístico del problema. Los problemas de clasificación basados en características son muy comunes en el mundo real. Entender los conceptos de causalidad y correlación para quienes se dedican a la minería de datos es esencial, dado que, para construir modelos fiables, se requiere analizar los datos con los que se cuenta, en aras de descubrir si las variables involucradas en un resultado en específico se encuentran tan intrínsecamente relacionadas que, podría decirse, causan un comportamiento, o bien, sólo influyen levemente en que ocurra o la manera en la que lo hace. El proyecto que aquí se está describiendo no ha carecido en absoluto de este tipo de análisis, ya que la pregunta que el equipo de investigadores trata de resolver es, precisamente, ¿cuáles de los datos que recaba la herramienta pueden ser de utilidad para intentar predecir errores? De ser así, ¿cuál es el modelo que lo sustenta? Si bien cabe hacer notar que el proyecto principal se encuentra todavía en una fase exploratoria de los datos, pues comprenderlos no ha sido una tarea sencilla, para efectos de este documento nos centraremos en uno de los archivos `.JSON` de salida, que conjunta, cuando menos, un condensado de los datos que, hasta el momento, se han considerado más relevantes. A partir de ahí, se ha considerado prudente implementar un modelo de regresión multivariada, que brinde una primera aproximación al problema, basada en lo aprendido durante el curso. Se ha elegido este modelo porque los modelos de regresión son capaces de predecir con base en características y son relativamente fáciles de evaluar y de ilustrar en este documento, si bien se prevén resultados no prometedores en vista de que, como ya se verá más adelante, no existen relaciones lineales entre las variables de entrada y la de salida. En el apartado de aprendizaje profundo, se implementará otro modelo, conocido como *K-Nearest Neighbors* para contrastar los resultados obtenidos con Redes Neuronales Convolucionales, que, si bien no se evalúa con el mismo detalle que la regresión multivariada en este documento, sí presenta resultados prometedores. Como ya se ha explicado en párrafos anteriores, los desarrolladores, con base en su experiencia, han indicado cuáles de las variables les parecen más importantes y, por ende, que merecen tener un lugar asegurado en el análisis realizado mediante modelos estadísticos. Por este motivo, la primera fase de la implementación del modelo consistió en limpiar los datos [14] [15] basándonos en tres premisas:

- Incluir los apartados indicados por los desarrolladores: `modelVersion`, `exachkTargetType`, `exachkType` y `NodeName`.
- Eliminar el resto de las columnas que, dada su naturaleza, no aportan mucho a un modelo de regresión, tales como mensajes de salida, variables que

describen cosas similares, logs y claves únicas de identificación, dejando también la columna `exachkTypeInHtml`".
– Eliminar los registros de chequeos que no forman parte de la ejecución, sino de la instalación, cuyo ID comienza con `GCHECKID`.

Asimismo, teniendo en consideración que los datos están desbalanceados, se tratará a la variable de salida como una salida dicotómica, pudiendo solamente indicar un `PASS` (con 1) o un `FAIL` (con 0), con lo que podremos alimentar al algoritmo con más información sobre el contexto de los fallos. En adelante, en caso de tener más datos disponibles, como un trabajo futuro, puede considerarse la necesidad de volver a establecer cinco variables de salida en lugar de dos. Luego de la limpieza de los datos, se prepararán [16] [17] [18] [19] por medio de la función `LabelEncoder` de `sklearn`, con lo que las etiquetas categóricas de todos los datos se convertirán en números, según corresponda. En el caso de la variable de salida, se establece que todo aquel estatus de salida que no sea `PASS`, será considerado `FAIL`. Se verifica que no existan datos faltantes antes de proceder con algunas de las métricas iniciales de evaluación, siendo el caso que no existen datos por imputar. Como puede observarse en las gráficas [20] de la figura 10, los datos, al provenir enteramente de variables categóricas, no presentan, a primera vista, una distribución normal. Esto se evaluará a través de métricas que se citarán posteriormente. Además de analizar la distribución, conviene observar si existe correlación entre las variables, usando tres tipos de métrica: Pearson, Spearman y Kendall. En la figura 12 se observa que, en términos de la de Pearson, la correlación entre las variables regresoras y la de salida es bastante baja, algo esperable en vista de que este tipo de correlación mide directamente relaciones lineales, que no se observan en los datos. Preocupantemente, se puede observar que, de manera contraria a lo que pasa con la variable de salida, entre las variables regresoras, sí existen correlaciones mucho más fuertes, como es el caso de `NodeName` y `exachkType` con `exachkTargetType`. Para intentar paliar esto, podrá elegirse la eliminación de las variables que se encuentren más correlacionadas entre sí y se construirá otro modelo con distinto número de variables regresoras, con el objetivo de verificar cómo se modifican las métricas de desempeño y ver si es posible mejorarlo. No obstante, como ya se mencionó, la correlación de Pearson no es la única que existe. Es por esta razón que se presentan aquí las tablas de valores de correlación de Spearman y Kendall:

