JITEL 2017
XIII Jornadas de Ingeniería Telemática
27-29 Septiembre 2017, Valencia (España)

Actas de las XIII Jornadas de Ingeniería Telemática (JITEL 2017), Valencia (España), 27-29 de Septiembre de 2017

ISBN: 978-84-9048-595-8
DOI: http://dx.doi.org/10.4995/JITEL2017.2017.6574

Aplicación de técnicas de detección de anomalías a escenarios de ciudades inteligentes

Irene Romero, Carolina Alonso, Víctor A. Villagrá, Luis Vázquez, Pilar Holgado
Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid
iromero@dit.upm.es, carolina.alonso.lopez@alumnos.upm.es. villagra@dit.upm.es, lvazquez@dit.upm.es, pholgado@dit.upm.es

***Resumen*- Una de las grandes preocupaciones en la actualidad de las empresas es la detección y prevención temprana de ataques de ciberseguridad. Para ello, existen los Sistemas de Detección de Intrusiones, herramientas que cuentan con sensores virtuales y que basan su detección en el análisis del tráfico de red. El problema surge cuando se dan ataques que estos sistemas no detectan. Una de las soluciones existentes a esta problemática es acudir a la minería de datos e intentar detectar anomalías en grandes volúmenes de datos, no pertenecientes únicamente al tráfico de red sino datos que puedan provenir de diversas fuentes. En este artículo se propone una solución enmarcada en el proyecto DHARMA haciendo uso de la técnica de agrupamiento, dentro de la disciplina de la minería de datos, en concreto del algoritmo DBSCAN.**

***Palabras clave* – DBSCAN, minería de datos, agrupamiento, detección de anomalías.**

I. INTRODUCCIÓN

Todos los ámbitos de las tecnologías de la información tienen un concepto en común: la innovación. A diario encontramos nuevos lenguajes de programación, nuevas y útiles librerías para aquellos lenguajes ya existentes, e incluso nuevos paradigmas de desarrollo. Y esto también se cumple en el campo de la ciberseguridad: nuevas vulnerabilidades, nuevos métodos de ataque y, por supuesto, formas de luchar contra ellos. Por otro lado, las nuevas tecnologías son clave en el desarrollo de soluciones de software innovadoras. En concreto, el análisis estadístico y la minería de datos son el segundo conocimiento más valorado en Linkedin para encontrar trabajo en 2017 [1], y la razón detrás de ello es el hecho de que son tecnologías emergentes cuyas posibilidades de aplicación todavía no están totalmente explotadas e incluso se desconocen.

Dado el auge de las comunicaciones a distancia y de Internet en concreto, una de las grandes preocupaciones de las empresas y de las organizaciones es la detección y prevención de ataques de ciberseguridad. Que una persona al otro lado del mundo pueda infiltrarse en tu sistema sin que seas consciente puede resultar extremadamente problemático. Además, no solo es necesario saber que tus sistemas están siendo atacados sino que hacerlo en el momento oportuno, en etapas tempranas del ataque, puede resultar clave.

La minería de datos toma parte en este apartado: la mayoría de las empresas, y sobre todo aquellas que trabajan online, cuentan volúmenes de datos recogidos durante años, muchas veces sin explotar, de los que se puede extraer información. Se puede aplicar al problema anterior y analizar qué sucesos pueden indicar que un ataque está en proceso, o incluso anticiparse y detectarlo en etapas no críticas, de forma que se puedan tomar las salvaguardas apropiadas antes de que haya causado daño.

Definir cuáles son los indicios de un ciberataque no supone una tarea simple. Existen sucesos evidentes, como la presencia de un nuevo usuario en nuestros sistemas accediendo a información crítica, pero la problemática surge en relacionar sucesos que puedan considerarse irrelevantes con los datos disponibles y detectar así una posible anomalía. Es este el problema para el que trata de plantear solución este artículo.

En este documento, la detección de sucesos anómalos se enmarca dentro de un proyecto de evaluación y gestión dinámica de riesgos denominado DHARMA [2], [3] (Dynamic Heterogeneous threAts Risk Management and Assessment).

El análisis y gestión de riesgo tradicionales se encargan de identificar activos y sus vulnerabilidades, amenazas sobre estos y probabilidad de ocurrencia (obteniendo el riesgo sobre un activo) y salvaguardas para reducir este riesgo. Para ello se siguen distintas metodologías como MAGERIT, ISO/IEC 27005, OCTAVE, etc. No obstante, estas metodologías son estáticas, pues únicamente se ejecutan en momentos determinados, y el contexto puede variar completamente entre ejecuciones.

