Identificación de las crisis en el sistema Zújar de la subdirección de análisis de información e investigación del fraude de la AEAT

RUBÉN DE LA TORRE MADRID
Grado de Ingeniería Informática

Humberto Andrés Sanz

10 de junio de 2022

Agradecimientos

En primer lugar, quiero agradecer a Humberto Andrés, consultor de este TFG, sus consejos para poder realizar con "éxito" este trabajo de fin de Grado.

Por otro lado, quiero agradecer el apoyo, ayuda y consejo de mis muy estimados compañeros de la AEAT: Carlos Abad, Vanesa Abadías, Maite Luque, Javier López y Miguel Terrero.

Por último, quiero agradecer a Lucía (el amor de mi vida) todo el apoyo que me ha brindado para superar este reto que tanto esfuerzo me ha supuesto. Y a nuestro hijo Gael (de un año recién cumplido), aunque no ha sido el que más ha ayudado, también se lo dedico a él ;-)

FICHA DEL TRABAJO FINAL

Título del trabajo:	Identificación de las crisis en el sistema Zújar de la subdirección de análisis de información e investigación del fraude de la AEAT
Nombre del autor:	**Rubén de la Torre Madrid**
Nombre del consultor:	**Humberto Andrés Sanz**
Fecha de entrega:	06/2022
Área del Trabajo Final:	**Business Intelligence**
Titulación:	*Grado de Ingeniería Informática*

Resumen del Trabajo (máximo 250 palabras):

Proyecto de aprendizaje computacional (o *machine learning*) destinado a identificar las crisis en el sistema Zújar del departamento de TAIIF de la AEAT a partir de los registros de actividad generados por las aplicaciones que consumen su información.

En el proyecto se valora el uso de los principales modelos de clasificación, con la intención de seleccionar aquel, o aquellos, modelos que obtengan las mejores métricas la clasificar registros como momentos de crisis o como momentos no de crisis.

Durante el proyecto se valora principalmente la métrica F1, pues esta pondera a su vez las métricas precisión (*precision*) y exhaustividad (*recall*) que son las más interesantes para el problema de identificar la mayor cantidad de crisis. La precisión mide el porcentaje de positivos identificados y la exhaustividad el porcentaje de positivos reales entre los registros clasificados como tal.

Asimismo, se hace uso de diversas técnicas con el objetivo de mejorar los pobres resultados obtenidos durante las primeras etapas de la fases modelado. Con estas técnicas se pretende paliar una serie de problemas identificados, principalmente la falta de balanceo entre los casos positivos y

negativos.

Como resultado del proyecto, además de esta memoria, se entrega una librería desarrollada en Python (debidamente documentada) preparada para poder evaluar los distintos modelos utilizados (utilidades desarrolladas), así como una guía de uso necesaria para poder utilizar adecuadamente dicha librería.

Abstract (in English, 250 words or less):

Machine learning project aimed at identifying crises in the Zújar system of the TAIIF department of the AEAT from the activity records generated by the applications that consume its information.

The project evaluates the use of the main classification models, with the intention of selecting the model or models that obtain the best metrics when classifying records as crisis or non-crisis moments.

During the project, the F1 metric is mainly evaluated, since it, in turn, weighs the precision and recall metrics, which are the most interesting for the problem of identifying the greatest number of crises. Precision measures the percentage of positives identified, and recall measures the percentage of real positives among the records classified as such.

In addition, several techniques are used to improve the poor results obtained during the first stages of the modeling phase. These techniques are intended to alleviate several identified problems, mainly the lack of balance between positive and negative cases.

As a result of the project, in addition to this report, a library developed in Python (duly documented) prepared to evaluate the different models used (developed utilities) is delivered, as well as a user's guide necessary for the proper use of this library.

Palabras clave (entre 4 y 8):

Datos, Aprendizaje computacional, modelos clasificación, árboles decisión

Índice

Lista de figuras

1. Introducción

Zújar es un sistema de análisis de datos de la subdirección TAIIF (Tecnologías de Análisis de la Información e Investigación del Fraude) del DIT (Departamento de Informática Tributaria) de la AEAT (Agencia Estatal de Administración Tributaria) compuesto por varias capas de software. Entre estas están las aplicaciones cliente, las cuales proveen a los usuarios los interfaces para realizar los trabajos de análisis de la información en los distintos departamentos, como por ejemplo inspección, recaudación, aduanas, etc.

Debido al uso cada vez más extensivo de estas aplicaciones cliente, y a unos perfiles de uso altamente variables, es frecuente las aparición problemas en el servicio, causando el mal funcionamiento de las aplicaciones anteriores y llegando a provocar saturación en los servidores derivando en crisis del servicio. Estas crisis provocan problemas en el comportamiento normal del sistema, que pueden derivar en rechazos de peticiones para intentar asegurar la calidad del servicio, sobrecarga de las aplicaciones que forman la capa de datos e incluso llegar a causar la caída de alguno de los servidores que dan soporte a estas.

1.1 Contexto y justificación del Trabajo

El sistema Zújar se ha convertido, en los últimos años, en una parte fundamental de las herramientas de análisis de la Agencia Tributaria. Por otro lado, desde la aparición de la aplicación Genio en 2013, el modo de explotar la información del sistema ha variado completamente, permitiendo ser consumida (además de por los propios usuarios) por distintos servicios y de formas diversas.

A pesar de que este cambio en el uso del sistema es evidentemente positivo, esto ha provocado que la carga del sistema sea altamente variable, llegando a producirse picos de uso muy pronunciados, en los que pueden sufrirse problemas en el suministro de servicio y por otro lado momentos de valle en los que el sistema en su conjunto está prácticamente ocioso.

Un problema derivado de esta variabilidad en los perfiles de uso, es la dificultad para localizar el problema (o problemas) concreto que origina las crisis en los sistemas, ya que en muchas ocasiones este no es evidente: Por ejemplo, puede estar provocado porque que el número de peticiones sea especialmente elevado o que alguna de las capas esté saturada, pero con el volumen de uso habitual es realmente difícil concretar dónde está el problema.

En la siguiente figura, se presenta la arquitectura actual del sistema Zújar, en la que se aprecia variedad de los clientes del Motor, cada uno de los cuales puede lanzar consultas arbitrariamente complejas de forma simultánea, lo que dificulta localizar la fuente de las crisis.

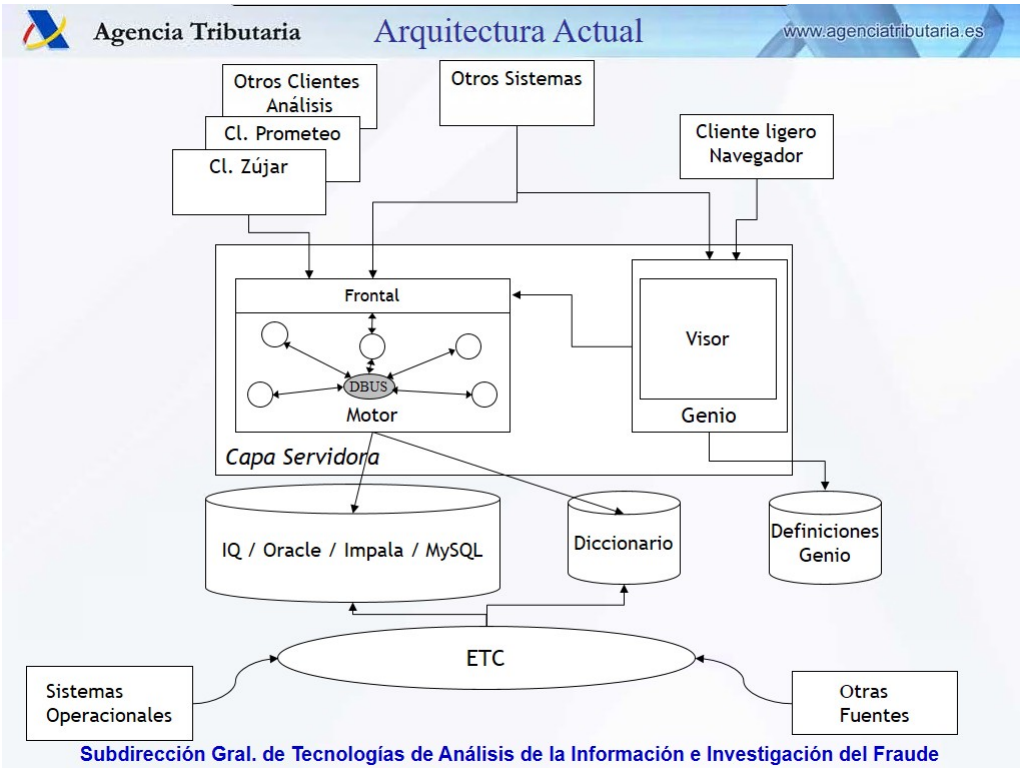


Figura 1: Arquitectura sistema Zújar

La lista de los clientes es:

- ClienteZújar: especializado en el análisis multidimensional de la información.
- Prometeo: especializado en el análisis contable de los datos suministrados por los contribuyentes en el curso de las actuaciones inspectoras.
- Teseo: especializado en el análisis de las relaciones entre los objetos de cuya información se dispone
- Dédalo: dedicado a la localización de contribuyentes no identificados.
- Genio: especializado en el diseño y ejecución de informes complejos.
- Aplicaciones operativas, que consultan a Zújar para incorporar la información elaborada por el sistema en sus transacciones.

Actualmente, la operativa para intentar encontrar las causas que provocan estas crisis y así poder resolverlas no está organizada. Las operativas son normalmente son iniciadas los responsables de sistemas, que son quienes observan las alertas enviadas por las herramientas de

monitorización. A su vez, estos se comunican con los distintos grupos en función de la naturaleza del problema, los cuales investigan de manera individual, el origen del problema. De esta manera, la información fluye en conversaciones individuales entre el grupo de sistemas y el resto de equipos tratando de resolver la crisis, pero de una manera improvisada.

1.2 Objetivos del Trabajo

Como resultado de este proyecto, se pretende disponer de una herramienta que permita identificar y analizar las crisis, de manera que se pueda conocer de manera automática el origen del problema y que de esta manera se pueda actuar con el foco puesto en la causa del problema. A continuación se detallan los objetivos de manera individualizada:

1.2.1 Identificación de crisis bajo enfoque forense

El objetivo de este trabajo de fin de grado es desarrollar un software capaz de identificar situaciones de crisis a partir del análisis de los datos de actividad que actualmente ya registran las aplicaciones que forman el núcleo del sistema Zújar, que son: Motor de datos (MotorZújar), ClienteZújar y Genio.

Para hacerlo, se analizará el ajuste de varios algoritmos de aprendizaje computacional o *machine learning*, con el objetivo de identificar las crisis producidas en un tramo de datos de este registro de actividad, a partir del aprendizaje previo realizado utilizando los valores históricos almacenados.

1.2.2 Identificar las premisas que determinan las crisis

Como objetivo secundario se pretende que, además de identificar las crisis, el sistema sea capaz de ofrecer las pautas en las que se ha basado para etiquetar como momento de crisis o como momento de no crisis unos datos determinados. Con ello se pretende ayudar en la identificación del origen de los problemas experimentados. Por ejemplo, el algoritmo podría indicar que se ha recibido un porcentaje elevado de peticiones de un cierto tipo superando cierto umbral.

1.2.3 Identificación de crisis bajo enfoque *on-line*

Por último, y en función de la viabilidad de realizarse sin perjuicio del funcionamiento del sistema, se contempla un último objetivo que consiste en que el algoritmo pueda ser alimentado periódicamente con datos *on-line* con el fin de detectar las crisis en un momento prematuro el cual permita a los administradores de estos sistemas y aplicaciones, emprender acciones para evitar que estas crisis terminen de producirse.

1.3 Enfoque y método seguido

Dada la complejidad del proyecto, este TFG se contempla como un piloto que trace la línea a seguir, de cara a conseguir los objetivos de poder identificar y analizar cualquier tipo de crisis que se pueda producir en el entorno Zújar. Por tanto, de cara a este proyecto se realizarán las siguientes simplificaciones:

1. Se considerarán únicamente los datos de actividad comprendidos entre las 7 horas de la mañana y las 15 horas de la tarde, puesto que en otras franjas horarias, el uso de los sistemas es altamente variable y sujeto a numerosos cortes controlados de servicio, por ejemplo, por la realización de pases de versiones de *software* al entorno de producción.
2. A pesar de disponer de más de 1000 tipos de alertas distintas del sistema (Zabbix) se considerarán solo aquellas relevantes para el funcionamiento de la aplicación Genio, entre las que destacan las siguientes:
    - *Ping* a la aplicación Genio supera el tiempo considerado
    - *Ping* a Tomcat supera el tiempo considerado
    - Tomcat fuera de servicio
3. Las aplicaciones involucradas en este análisis escriben su actividad de manera *on-line* en tablas transaccionales en tecnología MySQL y posteriormente estos datos son volcados, con una periodicidad diaria en tablas para el análisis en tecnología SybaseIQ. El *software* desarrollado en este TFC tendrá una aplicación forense, de manera que solo consultará datos consolidados en las tablas de análisis.
    - Sin embargo, se desarrollará para ser compatible con las tablas de datos transaccionales y poder ser usado en el futuro en tiempo real para identificar y analizar crisis que se estén produciendo en el momento actual.
    - Para ello, sería necesario analizar y posiblemente resolver, los problemas que puedan surgir, derivados de hacer lecturas de datos masivas para realizar los análisis.

En cuanto a la metodología utilizada, al tratarse de un proyecto de minería de datos, se ha empleado una metodología específica para este tipo de proyectos como es CRISP-DM (*Cross Industry Standard Process for Data Mining*)[1], en la que se identifican las siguientes etapas:

1. Entendimiento del negocio. Esta fase está destinada a recabar información sobre el cliente y sus necesidades, para resolver las dudas que puedan surgir y finalmente entender el alcance del proyecto. Para el desarrollo de esta fase, se parte con la ventaja de formar parte del equipo de desarrollo de la AEAT desde hace más de diez años.
2. Entendimiento de los datos. En esta etapa se debe realizar un análisis detallado de los datos que se desean explotar para, apoyados en el entendimiento recabado en la etapa anterior, ser

capaces de determinar la viabilidad del proyecto. Al igual que en la etapa anterior, para la elaboración de esta etapa se parte con la ventaja del conocimiento previo adquirido de los datos.

3. Preparación de los datos. Durante esta etapa, se realizan aquellos trabajos destinados a la preparación del conjunto de todos los datos necesarios para poder realizar el modelado de los mismos en la siguiente fase. Esta fase es crucial en el proyecto y el reto principal consiste en definir que es una crisis del sistema e identificarlas en los datos existentes.
4. Modelado. En esta fase, se aplican los análisis y modelos pertinentes sobre los datos preparados anteriormente con la consiguiente obtención de los primeros resultados, los cuales serán evaluados en la siguiente fase.
5. Evaluación. Durante esta etapa se comprueban los resultados obtenidos durante las ejecuciones de los modelos realizados en la etapa anterior. Las iteraciones entre esta fase y la anterior son frecuentes, para ir midiendo la calidad de los resultados en función de los cambios realizados en el modelado.
6. Despliegue. Una vez afinados los modelos con los cuales se obtienen unos resultados con la calidad esperada se procede a poner en marcha estos algoritmos en el entorno de producción del cliente. Sin embargo, es este caso, al tratarse de un proyecto con carácter formativo, esta fase será destinada a elaborar la memoria que recoja los resultados del análisis.

En la siguiente imagen, se muestra como se relacionan las seis etapas anteriores:

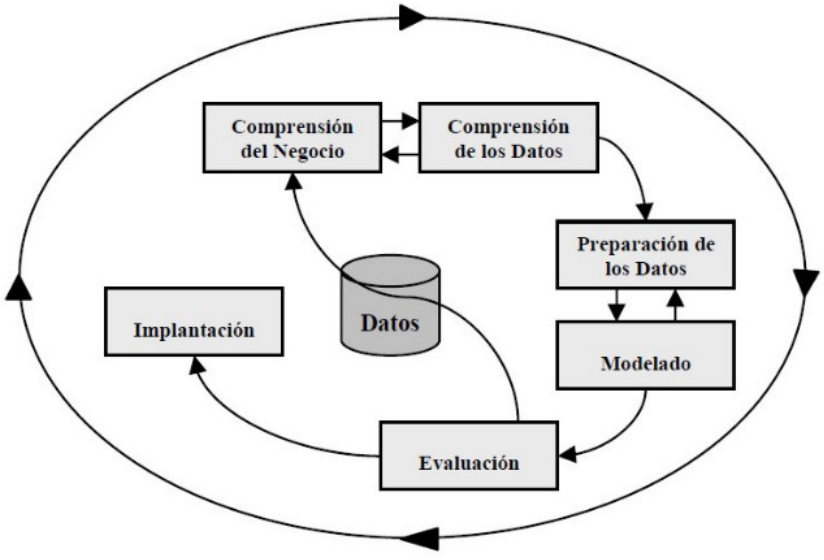


Figura 2: Modelo de proceso CRISP–DM ([CRISP-DM, 2000]).

A continuación se realizan algunas observaciones importantes en relación a la metodología del proyecto:

1. La planificación de este TFG, la cual se detalla en el apartado posterior, y en concreto el Gantt del proyecto, han sido elaborados guiados por del uso de esta metodología.
2. Como se mencionaba anteriormente, el despliegue de la solución no será contemplado como parte de este proyecto. De esta manera, el proyecto se considera concluido con la presentación de los modelos y resultados obtenidos al analizar nuevos datos, sin necesidad de integrarlos en una aplicación que lo explote.
3. Por otro lado, aunque esta metodología tiene un carácter claramente cíclico, al disponer de plazos cerrados para la entrega del proyecto, se eliminarán en la medida de lo posible los aspectos con más incertidumbre para evitar tener que volver a fases previas y poder cumplir así con los plazos establecidos.
4. Por último, indicar que a pesar de que las seis fases están marcadas temporalmente en el Gantt, la ejecución de tareas de una y otra fase no tendrán un orden estricto, así por ejemplo, durante la fase de entendimiento del cliente, se estarán realizando también aproximaciones al entendimiento de los datos y viceversa. Algo similar ocurrirá durante la ejecución de las fase 3 y 4 y 5 y 6, dando de esta manera el carácter cíclico propuesto por la metodología CRISP-DM.