Table 1. Correlación de Spearman

Variable regresora	r	$p - val$
modelVersion	-0.001	0.853
exachkTargetType	-0.041	4.587
exachkType	0.07	1.024
NodeName	0.078	2.135
exachkTypeInHtml	0.125	7.785

Table 2. Correlación de Kendall

Variable regresora	r	$p-val$
modelVersion	-0.001	0.853
exachkTargetType	-0.038	4.696
exachkType	0.064	1.254
NodeName	0.073	2.917
exachkTypeInHtml	0.110	6.275

Puede observarse que, aún intentando medir la correlación con otros paradigmas, no puede observarse un aumento significativo en los números obtenidos para el valor de r. Conviene recordar que la correlación de Spearman no es lineal, al menos no necesariamente, por lo que este caso es muy interesante. Se creará un modelo de regresión multivariada y se observará su desempeño. La salida de consola al crear el modelo es la siguiente:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           exachkStatus   R-squared:                       0.031
Model:                          OLS   Adj. R-squared:                  0.031
Method:               Least Squares   F-statistic:                     210.2
Date:              Sat, 19 Nov 2022   Prob (F-statistic):          1.95e-221
Time:                      17:33:35   Log-Likelihood:                -9574.6
No. Observations:             33092   AIC:                         1.916e+04
Df Residuals:                 33086   BIC:                         1.921e+04
Df Model:                         5
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const              1.0020      0.010    101.830      0.000       0.983       1.021
modelVersion      -0.0007      0.005     -0.159      0.874      -0.010       0.008
exachkTargetType  -0.0309      0.001    -23.479      0.000      -0.034      -0.028
exachkType        -0.0254      0.003     -9.603      0.000      -0.031      -0.020
exachkTypeInHtml   0.0326      0.002     20.763      0.000       0.030       0.036
NodeName           0.0025      0.001      1.849      0.064      -0.000       0.005
==============================================================================
Omnibus:                    13144.761   Durbin-Watson:                   0.816
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            39151.803
Skew:                          -2.187   Prob(JB):                         0.00
Kurtosis:                       6.044   Cond. No.                         52.5
==============================================================================