EDITORIAL UNIVERSITAT POLITÈCNICA DE VALÈNCIA

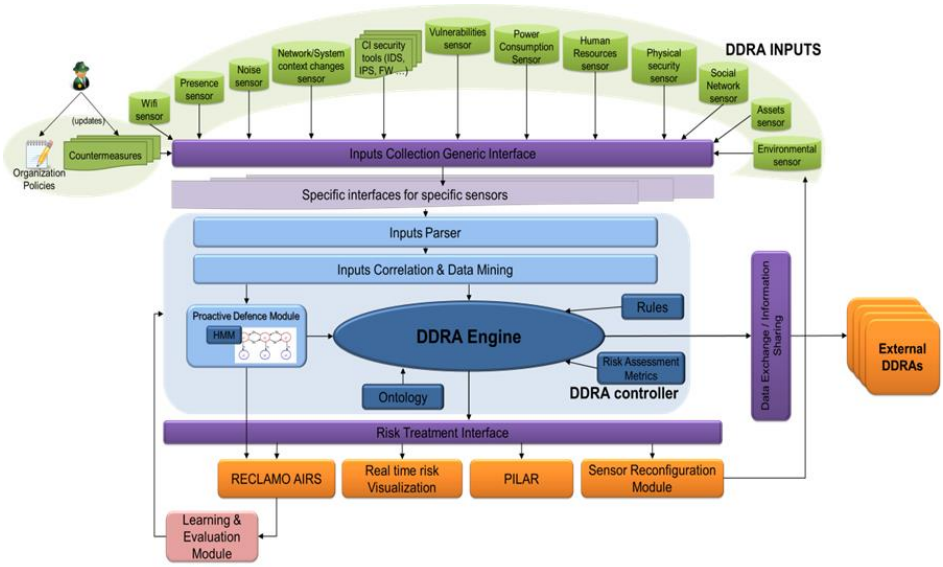

Fig. 1 Arquitectura DHARMA

Este sistema se encarga de dar respuesta a incidentes o ataques en un corto plazo de tiempo, pudiendo responder a esos cambios de contexto de una forma dinámica. Para ello se ha propuesto una arquitectura en la que el contexto de la organización se modela mediante un gran número de sensores heterogéneos (wifi, Bluetooth, presencia, estado de activos, crispación laboral, ambientales, actividad en redes sociales, etc.), como se observa en la figura 1. Los datos obtenidos por estos sensores serán analizados, correlados (el tema principal de este artículo) y enviados a un motor de análisis de riesgo que calculará el riesgo instantáneo de la organización y desplegará las contramedidas pertinentes (visualización en tiempo real, reconfiguración de los sensores, alimentación de un Sistema de Respuesta Automática a Intrusiones o herramientas clásicas de análisis de riesgo como PILAR, etc.).

La propuesta estudiada en este artículo en encuentra en el nivel "Inputs Correlation and Data Mining" dentro de la arquitectura DHARMA, como se observa en la figura 1.

La aplicabilidad de esta arquitectura y técnica se explorará a través del caso de uso de ciudades inteligentes.

II. Técnicas de detección de anomalías con minería de datos

Una anomalía es una observación significativamente diferente al resto de observaciones, de lo que se deduce que dicha observación puede haber sido generada por un mecanismo diferente del resto de observaciones.

La detección de anomalías surge como un área de trabajo importante en los algoritmos de aprendizaje. En algún caso la capacidad de detección de anomalías deriva de forma natural de los algoritmos de agrupamiento, que se estudiarán a continuación.

El agrupamiento es una disciplina de la minería de datos que consiste en la clasificación de las entradas de un set de datos en distintos grupos, pero sin tener ningún tipo de información a priori sobre éstos. Es decir, trata de buscar algún tipo de relación desconocida que pueda indicar que un subconjunto de nuestros datos tenga algo en común entre sí.

En este caso se explotará una de sus utilidades: la detección de anomalías. Al hacer agrupamiento, lo normal es obtener una serie de clústeres y, en función del tipo de algoritmo, ruido. El ruido está formado por aquellos puntos que no cumplen las condiciones necesarias para formar parte de ningún otro clúster; normalmente están demasiado lejos de otros puntos. De esta forma, se puede definir el ruido como puntos que están fuera de la norma, del comportamiento del resto del set de datos. En el caso de datos de uso de redes Wifi, una anomalía sería que el tiempo de conexión de un usuario fuese mayor a un día o que consumiese más ancho de banda del que debería.

Por otro lado, hay distintos métodos para agrupar datos: por distribución estadística, asignando un centroide a cada grupo y calculando la distancia a éste, basados en la densidad de puntos en el espacio bidimensional… A continuación se explica el funcionamiento de algunos de estos métodos.

A. Agrupamiento por centroide

En este modelo se asigna un centroide a cada clúster, de forma que todos los elementos que pertenecen a ese clúster estén más cerca de su centroide que del centroide de otros clústeres. El punto asignado al propio centroide depende de los elementos que haya en el clúster, y su valor no tiene por qué ser un punto del set de datos.