1.4 Planificación del Trabajo

1.4.1 Recursos necesarios

La dedicación ha sido 4 horas diarias de media (incluyendo fines de semana y festivos), en las que se ha ejercido de manera alternativa los roles de Jefe de Proyecto, Analista Programador y Analista de Datos. El número de días del proyecto es de 95 días lo que suponen un total de 380 horas de dedicación.

En la siguiente tabla se realiza una estimación[1] del coste de proyecto en función de las horas empleadas de cada uno de los roles anteriores:

	Horas	**Coste/Hora €**	**Total €**
Jefe de Proyecto	95	70	6.650 €
Analista Programador	70	45	3.150 €
Analista de Datos	215	52	11.180 €
	380		**20.980 €**

1.4.2 Actividades y tareas del proyecto

Se detallan en la siguiente tabla las actividades y tareas del proyecto, así como las fechas en las que están planificadas realizarse y sus duraciones:

1 Los costes de precio por hora considerados en la tabla siguiente son orientativos según el precio actual del mercado.

Fecha de inicio	Fecha finalización	Tarea	Jornadas
16/02/22	19/02/22	Elaboración de propuestas	4
16/20/22	18/02/22	Reuniones con AEAT para identificar posibles temas	3
19/02/22	19/02/22	Documentación propuestas	1
21/02/22	06/03/22	PEC1 - Plan de trabajo	14
21/02/22	25/02/22	Selección tema del proyecto	5
26/02/22	01/03/22	Planificación	4
02/03/22	06/03/22	Documentación PEC1	5
08/03/22	17/04/22	PEC2 - Análisis y preparación datos	41
08/03/22	12/03/22	*Feedback* PEC1	5
08/03/22	13/03/22	01. Entendimiento del negocio	6
08/03/22	10/03/22	Definición de requisitos y objetivos del proyecto	3
11/03/22	13/03/22	Diseño plan preliminar	3
14/03/22	28/03/22	02. Entendimiento de los datos	15
14/03/22	15/03/22	Estudio de algoritmos basados en series temporales	2
16/03/22	17/03/22	Investigación de estado del arte en algoritmos predictivos	2
18/03/22	20/03/22	Selección de tablas y atributos	3
21/03/22	25/03/22	Análisis de componentes principales	5
26/03/22	28/03/22	Identificación de crisis: casos positivos	3
29/03/22	17/04/22	03. Preparación de datos	20
29/03/22	31/03/22	Instalación y preparación del entorno	3
01/04/22	05/04/22	Selección, transformación y limpieza de datos	5
06/04/22	09/04/22	Selección y marcaje de positivos (casos de crisis)	4
10/04/22	12/04/22	Desarrollo ETL	3
13/04/22	17/04/22	Documentación PEC2 (fase 1 memoria)	5
19/04/22	23/05/22	PEC3 - Implementación	35
19/04/22	23/04/22	*Feedback* PEC2	5
19/04/22	08/05/22	04. Modelado	20
19/04/22	08/05/22	Implementación con algoritmos identificados	20
09/05/22	23/05/22	05. Evaluación	15
09/05/22	18/05/22	Pruebas y comparativa de resultados	10
19/05/22	23/05/22	Documentación PEC3 (fase 2 memoria)	5
24/05/22	09/06/22	Entrega final	17
24/05/22	28/05/22	*Feedback* PEC3	5
24/05/22	02/06/22	Redacción memoria (fase 3 memoria)	10
24/05/22	26/05/22	Preparación de la presentación para la defensa	3

En la tabla anterior se subrayan los bloques en los que se agrupan las tareas (según los plazos de entrega del TFG y sus PEC), y las fases correspondientes a la metodología CRISP-DM (detallada en el apartado 1.3).

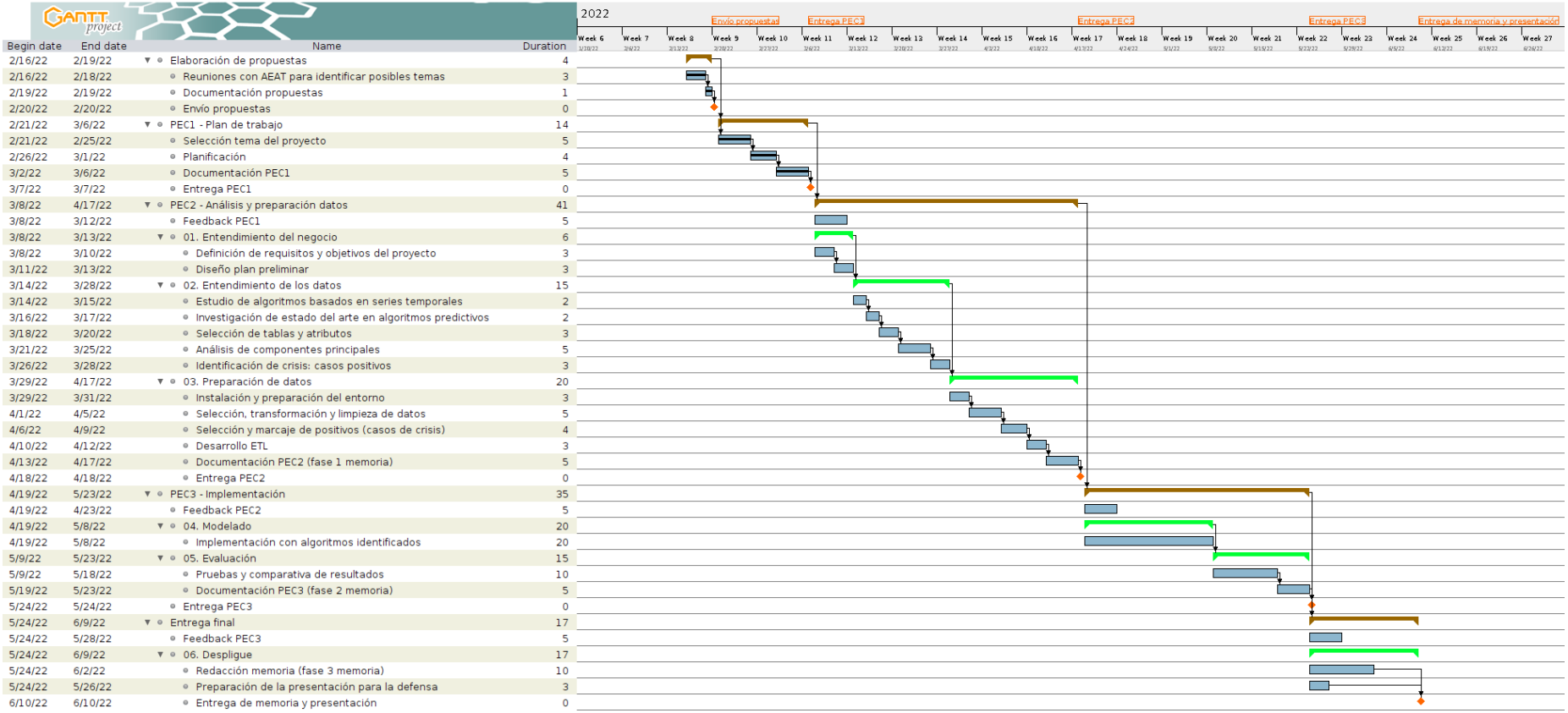
1.4.3 Hitos del proyecto

Se detallan en la siguiente tabla los hitos del proyecto

Fecha	Hito
20/92/22	Envío propuestas
07/03/22	Entrega PEC1
18/04/22	Entrega PEC2
24/05/22	Entrega PEC3
10/06/22	Entrega de memoria y presentación

1.4.4 Diagrama de Gantt

Se presenta a continuación el Gantt del proyecto



Begin date	End date	Name	Duration
2/16/22	2/19/22	Elaboración de propuestas	4
2/16/22	2/18/22	Reuniones con AEAT para identificar posibles temas	3
2/19/22	2/19/22	Documentación propuestas	1
2/20/22	2/20/22	Envío propuestas	0
2/21/22	3/6/22	PEC1 - Plan de trabajo	14
2/21/22	2/25/22	Selección tema del proyecto	5
2/26/22	3/1/22	Planificación	4
3/2/22	3/6/22	Documentación PEC1	5
3/7/22	3/7/22	Entrega PEC1	0
3/8/22	4/17/22	PEC2 - Análisis y preparación datos	41
3/8/22	3/12/22	Feedback PEC1	5
3/8/22	3/13/22	01. Entendimiento del negocio	6
3/8/22	3/10/22	Definición de requisitos y objetivos del proyecto	3
3/11/22	3/13/22	Diseño plan preliminar	3
3/14/22	3/28/22	02. Entendimiento de los datos	15
3/14/22	3/15/22	Estudio de algoritmos basados en series temporales	2
3/16/22	3/17/22	Investigación de estado del arte en algoritmos predictivos	2
3/18/22	3/20/22	Selección de tablas y atributos	3
3/21/22	3/25/22	Análisis de componentes principales	5
3/26/22	3/28/22	Identificación de crisis: casos positivos	3
3/29/22	4/17/22	03. Preparación de datos	20
3/29/22	3/31/22	Instalación y preparación del entorno	3
4/1/22	4/5/22	Selección, transformación y limpieza de datos	5
4/6/22	4/9/22	Selección y marcaje de positivos (casos de crisis)	4
4/10/22	4/12/22	Desarrollo ETL	3
4/13/22	4/17/22	Documentación PEC2 (fase 1 memoria)	5
4/18/22	4/18/22	Entrega PEC2	0
4/19/22	5/23/22	PEC3 - Implementación	35
4/19/22	4/23/22	Feedback PEC2	5
4/19/22	5/8/22	04. Modelado	20
4/19/22	5/8/22	Implementación con algoritmos identificados	20
5/9/22	5/23/22	05. Evaluación	15
5/9/22	5/18/22	Pruebas y comparativa de resultados	10
5/19/22	5/23/22	Documentación PEC3 (fase 2 memoria)	5
5/24/22	5/24/22	Entrega PEC3	0
5/24/22	6/9/22	Entrega final	17
5/24/22	5/28/22	Feedback PEC3	5
5/24/22	6/9/22	06. Despligue	17
5/24/22	6/2/22	Redacción memoria (fase 3 memoria)	10
5/24/22	5/26/22	Preparación de la presentación para la defensa	3
6/10/22	6/10/22	Entrega de memoria y presentación	0

Figura 3: Diagrama de Gantt del proyecto

- Los bloques de tareas agrupados en función de las PEC en marrón.
- Los hitos del proyecto, coincidentes con las finalizaciones de los bloques anteriores en rojo.
- Las fases del proyecto según la metodología CRISP-DM en verde.

1.5 Breve sumario de productos obtenidos

Los productos obtenidos en este proyecto, además de los propios requeridos por la elaboración y presentación del TFG, son los siguientes:

- Librería desarrollada en Python con los métodos necesarios (debidamente comentados) para poder obtener los datos objetos de análisis y para posteriormente poder evaluar los distintos modelos.
- Guía de uso de la librería anterior.
- Fichero CSV con el histórico de los resultados obtenidos con los métodos contemplados en el estudio a lo largo del trabajo.

1.6 Breve descripción de los otros capítulos de la memoria

La memoria está estructurada en los siguientes capítulos, los cuales corresponden a las fases del proyecto según la metodología empleada:

- Capítulo 2. Entendimiento del negocio. Se trazan los objetivos del proyecto y el diseño de la solución. También se presenta el plan del proyecto.
- Capítulo 3. Entendimiento de los datos. Se revisan los datos disponibles y la calidad de los mismos para obtener la información necesaria para realizar el modelado.
- Capítulo 4. Preparación de los datos. Se detallan los trabajos necesarios para la obtención de los datos.
- Capítulo 5. Modelado. Se describe el proceso de modelado para los algoritmos de clasificación contemplados en el estudio.
- Capítulo 6. Evaluación. Se revisan los resultados obtenidos por los modelos anteriores y se detallan las técnicas aplicadas para mejorar los resultados.
- Capítulo 7. Despliegue. Se detallan los métodos desarrollados de la librería entregada como parte del trabajo y se elabora la guía de uso de la misma.

2. Entendimiento del negocio

Durante la fase inicial del proyecto se persigue conocer las características del negocio, así como las necesidades del cliente. Por ese motivo, se consolida la información que se detalla en los siguientes subapartados, como resultado de varias reuniones mantenidas con el personal de la AEAT.

2.1. Objetivos

Se confirman los siguientes objetivos del proyecto:

1. Clasificar los momentos como crisis o no crisis. El sistema de análisis desarrollado debe ser capaz de identificar (para un intervalo de tiempo indicado, por ejemplo un día) en que momentos el sistema estuvo en crisis (clasificados como positivos) o en que momentos no estuvo en crisis (clasificados como negativos).
2. Identificar las causas de las crisis. Para los periodos de tiempo clasificados como positivos, el sistema debe indicar cuales son las causas que han provocado que sean catalogadas como tal. Para el cliente este requisito es especialmente importante, porque servirá de ayuda para localizar y solucionar los problemas del sistema. Como consecuencia de lo anterior, las causas que han llevado a considerar los momentos de crisis deben indicarse en un lenguaje entendible.
3. Funcionamiento *on-line*. Por defecto el proyecto tiene un enfoque forense, para poder analizar crisis que se hayan producido en el pasado, para solucionar posibles problemas y que no vuelvan a suceder. Por ejemplo, el día siguiente a una crisis, sería interesante ejecutar la herramienta para identificar que factores se dieron para que esta se produjera. Sin embargo, el sistema debe estar preparado para poder funcionar con datos *on-line*, de manera que se puedan analizar los datos que se están produciendo en el momento, para poder actuar sobre las crisis que se están produciendo.
4. Herramienta ejecutable. El sistema debe estar desplegado en el entorno de producción y debidamente documentado, de manera que pueda ser ejecutado en cualquier momento seleccionando el intervalo de fechas que se desea analizar.

2.2. Identificación de las crisis

Una de los principales dificultades que se prevén para el éxito de este proyecto es la realización de una correcta descripción del problema, de manera que se pueda identificar claramente cuales son los momentos de crisis, y con ello poder etiquetar los datos de actividad como registros normales (casos negativos) o registros de momentos de crisis (casos positivos).

Para realizar la de las crisis se dispone de la herramienta de monitoreo Zabbix mantenida por el grupo de sistemas de TAIIF. Concretamente se contemplan dos enfoques distintos:

- Utilizar registros de las alertas enviadas. El sistema envía a los grupos de interés mensajes vía mensajería instantánea alertando de los problemas sucedidos relevantes para cada uno de los grupos. Por tanto, el primer enfoque consiste en recopilar el histórico de las alertas enviadas al grupo Genio y considerar como crisis los momentos en los que se produjeron.
- Utilizar registros de las peticiones al informe *ping*. El sistema Zabbix realiza una petición cada 3 minutos a un informe Genio (el cual hace uso de Genio, MotorZújar y Bases de Datos) y el segundo enfoque consiste en recopilar los tiempos invertidos en estas peticiones y considerar como momentos de crisis aquellos en los que el informe tardó más de un cierto umbral establecido.

2.3. Técnicas y herramientas

El trabajo está enfocado como un proyecto de aprendizaje computacional y más concretamente como modelo de clasificación, mediante el cual se puedan clasificar los intervalos de tiempo como crisis o no crisis.

En esa línea, se estudiarán al menos los siguientes algoritmo de clasificación, para medir cual obtiene mejores resultados:

- K vecinos más próximos (KNN)
- Random Forest
- Árboles de decisión
- XGBoost
- AdaBoost

Para desarrollar la solución se usará la plataforma de código abierto Jupyter Notebook, así como las librerías requeridas para el tratamiento de datos y modelos elegidos, por ejemplo:

- Pandas
- Numpy
- Sklearn
- Matplotlib
- Etc.

Por otro lado, se usará la infraestructura de la AEAT de la suite Motor-Cliente-Genio para realizar la extracción de los datos y parte de la preparación de los mismo.

2.4. Métricas de calidad[2]

Para analizar las predicciones realizadas por los modelos entrenados, se usarán las métricas convencionales, las cuales se obtienen a partir de la matriz de confusión[3]:

		Predicción
Realidad	Verdaderos negativos (TN)	Falsos positivos (FP)
	Falsos negativos (FN)	Verdaderos positivos (TP)

- Precisión (*precision*) = TP / TP + FP - Con esta métrica se obtendrá el porcentaje de crisis identificadas que realmente lo son (se establece 0,7 como la precisión mínima aceptable).
- Exhaustividad (*recall*) = TP / TP + FN - Con esta métrica se obtendrá el porcentaje de crisis identificadas de las que se producen realmente (se establece 0,7 como la exhaustividad mínima aceptable).
- F1 = 2 x *precision* x *recall* / (*precision* + *recall*) - Con esta métrica se combinan los resultados de las dos anteriores para poder tener una visión conjunta de ambas (se establece 0,7 como el F1 mínimo aceptable).
- Exactitud (*accuracy*) = TP + TN / TP + TN + FP + FN - Con esta métrica se obtendrá el porcentaje de acierto total (se establece 0,7 como la exactitud mínima aceptable).

Sin embargo, dada la naturaleza del proyecto, se consideran las más interesantes las dos primeras (precisión y exhaustividad) pues miden la capacidad de modelo de clasificación para encontrar correctamente los casos positivos (crisis del sistema). Por lo tanto, se considera a partir de ahora F1 como la métrica principal, pues es la que combina los resultados de las dos métricas anteriores.

2.5. Diseño de la solución

El sistema desarrollado se plantea como una aplicación para ser ejecutada en un terminal o en un navegador web con Jupyter Notebook, mediante el cual se disponga de un conjunto de comandos que permitan al usuario aplicar los modelos que elija y para el intervalo de fechas que seleccione.

Una vez indicados los parámetros anteriores, la ETL integrada en el programa se encargará de descargar los datos y de prepararlos para a continuación ser ejecutados para el modelo seleccionado. El programa mostrará mensajes en pantalla que ayuden a comprender en que punto de su ejecución está el proceso.