```

Los resultados mostrados en la parte superior dan cuenta de la prueba de bondad de ajuste realizada. Se observa que el valor de F probabilístico es prácticamente cero, con lo que se rechaza la hipótesis nula. Las probabilidades T de las variables regresoras, asimismo, son cero, por lo que es posible aventurar que son

independientes entre sí -excepto una- y que aportan de manera potencialmente significativa al modelo. El valor de r^2 obtenido en el resumen del modelo no es nada significativo, lo que casa con lo antes mencionado sobre la falta de linealidad de los datos, que se puede observar de forma más detallada en la figura 11.
Como se mencionó anteriormente, hay métricas que ayudan a comprobar que

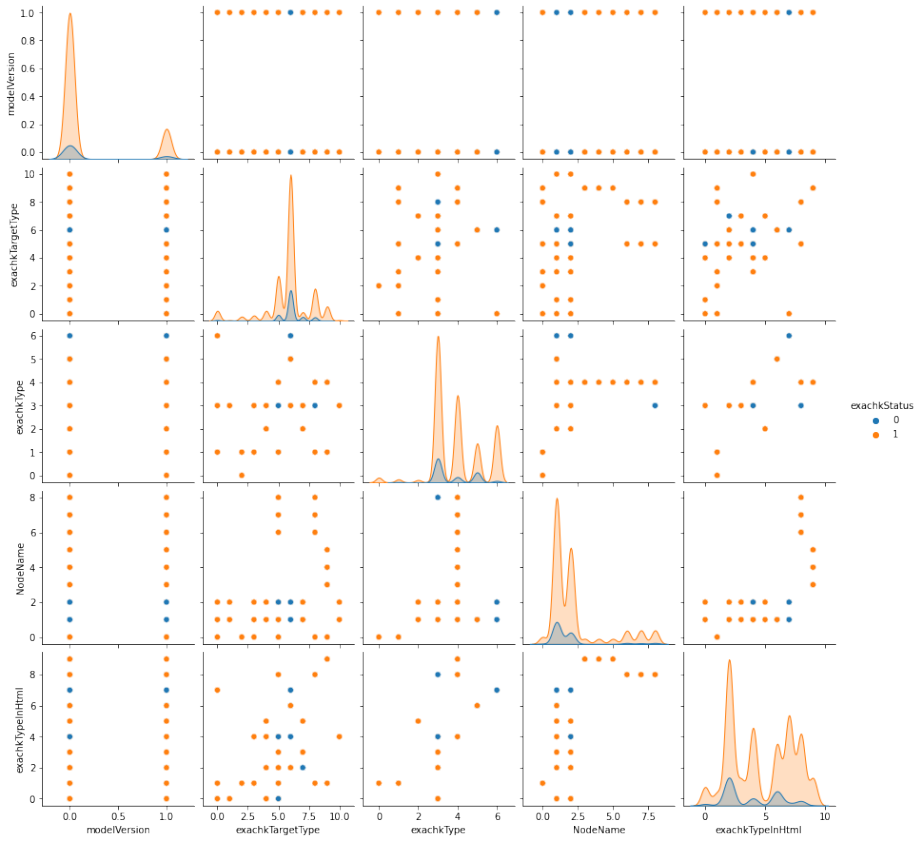


Fig. 11. Gráficas par a par realizadas sobre el conjunto sanitizado de datos. Elaboración propia.

los datos no siguen una distribución normal, -en virtud de que no siempre una primera impresión de los histogramas es prueba de ello-, y ayudan a visualizar mejor su comportamiento. *A priori*, conviene recordar las gráficas presentadas en la fase de exploración de datos de este documento, en el que se muestra que los datos se concentran principalmente en los extremos de las gráficas presentadas. Si se grafica su distribución, se observa lo siguiente: A partir de estas gráficas, puede suponerse, entonces, que los datos sí están distribuidos en dos puntos principales, ubicados en los extremos y no son, de ningún modo, normales. Sin

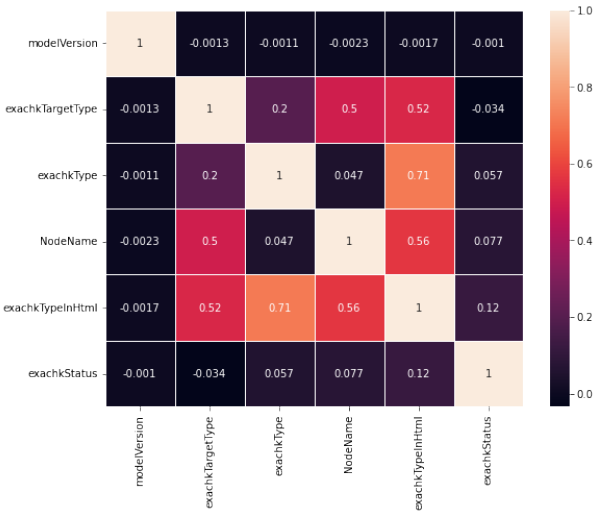


Fig. 12. Mapa de calor que ilustra la correlación de Pearson entre las variables. Elaboración propia.

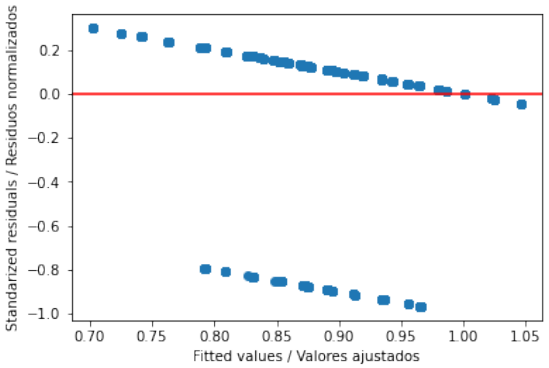


Fig. 13. Gráfico de residuos normalizados vs. valores ajustados. Elaboración propia.

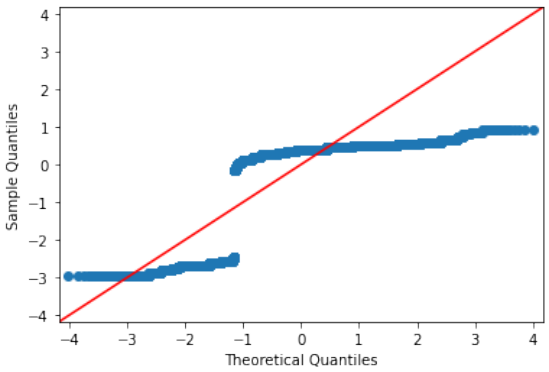


Fig. 14. Gráfica cuantil - cuantil. Elaboración propia.

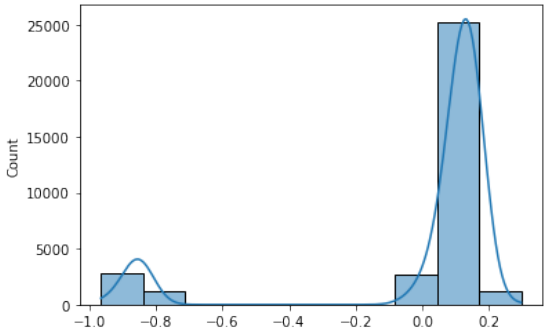


Fig. 15. Histograma de residuos. Elaboración propia.

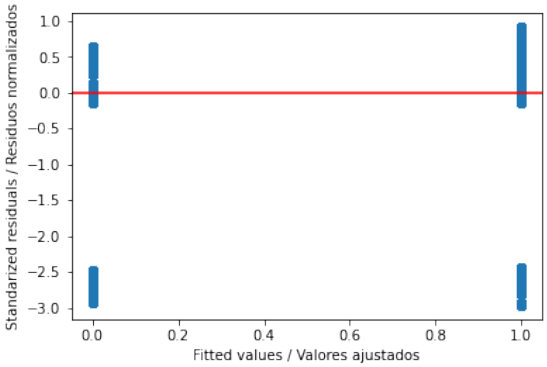


Fig. 16. Gráfica de residuos contra valores ajustados (validación cruzada). Elaboración propia.

embargo, se realizaron también otras pruebas estadísticas para comprobar esto, obteniendo los siguientes valores:

Table 3. Pruebas de normalidad

Prueba	r
Interpretación	
Normalidad de residuos	0.000
Al ser menor a 0.05, indica que los datos no provienen de una distribución normal.	
Heterocedasticidad de los residuos / varianza constante de los residuos	1.282e-84