Ejemplo: K-means. Se trata de un algoritmo de agrupamiento por centroide en el que el número de clústeres está prefijado, y viene dado por el valor k. De esta forma, para k=2 tendremos dos clústeres. El esquema que sigue es el siguiente:

1. Asignar k centroides de forma aleatoria.
2. Asignar cada elemento del set de datos al centroide que proporcione la mínima suma de distancias al centroide.
3. Recalcular los centroides.
4. Repetir hasta que converja.

Se considera que el algoritmo converge cuando durante el paso 3 los centroides no cambian, o cuando la distancia entre centroides de una iteración a la siguiente es menor que cierto parámetro de precisión predefinido.

B. Agrupamiento por distribución

Este modelo está basado en el ajuste de cada clúster a una distribución estadística, de forma que cada punto se asigna al clúster al que tiene la máxima probabilidad de pertenecer.

Ejemplo: algoritmo esperanza-maximización. Se trata de un método de agrupamiento por distribución en dos pasos: durante el primero, llamado paso de "esperanza", se calcula una función de la media de la función de verosimilitud de los actuales parámetros; durante el segundo, el de "maximización", se maximiza dicha función para buscar una solución óptima.

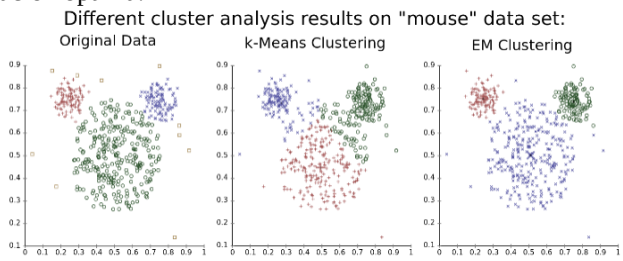

Fig. 2 Comparación del resultado de aplicar k=3 means respecto a esperanza-maximización

C. Agrupamiento por densidad

En este modelo se considera un grupo como una zona del espacio en el que la densidad de puntos es mayor que en el resto. Tienen la ventaja de que, si en una zona no hay suficiente densidad de puntos, los que se encuentren en ella se consideran ruido, que se puede interpretar como anomalías.

Ejemplo: DBSCAN: representa las siglas en inglés de "Agrupamiento Espacial Basado en Densidad de Aplicaciones con Ruido". En este algoritmo contamos con dos parámetros: **minPts** (número mínimo de puntos) y **ε** (épsilon, que en el código se representa como eps). Por otro lado, los puntos se clasifican en tres tipos: núcleos, no-núcleos y ruido. Un **núcleo** es un punto que tiene a una distancia épsilon a al menos minPts-1 puntos, por lo que juntos forman un vecindario de radio ε. Un **no-núcleo** es un punto vecino de un núcleo, pero que no es núcleo por sí mismo. Y finalmente, un punto es considerado **ruido** si no es vecino de ningún núcleo.

De esta manera, en DBSCAN un grupo de puntos forman un clúster si todos ellos son núcleos o vecinos de al menos un núcleo, y los puntos que no forman parte de ningún grupo de tamaño minPts son ruido. El esquema que sigue el algoritmo es el siguiente:

1. Elegir un punto aleatorio no visitado y moverlo a la lista de visitados.
2. Calcular su vecindario en un radio ε.
    a. Si no está formado por minPts elementos, etiquetar como ruido y volver a 1.
    b. Si está formado por al menos minPts elementos, iniciar un clúster y añadir a sus vecinos a la lista de puntos candidatos a formar parte de éste.
        i. Para cada punto candidato, si no ha sido visitado, calcular sus vecinos y comprobar si es núcleo.
            1. Si lo es, añadir sus vecinos a la lista de candidatos.
        ii. Para cada punto candidato, si no forma parte de ningún clúster, añadirlo al actual.
3. Si quedan puntos sin visitar, volver a 1.

A pesar de que tiene la ventaja de permitir encontrar anomalías, su principal punto en contra es que no es determinista. Un punto no-núcleo puede ser vecino de varios núcleos que a su vez formen parte de distintos clústeres, por lo que en el resultado final puede estar en cualquiera de ellos. Sin embargo, esta desventaja no afecta a esta solución, ya que en ella se utilizará DBSCAN sólo para obtener el ruido en un set de datos. DBSCAN también tiene una serie de características que pueden resultar interesantes al elegir un algoritmo de detección de anomalías: es determinista en cuanto a ruido y no requiere que haya una correlación alta entre variables, sólo que

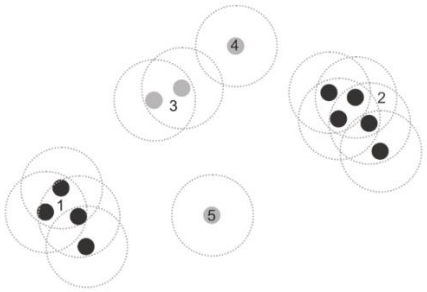

Fig. 3 Ejemplo de distintos tipos de puntos tras ejecutar DBSCAN para $minPts = 3$ y ϵ representada por las líneas discontinuas. 1 y 2 representan dos grupos, mientras que 3, 4 y 5 no cumplen la condición y se consideran ruido.

las variaciones en el set de datos sean suaves. En cuanto a sus desventajas. En cuanto a sus desventajas, dependiendo de la implementación puede ser lento, con una complejidad temporal entre O(*nlogn*) y O(n^2), sucediendo lo mismo en cuanto a memoria, con complejidad entre O(n) y O(n^2)[4]. También implica tener un conocimiento previo de los datos, ya que es complicado determinar los valores adecuados de ϵ y minPts al vuelo, y la utilidad de su aplicación a variables nominales es discutible, ya que no siempre es inmediato entender qué mide una distancia en este tipo de valores.