Finalmente mostrará un dossier con los resultados obtenidos. Estos resultados incluirán principalmente un indicador que informará de si en

algún momento del intervalo de tiempo indicado se produjo a no crisis. En caso afirmativo, si el algoritmo seleccionado lo contempla, mostrará además un resumen de las condiciones que han provocado que se haya considerado que los datos eran de un momento de crisis.

En la siguiente imagen se presenta, mediante el diagrama de componentes, la arquitectura de la solución detallada anteriormente:

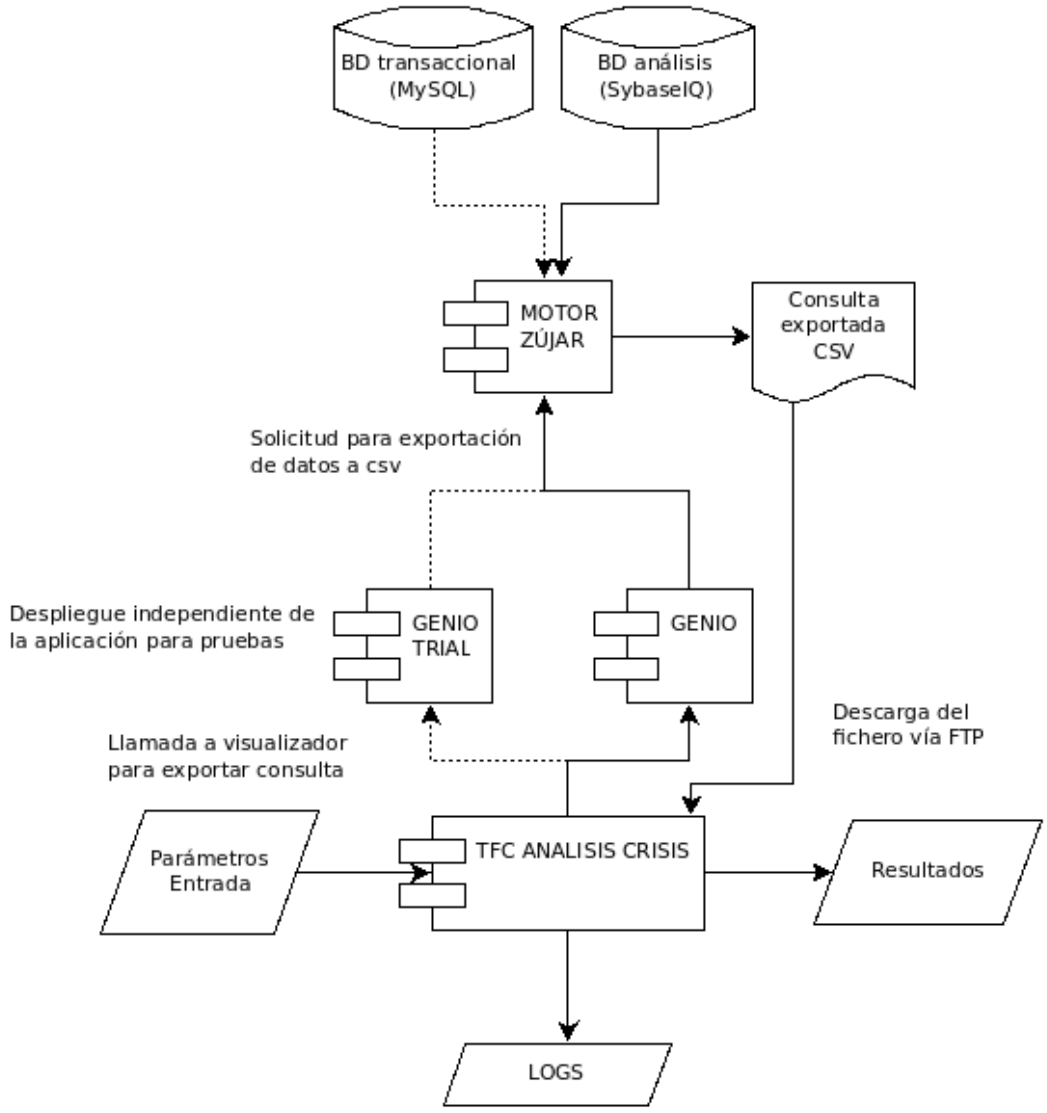


Figura 4: Diagrama de componentes

Existe el inconveniente en usar el sistema Genio para intentar detectar crisis de funcionamiento en el propio sistema Genio y es que este podría no responder (en cualquier caso le podría suponer más carga de la que ya tiene). Para resolverlo, se plantea inicialmente hacer uso del despliegue Genio Trial de la misma aplicación, la cual también obtiene los datos de producción.

3. Entendimiento de los datos

El capítulo se divide en dos secciones principales. En la primera se analizan los datos con la intención de reconocer lo que se considera una crisis en el sistema Genio. En la segunda, se analizan las fuentes de datos y las variables de las que se obtendrán la información para aplicar modelos de regresión con la intención de identificar, predecir y analizar nuevas crisis.

3.1. Identificación de lo casos de crisis

3.1.1 Fuentes de datos utilizadas

Para la obtención de los datos de las crisis se dispone de la herramienta Zabbix usada por el grupo de sistemas para monitorizar el servicio.

3.1.2. Clasificación de los datos de entrenamiento

Para clasificar los datos de entrenamiento como positivos o negativos se siguen varios enfoques con diferentes resultados:

1. Uso de los mensajes recibidos de Zabbix con las alertas Genio para marcar manualmente, en función de las gráficas del propio Zabbix, los periodos marcados como crisis.
2. Igual que el enfoque anterior, pero marcando de manera sistemática un intervalo de tiempo fijado por delante y por detrás de los momentos en los que se reciben las alertas.
3. Uso de los tiempos de respuesta de un informe de control (denominado informe *ping*) de Genio el cual se ejecuta periódicamente cada tres minutos y cuyos resultados se almacenan en la base de datos de Zabbix.

Los enfoques 1 y 2 no han sido útiles para la identificación de los clases, puesto que se dispone de pocos datos y de baja fiabilidad. Sin embargo, durante las fases modelado y evaluación del proyecto, se ha identificado que en la base de datos de Zabbix están registradas todas las alertas enviadas. Se detalla en los apartados 5 y 6 (correspondientes a estas fases) el uso que se ha hecho de esta información.

3.1.3. Retención de los datos de Zabbix

En la base de datos de Zabbix se almacena el histórico de las peticiones realizadas, sin embargo se realiza con una retención por defecto de 30 días. Por ese motivo, se solicita al principio del proyecto al grupo de sistemas que amplíe temporalmente la retención del tipo de alarmas "*Ping* Genio" a 90 días para poder tener un mayor volumen de datos con el que entrenar.

Por este motivo, durante la vida del proyecto, el intervalo de fechas usado para entrenar los modelos ha ido creciendo paulatinamente según avanzaban las semanas.

Concretamente la ampliación fue efectiva el pasado 21 de febrero, por lo que desde el pasado 22 de mayo se dispone de un histórico de 90 días para entrenar los modelos.

3.1.4. Uso crítico de Zabbix

El consumo de información de la base de datos Zabbix debe hacerse de manera responsable pues es un sistema crítico en el conjunto de aplicaciones del sistema Por ese motivo, por recomendación expresa del grupo de sistemas, se cachearán los datos consultados previamente para un mismo intervalo de fechas. Esto es posible porque los datos históricos no cambian.

3.1.5. Calidad de los datos

Se analiza la calidad de los datos, con la intención de buscar registros ausentes o columnas con valores nulos o blancos.

Datos de Zabbix. Los de la base de datos de Zabbix son altamente fiables pues no tienen valores blancos o nulos. Por otro lado, el número de registros ausentes es muy bajo. En la siguiente consulta, sobre la tabla *history* de Zabbix, se observa que el único día en los que no se dispone del al menos el 95% (456) del total de *pings* posibles (60/3*24=480) es el 18/03/2022.

```
2
3 select date(from_unixtime(clock)) as fecha, itemid, count(*)
4 from history where itemid in (216278,216279) and clock > unix_timestamp('2022-01-01 00:00:00')
5 group by fecha, itemid
6 having count(*) <= 456
7
```

history (3×4)

fecha	itemid	count(*)
2022-02-16	216.278	203
2022-02-16	216.279	203
2022-03-18	216.278	270
2022-03-18	216.279	270

Figura 5: Calidad datos informes ping Genio

> Es importante mencionar, que los periodos de los que no se disponga datos para poder clasificarlos no serán considerados en el entrenamiento.

3.2. Información de actividad

3.2.1 Fuentes de datos utilizadas

Las principales fuentes de datos utilizadas para la resolución del proyecto serán los registros de las tablas de actividad que escriben las aplicaciones involucradas: MotorZújar, ClienteZújar y Genio (cada aplicación escribe en una tabla separada). Sin embargo, estas tablas solo almacenan las últimas 48 horas de los datos, ya que en un proceso nocturno los datos más antiguos se mueven a una única tabla del ámbito de análisis en la cual se combinan los registros de las tres tablas anteriores.

Sin embargo, la infraestructura Zújar permite obtener los datos de ambas bases de datos de manera homogénea, ya que el MotorZújar habilita a sus clientes a acceder a los datos de distintas tecnologías de base de datos de manera transparente para el usuario. Por ese motivo, en uso de los datos transaccionales (MySQL) y el de los datos de análisis (SybaseIQ) puede ser idéntico, con apenas cambiar la identidad de la consulta en la que establece el origen de los datos.

Las variables utilizadas para el estudio se van a obtener del Zújar de peticiones histórico, el cual se nutre de los registros con la actividad que las aplicaciones involucradas escriben en sus correspondientes tablas MySQL. Este Zújar se recarga todas las noches con los registros generados en la jornada anterior, por ese motivo se le denomina Zújar histórico.

Por otro lado, también se dispone de zújares individuales *on-line* montados directamente sobre las tablas anteriores, pero estos no serán usados inicialmente, pues sus tablas se consideran críticas y su podría penalizar el funcionamiento del sistema. Sin embargo, las variables usadas del Zújar histórico van a tener su correspondencia en los zújares *on-line*, por lo que los modelos desarrollados podrán serán aplicados también sobre estos últimos.

Asimismo, es importante mencionar que gracias a la infraestructura Zújar, se puede cambiar la fuente de datos de manera transparente, tan solo con cambiar la consulta que obtiene los resultados, para que use las variables de un zújar o de los otros.

3.2.2 Selección de variables

En la siguiente figura se muestra el diálogo con la cual permite la aplicación ClienteZújar seleccionar las variables disponibles en el Zújar de peticiones histórico:

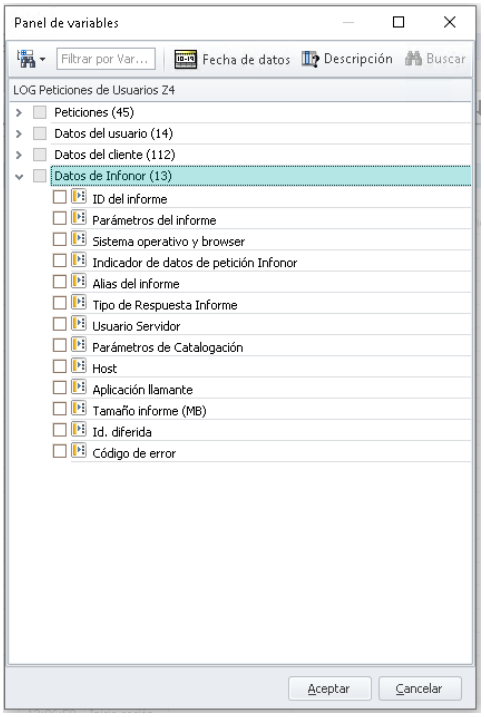


Figura 6: Columnas disponibles Zújar de Peticiones

Como se puede observar, hay un total de 184 variables (45+14+112+13), las cuales se agrupan en 4 carpetas (Peticiones, Datos del usuario, Datos del cliente, Datos de Infonor). Se analizan las 184 variables para decidir o no su uso en el presente estudio (en función a las consideraciones anteriores) y se preseleccionan inicialmente las siguientes:

#	Variable	Tipo	Descripción
1	ID	Ambos	Identificador único de la petición
2	ID Sesión	Ambos	Identificador único de la sesión (bajo una misma sesión se realizan varias peticiones de informes y a su vez de consultas)
3	Indicador Infonor	Ambos	Identifica si el registro en base de datos tiene información generada por Genio. Puesto que se va a revisar el funcionamiento de la aplicación Genio solo se van a considerar registros con este indicador activo.
4	Tipo Operación	Ambos	El tipo de operación, para los registros de Genio (con el indicador de Genio activo) admite los valores "PETICIÓN INFORME" y "CONSULTA MOTOR ZÚJAR"
5	Subtipo de operación	Consulta	El subtipo de operación solo está informado para los registros cuyo tipo de operación es "CONSULTA MOTOR ZÚJAR" y admite los siguientes valores: Principal, Nº total de registros de la consulta, Sumatorio, etc
6	Resultado	Ambos	Admite los valores "OK" y "ERROR" e indica si la consulta o el informe ha terminado correctamente o con error respectivamente.
7	Fecha inicio	Ambos	Fecha en la que se solicita el informe o consulta

#	Variable	Tipo	Descripción
8	Hora inicio	Ambos	Hora en la que se solicita el informe o consulta
9	Fecha fin	Ambos	Fecha en la que se finaliza el informe o consulta
10	Hora fin	Ambos	Hora en la que se finaliza el informe o consulta
11	ID del informe	Informe	Identificador del informe
12	ID de la consulta	Consulta	Identificador de la consulta
13	Versión de Genio	Ambos	Versión de la aplicación Genio durante la solicitud de la consulta o informe
14	Usuario	Ambos	Identificador del usuario que solicita el informe o consulta
15	En caché cliente	Consulta	Indica si la consulta se resolvió en caché de Genio
16	En caché motor	Consulta	Indica si la consulta se resolvió en caché del Motor
17	Host	Ambos	Identifica el servidor de Genio en el que se resolvió el informe o la consulta (actualmente admite los valores tomcat-1, tomcat-2, tomcat-3 y tomcat-4)
18	Aplicación llamante	Informe	Identificador de la aplicación que realiza la petición del informe Genio.
19	Tamaño del informe	Informe	Tamaño en bytes que supone la respuesta del informe Genio
20	Formato del informe	Informe	Formato en el que se solicita el informe Genio (html, pdf, rtf, etc.)
21	Tiempo total (ms)	Ambos	Tiempo (en mili-segundos) que se invierte desde Genio para la obtención de cada consulta (incluye tiempo de servidor más tiempo transmisión) o de cada informe (incluye la suma de los tiempos de todas sus consultas más el del propio proceso del informe)
22	Tiempo total servidor	Consulta	Tiempo (en mili-segundos) que invierte el Motor, que incluye el tiempo de base de datos, el de proceso del Motor y de su servicio web.
23	Tiempo transmisión	Consulta	Tiempo (en mili-segundos) de transmisión entre Genio y el Motor
24	Tiempo BBDD	Consulta	Tiempo (en mili-segundos) dedicado a obtener la consulta de la base de datos
25	Árbol	Consulta	En el campo árbol se almacena el código del árbol. El árbol es equivalente al zújar sobre el que está montada la consulta. La consulta puede depender de más de un árbol (consultas combinadas de varios zújares), por tanto, este campo es una lista de arboles separados por comas, como por ejemplo: "CT4CTE,ZJ4RXX,AD425C1P"
26	Nodo motor	Consulta	Es el nodo del motor donde se procesa la consulta. En principio admite los valores "Motor-1" y "Motor-2", correspondiendo a los dos motores en funcionamiento en alta disponibilidad.
27	Identificador diferida Genio	Informe	Identificador de petición creada de manera diferida para que el sistema la procese idealmente en momentos de carga baja. Aunque esta opción es nueva en el entorno Genio y posiblemente no aporte soluciones al funcionamiento actual, se considera porque probablemente si lo haga a medio plazo.

3.2.3. Análisis de componentes principales[4]

Con el análisis de componentes principales o PCA (*Principal Component Analysis*) se pretende reducir la dimensión de los datos a analizar reduciendo el número de columnas consideradas. Para ello, se buscan correlaciones o dependencias entre las variables de origen para intentar expresar la misma información en un número menor de variables pero conservando en un alto porcentaje toda la información.

Sin embargo, de las variables seleccionadas anteriormente, solo las siguientes son variables numéricas y por lo tanto susceptibles de aplicar sobre ellas el análisis de componentes principales:

- Tamaño del informe
- Tiempo total (ms)
- Tiempo transmisión

Por lo tanto, no merece la pena hacer este estudio pues apenas se puede reducir las dimensionalidad, y sobre todo cuando estas variables son fundamentales para el estudio y es importante no perder nada de información sobre ellas.

Por otro lado, uno de los objetivos del proyecto es la explicabilidad de los resultados, la cual se vería perjudicada si se aplicase esta técnica para agrupar variables.

3.2.4. Tratamiento agrupado de los datos

En el estudio se tratarán los datos de actividad de manera agrupada (probablemente por minuto o por grupos de tres minutos, para coincidir con cadencia de ejecución del informe de control), considerando de esta manera periodos de tiempo en lugar de instantes individuales.

Para realizar esta agrupación se dispone de dos opciones, agrupar en la elaboración de la consulta Zújar de manera que los datos se descarguen ya agrupados o agrupar los datos en Python una vez estos sean descargados.