Al ser menor a 0.05, indica que los residuos no tienen una varianza constante.	

Puede concluirse entonces que el modelo propuesto no tendrá, claramente, un buen desempeño y que lo que se observaba en principio al explorar los datos sí los describe en la realidad. La falta de normalización y de linealidad hace que éste no funcione adecuadamente, algo que también se ve reflejado en el valor de r ajustado obtenido en el resumen del modelo. No obstante, se destaca el hecho de que este análisis también da un precedente sobre que, aunque las pruebas de correlación no muestren que existe una relación de esta índole entre las variables regresoras y la de salida, lo cierto es que numéricamente, los valores de f estadístico parecen mostrar que las variables sí pueden estar relacionadas. Si se construye el modelo sin incluir las variables correlacionadas entre sí -que convergen en `exachkTargetType` y `exachkTargetTypeInHtml`-, se obtiene lo siguiente:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           exachkStatus   R-squared:                       0.009
Model:                          OLS   Adj. R-squared:                  0.009
Method:               Least Squares   F-statistic:                     98.08
Date:              Sun, 27 Nov 2022   Prob (F-statistic):           3.35e-63
Time:                      00:47:37   Log-Likelihood:                -9945.5
No. Observations:             33092   AIC:                         1.990e+04
Df Residuals:                 33088   BIC:                         1.993e+04
Df Model:                         3
Covariance Type:          nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.7940      0.006    125.581      0.000       0.782       0.806
modelVersion  -0.0007      0.005     -0.143      0.886      -0.010       0.009
exachkType     0.0145      0.001      9.813      0.000       0.012       0.017
NodeName       0.0122      0.001     13.588      0.000       0.010       0.014
==============================================================================
Omnibus:                    13679.148   Durbin-Watson:                   0.825
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            42446.933
Skew:                          -2.264   Prob(JB):                         0.00
```

```
Kurtosis:                       6.206   Cond. No.                         17.5
==============================================================================
```