Finalmente, a pesar de ser relativamente antiguo, al haber sido publicado originalmente en 1996, DBSCAN es un algoritmo que sigue utilizándose en un amplio campo, recibiendo incluso en 2014 el *Test of Time Award* de *KDD*[5], y habiendo probado anteriormente[6-7] su utilidad en la detección temprana de riesgos de seguridad.

III. Caso de uso: Ciudades Inteligentes

El término "Ciudad Inteligente" (o Smart City) es el nombre que se da a una ciudad que utiliza las tecnologías de la información y el internet de las cosas para facilitar servicios a sus residentes y mejorar el funcionamiento global de la ciudad. Se basan en la presencia de sensores por distintas zonas, como pueden ser cámaras, sensores de humedad o de presencia en las aceras.

Las tecnologías de las Smart Cities tienen gran cantidad de aplicaciones, principalmente enfocadas a la prestación de servicios eficientes y útiles a sus habitantes. Son un modo de mejorar el nivel de vida de la población, y el hecho de que a día de hoy todo el mundo esté conectado en todo momento, es especialmente fácil encontrar aplicaciones con las que podrán interactuar personalmente.

La tendencia a evolucionar hacia ciudades inteligentes, dada la velocidad de crecimiento de las áreas urbanas, es cada

vez más acusada. Teniendo en cuenta que se prevé que el 70% de los seres humanos habiten en centros urbanos en 2050, es necesario avanzar hacia ciudades que permitan una gestión menos manual y más sostenible. De esta forma, todos los elementos que conforman la ciudad han de evolucionar hacia plataformas inteligentes para lograr una gestión eficiente y crear ciudades formadas por una red compleja de sensores que proporcionarán información clave para el funcionamiento y la toma de decisiones.

Esta compleja red de sensores que forman las plataformas de ciudades inteligentes pueden ser utilizadas para múltiples ámbitos: uno de ellos es la ciberseguridad, o cómo aprovechar el gran volumen de datos que estos sensores proporcionan para poder correlar sus anomalías con posibles intrusiones en entornos de ciberseguridad en sistemas y plataformas albergadas en estas ciudad inteligente. Se trata de un enfoque holístico: la combinación de detección de anomalías en el entorno físico (proporcionado por los sensores de la ciudad inteligente) y entorno lógico (proporcionado por los sensores propios de ciberseguridad como sistemas de detección de intrusiones, etc.)

El uso de técnicas de detección de anomalías en las Smart Cities resulta de mucha utilidad para extraer información valiosa o detectar comportamientos anómalos que puedan generarse en estas ciudades. Dada la cantidad de sensores con los que contará una Smart City, es necesario implementar una técnica que permita analizar el elevado volumen de datos generado y, de esta forma, que la gestión de dicha ciudad sea también inteligente. Los comportamientos anómalos detectados podrían pertenecer a ataques y la sola gestión manual de un operario no sería suficiente para detener dicho ataque.

El escenario donde se enmarca este trabajo es una plataforma de procesamiento de datos procedentes de la City of the Future de la Universidad Politécnica de Madrid. Esta iniciativa, creada y mantenida tanto por docentes como alumnos, está centrada en la investigación, desarrollo e innovación en el campo de las City Sciences, y aborda temas tan amplios como el control de tráfico, la gestión de emergencias o las *smart grids*.

La City of the Future es el nombre que se le da al conjunto de sensores y fuentes de datos que tiene la UPM por toda la ciudad de Madrid, con mayor presencia en el área de Ciudad Universitaria y Moncloa. Está formada por sensores de clima, de tráfico, de conexiones Bluetooth, etc. Supone un proyecto enfocada a la aplicación de las tecnologías de la información para crear una Smart City en Madrid. Al tratarse de un proyecto universitario, su principal objetivo es la investigación y el desarrollo de técnicas nuevas en un ámbito docente, proporcionando a los estudiantes acceso a proyectos diversos con los que enriquecer sus conocimientos, además de ayudar a la prestación de servicios inteligentes a los habitantes de la propia ciudad.

Este trabajo se centra en el tratamiento de la información generada por sistemas ya implementados para obtener información relevante mediante minado de datos.