Se analizan a continuación los pros y los contras de ambas opciones:

a) Se identifican los siguientes aspectos positivos de realizar el tratamiento en la consulta que se usa para realizar la exportación de los datos desde la base de datos:

- Disminuir el volumen de datos manejados al agrupar la consulta en el origen. Por ejemplo, en lo que ha transcurrido desde el 1 de Enero hasta el 22 de Marzo de 2022 el número medio de peticiones por minuto es de 385.
- Se realiza una prueba de descarga en formato CSV de los datos para las variables consideradas en este documento y para un

intervalo de tiempo de dos meses y el tamaño del fichero generado es de 12,3 Gb.
- Conocimiento previo (y sencillez) para obtener los totales en las consultas Zújar

b) Se identifican los siguientes aspectos positivos de realizar el tratamiento en Jupyter Notebook:

- Se dispone de un conjunto superior de funciones para tratamiento de periodos (por ejemplo *period* o *resample* de la librería Pandas). De hecho, es posible que en Zújar no sea posible agrupar los datos por grupos de tres minutos (que es una de las opciones barajadas) mientras que con Pandas es inmediato. Se muestra un ejemplo en la siguiente consulta:
- Centralizar todo el tratamiento de los datos en la herramienta Jupyter Notebook especializada en ello.
- Flexibilidad para hacer cambios en base a los resultados parciales obtenidos.
- Se reduce la carga del sistema Genio/Zújar al evitar lanzar una consulta, que debido al uso de funciones sobre datos agrupados, que es probable que sea pesada para el sistema.

Finalmente se opta por una solución intermedia, mediante la cual se realiza una primera agrupación (por minuto) en el ClienteZújar (de manera que se reduce tamaño de la información a mover) y posteriormente una segunda agrupación en Python (mediante las funciones avanzadas) por bloques de tres minutos.

3.2.7. Listado de variables agrupadas

Se usarán inicialmente la siguientes variables agrupadas por minuto:
- Número de consultas solicitadas
- Número de consultas correctas
- Número de informes solicitados
- Número de informes correctos
- Número de consultas combinadas (consultan más de un Zújar)
- Tiempo medio de las consultas generadas (ms)
- Tiempo medio de los informes generados (ms)
- Tamaño medio de los informes generados (bytes)
- Número de consultas no resueltas en caché cliente
- Número de consultas no resueltas en caché motor
- Número de informes diferidos
- Número de peticiones de los principales tipos de consultas (ZSQL, exportación datos, principal, nº de registros, etc.)
- Número de consultas por cada uno de los 4 servidores Tomcat de Genio
- Número de informes de cada uno de los formatos existentes (HTML, PDF, RTF, XLS, etc)

- Número de informes solicitados por determinadas aplicaciones consideras críticas
- Número de consultas sobre determinados zújares debido a su criticidad o a su volumen

3.2.6. Consultas pesadas

Las consultas desarrolladas para obtener los datos de actividad son pesadas para el sistema, por lo que su consumo también será cacheado, de manera que los datos de actividad descargados para una fecha no deberán volver a ser consultados al MotorZújar pues se dispondrá de una copia de los mismos.

3.2.7. Calidad de los datos

Se analiza la calidad de los datos, con la intención de buscar registros ausentes o columnas con valores nulos o blancos. Los datos de actividad son altamente fiables, pues no se han encontrado en el intervalo de fechas que ha durado el estudio variables con valores blancos o nulos. En cuanto a registros perdidos, el sistema no tiene reportadas incidencias a este respecto, por lo que no se cuenta con la ausencia de peticiones. Sin embargo, no hay forma de confirmarlo al igual que se ha hecho con las peticiones de Zabbix en la que el número de peticiones esperadas es conocido.

4. Preparación de los datos

4.1. Obtención de los datos de resultado

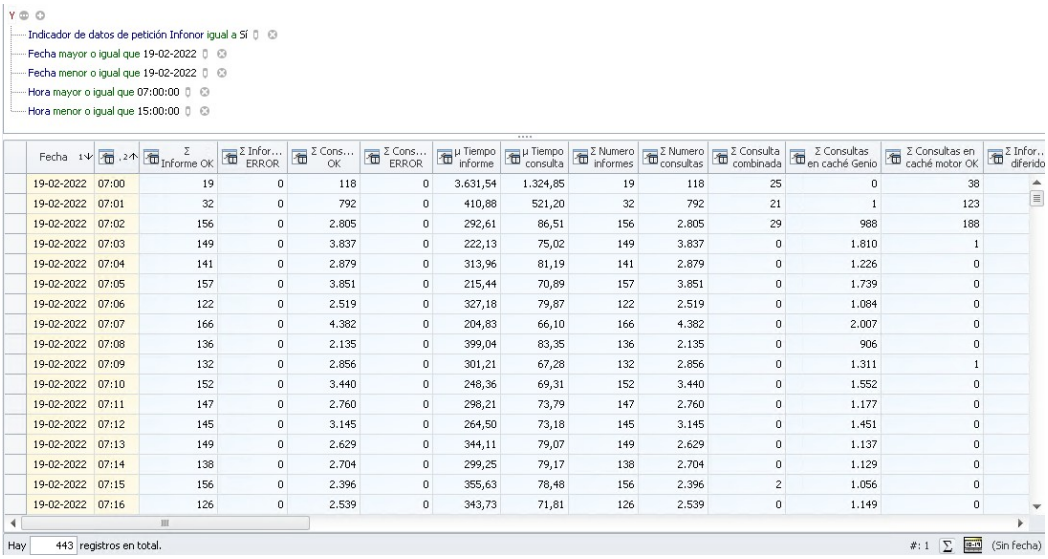
Para la obtención de los resultados positivos se hace uso de los registros contenidos en la tabla *history* de la base de datos de Zabbix. Concretamente se usan los tiempos de respuesta (y velocidad de descarga) de las peticiones periódicas realizadas a un informe Genio llamado informe *ping* (el cual es ejecutado cada 3 minutos).

Para ejecutar la consulta desde Python y descargar los resultados en un *dataframe* se hace uso del conector *mysql.connector*.

4.2. Obtención de los datos de actividad

Para obtener los datos de actividad se realiza una consulta con la aplicación ClienteZújar que permite obtener los datos con los filtros, expresiones y agrupaciones necesarias.

En la siguiente figura se muestra un ejemplo de resultado de la ejecución de la consulta Zújar (en la imagen solo se aprecian las primeras 14 columnas de las 46 existentes):


Y
Indicador de datos de petición Infonor igual a Sí
Fecha mayor o igual que 19-02-2022
Fecha menor o igual que 19-02-2022
Hora mayor o igual que 07:00:00
Hora menor o igual que 15:00:00

Fecha	Σ Informe OK	Σ Infor... ERROR	Σ Cons... OK	Σ Cons... ERROR	μ Tiempo informe	μ Tiempo consulta	Σ Numero informes	Σ Numero consultas	Σ Consulta combinada	Σ Consultas en caché Genio	Σ Consultas en caché motor OK	Σ Infor.. diferido
19-02-2022 07:00	19	0	118	0	3.631,54	1.324,85	19	118	25	0	38	
19-02-2022 07:01	32	0	792	0	410,88	521,20	32	792	21	1	123	
19-02-2022 07:02	156	0	2.805	0	292,61	86,51	156	2.805	29	988	188	
19-02-2022 07:03	149	0	3.837	0	222,13	75,02	149	3.837	0	1.810	1	
19-02-2022 07:04	141	0	2.879	0	313,96	81,19	141	2.879	0	1.226	0	
19-02-2022 07:05	157	0	3.851	0	215,44	70,89	157	3.851	0	1.739	0	
19-02-2022 07:06	122	0	2.519	0	327,18	79,87	122	2.519	0	1.084	0	
19-02-2022 07:07	166	0	4.382	0	204,83	66,10	166	4.382	0	2.007	0	
19-02-2022 07:08	136	0	2.135	0	399,04	83,35	136	2.135	0	906	0	
19-02-2022 07:09	132	0	2.856	0	301,21	67,28	132	2.856	0	1.311	1	
19-02-2022 07:10	152	0	3.440	0	248,36	69,31	152	3.440	0	1.552	0	
19-02-2022 07:11	147	0	2.760	0	298,21	73,79	147	2.760	0	1.177	0	
19-02-2022 07:12	145	0	3.145	0	264,50	73,18	145	3.145	0	1.451	0	
19-02-2022 07:13	149	0	2.629	0	344,11	79,07	149	2.629	0	1.137	0	
19-02-2022 07:14	138	0	2.704	0	299,25	79,17	138	2.704	0	1.129	0	
19-02-2022 07:15	156	0	2.396	0	355,63	78,48	156	2.396	2	1.056	0	
19-02-2022 07:16	126	0	2.539	0	343,73	71,81	126	2.539	0	1.149	0	

Hay 443 registros en total. #: 1 (Sin fecha)

Figura 7: Consulta almacenada agrupando por minutos

Debido a su complejidad, consulta ha sido revisada, con ayuda del grupo de bases de datos de TAIIF, y se ha determinado que su ejecución no es excesivamente pesada para el sistema, puesto que la consulta filtra por 3 índices existentes en la tabla en cuestión. Por tanto, no hay problema en ejecutar esta consulta puntualmente para obtener los datos de actividad de esta fuente y con esta consulta. No obstante, como se

detalla más adelante, los resultados de esta consulta son cacheados para evitar ejecuciones repetidas.

Para obtener una exportación de los datos obtenidos por una consulta Zújar y cargarlos en un *dataframe*, se dispone de la función *exportacion_genio* de la librería *connect2motor* desarrollada por el grupo de Análisis de Datos de TAIIF.

4.3. Preparación y ensamblaje de los datos anteriores

Una vez se dispone de los datos de datos de actividad y de los datos de resultados, queda pendiente la preparación de los datos para establecerlos en el formato adecuado y unificarlos en un único *dataframe* para poder aplicar los modelos sobre ellos. Para hacerlo se realizan varias operaciones en Jupyter Notebook, entre las que destacan las siguientes:

1. Indexar el conjunto de datos por la fecha y hora
2. Se marcan como positivos aquellos registros cuyos tiempos de ejecución son iguales o superiores a 3 segundos y negativos el resto.
3. Realizar una nueva agrupación para disponer de la información por bloques te tres minutos con la función *resample* (usando la función de agregación *mean* o *sum* según interese en cada caso)
4. Combinar los dos *dataframe* obtenidos mediante la función *merge* de Pandas. Es importante considerar el problema de ciertos datos inexistentes en el *dataframe* de datos de resultados (tal y como se adelantaba en el capítulo "04. Entendimiento de los datos"), ya que para ciertos días el número de registros no coinciden con el mínimo número de datos esperados. Para solucionar este problema la combinación se realizará de modo interna ó *inner*, de manera solo se tomarán en consideración los registros cuyos periodos se encuentren en ambos set de datos.

4.4. Automatización del proceso

Para facilitar la utilización de las preparación de los datos, en base a los pasos descritos anteriormente, se ha implementado un función llamada *get_prepared_dataframe* dentro de una libreria llamada *aeat_activity_machine_learning* que encapsula todas estas actividades. A continuación se realizan una serie de consideraciones de interés:

- La función recibe los parámetros *date_from* y *date_to* para poder indicar el intervalo del fecha sobre el que se desea obtener los datos
- Para disminuir la carga del sistema y puesto que los datos almacenados en las tablas de análisis no cambian, si ya se ha invocado la función previamente con unos determinados parámetros, entonces esta no vuelve a consultar la tabla de

actividad, sino que devuelve los datos directamente desde una copia local realizada durante la primera ejecución.

- De manera análoga, los datos los resultados solo se consultan la primera vez para un mismo intervalo de fechas, puesto que también se cachean los resultados para no tener que volver a repetir la consulta a base de datos.
- En la siguiente figura, se muestra un ejemplo de utilización de la función anterior desde Jupyter Notebook:

```
from aeat_activity_machine_learning import get_prepared_dataframe
df = get_prepared_dataframe('2022-03-20', '2022-03-24')
```

Figura 8: Código Python para obtener dataframe con consulta almacenada

Junto a esta memoria se entrega el código fuente desarrollado, así como una guía de uso de la misma. En el presente apartado solo se describe el método utilizado para obtener el *dataframe* con los datos fusionados, (este método no existe en la versión final de la librería pues ha sido reemplazado por otros).

4.5. Caché de resultados

Los métodos anteriores para descargar los datos de actividad y los de resultados implementan un sencillo sistema de caché en disco para evitar peticiones repetidas al sistema. Esto es posible porque los datos consultados son históricos y no cambia y con ello se evitan peticiones redundantes al sistema que pueden ser bastante pesadas.

5. Modelado

Se han utilizado 10 métodos de clasificación que se detallan a continuación, evaluando los datos preparados en la fase anterior, para intentar conseguir los resultados fijados como objetivos del proyecto.

5.1. K vecinos más próximos o K Nearest Neighbours (KNN)

Se evalúa la calidad de predicción del algoritmo KNN para identificar los periodos de crisis en el conjunto de datos preparado en la fase anterior. Para ello, se utiliza la función *KNeighborsClassifier* de Scikit klearn[5].

Sin embargo, antes de aplicar KNN es necesario realizar un tratamiento adicional sobre los datos para normalizarlos, ya que este algoritmo trabaja con distancias y es por lo tanto necesario disponer de todas las variables en la misma escala para que unas variables no tengan más peso que otras.

En la siguiente figura, se muestra el resultado de la ejecución del algoritmo para los primeros 15 valores de K:

```
Para KNN=1 la exactitud (accuracy) del modelo es: 0.9342301943198804
Para KNN=2 la exactitud (accuracy) del modelo es: 0.953662182361734
Para KNN=3 la exactitud (accuracy) del modelo es: 0.9476831091180867
Para KNN=4 la exactitud (accuracy) del modelo es: 0.9521674140508222
Para KNN=5 la exactitud (accuracy) del modelo es: 0.953662182361734
Para KNN=6 la exactitud (accuracy) del modelo es: 0.953662182361734
Para KNN=7 la exactitud (accuracy) del modelo es: 0.953662182361734
Para KNN=8 la exactitud (accuracy) del modelo es: 0.9566517189835575
Para KNN=9 la exactitud (accuracy) del modelo es: 0.9551569506726457
Para KNN=10 la exactitud (accuracy) del modelo es: 0.9566517189835575
```

Figura 9: Valores de exactitud para KNN

Como se puede observar los resultados parecen excelentes. Sin embargo, al consultar la matriz de confusión obtenida para el valor K=3 (que es con el que se obtiene un mejor resultados), se puede observar que los resultado

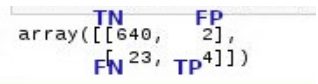


Figura 10: Matriz de confusión

En la imagen se han marcado los valores de cada posición de la matriz (verdaderos negativos o *true negative* (TN), falsos positivos o *false positive* (FP), falsos negativos o *false negative* (FN) y verdaderos positivos o *true positive* (TP)). Se puede observar que los resultados no son tan buenos como parecían, puesto que el número de positivos acertados es muy bajo (4 de un total de 27) y por tanto el buen resultado

para el valor de exactitud, está determinado por el bajo número de casos de positivos existentes en el conjunto de datos, es decir que los casos positivos y negativos están desbalanceados.

Por ese motivo, en el apartado "2.4. Métricas de calidad" se estableció como métrica de calidad a utilizar la métrica F1. Se calculan a continuación el resto de medidas obtenidas por el algoritmo KNN para K=3:

Precision	Recall	F1	Accuracy	Support
0.67	0.15	0.24	0.96	669

Figura 11: Métricas KNN para K=3

Con el resultado anterior, se confirma el problema balanceo de los datos. Al contrario de lo que parecía con los resultados de exhaustividad, las valores para las métricas *precision* (porcentaje de los marcados como positivos son realmente positivos) y *recall* (porcentaje de los positivos existentes identificados como positivos), que son las más importantes para este modelo, son realmente bajos.

A continuación, se evalúa el resultado de la métrica F1 para la K tomando los primeros 15 valores enteros y se obtiene resuelve que el mejor resultado obtenido es para K=3. Por tanto, los valores de la figura anterior son una muestra válida de los pobres resultados que obtiene KNN para clasificar el conjunto de datos. Se recuerda que el objetivo se sitúa en 0,7 para las 4 métricas en cuestión.

Por último, comentar que se ha desarrollado una función llamada *model_knn* que realiza la operativa anterior a partir de un *dataframe* obtenido como parámetro.

5.2. Random Forest

Se aplica el modelo de clasificación *RamdomForestClassifier* de Sklearn[6]. En esta caso también es preciso hacer una normalización de los datos la cual se realiza con la utilidad *MaxMinScaller* de Sklearn. Asimismo, se prueba con varias opciones para los parámetros *random_state* y *num_estimators*. En la siguiente figura se muestran los resultados obtenidos con la selección de parámetros más positiva:

Precision	Recall	F1	Accuracy	Support
0.67	0.07	0.13	0.96	669

Figura 12: Métricas Random Forest

Los resultados obtenidos por este modelo son también muy pobres, probablemente por el mismo problema de datos positivos y negativos no balanceados.

Por último, comentar que se ha desarrollado una función llamada *model_random_forest* que realiza la operativa anterior a partir de un *dataframe* obtenido como parámetro.

5.3. Árboles de decisión

Se aplica el modelo de clasificación *DecisionTreeClassifier* de Sklearn[7], el cual está basado en árboles de decisión. En este caso, no es preciso realizar normalización de los datos, puesto que el algoritmo no funciona con distancias. En la siguiente imagen se muestran los resultados obtenidos tras el modelo:

Precision	Recall	F1	Accuracy	Support
0.18	0.15	0.16	0.94	669

Figura 13: Métricas árboles de decisión

Como se puede observar, los resultados obtenidos por este modelo son igualmente muy pobres. Al igual que para los modelos anteriores, se ha encapsulado el código necesario para aplicar el modelo sobre un *dataframe* de entrada una nueva función llamada *model_decission_tree*.

5.4. XGBoost

Se aplica el modelo de clasificación XGBoost con la utilidad *XGBRFClassifier*[8]. En este caso, no se realiza normalización de los datos, puesto que el algoritmo el algoritmo obtiene idénticos resultados para el conjunto de datos de estudio. En la siguiente figura se muestran los resultados obtenidos al ejecutar el modelo para el conjunto de datos:

Precision	Recall	F1	Accuracy	Support
0.67	0.07	0.13	0.96	669

Figura 14: Métricas XGBoost

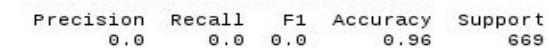
Como se puede observar, los resultados obtenidos por este modelo son igualmente muy pobres. Al igual que para los modelos anteriores, se ha encapsulado el código necesario para aplicar el modelo sobre un *dataframe* de entrada una nueva función llamada *model_xgboost*.