Demostrando que aún pueden realizarse más pruebas en trabajos futuros, quizá incluso con más datos disponibles, para comprobar la correlación real que existe entre las variables, pues el modelo empeora al quitar algunas de las variables regresoras, incluso si éstas están correlacionadas entre sí. Adicionalmente, plantea retos adicionales de balanceo de datos que se discutirán en profundidad en el siguiente capítulo y algunos hallazgos interesantes relacionados con los puntos de influencia, que conducen a la eliminación de datos atípicos y, que, en este ejercicio, también resultan en que el modelo tenga, incluso, un peor desempeño que el original. Cabe recordar en este punto que se ha realizado también un modelo simple de índole *K-Nearest Neighbors*, que basa su funcionamiento en premisas geométricas, obteniendo un puntaje de *accuracy* o precisión de 0.882, medido en función de su desempeño con un conjunto de datos de prueba no sanitizado del mismo tamaño que el que se describió en capítulos previos. Dicho desempeño es visiblemente mejor que el del modelo de regresión, algo probablemente relacionado con la forma en la que se distribuyen los datos al graficarse.

5 Construcción e implementación de modelos de aprendizaje profundo

En el apartado anterior se pudo dar cuenta de la implementación de un modelo de clasificación usando regresión con múltiples variables. Fue evidente que su formulación, si bien sirve como un buen punto de partida para entender el problema y los datos con los que se cuenta, no siempre puede resultar tan eficaz como se desearía. Debido a esto, surge la necesidad de sugerir métodos aún más sofisticados de aprendizaje de máquina, que permitan la formulación de un modelo matemático que resuelva el problema y permita, en este caso concreto, hacer predicciones sobre los estados de error del sistema. Para ilustrar este apartado, se implementó una red neuronal convolucional de entre tres y cuatro capas, incluyendo una o dos ocultas, que permite ajustar una función matemática mediante retropropagación, con resultados mucho mejores a los obtenidos mediante modelos de regresión. De manera distinta a como se abstrajo el modelo estadístico, en esta implementación se realizaron cuatro sub-implementaciones, que contemplan más o menos capas y variables de salida dicotómicas y múltiples, con el objetivo de contrastar el desempeño de todas ellas y decidir cuáles son los parámetros más adecuados para el problema presentado. Dado que este tipo de redes reciben datos en forma de matrices o tensores, se realizó una pequeña implementación de `reshape` mediante la librería `NumPy`, que permitió ingresar los datos dentro de la red. En la construcción de estos modelos, se usó el ya popular y reputado framework `TensorFlow-Keras` [21], que permite la creación de este tipo de redes neuronales mediante Python. Se dividieron los datos en conjuntos de entrenamiento, prueba y validación, en los siguientes porcentajes: 85 para los datos de validación y prueba en x y y y 15 para los datos de entrenamiento

y validación en x y y. Los cuatro modelos que se construyeron en la primera iteración cuentan con los siguientes parámetros:

- Variable de salida dicotómica (`PASS` y `FAIL`), una sola capa Dense de procesamiento con función de activación `relu` y una capa Dense de salida con función de activación sigmoidal. Se compila con una función de pérdida `binary_crossentropy`, optimizador de compilación `Adam` y *accuracy* como métrica principal -entendiéndose ésta como un porcentaje de las observaciones, tanto positivas como negativas, que fueron clasificadas correctamente- . 171 parámetros entrenables, cien épocas y tamaño de `batch` equivalente a 32:
- Variable de salida dicotómica (`PASS` y `FAIL`), dos capas Dense de procesamiento con función de activación `relu` y una capa Dense de salida con función de activación sigmoidal. Se compila con una función de pérdida `binary_crossentropy`, optimizador de compilación `Adam` y *accuracy* como métrica principal. 171 parámetros entrenables, cien épocas y tamaño de `batch` equivalente a 32:
- Variable de salida múltiple (`PASS`, `INFO`, `WARNING`, `CRITICAL` y `FAIL`), una sola capa Dense de procesamiento con función de activación `relu` y una capa Dense de salida con función de activación `softmax`. Se compila con una función de pérdida `categorical_crossentropy`, optimizador de compilación `Adam` y *accuracy* como métrica principal. 61 parámetros entrenables, cien épocas y tamaño de `batch` equivalente a 64:
- Variable de salida múltiple (`PASS`, `INFO`, `WARNING`, `CRITICAL` y `FAIL`), dos capas Dense de procesamiento con función de activación `relu` y una capa Dense de salida con función de activación `softmax`. Se compila con una función de pérdida `categorical_crossentropy`, optimizador de compilación `Adam` y *accuracy* como métrica principal. 61 parámetros entrenables, cien épocas y tamaño de `batch` equivalente a 64:

Luego de tal iteración, se observaron los resultados y se realizaron los siguientes cambios para la siguiente:

- Implementación de *early stopping* para evitar sobre entrenamiento del modelo.
- Eliminación definitiva de capas adicionales, que sólo entorpecen el desempeño.
- Implementación de técnicas de regularización *ridge* y *lasso* para asegurar un buen desempeño a través de las épocas de los conjuntos de validación y prueba.