Dado que los datos procederán de fuentes heterogéneas, será importante crear una plataforma que pueda tratar con datos de distintos tipos (no sólo numéricos), de gran tamaño o de distintos formatos. También será fundamental que se trate de un sistema multiplataforma y ligero para el usuario, además de poder hacer un volcado de los resultados para posibles aplicaciones futuras.

Como ya se ha comentado, la aplicación estará enfocada al tratamiento de los datos generados por estas aplicaciones, de los que se intentará extraer información subyacente. El proyecto más relevante es el Smart CEIM.

Como se puede ver en la figura 4, la plataforma Smart CEIM está formada por una red de sensores y actuadores conectados a la plataforma de almacenamiento y al cuadro de control mediante IP.

Los datos recogidos por dicha plataforma arrojan información ambiental e información sobre las redes WiFi y su uso, por lo que sólo detectaremos comportamiento anómalo en dichos datos, identificando posibles amenazas en estas dimensiones.

Los datos ambientales provienen de sensores ambientales colocados en las bibliotecas de las distintas escuelas, así como en exteriores (en zonas resguardadas para evitar daños por lluvia o sol). Estos sensores proporcionan valores en tiempo real de temperatura ambiente, humedad, nivel de luz, batería del dispositivo, monóxido de carbono, dióxido de nitrógeno y nivel de ruido. Además, cada valor incluye una marca de tiempo que permite conocer cómo varía cada variable a lo largo del día.

En cuanto a los sensores WiFi, se cuenta con los datos de uso de redes WiFi en las bibliotecas de las escuelas de Aeronáuticos, Industriales y Telecomunicaciones. En la escuela de Telecomunicaciones, además, se contará con acceso a los datos de varias subredes repartidas por los edificios. Dichos sensores están colocados en zonas de mucho tránsito, como bibliotecas, ya que su finalidad original es la monitorización de personas.

La información que proporcionan estos sensores incluye la relativa a los dispositivos conectados (dirección MAC, canal al que se han conectado, potencia, etc.) así como estadísticas que pueden ser de utilidad (número total de dispositivos conectados, número total de conexiones simultáneas a la red, etc.).

Así pues, la detección de anomalías se realizará únicamente sobre estos dos conjuntos de datos, permitiendo detectar comportamientos anómalos en el uso de las redes WiFi o en los datos ambientales; ya que ambos pueden ser indicadores de un problema mayor.

IV. VALIDACIÓN

La implementación de este sistema se ve dividida en distintos módulos, que se exponen a continuación:

A. Preprocesado

Como se ha comentado, la aplicación cuenta con un apartado de preprocesado, mediante el cual se consiguen dos cosas: en primer lugar, darle información al algoritmo sobre cómo son los datos con los que va a tratar, y en segundo lugar modificar los datos para que DBSCAN proporcione información más relevante, sin modificar su significado.

El flujo básico de la aplicación es la siguiente: un usuario sube datos en cualquier momento al servidor, indicando en la pantalla de preprocesado los parámetros necesarios, como el tipo de las variables. Estos parámetros se almacenan en un archivo de configuración, que se utilizará cada vez que el usuario decida entrar de nuevo en la aplicación. En la vista principal se muestra una lista con todos los sets de datos disponibles y sus archivos de configuración correspondientes, de forma que cuando el usuario lo desee, puede seleccionarlos y ejecutar los cálculos que ha indicado.

Una vez recibidos los resultados, se pasa a la vista de visualización de resultados, en la que el usuario puede detectar las anomalías entre cada par de variables.

B. Detección de anomalías

Para la detección de anomalías se ha utilizado R, dado su versatilidad en lo que a minería de datos se refiere.

En cuanto a herramientas adicionales, se han utilizado varias librerías para funciones no disponibles por defecto en R: en primer lugar se encuentra *devtools*, utilizada para crear e instalar funciones personalizadas dentro del entorno de ejecución. En segundo lugar se encuentra *OpenCPU*, que proporciona acceso a dichas funciones mediante peticiones HTTP. La ventaja de usar estas dos herramientas yace en el hecho de que se está realizando minería de datos, por lo que conviene dedicar su propio servidor a esta herramienta, evitando así quedarse sin recursos, tanto con potencia de procesamiento como con memoria.

Entrando en la implementación de la detección de anomalías, hay que tener en cuenta una serie de consideraciones:

- En primer lugar, el algoritmo DBSCAN sólo puede recibir valores completos. En caso de recibir datos *null*, *undefined* o vacíos, salta un error y para la ejecución. Esto se soluciona fácilmente en R mediante la función *complete.cases(datos)*, que filtra los datos incompletos.
- En segundo lugar, limpiar los datos de valores no completos tiene una consecuencia: una vez se tengan los resultados, éstos no serán aplicables al set de datos original. Por meros estándares de calidad, es conveniente que los resultados se generen de forma concordante con los datos recibidos. La solución para este problema consiste en considerar que estos datos incompletos son anomalías en sí, e intercalar los resultados de DBSCAN con estas consideraciones en el orden adecuado.
- En tercer lugar, los valores deben estar normalizados. Es uno de los errores más comunes cuando se agrupa utilizando medidas de distancia: no se tiene en cuenta que distintas variables representan métricas distintas, por lo que, si una tiene un rango mucho mayor, la medida de distancia sólo será representativa para ella. Como se utiliza DBSCAN para detectar anomalías con pares de variables, esto hará que se obtengan resultados erróneos al tratar con un par de variables de rangos muy distintos. Normalizándolas entre 0 y 1 este apartado queda solucionado.
- En cuarto lugar, como se ha comentado, DBSCAN hace uso de dos parámetros: $minPts$ y ϵ. Por comodidad, siendo lo más común, se fija el valor de $minPts$ en 3, de forma que podamos calcular ϵ en función de los datos con los que se esté tratando en cada momento. Para ello se utiliza una función de la desviación estándar de ambas variables.
- Finalmente, no se detectarán anomalías para variables nominales, pues es de dudosa utilidad medir la distancia entre este tipo de variables, por lo que directamente se descartan. Por simplificación del código, simplemente se oculta el botón correspondiente en la vista, de forma que no haya que realizar comprobaciones de los tipos de todas las variables en cada llamada a la función

De esta manera, el esquema que sigue la función de detección de anomalías es la siguiente:

- Lectura de datos
- Filtrado de valores incompletos
- Normalización
- Cálculo de ϵ a partir de las desviaciones estándar
- Ejecución de DBSCAN y obtención de anomalías
- Ampliación de resultados a set de datos completos
- Devolución de resultados

C. Detección de anomalías en tiempo real

El sistema permite también la detección de anomalías en tiempo real, dentro del proyecto DHARMA, que utiliza esta propuesta como un indicador de posibles ataques en fases tempranas.

Esta nueva idea sigue el siguiente esquema: se cuenta con varios flujos de datos constantes, cuya frecuencia de actualización depende de las fuentes. En el momento en el que se reciban datos nuevos, un *watchdog* los detecta y preprocesa para ser almacenados en distintos sets de datos. Dónde se almacenan depende de parámetros predefinidos de actividad, esto se debe a que no es lo mismo tener tráfico alto en un sistema en jueves por la mañana, en horario laboral, que en la noche del sábado al domingo. Es importante diferenciar entre períodos de alta y baja actividad, así que mediante el preprocesado se indicarán los rangos de cada uno. Una vez se tiene esta información, se analiza y se almacena en los sets de datos correspondientes, cambiado el formato de las marcas temporales para favorecer la ejecución del algoritmo de agrupamiento. Finalmente, en el momento en el que se ha terminado de procesar toda la información, se ejecuta DBSCAN para todos los sets de datos con nuevas entradas, y, en el caso de que haya nuevas anomalías, se informa a DHARMA como si se tratase de un indicio inicial de ataque.

Teniendo en cuenta estas consideraciones, el preprocesado tiene dos fases: una inicial en la que se crea un fichero de configuración con información sobre los datos (nombres de variables y sus tipos, cuáles son marcas temporales y sus formatos, y los periodos de alta y baja actividad), y una segunda que tiene lugar cada vez que se reciben datos nuevos, en la que se procesan los datos, ajustando su formato para que sea más adecuada su visualización y para almacenarlos donde corresponda.

Cabe destacar que el preprocesado en esta implementación es distinto al anterior: tiene lugar en el momento en el que se configura el servidor para recibir datos de una fuente, ya que hasta que no se cuente con la información de preprocesado no se podrá ejecutar el código de detección de anomalías.

En cuanto a los cambios de formato, un ejemplo es el de las marcas temporales. En primer lugar, se pasan a formato UNIX, dado que éste permite tener variaciones constantes entre instantes consecutivos. Esto se debe a que los sets de datos con los que se ha trabajado durante el desarrollo del proyecto siguen formatos similares a ISO 8601, en los que se emplean directamente la fecha y la hora. La desventaja de estas configuraciones es que los rangos que sigue cada uno de los elementos no son decimales: la hora varía entre 00 y 23, el mes entre 1 y 12, etc. y a la hora de representarlos en un eje temporal se producen saltos que harán que al ejecutar DBSCAN se identifiquen como grupos distintos valores que realmente son cercanos.

Otra consideración en lo que a cambio de formato se refiere es cómo asegurarse de que se utilizan correctamente los rangos de actividad. Su finalidad es que, en el momento de detectar anomalías, estemos comparando los nuevos datos

con otros equivalentes, como puede ser datos de uso de redes Wifi en una universidad durante el curso. Dados los horarios utilizados en estos tipos de instituciones, cabría deducir que el tráfico durante todos los martes debería ser similar a las mismas horas. Por tanto, el formato de la marca de tiempo se ha cambiado para indicar sólo la hora.