5.5. Support vector classifier (SVC)

Se aplica el modelo SVC de Sklearn[9]. el cual implementa un algoritmo de clasificación basado en máquinas de vector de soporte. En este caso, es preciso realizar normalización de los datos, puesto que en caso

contrario la ejecución del algoritmo para el conjunto de datos utilizado no termina en un tiempo razonable.

Se han probado distintos valores para los parámetros *kernel*, *gamma*, *random_state*, etc y en todos los casos se ha obtenido el mismo resultados. En la siguiente figura se muestran los resultados obtenidos al ejecutar el modelo para el conjunto de datos con los parámetros con los que se obtiene mejores resultados:



Precision	Recall	F1	Accuracy	Support
0.0	0.0	0.0	0.96	669

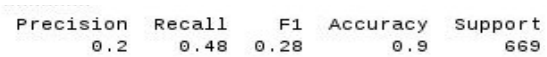
Figura 15: Métricas SVC

Los resultados obtenidos por este modelo son muy malos, obteniendo por ejemplo un resultado de 0,0 en *precision* y en *recall*. Al igual que para los modelos anteriores, se ha encapsulado el código necesario para aplicar el modelo sobre un *dataframe* de entrada una nueva función llamada *model_scv*.

5.6. Multi-layer Perceptron Classifier (MLP)

Se aplica el modelo *MLPClassifier* de Sklearn[10] el cual implementa un algoritmo de clasificación basado en redes neurológicas. En este caso, no se realiza normalización de los datos, puesto que al hacerlo se obtiene peores resultados.

Se han probado distintos valores para los parámetros *solver*, *alpha*, *hidden_layer_sizes*, etc y los mejores resultados se han obtenido con los mostrados en la imagen anterior. A continuación, se muestran los mejores resultados obtenidos al ejecutar el modelo para el conjunto de datos:



Precision	Recall	F1	Accuracy	Support
0.2	0.48	0.28	0.9	669

Figura 16: Métricas MLP

Los resultados obtenidos por este modelo continúan siendo muy pobres, a pesar de ser los mejores hasta el momento. Al igual que para los modelos anteriores, se ha encapsulado el código necesario para aplicar el modelo sobre un *dataframe* de entrada una nueva función llamada *model_mlp*.

5.7. Gaussian Process

Se aplica el modelo *GaussianProcessClassifier* de Sklearn[11] basado en probabilidad. En este caso, no se realiza normalización de los datos, puesto que al hacerlo se obtienen los mismos resultados. A continuación, se muestran los resultados obtenidos al ejecutar el modelo para el conjunto de datos:

Precision	Recall	F1	Accuracy	Support
0.2	0.07	0.11	0.95	669

Figura 17: Métricas gaussian process

Los resultados obtenidos por este modelo continúan siendo muy pobres. Al igual que para los modelos anteriores, se ha encapsulado el código necesario para aplicar el modelo sobre un *dataframe* de entrada una nueva función llamada *model_gaussian_process*.

5.8. AdaBoost

Se aplica el modelo *AdaBoostClassifier*[12] de Sklearn el cual implementa un algoritmo de clasificación basado reglas de inferencia a partir de las variables de entrada. En este caso, no se realiza normalización de los datos, puesto que al hacerlo se obtiene los mismos resultados que sin hacerlo.

Se han probado distintos valores para los parámetros *random_state*, *num_estimators*, etc y los mejores resultados se han obtenido con los mostrados en la imagen anterior. A continuación, se muestran los resultados obtenidos al ejecutar el modelo para el conjunto de datos:

Precision	Recall	F1	Accuracy	Support
0.71	0.19	0.29	0.96	669

Figura 18: Métricas AdaBoost

Los resultados obtenidos por este modelo continúan siendo muy pobres, a pesar de ser los mejores hasta el momento. Al igual que para los modelos anteriores, se ha encapsulado el código necesario para aplicar el modelo sobre un *dataframe* de entrada una nueva función llamada *model_ada_boost*.

5.9. Gaussian Naive Bayes

Se aplica el modelo *GaussianNB* de Sklearn[13] el cual implementa un algoritmo de clasificación basado la relación de manera independiente de las variables de entrada y la variable resultado. En este caso, no se

realiza normalización de los datos, puesto que al hacerlo se obtiene los mismos resultados que sin hacerlo. A continuación, se muestran los resultados obtenidos al ejecutar el modelo para el conjunto de datos:

Precision	Recall	F1	Accuracy	Support
0.07	0.7	0.14	0.64	669

Figura 19: Métricas gaussian NB

Los resultados obtenidos por este modelo son también muy pobres. Al igual que para los modelos anteriores, se ha encapsulado el código necesario para aplicar el modelo sobre un *dataframe* de entrada una nueva función llamada *model_gaussian_nb*.

5.10. Quadratic Discriminant Analysis

Se aplica el modelo *QuadraticDiscriminantAnalysis* de Sklearn[14] el cual implementa un algoritmo de clasificación basado reglas bayesanas con un límite de decisión cuadrático. En este caso, no se realiza normalización de los datos, puesto que al hacerlo se obtiene los mismos resultados.

Precision	Recall	F1	Accuracy	Support
0.08	0.63	0.14	0.69	669

Figura 20: Métricas QDA

Los resultados obtenidos por este modelo continúan siendo muy pobres. Se ha encapsulado el código necesario para aplicar el modelo sobre un *dataframe* de entrada una nueva función llamada *model_quadratic_discriminant_analysis*.

5.11. Conclusiones

Como se puede apreciar los resultados obtenidos son muy pobres en todos los modelos, lo que indica que existe algún problema general que lo provoca. En el siguiente apartado se identifican estos posibles problemas y se buscan soluciones para los mismos.

6. Evaluación

6.1. Histórico de evaluaciones

Con el objetivo de disponer de un histórico con el resultado de las distintas evaluaciones de los modelos, se implementa un registro de resultados históricos el cual se actualiza automáticamente cada vez que se realiza una evaluación de todos los modelos con el método *eval_all_models*. En la siguiente figura se muestra un fragmento con las dos primeras ejecuciones registradas:

evaluation_date	2022/04/21 11:17:24	2022/04/21 12:35:02
ada_boost_accuracy	0.96	0.99
ada_boost_f1	0.29	0.25
ada_boost_precision	0.71	1
ada_boost_recall	0.19	0.14
comment	Prueba fase 1: ejecución con consulta almacenada	Prueba fase 2: 1ª ejecución con zsql
date_from	2022-02-19	2022-02-21
date_to	2022-04-01	2022-04-01
decision_trees_accuracy	0.94	0.97
decision_trees_f1	0.14	0
decision_trees_precision	0.18	0
decision_trees_recall	0.11	0
gaussian_nb_accuracy	0.64	0.92
gaussian_nb_f1	0.14	0.14
gaussian_nb_precision	0.07	0.08
gaussian_nb_recall	0.7	0.43

Figura 21: Ejemplo histórico de resultados

Como se puede observar, para cada registro se almacena la fecha de evaluación, un comentario para identificar la prueba en cuestión, las fechas de inicio y fin para las que se realizó la evaluación. Por otro lado, para cada uno de los 10 modelos analizados se registran las 4 métricas consideradas en el estudio (*precision*, *recall*, *f1* y *accuracy*).

Junto con esta memoria, se entrega un fichero CSV con el histórico de todas las evaluaciones realizadas.

6.2. Problemas identificados

6.2.1. Problema de balanceo

En la siguiente figura se muestra el porcentaje de datos positivos y negativos respecto del total (para el *dataframe* del ejemplo que contempla los datos entre el 19/02/2022 y 01/04/2022):

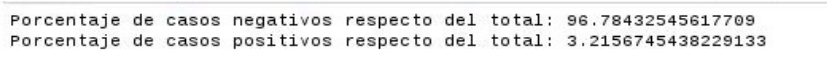


Figura 22: Problema balanceo clases

Como se puede apreciar, el porcentaje de positivos frente al total es del 3,21% lo que provoca problemas para entrenar los algoritmos de clasificación[15].

6.2.1. Agrupación incorrecta de los datos

La agrupación de los datos utilizada no considera el tráfico acumulado. Con la consulta que obtiene los datos de actividad que se ha usado para analizar los modelos se obtienen los informes y consultas que se han lanzado en el periodo de tiempo en cuestión, sin embargo, en estos periodos de tiempo no se consideran los informes o consultas que se lanzaron en periodos anteriores y que todavía están en curso.

6.2.3. Problema en la selección de variables

Hay columnas que tienen el mismo valor para todos los registros. Asimismo, se han identificado otras variables que podrían aportar valor y por el contrario otras variables que parecen estar aportando muy poco.

6.2.4. Datos de los fines de semana

En la consulta utilizada hasta la fecha se obtienen los datos de los fines de semana, los cuales no representan un tráfico significativo para el análisis de las crisis que se desean identificar. Por lo que se considera que es preferible obviar estos datos.

6.2.5. Problema en la clasificación de las crisis

Por la distribución muy dispersa de los datos de las crisis, existe la sospecha de que el informe *ping* utilizado para clasificar los momentos como crisis o como no crisis no es acertada.

6.3. Técnicas utilizadas para mejorar los resultados

Se describen a continuación las técnicas aplicadas (en orden cronológico) con la finalidad de resolver los errores destacados anteriormente.

6.3.1. Eliminación de los datos de los fines de semana

Tal y como se indica en el apartado anterior ("6.2. Problemas identificados"), se decide eliminar del estudio los datos de los fines de

semana, puesto que no tienen un perfil de uso similar al que se desea analizar. Este cambio se desarrolla junto con el siguiente (cambio de consulta almacenada a consulta de tipo ZSQL). Por tanto, los resultados obtenidos se revisan en el siguiente punto.

6.3.2. Cambios en la consulta para considerar las peticiones en curso

Se estudia la manera de mejorar la consulta Zújar que extrae los datos de actividad para que incluya el número de peticiones (consultas e informes) en curso, puesto que se considera que los periodos de crisis no están determinados por las peticiones lanzadas en un intervalo de tiempo, sino por las peticiones en curso en ese intervalo de tiempo. Sin embargo, tras consultarlo con el equipo de desarrollo de la aplicación ClienteZújar, se confirma que no hay forma de trasladar esta información a una consulta.

Por ese motivo, se decide cambiar el enfoque y dejar de usar consultas creadas por el ClienteZújar y usar en su lugar consultas de tipo ZSQL (variante de SQL de uso interno en la AEAT) con las que si sea posible realizar esta operativa. Se desarrolla la nueva consulta ZSQL[2] para que obtenga las mismas columnas que las que obtenía la consulta anterior pero con el nuevo enfoque.

Afortunadamente, el grupo de Analytics de la AEAT ya dispone de una utilidad que permite realizar la exportación de datos de una consulta (de manera equivalente a la que se ha usado anteriormente para exportar los datos de una consulta almacenada) a partir de una ZSQL. De esta manera, el proceso de preparación de los datos no sufrirá muchos cambios, más allá de usar esta nueva utilidad para consultar usando la nueva ZSQL (el límite de filas que se puede obtener con este nuevo método es menor, pero sigue siendo suficiente para el número de registros que se manejan en el estudio gracias a estar agrupados por minuto).

Sin embargo, pese al cambio de consulta, los resultado obtenidos no mejoran con respecto a los obtenidos con la consulta almacenada. En la siguiente figura se muestran los indicadores F1 obtenidos en todos los modelos para la última ejecución con consulta almacenada y la primera con consulta ZSQL:

2 Se decide no compartir en este trabajo el código de la consulta ZSQL elaborada porque utiliza sintaxis de uso privado en la AEAT y no se desea publicar detalles de su naturaleza.

evaluation_date	2022/04/21 11:17:24	2022/04/21 12:35:02
ada_boost_f1	0.29	0.25
comment	Prueba fase 1: ejecución con consulta almacenada	Prueba fase 2: 1ª ejecución con zsql
decision_trees_f1	0.14	0
gaussian_nb_f1	0.14	0.14
gaussian_process_f1	0.11	0
knn_f1	0.24	0
mlp_f1	0.28	0
qda_f1	0.14	0.01
random_forest_f1	0.13	0
svc_f1	0	0
xgboost_f1	0	0

Figura 23: Comparativa resultados consulta almacenada y ZSQL

El resultados es que se obtienen peores resultados con el nuevo enfoque. Buscando el motivo que pueda estar causando el problema, se identifica que el número de casos positivos obtenidos por la nueva consulta es muy inferior, tal y como se observa en la siguiente figura

```
dfca['class'].value_counts()
```

```
0    6466
1     215
Name: class, dtype: int64
```

```
dfzsql['class'].value_counts()
```

```
0    4666
1      71
Name: class, dtype: int64
```

Figura 24: Aumento en la diferencia de balanceo entre clases

Se comprueba que el problema es que se han dejado de considerar las peticiones de los fines de semana que es donde se concentran más casos positivos. De manera, se han desbalanceado más los casos negativos frente a los positivos:

dow	count(*)
1	183
2	121
3	130
4	118
5	143
6	131
7	198

Figura 25: Casos positivos por día de la semana (1 domingo, 7 sábado)

Por este motivo, se modifica la consulta ZSQL para volver a considerar los datos correspondientes a los días de fines de semanas. Se vuelve a ejecutar el modelo comprobando que ahora los datos si mejoran ligeramente respecto a los obtenidos en la ejecución con consulta almacenada, tal y como se observa en la siguiente figura:

evaluation_date	2022/04/21 11:17:24	2022/04/21 21:02:49
ada_boost_f1	0.29	0.50
decision_trees_f1	0.14	0.24
gaussian_nb_f1	0.14	0.10
gaussian_process_f1	0.11	0.00
knn_f1	0.24	0.40
mlp_f1	0.28	0.00
qda_f1	0.14	0.14
random_forest_f1	0.13	0.53
svc_f1	0.00	0.00
xgboost_f1	0.00	0.53

Figura 26: Evaluación con ZSQL y sin eliminar fines de semana

6.3.3. Problemas con la columna consultas sobre colectivos

Se produce un nuevo error (como consecuencia de ejecutar varias veces la consulta nueva) al agotar el espacio temporal para la conexión de la base de datos. Tras analizarlo con el equipo de Base de Datos, se identifica que el problema lo provoca la expresión utilizada para obtener el número de consultas de tipo colectivo en curso. Se decide eliminar esta variable del estudio, al no encontrar ninguna opción alternativa para obtener esta información. En su lugar se comienza a utilizar el número de consultas ejecutadas en cada periodo de tiempo.

6.3.4. Refinamiento en la selección de columnas

Se utiliza la opción *feature_importantes* (que provee el clasificador *RandomForestClassifier*[16] de Sklearn) para identificar las variables más significativas. En la siguiente tabla se muestran, en orden decreciente de importancia, las variables incluidas en la consulta ZSQL:

VARIABLE	IMPORTANCIA
TIEMPO_CONSULTA	0,129298
TIEMPO_BD	0,112463
TIEMPO_INFORME	0,074348

TAMANNO_INFORME	0,056175
TIEMPO_TRNS	0,042501
INFORME_DIFERIDO	0,039307
CONSULTAS_NUEVAS	0,034645
NUMERO_CONSULTAS	0,034111
INFORME_TOMCAT_3	0,034077
CONSULTA_PRINCIPAL	0,031819
NUMERO_INFORMES	0,030957
INFORME_HTML	0,028720
CONSULTA_COUNT	0,028100
INFORME_TOMCAT_4	0,027136
INFORMES_NUEVOS	0,022314
INFORME_TOMCAT_2	0,020794
CONSULTA_COMBINADA	0,020206
CONSULTA_OTRO_TIPO	0,019461
INFORME_TOMCAT_1	0,019258
CONSULTA_CT4CTE	0,018638
CONSULTA_CACHE_MOTOR	0,016718
CONSULTA_ZJ4COLECTI	0,015760
CONSULTA_ZJ4INPAD	0,014057
INFORME_PDF	0,011984
CONSULTA_ZJ4NCIH	0,011923
CONSULTA_CACHE_GENIO	0,011036
INFORME_SW	0,010916
INFORME_TOMCAT_OTRO	0,010338
INFORME_PDFCATALOG	0,010171
INFORME_GENIOJS	0,009355
CONSULTA_ZSQL	0,007431
INFORME_APP_SRV	0,007065
INFORMES_ERROR	0,006620
CONSULTA_COLECTIVO	0,006163
CONSULTA_EXPORTACION	0,004088
CONSULTA_ZJ4M200	0,003582
INFORME_RTF	0,003045
CONSULTAS_ERROR	0,002767
CONSULTA_ZJ4DEU	0,002668
CONSULTA_ZJ4ZICBA	0,002601
INFORME_JSON	0,002488
CONSULTA_ZJ4SEGZ4PE	0,002418
INFORME_XLS	0,001866
INFORME_APP_BIGDATA	0,000612
INFORME_APP_ANALYTICS	0,000000

A continuación, se muestra de manera gráfica la importancia de las variables mostrada en la tabla anterior:

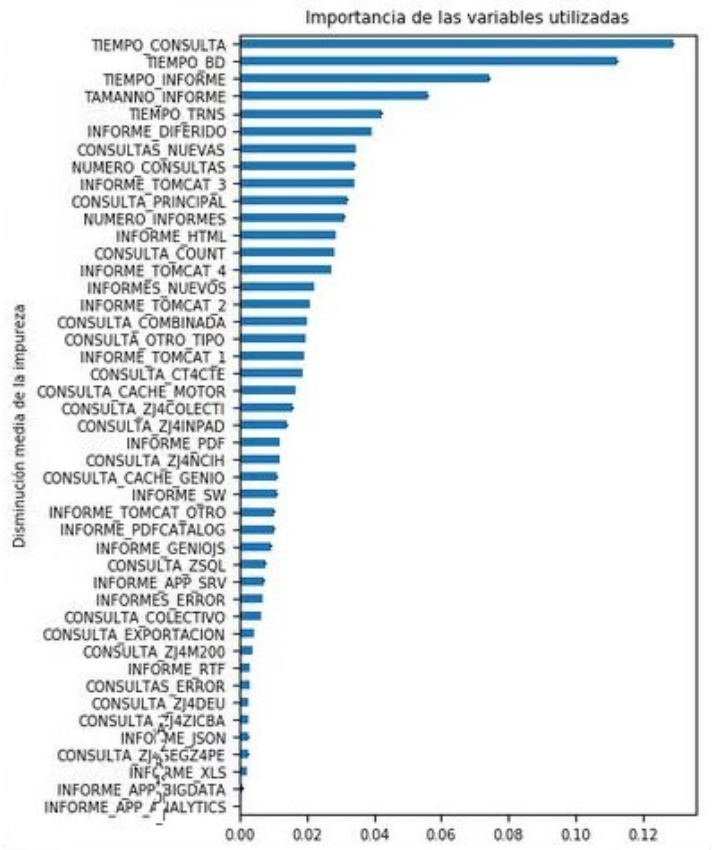


Figura 27: Importancia de las variables

Se realiza una evaluación de todos los modelos para analizar el comportamiento suprimiendo las variables con una importancia menor de 0,01 según la tabla anterior.

evaluation_date	2022/04/21 21:02:49	2022/04/24 09:29:01
ada_boost_f1	0.50	0.20
decision_trees_f1	0.24	0.17
gaussian_nb_f1	0.10	0.11
gaussian_process_f1	0.00	0.00
knn_f1	0.40	0.05
mip_f1	0.00	0.13
qda_f1	0.14	0.08
random_forest_f1	0.53	0.05
svc_f1	0.00	0.00
xgboost_f1	0.53	0.06

Figura 28: Evaluación sin considerar las variables menos significativas

Sin embargo, la evaluación de los modelos con este nuevo conjunto de datos empeora en general para todos los modelos. Se concluye, que

aunque no sean muy significativas, las variables aportan su significado a los modelos y que es por tanto preferible no eliminarlas. Por otro lado, el volumen de datos manejados no es muy grande, por lo cual el tamaño del conjunto de datos no necesita ser reducido.