En la figura 17, se muestra el contraste de pérdida y exactitud de los modelos realizados. A continuación, se enlistan algunos detalles observados que permiten entender mejor qué sucedió, cuál modelo podría funcionar mejor y por qué y, luego, se mencionará cuál podría ser el trabajo futuro para mejorar el modelo:

- En cuestión de modelos convolucionales, lo más complejo no siempre es lo más recomendable. Los gráficos y las matrices de confusión obtenidas muestran que aquellos modelos con más de una capa oculta funcionan mucho peor

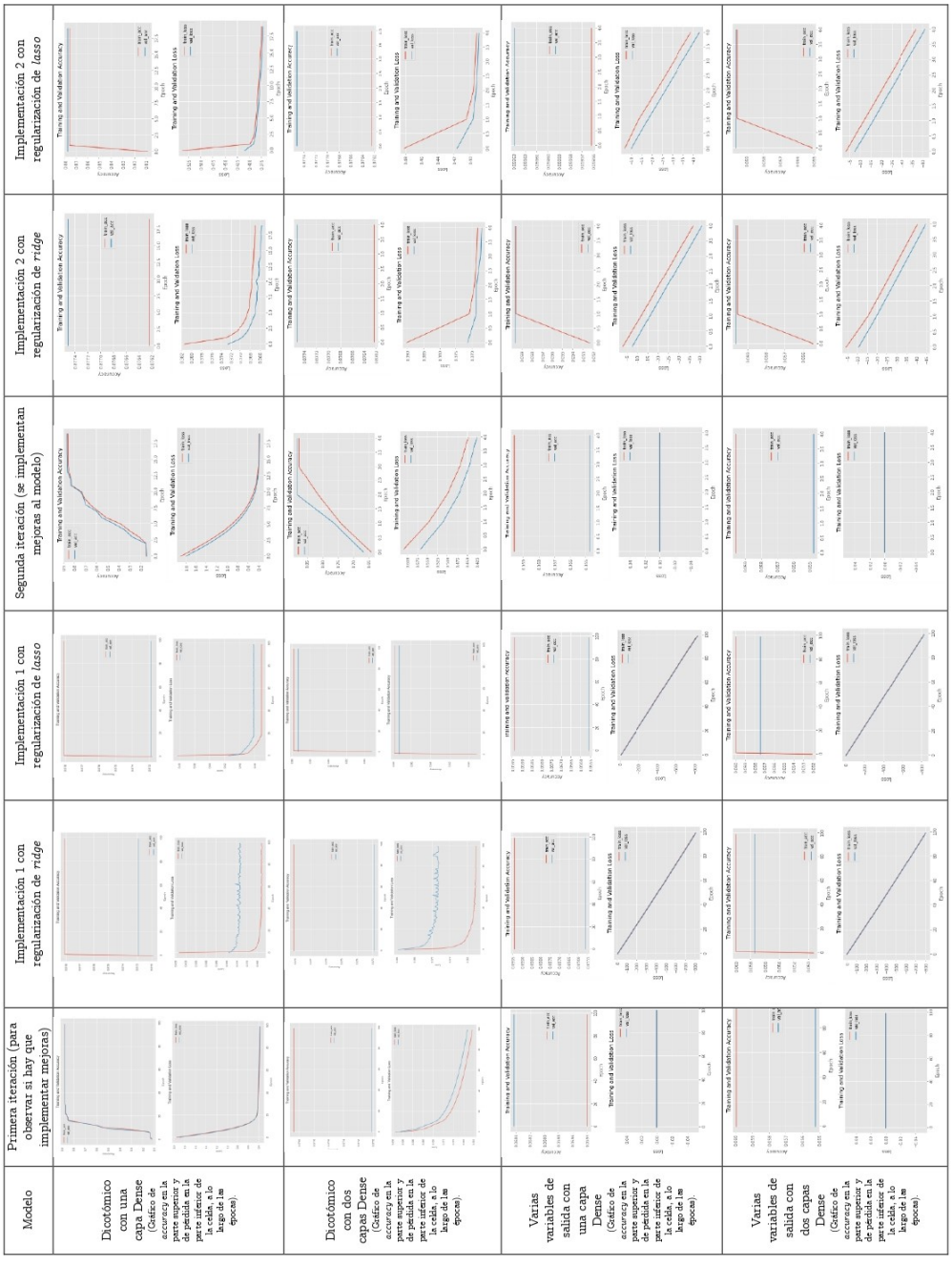


Fig. 17. Tabla de contraste de desempeño de modelos de CNN. Elaboración propia.

que aquellos que sólo incluyen una. Evidentemente, las redes neuronales deberán modelarse en función del problema y los datos con los que se cuenta. Si bien existen modelos sumamente profundos, modelos pre entrenados y otras herramientas más de esta índole, en el caso de nuestros datos parece ser que lo más simple es lo más funcional. En realidad, el desempeño de la red neuronal, tomando en cuenta su exactitud como punto referencial, es bastante aceptable con una sola capa oculta y muy pocas épocas. La técnica de *early stopping* funcionó muy bien para evitar el sobre entrenamiento del modelo y el cómputo innecesario de épocas que, además, no mejoraban nada el desempeño.