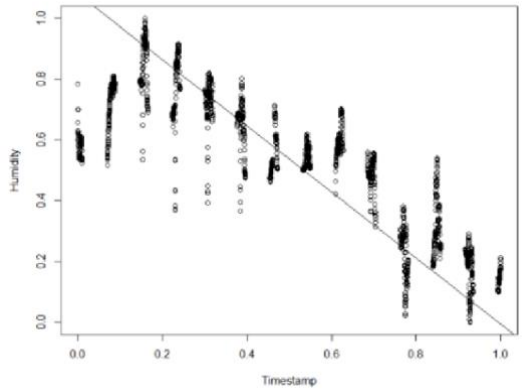

Fig. 5 Ejemplo de visualización de datos con eje temporal con formato ISO 8601

Una vez realizados estos procesos, los datos se encuentran ya en un formato adecuado para la detección de anomalías.

D. Postprocesado

Una vez se han recibido los datos de R, hay que realizar un postprocesado. La razón por la que sucede esto es que mediante DBSCAN se van a detectar anomalías en todo el set de datos, incluso anomalías antiguas que ya no son motivo de alarma.

Para ello, una vez recibidos los resultados se comparan con los anteriores, que estarán almacenados. En caso de haber nuevas anomalías, se notifica a DHARMA. Por otro lado, en todos los casos se almacenan los nuevos resultados, para utilizarlos en la próxima llamada.

También se tiene en cuenta el caso inicial, en el que no hay resultados antiguos. La decisión tomada para este apartado es simple: se notifica a DHARMA de todas las anomalías presentes, ya que, si en el sistema no se encuentra en proceso un ataque, será DHARMA quien lo deducirá.

V. PROTOTIPO: APLICACIÓN WEB

En un principio, la idea a implementar consiste en una aplicación web que permite al usuario subir sus sets de datos, realizar el preprocesado manualmente y, una vez almacenados, calcular la correlación entre distintas variables, visualizarlas en un diagrama de dispersión y detectar anomalías.

Para ello se ha hecho uso de las siguientes tecnologías:

- HTML, CSS, AngularJS y D3.js para el front end.
- Node.js y Express para el back end.
- R para el motor de cálculo.

Por otro lado, R no sólo se utiliza mayormente para realizar minado de datos, sino que, en el caso de querer ampliar las funcionalidades de la aplicación, permitiría tener acceso a gran cantidad de funciones pre implementadas y ampliar el alcance de manera rápida.

En cuanto al diseño de la aplicación, la forman tres vistas que se detallarán a continuación.

A. Vista principal

En la vista principal de la aplicación se permite al usuario seleccionar un set de datos existente o subir uno nuevo.

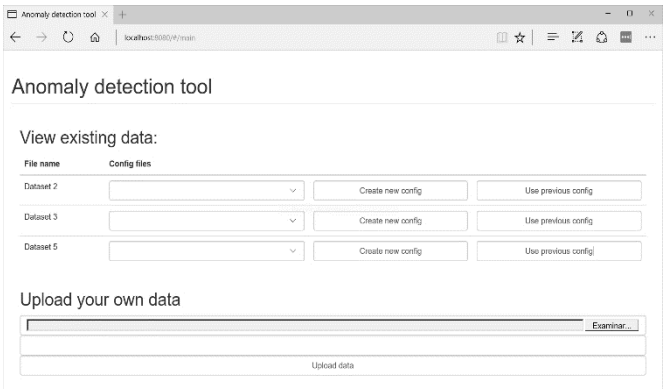

Fig. 6 Vista principal

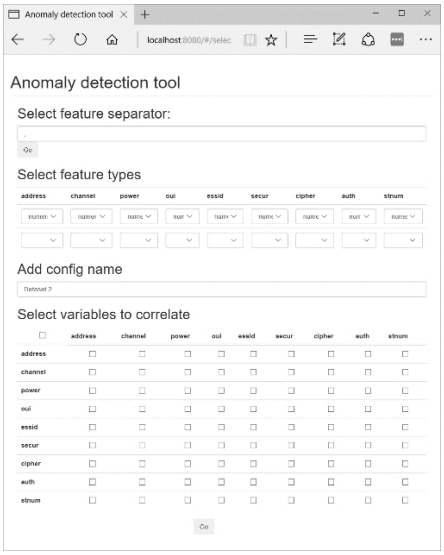

Fig. 7 Vista de selección de parámetros de preprocesado

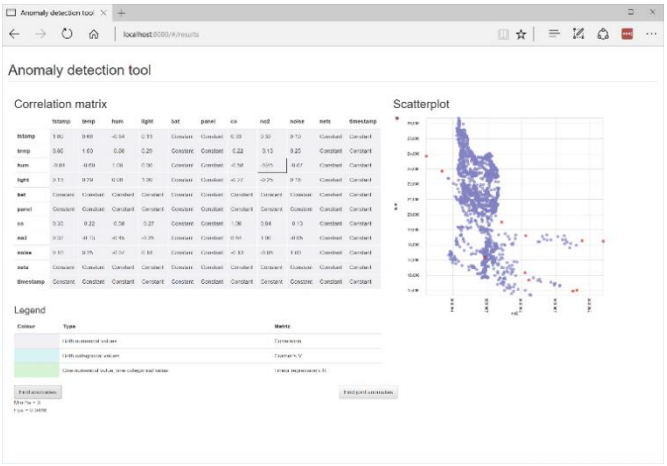

Fig. 8 Vista de visualización de resultados con las anomalías representadas en rojo

B. Vista de preprocesado

En esta vista se permite al usuario definir ciertos parámetros para el correcto funcionamiento del algoritmo de agrupamiento.