Por todo lo anterior, se decide añadir nuevas variables al estudio que puedan enriquecer los resultados. Se identifican las siguiente variables que se incluirán en la consulta:

- INFORMES_INICIADOS que obtiene el número de informes que se han lanzado en el intervalo. Es decir el dato que se agrupaba la consulta anterior, ya que ahora todas las variables se obtienen para los informes en curso en cada intervalo.
- CONSULTAS_INICIADAS que obtiene el número de consultas que se han lanzado en el intervalo (de manera análoga a lo explicado para la columna anterior.
- INFORMES_FINALIZADOS que obtiene el número de informes que se han concluido en el intervalo de tiempo
- CONSULTAS_FINALIZADAS que obtiene el número de consultas que se han concluido en el intervalo de tiempo
- TAMANNO_INFORME que obtiene la media de los tamaños de los informes generados en cada minuto. Esta información ya había sido identificada en la fase 2 de entendimiento de los datos, pero no se había incluido por error.
- TIEMPO_BD que obtiene la media de los tiempos de las consultas generadas en cada minuto. Esta información ya había sido identificada en la fase 2 de entendimiento de los datos, pero no se había incluido por error.
- TIEMPO_MOTOR que obtiene la media de los tiempos empleados en el núcleo del MotorZújar para el procesamiento de las consultas generadas en cada minuto. El tiempo del MotorZújar incluye el tiempo de base de datos.
- TIEMPO_SW que obtiene la media de los tiempos empleados en el servicio web del MotorZújar para el procesamiento de las consultas generadas en cada minuto.
- TIEMPO_SERV que obtiene la media de los tiempos empleados por el MotorZújar (suma de tiempos empleados por su núcleo y por su servicio web) para el procesamiento de las consultas generadas en cada minuto.
- NUMERO_SESIONES que obtiene el número de sesiones distintas en utilizadas en cada intervalo de tiempo.
- Se añade una nueva columna llamada CONSULTA_NODO1_MOTOR que calcula el número de consultas procesadas en cada intervalo de tiempo por el nodo 1 del MotorZújar.
- CONSULTA_NODO2_MOTOR que calcula el número de consultas procesadas en cada intervalo de tiempo por el nodo 2 del MotorZújar
- INFORME_HIJO que obtiene el número de informes, lanzados a su vez por otros informes, en cada minuto de tiempo

- INFORME_100_PSD_GENERAL que obtiene el número de informes de este tipo generados en el intervalo de tiempo.
- INFORME_DISCREP_IRPF_2EXP que obtiene el número de informes de este tipo generados en cada minuto.
- INFORME_DETALLEINGRESOSV3 que obtiene el número de informes de este tipo generados en el intervalo de tiempo.
- INFORME_DETALLEGASTOSV3 que obtiene el número de informes de este tipo generados en cada minuto.
- INFORME_FICHAIVA que obtiene el número de informes de este tipo generados en el intervalo de tiempo.
- INFORME_INGRESOSYPAGOS que obtiene el número de informes de este tipo generados en cada minuto.
- INFORME_HERMESCALC que obtiene el número de informes de este tipo generados en cada minuto.
- INFORME_HERMESINDICE que obtiene el número de informes de este tipo generados en el intervalo de tiempo.
- INFORME_ONIFM190_SIMPLIF que obtiene el número de informes de este tipo generados en cada minuto.
- INFORME_VISUALIZACONSULTA que obtiene el número de informes de este tipo generados en el intervalo de tiempo.

Con estas últimas variables añadidas, los resultados mejoran de manera general para todos los modelos, aunque continúan sin ser suficientemente positivos:

evaluation_date	2022/04/24 09:29:01	2022/04/27 12:02:22
ada_boost_f1	0.20	0.27
decision_trees_f1	0.17	0.31
gaussian_nb_f1	0.11	0.13
gaussian_process_f1	0.00	0.05
knn_f1	0.05	0.27
mlp_f1	0.13	0.00
qda_f1	0.08	0.13
random_forest_f1	0.05	0.27
svc_f1	0.00	0.00
xgboost_f1	0.06	0.20

Figura 29: Evaluación tras añadir nuevas variables

6.3.5 Matriz de correlación[17]

Se sospecha que la relación entre la variable clase (obtenida con los tiempo de respuesta del informe control) y el resto de variables usadas en el estudio es muy baja. Con ayuda de la matriz de correlación se confirma esta hipótesis. Se observa en la siguiente figura que la relación del atributo *class* con el resto de variables es muy baja, siendo la gama de colores es casi negra:

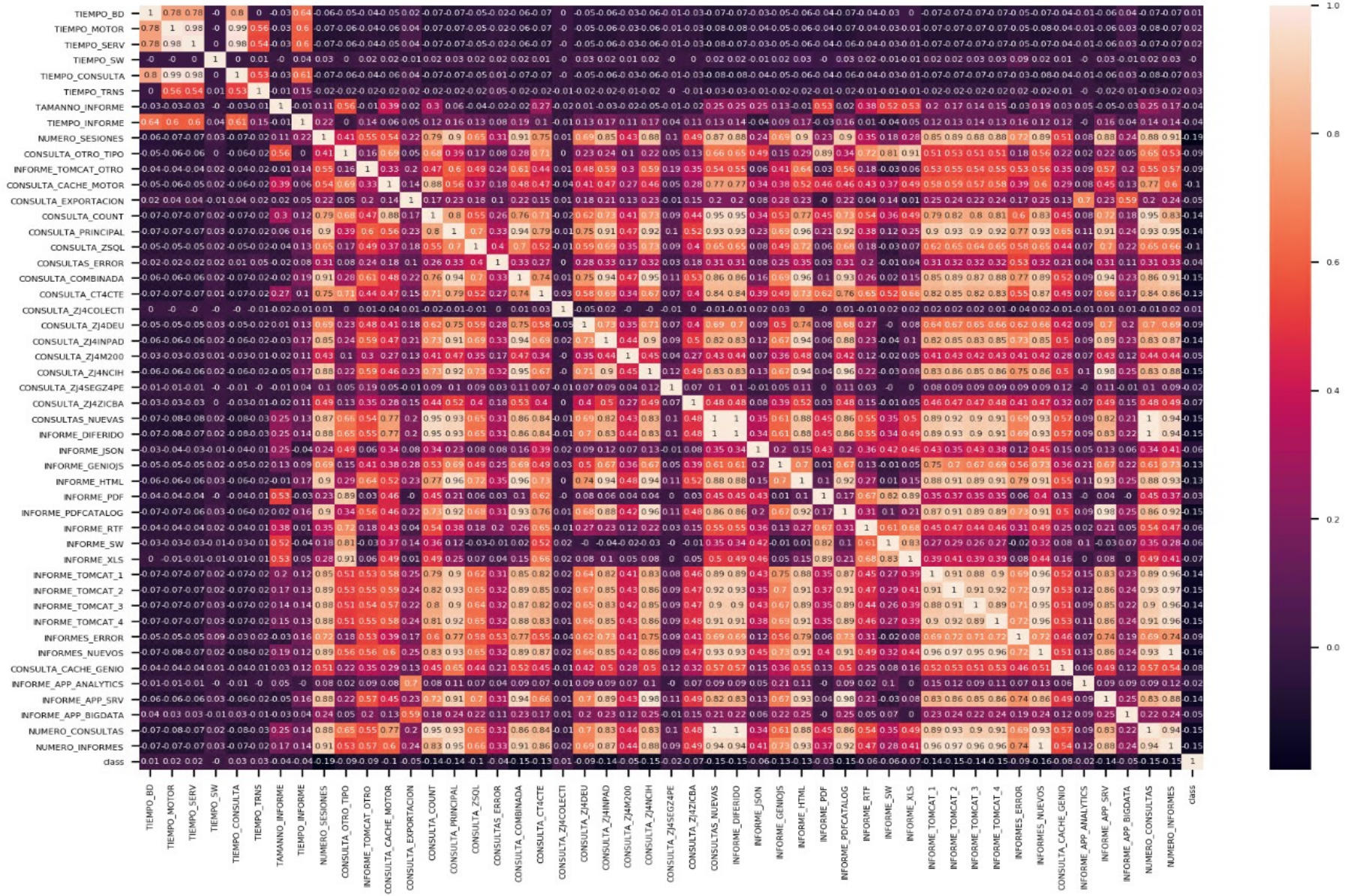


Figura 30: Matriz de correlación

Consecuentemente, se plantea la opción de buscar nuevas fuentes de datos, alternativas al *ping* del informe genio, para localizar los casos positivos.

6.3.6. Normalización de datos

Se evalúa la distribución de los datos de las columnas consideradas por si es necesario aplicar alguna transformación sobre ellos que pudiera hacer que los resultados de las clasificaciones de los modelos mejoren.

Para ello se utiliza el método *normalizar* de la clase *Analisis* desarrollado por el grupo de Analytics de la AEAT, cuyo resultado indica que no se obtienen mejoras al aplicar alguna de las transformaciones consideradas (*lambert*, *log*, *sqrt*, *cbrt*, *asinh*, *boxcox*, *yeo-johnson*, *ordnorm*). Por tal motivo, no se aplicarán transformaciones sobre los datos de ninguna de las variables utilizadas.

Se incluye el resultado completo de ejecutar el método anterior en el anexo "11.1. Resultado de la normalización de las variables del estudio" de la presente memoria.

6.3.7. Solución al problema de falta de balanceo

Para resolver el problema de balanceo de datos se realizan las siguientes aproximaciones.

En primer lugar se realiza un reajuste en las condiciones para clasificar los registros de entrenamiento como positivos y negativos. Concretamente, se pasa a considerar casos negativos aquellas peticiones del informe que tardan menos de un segundo, se consideran casos positivos los correspondientes a aquellas peticiones que tardan más de 3 segundos y se eliminan los casos intermedios (con la intención de eliminar los casos que dificulten la clasificación al ser casos cerca de la frontera).

Se ejecutan los modelos con este nuevo enfoque y se obtienen los siguientes resultados:

evaluation_date	2022/04/27 12:02:22	2022/04/28 13:22:05
ada_boost_f1	0.27	0.30
decision_trees_f1	0.31	0.31
gaussian_nb_f1	0.13	0.19
gaussian_process_f1	0.05	0.03
knn_f1	0.27	0.27
mlp_f1	0.00	0.02
qda_f1	0.13	0.21
random_forest_f1	0.27	0.27
svc_f1	0.00	0.00
xgboost_f1	0.20	0.27

Figura 31: Evaluación tras cambiar condiciones para clasificar los casos

Los resultados mejoran sensiblemente, por lo tanto se mantendrán estos cambios.

En segundo lugar, se utiliza otra técnica para solucionar el problema de balanceo que consiste en eliminar casos negativos, haciendo coincidir la cantidad de individuos de ambas clases. Se realizan los cambios necesarios para realizar este equilibrado de casos mediante el desarrollo de la función *equilibrate_df_droping_negatives*. En la siguiente figura se muestran los resultados obtenidos tras ejecutar de nuevo la clasificación de todos los modelos con este nuevo enfoque:

evaluation_date	2022/05/04 18:00:58
ada_boost_f1	0.74
decision_trees_f1	0.72
gaussian_nb_f1	0.65
gaussian_process_f1	0.00
knn_f1	0.71
mlp_f1	0.66
qda_f1	0.68
random_forest_f1	0.76
svc_f1	0.75
xgboost_f1	0.77

Figura 32: Evaluación equilibrando casos eliminando negativos

Como se puede apreciar, los datos obtenidos son muy buenos y de hecho se supera el límite de 0,7 situado como objetivo para la métrica F1 para 6 modelos distintos.

Sin embargo, se descubre al final del proyecto, que se ha cometido un error de concepto en el desarrollo del método *equilibrate_df_droping_negatives* que provocaba que los datos fueran tan positivos. Concretamente se estaban eliminando casos positivos del toda la muestra de manera aleatoria cuando en realidad solo se debía eliminar casos del conjunto de entrenamiento pero no del conjunto de pruebas.

En tercer lugar, y a pesar de haber obtenido buenos resultado con el enfoque anterior, se prueba la opción de asignar pesos a las clases en el momento de entrenar los modelos, de manera que estos consideren más importantes los aciertos sobre unas determinadas clases. Para hacerlo, se revisan uno a uno los distintos algoritmos implementados, puesto que esta opción se configura de manera específica en cada caso (de hecho, para determinados modelos no existe esta opción).

Se evalúan de nuevo los modelos, pero ahora utilizando esta nueva opción de balancear los pesos de las clases y se obtienen los siguientes resultados. Se muestran junto con los resultados obtenidos con la opción sin balancear para el mismo intervalo de fechas para poder compararlos:

evaluation_date	2022/05/07 11:44:24	2022/05/09 17:24:53
ada_boost_f1	0.28	0.39
comment	ZSQL ampliando fechas y sin eliminar casos (mo...	Establece peso a las clases para paliar el pro...
decision_trees_f1	0.24	0.3
gaussian_nb_f1	0.24	0.23
gaussian_process_f1	0.08	0.08
knn_f1	0.27	0.27
mip_f1	0.19	0.19
qda_f1	0.28	0.28
random_forest_f1	0.27	0.17
svc_f1	0	0.29
xgboost_f1	0.3	0.43

Figura 33: Evaluación con la opción de balanceo de clases

Como se puede observar, los resultados de los modelos en general mejoran (en AdaBoost se pasa de F1 igual 0,28 a 0,39, en en árboles de decisión de 0,24 a 3, en SVC de 0 a 0,29 y en XGBoost de 0,3 a 0,43) a excepción de Random Forest y GaussianNB que empeoran. Hay que

considerar que en determinados modelos no se ha aplicado ningún cambio porque no admiten opción de establecer pesos a las clases.

6.3.8. Obtención de nuevos casos de crisis

A tenor de lo observado en la matriz de correlación presentada en el apartado "6.3.5 Matriz de correlación", se pretende mejorar la identificación de los casos positivos, pues se desprende que el tiempo de respuesta del informe *ping* de Genio no es un buen indicador del estado del sistema.

Para hacerlo, se considera la opción de utilizar de manera complementaria las alertas que envía el sistema Zabbix avisando de los problemas en los distintos servicios. Se identifica que esta información está disponible en la tabla *alerts* de la base de datos de Zabbix. Se elabora una consulta SQL que obtiene las alertas enviadas a los grupos MotorZújar y Genio y se modifica el método *get_class_results_df* para que considere positivos aquellos momentos en los que el informe *ping* de Genio ha tardado más de 3 segundos o en los que se ha recibido alguna alerta de Zabbix.

En la siguiente figura se muestran los resultados obtenidos tras evaluar los modelos con este nuevo planteamiento:

evaluation_date	2022/05/10 09:00:06	2022/05/10 16:57:13
ada_boost_f1	0.29	0.34
decision_trees_f1	0.28	0.33
gaussian_nb_f1	0.24	0.25
gaussian_process_f1	0.08	0.04
knn_f1	0.27	0.35
mlp_f1	0.28	0
qda_f1	0.27	0.31
random_forest_f1	0.22	0.33
svc_f1	0	0.02
xgboost_f1	0.34	0.38
class_distribution	[10464, 670]	[10338, 819]

Figura 34: Evaluación considerando las alertas de Zabbix

La métrica F1 ha mejorado de manera generalizada (a excepción de MLP, Gaussian Process y SVC) para todos los modelos. Como se puede observar en la figura anterior, se ha incluido un nuevo atributo en los registros históricos con la distribución de individuos en cada clase

Se hace una nueva prueba pero balanceando los pesos de las clases obteniéndose los siguientes resultados:

evaluation_date	2022/05/10 16:57:13	2022/05/10 17:06:27
ada_boost_f1	0.34	0.4
decision_trees_f1	0.33	0.28
gaussian_nb_f1	0.25	0.25
gaussian_process_f1	0.04	0.04
knn_f1	0.35	0.35
mlp_f1	0	0
qda_f1	0.31	0.31
random_forest_f1	0.33	0.3
svc_f1	0.02	0.29
xgboost_f1	0.38	0.49
class_distribution	[10338, 819]	[10338, 819]

Figura 35: Evaluación considerando las alertas y balanceando casos

Como sucedía anteriormente, los resultados son muy variables y algunos modelos empeoran y otros mejoran. En particular se puede apreciar que XGBoost mejora sustancialmente alcanzando 0,49 en el indicador F1.