- Hablando de los datos que se poseen, es sumamente complejo establecer si el análisis realizado podría ser útil. Si bien la cantidad de registros no es tan pequeña, el desbalanceo de clases puede resultar un problema muy serio. No obstante, aplicar técnicas de *data augmenting* puede ser un arma de doble filo, pues, en realidad, el modelado del fenómeno es bastante acertado. En la realidad, los sistemas de Exadata suelen tener un número muy bajo de errores y un catálogo extensísimo de chequeos con estatus usual de `PASS`, como se ha observado gracias a la herramienta basada en `CLI` de uso interno. Tratar de nivelar los datos es, entonces, un poco incoherente con el fenómeno *per se*, por lo que, en adelante, el trabajo futuro consistirá en recabar más datos y en buscar una técnica que permita mejorar el desempeño del modelo, basándose en el tratamiento de los datos *a priori*.
- Siguiendo el hilo del punto anterior, los modelos dicotómicos se comportan de manera más adecuada y resuelven mejor el problema que aquellos que cuentan con varias salidas, una diferente para cada tipo de error. Tomemos el ejemplo de un `exachkStatus` de `CRITICAL`. Como ya se ha mencionado previamente, sólo los chequeos del switch pueden encontrarse en este estado, un porcentaje muy bajo del total de *check* con los que se cuenta. Al modelar con varias salidas, en casi todas las ocasiones, excepto en un caso, la variable se pierde. Es decir, no es sólo que en la matriz de confusión termine en ceros -se ejemplifican en la parte inferior dos de las matrices de confusión obtenidas, tanto con salidas dicotómicas como múltiples-, sino que la red termina eliminándola del análisis al recomputar sus parámetros mediante retropropagación. Por ende, se considerará mejor el modelo de dos salidas por, además, contar con mejor desempeño.

Matriz de confusión de una salida múltiple.
Se observa que la matriz termina en ceros para precision y recall en varias salidas. Nótese que los dígitos corresponden a la etiqueta de datos para cada clase codificada con el encoder, con dos salidas desaparecidas en la matriz, 0 y 4:

	precision	recall	f1-score	support
1	0.06	1.00	0.11	296
2	0.00	0.00	0.00	181

```
              3       0.00      0.00      0.00      4374

       accuracy                           0.06      4964
      macro avg       0.01      0.20      0.02      4964
   weighted avg       0.00      0.06      0.01      4964
```

Matriz de confusión de una salida dicotómica.
Se observa que la matriz termina en ceros en precision y recall para FAIL.

```
              precision    recall  f1-score   support

        FAIL       0.00      0.00      0.00       631
        PASS       0.87      1.00      0.93      4333

    accuracy                           0.89      4964
   macro avg       0.44      0.50      0.47      4964
weighted avg       0.76      0.87      0.81      4964
```

6 Resultados

Después de la implementación de tres modelos: una regresión de múltiples variables, un *K-Nearest Neighbors* y una red neuronal convolucional, se obtuvieron los siguientes valores para R cuadrada / exactitud en cada uno, en su implementación con mejoras añadidas, cambios de parámetros y demás elementos descritos en párrafos previos y luego de probarlo en un conjunto de datos que toma parte de los datos sanitizados y no sanitizados de uso interno. Se usará este valor como referencia para poder comparar su desempeño, en vista de que son modelos de implementaciones y fundamentos diferentes:

Table 4. Contraste de desempeño de modelos

Variable regresora	**Exactitud de predicción**
Regresión multivariada	0.031
K-Nearest Neighbors con siete vecinos	0.882
Red Neuronal Convolucional CNN (dicotómica, una sola capa oculta)	0.888

Gracias a esta comparación, así como a los resultados del modelo de Red Neuronal Convolucional en términos de precisión y *recall* de precisión, -cuyos valores finales son 0.87 y 1, respectivamente- se deduce que la implementación de CNN es la más apropiada para resolver el problema, pues es capaz de predecir con una exactitud aceptable el resultado. Una de sus fallas mayores está en la

medida de sensitividad (0.13), dado que el desbalanceo de datos provoca una pérdida muy importante en este rubro. No obstante, una de las ventajas que ofrece es la capacidad de aprendizaje y retropropagación, que hará que se pueda continuar optimizando si se añaden datos nuevos al set de entrenamiento.