C. Vista de visualización de resultados

Aquí el usuario puede visualizar la matriz de correlación de su set de datos, junto con un diagrama de dispersión al hacer hover sobre sus distintos valores. Haciendo click en cualquiera de ellos podrá bloquear la gráfica y seleccionar la opción de detectar anomalías.

VI. CONCLUSIONES

En este artículo se ha analizado la aplicación de un algoritmo de detección de anomalías, en concreto de un algoritmo de agrupamiento. La aplicabilidad del mismo se ha estudiado a través del caso de uso de ciudades inteligentes donde la gestión de la misma ha de ser inteligente y son necesarios algoritmos como el empleado.

Una vez terminada la implementación, los resultados obtenidos tras probarlos son positivos. No se puede hablar de términos como precisión, ya que no se ha contado con información sobre si un dato es anómalo o no, pero tras la ejecución del algoritmo y mediante la visualización se puede observar cómo aquellos datos que distan de los demás se han detectado correctamente como anomalías.

Esto es también aplicable a la solución en tiempo real, para la que se han proporcionado unos parámetros más laxos en la implementación de DBSCAN. La razón es su propia aplicación: al tratarse de una herramienta para detección temprana de ataques, es importante no dejar pasar ningún falso negativo, de forma que es preferible que salten falsas alarmas, que DHARMA se encargará de filtrar posteriormente.

Como se ha comentado a lo largo del documento, detectar ataques en fases tempranas puede resultar crítico en sistemas en los que la seguridad es importante, y utilizar tecnologías nuevas, como es el minado de datos, puede resultar extremadamente útil y resultar en soluciones innovadoras con múltiples aplicaciones.

Finalmente, ha destacado la importancia de investigar en el ámbito del minado de datos: que un algoritmo tan antiguo haya servido para añadir funcionalidades extra en un ámbito como la detección de ataques de ciberseguridad en ciudades inteligentes es un indicador de que las posibilidades de crecimiento aplicando otros conceptos del minado de datos pueden resultar muy satisfactorias.

Como línea futura, se propone la extensión de dicha técnica de detección de anomalías no solo a los datos procedentes de los sensores ambientales y de WiFi, sino a todos los sensores presentes en la ciudad inteligente (correlando la información entre ellos) y de esta forma completar el módulo de minería y correlación de datos para que arroje información sobre comportamientos anómalos en todo el sistema.

VII. AGRADECIMIENTOS

Este trabajo ha sido financiado en parte con el apoyo del MINECO español (proyecto DHARMA, Dynamic Heterogeneous Threats Risk Management and Assessment, con código TIN2014-59023-C2-2-R) y por la Comisión Europea (FEDER/ERDF).

REFERENCIAS

[1] Catherine Fisher, “Linkedin Unveils the Top Skills That Can Get You Hired In 2017”, en *https://blog.linkedin.com/2016/10/20/top-skills-2016-week-of-learning-linkedin*, Octubre 2016.

[2] Proyecto DHARMA: http://dharma.inf.um.es/

[3] Pilar Holgado, Víctor A. Villagrá: "Sistema de detección de fases de ataque basado en modelos ocultos de Markov," en *Proceedings of the II Jornadas Nacionales de Investigación en Ciberseguridad*, pp. 25-28, Granada (Spain), 15-17 June 2016.

[4] Martin Ester, Hans-Peter Kriegel, Jiirg Sander, Xiaowei Xu: “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, en *KDD-96*, 1996.

[5] KDD, “2014 SIGKDD Test of Time Award” in *ACM SIFKDD*, August 2014.

[6] Xiaohua Yan, Joy Ying Zhang, “Early Detection of Cyber Security Threats using Structured Behavior Modeling” in ACM Transactions on Information and System Security, January 2013.

[7] Li Xue-yong, Gao Guo-hong, Sun Jia-xia: “A New Intrusion Detection Method Based on Improved DBSCAN”, en *WASE International Conference on Information Engineering*, August 2010.

[8] Pilar Holgado, Víctor A. Villagrá: "Evolving from a static toward a proactive and dynamic risk-based defense strategy” en *Proceedings of the I Jornadas Nacionales de Investigación en Ciberseguridad*, pp. 129-136, León (Spain), 14-16 September (2015).