6.3.9. Mejora en la selección de parámetros de los modelos

Se utiliza GridSearchCV de Sklearn[18][19] para seleccionar los parámetros que optimicen los resultados de la clasificación de los modelos ejecutados. Con el algoritmo de GridSearchCV se consigue identificar la mejor parametrización (para la métrica indicada, en este caso F1) entre todas las combinaciones posibles para los valores de los parámetros indicados.

Además, con GridSearchCV se realiza *cross validation* con KFold[20], lo que permite realizar varias evaluaciones con un mismo set de datos, seleccionando en cada evaluación conjuntos (o pliegues) de entrenamiento y pruebas distintos.

Las ejecuciones de GridSearchCV pueden ser procesos muy pesados, sobre todo para ciertos modelos en los que se desea probar con muchas combinaciones de valores de parámetros distintos y para un número significativos de pliegues. Por ejemplo, para el modelo KNN se han valorado las siguientes opciones para los parámetros:

- *n_neighbors*: 2, 3, 4, 5, 6, 7, 8, 9, 10

- *weight*: "uniform", "distance"
- *leaf_size*: 15, 20, 25, 35

Se ha ejecutado *cross validation* con 5 pliegues con lo que se han realizado un total de 360 evaluaciones. La ejecución GridSearchCV para determinados modelos ha supuesto varias horas, e incluso días completos. A continuación, se muestra un resumen con los resultados de los distintos modelos tras optimizar los valores de los parámetro con GriSearchCV:

```
******************************
* K Nearest Neighbours (KNN) *
******************************
 Precision  Recall    F1  Accuracy  Support
      0.32    0.24  0.28       0.9     1223

******************
* DECISION TREES *
******************
 Precision  Recall    F1  Accuracy  Support
      0.27    0.64  0.38      0.83     1223

******************
* RANDOM FOREST *
******************
 Precision  Recall    F1  Accuracy  Support
       0.3    0.65  0.41      0.86     1223

***********
* XGBOOST *
***********
 Precision  Recall    F1  Accuracy  Support
      0.35    0.48  0.41      0.89     1223


*********************************
* Support vector machines (SVC) *
*********************************
 Precision  Recall    F1  Accuracy  Support
      0.16    0.72  0.26      0.68     1223

*******
* MLP *
*******
 Precision  Recall   F1  Accuracy  Support
       0.3    0.29  0.3      0.89     1223

*********************
* GAUSSIAN PROCESS *
*********************
 Precision  Recall    F1  Accuracy  Support
       0.5    0.11  0.19      0.92     1223


*************
* ADA BOOST *
*************
 Precision  Recall    F1  Accuracy  Support
      0.26    0.69  0.37      0.82     1223

************************
* GAUSSIAN NAIVE BAYES *
************************
 Precision  Recall    F1  Accuracy  Support
      0.13     0.6  0.21      0.64     1223

***********************************
* QUADRATIC DISCRIMINANT ANALYSIS *
***********************************
 Precision  Recall    F1  Accuracy  Support
      0.16    0.61  0.25      0.71     1223
```

Figura 36: Evaluación tras aplicar GridSearchCV

Se ha conseguido cierta mejoría en los resultados del indicador F1 de los distintos clasificadores, pero estos siguen sin acercarse a los valores establecidos en los objetivos (0,7).

Es importante mencionar que estos últimos datos son ligeramente peores que los obtenidos anteriormente. Esto es debido a que se ha realizado entre medias un cambio en la selección de los casos de entrenamiento y los casos de prueba: se ha comenzado a utilizar el parámetro *stratify* de la función *train_test_split[21]*, para conseguir que la proporción de casos negativos y positivos seleccionados en cada conjunto esté equilibrado. Con este cambio se obtienen peores resultados, sin embargo, es importante seleccionar de esta manera los conjuntos para que los resultados sean más realistas.

6.3.10. Resumen

Se muestra a continuación una tabla con el resumen los resultados de la métrica F1 obtenidos en las clasificaciones realizadas durante la fase de evaluación de los modelos:

	ada boost	**dec. trees**	**gaus. nb**	**gauss. proc.**	**knn**	**mlp**	**qda**	**rnd. forest**	**svc**	**xg boost**
21/04 11:17	0,29	0,14	0,14	0,11	0,24	0,28	0,14	0,13	0,00	0,00
21/04 12:35	0,25	0,00	0,14	0,00	0,00	0,00	0,01	0,00	0,00	0,00
21/04 13:00	0,24	0,24	0,10	0,20	0,16	0,25	0,12	0,11	0,00	0,06
21/04 20:45	0,14	0,05	0,09	0,12	0,17	0,00	0,08	0,19	0,00	0,00
21/04 21:02	0,50	0,24	0,10	0,00	0,40	0,00	0,14	0,53	0,00	0,53
24/04 09:29	0,20	0,17	0,11	0,00	0,05	0,13	0,08	0,05	0,00	0,06
24/04 09:49	0,15	0,20	0,10	0,00	0,13	0,00	0,11	0,06	0,00	0,06
24/04 10:09	0,15	0,17	0,10	0,00	0,13	0,00	0,11	0,06	0,00	0,06
24/04 10:12	0,21	0,26	0,26	0,00	0,22	0,00	0,32	0,24	0,00	0,29
26/04 16:34	0,24	0,20	0,10	0,20	0,16	0,25	0,12	0,11	0,00	0,06
27/04 12:02	0,27	0,31	0,13	0,05	0,27	0,00	0,13	0,27	0,00	0,20
27/04 16:35	0,24	0,30	0,13	0,05	0,27	0,24	0,13	0,23	0,00	0,16
28/04 08:57	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28/04 13:22	0,30	0,31	0,19	0,03	0,27	0,02	0,21	0,27	0,00	0,27
28/04 15:39	0,37	0,38	0,17	0,13	0,23	0,08	0,13	0,38	0,00	0,27
30/04 12:50	0,34	0,23	0,17	0,07	0,20	0,00	0,19	0,26	0,00	0,33
01/05 11:06	0,34	0,18	0,17	0,07	0,20	0,00	0,19	0,26	0,00	0,33
01/05 12:13	0,32	0,25	0,17	0,07	0,22	0,00	0,19	0,34	0,00	0,33
02/05 10:04	0,38	0,29	0,17	0,07	0,25	0,00	0,20	0,31	0,00	0,33
04/05 16:55	0,22	0,26	0,18	0,09	0,17	0,06	0,19	0,22	0,00	0,16
04/05 17:22	0,29	0,29	0,17	0,04	0,24	0,00	0,20	0,19	0,00	0,29
04/05 18:00	*0,74*	*0,72*	*0,65*	*0,00*	*0,71*	*0,66*	*0,68*	*0,76*	*0,75*	*0,77*

	ada boost	dec. trees	gaus. nb	gauss. proc.	knn	mlp	qda	rnd. forest	svc	xg boost
07/05 11:44	0,28	0,24	0,24	0,08	0,27	0,19	0,28	0,27	0,00	0,30
09/05 17:24	0,39	0,30	0,23	0,08	0,27	0,19	0,28	0,17	0,29	0,43
10/05 09:00	0,29	0,28	0,24	0,08	0,27	0,28	0,27	0,22	0,00	0,34
10/05 16:57	0,34	0,33	0,25	0,04	0,35	0,00	0,31	0,33	0,02	0,38
10/05 17:06	0,40	0,28	0,25	0,04	0,35	0,00	0,31	0,30	0,29	0,49
11/05 13:08	0,34	0,36	0,24	0,00	0,31	0,37	0,30	0,32	0,05	0,35
11/05 15:52	0,21	0,27	0,21	0,00	0,13	0,16	0,25	0,26	0,00	0,25
11/05 17:09	0,33	0,29	0,22	0,02	0,23	0,30	0,27	0,35	0,02	0,31
14/05 11:51	0,27	0,25	0,21	0,04	0,23	0,20	0,16	0,28	0,00	0,33
22/05 19:08	0,37	0,38	0,21	0,02	0,28	0,30	0,25	0,41	0,26	0,41
23/05 00:05	0,38	0,34	0,20	0,30	0,23	0,00	0,14	0,41	0,27	0,40
23/05 13:53	*0,79*	*0,79*	*0,61*	*0,19*	*0,68*	*0,66*	*0,36*	*0,80*	*0,74*	*0,79*
24/05 16:52	*0,79*	*0,74*	*0,68*	*0,11*	*0,63*	*0,76*	*0,11*	*0,78*	*0,71*	*0,82*
24/05 17:03	*0,50*	*0,80*	*0,00*	*0,80*	*0,40*	*0,33*	*0,00*	*1,00*	*0,67*	*0,44*
27/05 12:28	*0,78*	*0,75*	*0,67*	*0,02*	*0,64*	*0,39*	*0,36*	*0,78*	*0,76*	*0,79*
27/05 12:28	0,09	0,10	0,00	0,00	0,00	0,02	0,00	0,00	0,00	0,06
27/05 12:57	0,40	0,41	0,22	0,15	0,32	0,28	0,09	0,37	0,22	0,39
27/05 13:01	0,39	0,34	0,22	0,57	0,41	0,30	0,00	0,39	0,21	0,42
31/05 08:42	0,35	0,36	0,22	0,22	0,22	0,20	0,18	0,39	0,26	0,38
31/05 08:42	0,35	0,36	0,22	0,22	0,22	0,20	0,18	0,39	0,26	0,38

Es importante destacar, que los resultados de las filas cuyos textos están en gris y cursiva corresponden a evaluaciones inválidas tal y como se ponía de manifiesto en el apartado anterior "6.3.7. Solución al problema de falta de balanceo", al haberse aplicado la técnica de eliminar casos negativos erróneamente.

Junto con esta memoria se entrega un archivo CSV con el histórico de resultados completo en el que se incluyen el resto de métricas, así como el intervalo de fechas para el que se realizó la evaluación y la distribución de las clases en el conjunto de datos para el que fue aplicado.

7. Despliegue

Tal y como se especificó en el plan preliminar del proyecto, la fase de despliegue para este TFG ha sido dedicada a la elaboración de esta memoria. La integración del desarrollo realizado en algún software que facilite su explotación y su posible posterior despliegue y mantenimiento quedan en manos de la AEAT.

Sin embargo, debido a los problemas encontrados para conseguir que los modelos de clasificación obtuvieran los mínimos de calidad esperada, la implementación de los siguientes métodos ha sido realizada durante esta fase al no haberse podido realizar con antelación:

- Método de evaluación para una determinada fecha
- Método que obtiene los detalles de los casos positivos

7.1. Evaluación para una determinada fecha

Se modifica el método *eval_all_models*, así como los métodos individuales que realizan la evaluación de cada uno de los modelos (por ejemplo, *model_randon_forest*) para que admitan un parámetro opcional (llamado *target_date*) de manera que, si se informa este parámetro, la selección del conjunto de pruebas toma los registros correspondientes a la fecha indicada y por tanto se realiza la clasificación entrenando con los registros correspondientes al resto de días.

Con el desarrollo de esta funcionalidad, se cubre el requisito del cliente de poder realizar la clasificación para un fragmento de tiempo determinado. Este requisito fue definido durante el plan de trabajo inicial y confirmado durante el plan preliminar.

Pueden encontrarse más detalles del funcionamiento de este método en la guía de uso entregada junto a esta memoria.

7.2. Obtención de los detalles de los casos positivos

Para cubrir el requisito de informar de las causas que han llevado a clasificar un determinado registro como positivo (crisis), se modifica el método *model_decision_tree* para que admita un parámetro opcional llamado *verbose*. Si se informa este parámetro, el algoritmo además de realizar la clasificación, muestra por pantalla los motivos (decisiones del árboles de decisión) que han provocado que se considere cada uno de los momentos clasificados como crisis. En la siguiente figura se muestra un ejemplo del resultado de la ejecución del método anterior en el que se presentan los motivos por los cuales se han clasificado dos registros como crisis:

```
Se analiza el positivo de la fila 206 (2022-03-27 08:51:00):
Decisión nodo 0 : (X_test[206, CONSULTAS_FINALIZADAS] = 7) <= 101.5)
Decisión nodo 1 : (X_test[206, TIEMPO_BD] = 305.1333333333333) <= 435.5416717529297)
Decisión nodo 2 : (X_test[206, TIEMPO_MOTOR] = 347.4666666666667) > 346.5598449707031)
Decisión nodo 6 : (X_test[206, CONSULTA_COMBINADA] = 0.0) <= 0.5)

Se analiza el positivo de la fila 207 (2022-04-16 09:39:00):
Decisión nodo 0 : (X_test[207, CONSULTAS_FINALIZADAS] = 36) <= 101.5)
Decisión nodo 1 : (X_test[207, TIEMPO_BD] = 913.1071428571444) > 435.5416717529297)
Decisión nodo 9 : (X_test[207, TIEMPO_BD] = 913.1071428571444) > 791.8948059082031)
Decisión nodo 13 : (X_test[207, INFORME_APP_BIGDATA] = 0) <= 0.5)
```

Figura 37: Explicación de casos positivos

Como se puede observar, el registro correspondiente a las 08:51 del día 27 de marzo del 2022 fue catalogado como crisis porque se cumplieron las 4 condiciones que se muestran.

Para satisfacer este requisito, se ha decidido utilizar la potencia de los árboles de decisión los cuales ofrecen explicabilidad de resultados, concretamente las opciones que ofrece el método *decision_path* del clasificador DecissionTreeClassifier[22].

7.3. Detalle de los métodos implementados

A continuación, se describe el funcionamiento de los métodos incluidos en la librería desarrollada.

- **eval_all_models**: método principal que realiza la evaluación de todos los modelos a partir de un *dataframe* obtenido anteriormente con alguno de los métodos para la descarga del conjunto de datos para el análisis (*get_prepared_dataframe_from_zsql* o *get_prepared_dataframe_from_ca*). El resultado se muestra por pantalla, se registra en el fichero CSV histórico de resultados. En último lugar se devuelve a la llamada un *dataframe* con la clase de cada elemento del conjunto de *test* junto con la predicción de cada uno de los modelos. Si se indica el parámetro opcional *target_date* el *split* de entrenamiento y pruebas se realiza considerando los registros de la fecha indicada como casos de test y el resto de entrenamiento (si no se manera aleatoria).
- ***get_prepared_dataframe_from_ca***: método que obtiene un *dataframe* con los datos para el intervalo de fechas indicado en los parámetros *date_from* y *date_to* a partir de una consulta almacenada creada en el ClienteZújar. Este fue el primer enfoque para obtener el conjunto de datos de estudio pero fue sustituido por el desarrollado en método que se detalla a continuación tal y como se ha descrito previamente en este documento. Por rendimiento, este método cachea los datos, de manera que si ya se ha ejecutado previamente para los mismos parámetros los resultados se devuelven directamente de esta caché sin necesidad de consultar al MotorZújar.

- ***get_prepared_dataframe_from_zsql***: método que obtiene un *dataframe* para el intervalo de fechas indicado mediante los parámetros *date_from* y *date_to* a partir de una consulta de tipo ZSQL, para la cual ha sido necesario especificar su código el cual se recoge en el método *get_zsql_from_dates* detallado más adelante. Por rendimiento, este método cachea los datos, de manera que si ya se ha ejecutado previamente para los mismos parámetros los resultados se devuelven directamente de esta caché sin necesidad de consultar al MotorZújar.
- ***process_new_result***: método auxiliar de uso interno por el método *eval_all_models* para procesar los resultados (predicciones) realizadas por los modelos. En primer lugar, añade la predicción realizada por el modelo en cuestión al *dataframe* con las predicciones de todos los modelos que se devolverá como resultado. En segundo lugar muestra en pantalla las métricas obtenidas por el modelo (usando el resultado del método *get_metrics*). Y por último añade estas métricas al registro que se está componiendo con las métricas de todos los modelos para escribirse escribirse finalmente en el histórico de evaluaciones en formato CSV.
- ***my_train_test_split***: método que parte horizontalmente el conjunto de datos en conjunto de entrenamiento y conjunto de pruebas. Y a su vez, cada uno de estos conjuntos los parte verticalmente en variables con los atributos y variable con el resultado o clase. El método admite el parámetro opcional *target_date* de manera que si se omite, la partición inicial se hace aleatoriamente asignando el 90% de los datos al conjunto de entrenamiento y el 10% restante al conjunto de pruebas (utiliza la función *train_test_split* de Sklearn para hacerlo). Si se indica el parámetro *target_date* se asignan los registros correspondientes a la fecha indicada al conjunto de pruebas y el restante al conjunto de entrenamiento.
- ***get_df_results***: método que obtiene un *dataframe* con los resultados almacenados el el fichero CSV con el histórico de las ejecuciones realizadas.
- ***get_df_main_models_results***: igual que el método anterior pero solo obtiene la información referente a los modelos considerados más importantes en este estudio: Árboles de decisión, Random Forest, XGBoost y AdaBoost.
- ***get_community_operations_df***: método de uso interno que se ocupa de obtener un *dataframe* con el número de consultas de tipo colectivo iniciadas por cada minuto en el intervalo de tiempo indicado mediante los parámetros *date_from* y *date_to*. Esta información ha sido necesaria obtener en una consulta separada por problemas rendimientos, como ha sido descrito anteriormente en este documento. Por rendimiento, este método cachea los datos, de manera que si ya se ha ejecutado previamente para los mismos parámetros los resultados se devuelven directamente de esta caché sin necesidad de consultar al MotorZújar.