7 Conclusiones

A manera de conclusión, se resalta la enorme influencia y poder del análisis de tendencias de datos en el rumbo de las grandes compañías. Los intereses del mundo entero están puestas en aquello que los científicos de datos pasan el día completo analizando. Se trata de una labor que requiere paciencia, perspicacia e, incluso, un poco de audacia: atreverse a formular hipótesis, incluso en ocasiones luchando con lo que la propia literatura dice, para poder llegar al resultado esperado. El estado del arte de la minería de datos se transforma, evoluciona constantemente. Del curso académico se aprendió el enorme valor del tiempo: de que no existe un único método para llegar a una solución y de que sólo se requiere tener cierta experiencia para lograr construir modelos cada vez más certeros por medio de cambios en los parámetros que, muchas veces, incluso carecen de fundamento, pero, increíblemente, funcionan. La implementaciones aquí descritas constituyen un fundamento sólido en materia de descripción y entendimiento de los datos y medidas de desempeño y, si bien no resuelven el problema aún, sí pueden aumentar el banco de conocimiento con el que se trabaja para seguir proponiendo modelos.

8 Trabajo futuro

Entre las cuestiones más interesantes de este caso de estudio, se encuentra el hecho de que, en realidad, lo aquí expuesto es sólo parte de una pequeña primera iteración independiente e individual realizada por la autora del documento. El proyecto *per se* tiene aún futuro y posibilidad de ampliar horizontes. La herramienta analizada forma parte de una suite de software líder y, por ende, inmensa, que seguramente esconde aún muchos secretos por descubrir. ¿Qué es la minería de datos sino el mero interés de encontrar joyas en el medio de cantidades ingentes de información? Definitivamente, todo lo que queda aún por descubrir y documentar resultará interesante y, probablemente, ayude a mejorar en demasía lo aquí mostrado. Adicionalmente, los datos siguen produciéndose en algún lugar, el software evoluciona y el proyecto avanza. Muchas de las implementaciones sugeridas sólo requieren de un poco más de datos o de mentes abiertas dispuestas a aportar ideas nuevas y es lo que se espera de esto en los próximos meses y años.

References

1. S. Zhang and M. D. Ernst, “Automated diagnosis of software configuration errors,” in *ICSE 2013, Proceedings of the 35th International Conference on Software Engineering*, San Francisco, CA, USA, May 2013, pp. 312–321.

2. M. Attariyan and J. Flinn, "Using causality to diagnose configuration bugs." 01 2008, pp. 281–286.
3. Oracle. (2016) Real application clusters administration and deployment guide: Oracle rac configuration audit tool. [Online]. Available: https://docs.oracle.com/cd/E11882_01/rac.112/e41960/racchk.htmRACAD8365
4. ——. (2016) Oracle engineered systems. [Online]. Available: https://www.oracle.com/mx/engineered-systems/
5. Y. Kumar, N. Basha, K. M, B. Sharma, and K. Kerekovski, *Oracle High Availability, Disaster Recovery, and Cloud Services: Explore RAC, Data Guard, and Cloud Technology.* Apress, 2019. [Online]. Available: https://books.google.com.mx/books?id=GOWYDwAAQBAJ
6. Oracle. (2016) Oracle exadata. [Online]. Available: https://www.oracle.com/mx/engineered-systems/exadata/
7. ——. (2016) Installation guide for hp-ux itanium: Downloading and installing the orachk health check too. [Online]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/hpdbi/downloading-and-installing-the-orachk-health-check-tool.htmlGUID-19914928-49B2-444A-8F1B-D4398C264AAD
8. ——. (2016) Ejecución de comprobaciones de estado de orachk (cli). [Online]. Available: https://docs.oracle.com/cd/E78252_01/html/E78266/gqteq.html
9. ——. (2016) Database concepts: Introduction to oracle database. [Online]. Available: https://docs.oracle.com/cd/E11882_01/server.112/e40540/intro.htmCNCPT001
10. ——. (2016) Oracle enterprise manager orachk healthchecks plug-in user's guide, 13.2.1.0. [Online]. Available: https://docs.oracle.com/cd/E73210_01/PICHK/PICHK.pdf
11. S. Pane, "Being proactive and keeping your databases healthy," *Database Trends Applications*, vol. 32, no. 4, p. 40, 2018. [Online]. Available: https://link.gale.com/apps/doc/A558368471/AONE?u=googlescholarsid=sitemapxid=0775bd13
12. P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. P. Reinartz, C. Shearer, and R. Wirth, "Crisp-dm 1.0: Step-by-step data mining guide," 2000.
13. Oracle. (2016) Oracle exadata database machine technical architecture. [Online]. Available: https://docs.oracle.com/en/engineered-systems/exadata-database-machine/edbid/
14. C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2
15. T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134
16. P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

17. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
18. S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.
19. M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. [Online]. Available: https://doi.org/10.21105/joss.03021
20. J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
21. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16. USA: USENIX Association, 2016, p. 265–283.