- ***make_commons_ca_df_updates***: método de uso interno que se ocupa de hacer un tratamiento sistemático que es necesario realizar sobre los *dataframe* obtenidos a partir de una consulta almacenada Zújar (se desarrolla este método únicamente para no tener que duplicar código).
- ***merge_df_data_and_results***: método de uso interno que se ocupa de unir los *dataframe* parciales utilizados para obtener los datos de actividad, los datos de consultas de colectivos y los datos de resultados (atributo *class*)
- ***get_zsql_from_dates***: método de uso interno que se ocupa de obtener la ZSQL necesaria para consultar los datos del Zújar de actividad agrupados por minuto para el intervalo de fechas indicado mediante los parámetros *date_from* y *date_to*.
- ***get_zabbix_alerts_df**:* método de uso interno que obtiene un *dataframe* con las alertas de Zabbix enviadas en el intervalo de fechas indicadas a un miembro del grupo Genio. Para obtener los datos, estos se consultan directamente a la base de datos Zabbix. Por rendimiento, este método cachea los datos, de manera que si ya se ha ejecutado previamente para los mismos parámetros los resultados se devuelven directamente de esta caché sin necesidad de consultar al MotorZújar.
- ***get_class_results_df***: método de uso interno que obtiene un *dataframe* con los resultados para cada minuto pertenecientes al intervalo de fechas indicado mediante los parámetros *date_from* y *date_to*. Este es el método que se usa finalmente el cual combina las consultas SQL que se obtiene mediante los dos siguientes métodos, lo cuales obtienen los valores de clase de los *pings* de *Zabbix* y de las alertas de Zabbix respectivamente.
- ***get_pings_sql_from_dates***: método de uso interno que obtiene la SQL necesaria para obtener la información de los *pings* de Zabbix al informe de control Genio el cual se envía cada 3 minutos. La SQL está filtrada por las fechas indicadas mediante los parámetros *date_from* y *date_to*.
- ***get_alerts_sql_from_dates***: método de uso interno que obtiene la SQL necesaria para obtener la información de las alertas de Zabbix enviadas al grupo Genio cuando se produce algún problema relacionado. La SQL está filtrada por las fechas indicadas mediante los parámetros *date_from* y *date_to*.
- ***print_header***: método auxiliar de uso interno utilizado desde todos los métodos de modelado para imprimir en pantalla una cabecera (texto entre asteriscos) cuando se evalúa un modelo.
- ***model_knn***: método utilizado para modelar con KNN.
- ***model_decision_trees***: método utilizado para modelar con árboles de decisión.
- ***decision_trees_explain_value***: método auxiliar utilizado desde *model_decision_trees* para mostrar en pantalla los motivos para clasificar un elemento como positivo o como negativo según el árbol generado durante el entrenamiento.
- **model_random_forest**: método utilizado para modelar con Random Forest.

- ***print_feature_importances***: método que muestra la importancia de cada una de las variables del modelo de clasificación Random Forest. Asimismo, muestra una gráfica ordenando estas variables en orden creciente de importancia.
- ***model_xgboost***: método utilizado para modelar con XGBoost.
- ***model_ada_boost***: método utilizado para modelar con AdaBoost.
- ***model_svc***: método utilizado para modelar con SCV.
- ***model_mlp***: método utilizado para modelar con MLP.
- ***model_gaussian_process***: método utilizado para modelar con Gaussian Process.
- ***model_gaussian_nb***: método utilizado para modelar con Gaussian Naive Bayes.
- ***model_quadratic_discriminant_analysis***: método utilizado para modelar con Quadratic Discriminant Analysis.
- ***get_samples_weight***: método de uso interno que obtiene para cada muestra del conjunto de entrenamiento recibido como parámetro el peso de la misma. Usado para ponderar la importancia de las clases de manera inversa al número de ocurrencias de cada una de ellas. Se utiliza para hacerlo la siguiente fórmula: muestras negativas / muestras positivas.
- ***eval_models_from_zsql***: método que evalúa todos los modelos contemplados a partir de ZSQL para el intervalo de fechas indicado. Utiliza el parámetro *message* para añadir un texto indicativo de la ejecución al fichero histórico de ejecuciones.
- ***eval_models_from_ca***: método equivalente al anterior pero usando los datos de actividad obtenidos desde una consulta almacenada en lugar de hacerlo desde la ZSQL. Este método está en desuso ya que se encontraron bastantes deficiencias que fueron solucionadas en el método anterior.
- ***decision_trees_explain_value***: método de uso interno (desde *model_decision_trees*) para explicar las decisiones en el árbol para elemento clasificado. Código basado en el ejemplo de: https://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html#sphx-glr-auto-examples-tree-plot-unveil-tree-structure-py.
- **equilibrate_df_droping_negatives**: método auxiliar utilizado devolver el *dataframe* de entrada pero eliminando registros de clase negativa de manera aleatoria hasta disponer en el *dataframe* del mismo número de individuos de ambas clases. Si se indica una fecha se equilibra únicamente el conjunto de entrenamiento y el conjunto de test (correspondiente a los registros de esa fecha) se mantienen todos.

Es importante mencionar, que junto a esta memoria, se entrega una guía de uso con ejemplo los métodos anteriores.

8. Conclusiones

Las determinación del estado del sistema Zújar a partir de los datos de actividad (generados por las aplicaciones involucradas) para concluir si este está o no en un momento de crisis, no ha podido realizarse con la precisión fijada como objetivo inicial.

Para intentar lograr este objetivo, se han utilizado diez algoritmos de clasificación distintos y se han aplicado diversas técnicas para intentar solucionar los problemas identificados tras realizar las primeras evaluaciones de los modelos.

El principal motivo que ha provocado que los resultados no hayan sido positivos ha sido la incapacidad para identificar y catalogar adecuadamente los momentos de crisis, hecho que ha podido comprobarse mediante la matriz de correlación entre el atributo clase y el resto de variables.

8.1. Logro de objetivos

Se revisa a continuación si se han cumplido o no los objetivos planteados durante el plan preliminar del trabajo:

1. Identificar momentos de crisis en el sistema. Como se mencionaba al principio de este capítulo, **este objetivo no se ha cumplido,** ya que esperaba obtener al menos un 0,7 en cada una de las métricas consideradas (*precision*, *recall*, F1 y *accuracy*) y los mejores resultados obtenidos son aproximadamente de 0,4.

2. Identificar las causas de las crisis detectadas. **Este objetivo se ha cumplido utilizando** la explicabilidad que ofrece el método de clasificación de árboles de decisión. En la situación actual esta funcionalidad no tiene mucho valor porque hay poca certeza de identificar correctamente los casos de crisis, pero cuando se mejore este aspecto, esta funcionalidad será realmente útil.

3. Posibilidad de funcionamiento *on-line*. Mediante este objetivo se pretende que el sistema desarrollado sea compatible para poder ejecutarse con registros de actividad *on-line*. **Este objetivo se ha cumplido** puesto que la arquitectura se ha diseñado para tal fin y con cambiar únicamente la consulta para que obtenga los datos del zújar *on-line* en lugar del zújar histórico es suficiente.

4. Herramienta ejecutable. **Este objetivo se ha cumplido** desarrollando la librería "aeat_activity_machine_learning.py" con el lenguaje de programación Python (debidamente comentada) con la cual se pueden realizar las operaciones necesarias para obtener los datos, prepararlos y realizar las clasificaciones de los distintos modelos. Junto con esta memoria, se aporta una guía de uso de la librería desarrollada.

8.2. Seguimiento de planificación y metodología

La planificación realizada al inicio del proyecto ha resultado ser adecuada y se ha podido seguir casi a la perfección. Únicamente ha existido una desviación en la fase de evaluación del proyecto, debido a los problemas para conseguir que los modelos obtuvieran los resultados esperados. Como consecuencia, el desarrollo de las funcionalidades siguientes ha tenido que ser asumido en la fase de despliegue:

- Evaluación para una determinada fecha
- Obtención de los detalles de los casos positivos

En cuanto a la metodología utilizada en el proyecto (CRISP-DM) ha resultado ser un acierto. Se seleccionó esta metodología por estar orientada para proyectos de este tipo y se ha podido comprobar, que con ella se han contemplado las fases requeridas del proyecto y que ofrece una flexibilidad para permitir transiciones entre fases que son muy necesarias, como por ejemplo entre las fases de preparación de los datos y de modelado.

8.3. Líneas de trabajo futuro

Debido a su complejidad, este TFG se planteaba originalmente como una prueba piloto para identificar las crisis de la AEAT. En esta línea, se considera que con este trabajo se han establecido las bases para conseguir este objetivo a largo plazo. Los siguientes pasos del proyecto, ya fuera del ámbito de este proyecto deberían ser los siguientes:

- Identificar nuevas formas de catalogar los momentos de crisis, pues se ha puesto de manifiesto que las consideradas en este proyecto no son fiables
- Conseguir el nivel de calidad de los métodos de evaluación trazados en este proyecto, especialmente para la métrica F1.
- Diseñar la consulta y adaptar los métodos necesarios para el funcionamiento del sistema con datos en tiempo real (Zújar de actividad on-line).
- Integrar el desarollo con algún tipo de ejecución programada que pueda sistematizar su uso con el objetivo de identificar nuevas crisis (y conocer sus causas) en un momento prematuro.

9. Glosario de términos

Término	Descripción
Accuracy	Es una métrica utilizada en las ciencias de datos que mide el número de aciertos totales respecto del número de predicciones realizadas (en castellano exactitud).
Caché	Es un sistema de memoria intermedia que evita realizar consultas innecesarias al origen de los datos si se dispone de en la memoria intermedia de la información solicitada.
ClienteZújar	Aplicación del sistema Zújar de la AEAT utilizada para consulta de datos a través del MotorZújar.
CRISP-DM	Siglas correspondientes a "Cross Industry Standard Process for Data Mining" referidas al modelo de desarrollo (orientado a proyectos de mineria de datos) utilizado en este TFG.
Cross validation	Es una técnica del análisis de datos utiliada para seleccionar el conjunto de entrenamiento y el conjunto de pruebas. Con esta técnica se asegurar que la selección de estos es independiente al realizarse la partición repetidas veces y seleccionando siempre conjuntos distintos (en castellano validación cruzada)
CSV	Siglas correspondientes a "Comma-Separated Values" referidas a una extensión de ficheros para contener datos en forma de tabla, en los que cada línea del fichero se corresponde con una fila de la tabla y las columnas se representan mediante algún caracter, como por ejemplo la coma o el punto y coma.
Dataframe	Estructura de datos de dos dimensiones (forma de tabla) utilizada en las ciencias de datos.
ETL	Siglas correspondientes a "Extract, transform and load" que es la técnica utilizada para la extracción, transformación y carga de datos procedente de diversos de orígenes en una base de datos.
F1	Es una métrica utilizada en las ciencias de datos que pondera los resultados de las otras dos métricas *precision* y *recall* (incluidas también en este glosario) con la siguiente fórmula: 2 x *precision* x *recall* / (*precision* + *recall*).
Genio	Suite de aplicaciones del sistema Zújar de la AEAT utilizado principalmente para la elaboración y ejecución de informes.
GridSearchCV	Es una utilidad de Scikit learn que permite seleccionar la parametrización óptima entre las opciones seleccionados para la evaluacón de un modelo. Utiliza para conseguirlo validación cruzada o *cross validation*.
Jupyter Notebook	Es una aplicación web que permite interpretar código Python y que es ampliamente utilizada en los proyectos de ciencias de datos.
KFold	Es un algoritmo de validación cruzada o *cross validation* mediante el cual se divide el conjunto de datos en número K de veces. Se evalúa el modelo ese número de veces y en cada ocasión el conjunto de datos de pruebas (de tamaño Total casos/K) va cambiado de manera que todos los datos son utilizados una vez.
MotorZújar	Aplicación núcleo del sistema Zújar encargada de procesar las peticiones, aplicar las restricciones de seguridad, obtener los datos de las fuentes correspondientes y devolverlos a la llamada.
MySQL	Es una base de datos relacional ampliamente extendida y perteneciente a Oracle.
Numpy	Es una biblioteca para el lenguaje de programación Python orientada al manejo de estructuras de datos que es ampliamente utilizada en proyectos de ciencias de datos.
Pandas	Es una biblioteca para el lenguaje de programación Python orientada

	al análisis y manipulación de datos ampliamente utilizada en proyectos de ciencias de datos.
PEC	Siglas correspondientes a Prueba de Evaluación Continua utilizadas para referirse a los trabajos evaluables de la UOC (Universidad Oberta de Catalunya).
Ping	Comando informático que tiene como objetivo comprobar el estado de una aplicación o sistema.
Precision	Es una métrica utilizada en las ciencias de datos que mide la proporcuón de positivos reales entre los individuos clasificados como positivos.
Python	Lenguaje de programación ampliamente utilizado en ciencia de datos.
Recall	Es una métrica utilizada en las ciencias de datos que mide la proporcuón de positivos identificados entre los positivos existentes en la muestra.
Sklearn	Scikit learn es una biblioteca para el lenguaje de programación Python que provee utilidades de aprendizaje automático.
SQL	Siglas correspondientes a "Structured Query Language" (en castellano lenguaje de consulta estructurada) referidas al principal lenguaje de consulta de base de datos relacionales.
SybaseIQ	Es una base de datos relacional optimizada para la consulta de datos y perteneciente a SAP.
Tomcat	Es un servidor de aplicaciones web.
Zabbix	Sistema de monitorización de redes.
ZSQL	Dialecto de SQL propiedad de la AEAT utilizado para realizar consultas del datos a través del MotorZújar.
Zújar	Sistema de análisis de datos de la AEAT.

10. Bibliografía

1. José Alberto Gallardo Arancibia (2009). Metodología para la definición de requisitos en proyectos de data mining (er-dm). Universidad Politécnica de Madrid

2. Jose Martinez Heras (2020): "Precision, Recall, F1, Accuracy en clasificación". Obtenido de: https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/

3. Wikipedia (2022): "Matriz de confusión". Obtenido de: https://es.wikipedia.org/wiki/Matriz_de_confusi%C3%B3n

4. Wikipedia (2021): "Análisis de componentes principales". Obtenido de: https://es.wikipedia.org/wiki/An%C3%A1lisis_de_componentes_principales

5. Scikit-learn (2022): "1.6. Nearest Neighbors — scikit-learn 1.1.1 documentation". Obtenido de: https://scikit-learn.org/stable/modules/neighbors.html

6. Scikit-learn (2022): "sklearn.ensemble.RandomForestClassifier". Obtenido de: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

7. Scikit-learn (2022): "https://scikit-learn.org/stable/modules/tree.html". Obtenido de: 1.10. Decision Trees — scikit-learn 1.1.1 documentation

8. Xgboost developers (2021): "XGBoost Parameters". Obtenido de: https://xgboost.readthedocs.io/en/stable/parameter.html

9. Scikit-learn (2022): "sklearn.svm.SVC — scikit-learn 1.1.1 documentation". Obtenido de: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

10. Scikit-learn (2022): "https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html". Obtenido de: sklearn.neural_network.MLPClassifier

11. Scikit-learn (2022): "sklearn.gaussian_process.GaussianProcessClassifier". Obtenido de: https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessClassifier.html

12. Scikit-learn (2022): "sklearn.ensemble.AdaBoostClassifier". Obtenido de: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

13. Scikit-learn (2022): "sklearn.naive_bayes.GaussianNB". Obtenido de: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

14. Scikit-learn (2022): "sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis". Obtenido de:

https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis.html

15. Na8 (2019): "Clasificación con datos desbalanceados". Obtenido de: https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/

16. Ajitesh Kumar (2020): "Feature Importance using Random Forest Classifier - Python". Obtenido de: https://vitalflux.com/feature-importance-random-forest-classifier-python/

17. Data to Fish (2020): "How to Create a Correlation Matrix using Pandas". Obtenido de: https://datatofish.com/correlation-matrix-pandas/

18. Scikit-learn (2022): "3.2. Tuning the hyper-parameters of an estimator - Scikit-learn". Obtenido de: https://scikit-learn.org/stable/modules/grid_search.html

19. Scott Okamura (2020): "GridSearchCV for Beginners". Obtenido de: https://towardsdatascience.com/gridsearchcv-for-beginners-db48a90114ee

20. Data Science Team (2020): "Validación cruzada K-Fold". Obtenido de: https://datascience.eu/es/aprendizaje-automatico/validacion-cruzada-de-k-fold/

21. Scikit-learn (2022): "sklearn.model_selection.train_test_split". Obtenido de: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

22. Scikit-learn (2022): "Understanding the decision tree structure". Obtenido de: https://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html

11. Anexos

11.1. Resultado de la normalización de las variables del estudio

```
Se analiza la distribución de TIEMPO_BD
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de TIEMPO_MOTOR
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de TIEMPO_SERV
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de TIEMPO_SW
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de TIEMPO_CONSULTA
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de TIEMPO_TRNS
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de TAMANNO_INFORME
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de TIEMPO_INFORME
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de NUMERO_SESIONES
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_OTRO_TIPO
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_TOMCAT_OTRO
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_CACHE_MOTOR
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_NODO1_MOTOR
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_NODO2_MOTOR
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_EXPORTACION
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_COUNT
```

```
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_PRINCIPAL
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_ZSQL
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTAS_ERROR
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_COMBINADA
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_CT4CTE
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_ZJ4COLECTI
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_ZJ4DEU
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_ZJ4INPAD
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_ZJ4M200
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_ZJ4NCIH
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_ZJ4SEGZ4PE
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_ZJ4ZICBA
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTAS_INICIADAS
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTAS_FINALIZADAS
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_DIFERIDO
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_HIJO
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_JSON
```

```
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_100_PSD_GENERAL
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_DISCREP_IRPF_2EXP
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_DETALLEINGRESOSV3
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_DETALLEGASTOSV3
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_FICHAIVA
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_INGRESOSYPAGOS
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_HERMESCALC
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_HERMESINDICE
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_ONIFM190_SIMPLIF
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_VISUALIZACONSULTA
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_GENIOJS
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_HTML
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_PDF
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_PDFCATALOG
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_RTF
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_SW
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_XLS
```

```
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_TOMCAT_1
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_TOMCAT_2
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_TOMCAT_3
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_TOMCAT_4
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORMES_ERROR
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORMES_INICIADOS
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORMES_FINALIZADOS
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de CONSULTA_CACHE_GENIO
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_APP_ANALYTICS
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_APP_SRV
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de INFORME_APP_BIGDATA
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de NUMERO_CONSULTAS
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de NUMERO_INFORMES
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
Se analiza la distribución de class
([('sin_tranformar', -0.05), ('lambert', -0.05), ('log', -0.05), ('sqrt', -0.05),
('cbrt', -0.05), ('asinh', -0.05), ('boxcox', -0.05), ('yeo-johnson', -0.05),
('ordnorm', -0.05)], 'sin_tranformar')
```