Actividad Evaluable 3

Descripción

MÓDULO	**Seminario Internacional en Herramientas y Técnicas de Detección de Ciberamenazas**
ASIGNATURA	**Data Science Aplicado a la Ciberseguridad**
Fecha Límite de Entrega	30 de Abril de 2023, a las 23:59
Puntos	25% de la Nota Total
Carácter	Grupo (max 2 personas)

Enunciado:

En esta actividad se planteará una serie de preguntas relacionadas con los últimos temas vistos en las sesiones síncronas. El estudiante debe responder a tales preguntas de este documento, de forma clara y concisa. **El resultado final debe estar disponible en un repositorio de código creado en Github.**

El repositorio de código debe ser público, es decir, estar configurado de forma que sea accesible para otras personas más allá del estudiante. Adicionalmente, se requiere entregar un documento de texto que contendrá el enlace al repositorio de código a través del campus virtual hasta la fecha límite de entrega, incluyendo nombre y DNI/NIE/Pasaporte en este documento. Se considerará tanto la corrección de las soluciones como su presentación.

Parte de esta actividad implica programar código R. **Tal código debe ser entregado en un documento de código** (formato de fichero *.R o .RMD*), tal que el código debe poderse ejecutar directamente sobre un terminal nuevo en R. El código es imprescindible para la corrección del ejercicio y deberá estar incluido en el contenido del repositorio de código. **Adicionalmente, para esta entrega también deberá acompañarse el código de el documento RMarkdown renderizado en formato HTML o PDF.**

Las entregas tardías serán marcadas como “tarde”, y pueden NO ser evaluadas.

Análisis de logs de servidor usando R (II)

Obtención de Datos:

Queremos programar un script con el que podamos hacer una investigación forense sobre un fichero de logs de un servidor de tipo Apache. Los datos del registro del servidor están en el formato estándar e incluyen miles de registros sobre las distintas peticiones gestionadas por el servidor web.

Nuestro programa ha de ser capaz de obtener las respuestas de forma dinámica a las siguientes preguntas utilizando instrucciones de código en R:

1. **Descomprimir el fichero comprimido que contiene los registros del servidor, y a partir de los datos extraídos, cargar en data frame los registros con las peticiones servidas.** El primer paso para realizar tareas de análisis consiste en la carga de los datos, la exploración inicial y limpieza de los datos (filtrado, corrección de tipo de datos, ajuste de la forma de los datos, etc.). Para esto usaremos la capacidad de R y de sus librerías (*tidyr* y *dplyr*) para modificar los data frames cargados y adaptarlos a la estructura de datos elegantes: cada fila se corresponderá a una observación y cada columna únicamente una variable.
   De esta forma, se facilita el análisis y la posterior modificación de la forma del data frame, adaptando este para que pueda ser usado en la mayoría de funciones de R.

   Pistas:
   - Aseguraros de que cada columna del data frame obtenido como resultado de la carga de datos tiene cada columna con el tipo de datos adecuado según la variable en cuestión (integer, boolean, factor, string) y que cada columna contiene información sobre una única variable.
   - Aseguraros de gestionar bien cualquier posible valor no disponible (*NA*). Suele ser más sencillo si nos aseguramos de cargar debidamente los datos con los parámetros correctos para `read_table()`.

Exploración de Datos

2. **Identificar el *número único de usuarios* que han interactuado directamente con el servidor de forma segregada según si los usuarios han tenido algún tipo de error en las distintas peticiones ofrecidas por el servidor.**

   Hacer la distinción (*break down*) del número de usuarios en función de si estos han tenido algún tipo de error durante las interacciones con el servidor y el tipo de error, es decir, ofrecer el número de usuarios que no han tenido ningún error en una de las peticiones gestionadas por el servidor y el caso contrario, el número de usuarios que sí han experimentado algún error para una petición servida según la tipología del error.

   Consultar Wiki para más detalles sobre cuales han sido incorrectamente servidas.

   Describir en el documento RMarkdown los distintos tipos de errores existentes en la muestra de datos y el número único de usuarios.

Análisis de Datos

3. **Analizar los distintos tipos de peticiones HTTP (GET, POST, PUT, DELETE) gestionadas por el servidor, identificando la frecuencia de cada una de estas. Repetir el análisis, esta vez filtrando previamente aquellas peticiones correspondientes a recursos ofrecidos de tipo imagen.**

   Pistas:

   - Para poder analizar los distintos tipos de peticiones HTTP, dependiendo de que valores se hayan pasado a los parámetros usados en la función de carga de datos y las tareas ejecutadas durante la limpieza y transformación en datos elegantes, debería existir una columna con el tipo de petición HTTP. Si no es el caso, será necesario crear una columna para tal información. Para ello, en el caso que esta información este contenida en una lista para cada fila de la columna en cuestión, puede resultar útil ejecutar un sapply con la función de selección `[`:

     i. `dataset$http_req_type <- sapply(http_res_prot, '[', 1)`
     ii. `dataset$req_resource <- sapply(http_res_prot, '[', 2)`

   - Para poder excluir de las observaciones aquellas peticiones relativas a archivos de tipo imagen, es recomendable usar la función filter de la librería dplyr, por ejemplo con la función grepl, la cual devuelve un booleano según si el contenido de la columna en cuestión hace *match* con la expresión regular. Así pues, basta con crear una regex que haga match con distintos tipos de extensiones comunes para imágenes.
     i. `!grepl(pattern = ".*[png|jpg|gif|ico]$", x = req_resource)`

Visualización de Resultados

4. **Generar un gráfico que permita visualizar las respuestas del servidor, es decir, la distribución de peticiones según el código de respuesta de esta. Probad distintos tipos de gráficos (por lo menos 2 distintos e incluid estos en el documento RMarkdown).**

Clústering de datos

5. **Utilizando un algoritmo de aprendizaje no supervisado, realizad un análisis de clústering con k-means para los datos del servidor.**

   - Para este análisis debéis repetir la ejecución del modelado con distintos valores de *k* (número de clústeres) con al menos 2 valores diferentes de k.
   - A fin de retener algo de información sobre el recurso servido, generad una columna numérica derivada de esta con el número de caracteres de la URL servida.

   Pista:
   La función one_hot de la librería mlops permite convertir fácilmente una columna factor a distintas columnas numéricas que representan el valor de la variable factor de forma que pueda usarse en algoritmos que trabajan únicamente con valores numéricos.

```
library(mltools)
library(data.table)
epa_http_one_hot <- one_hot(as.data.table(epa_http), sparsifyNAs = TRUE)
```

   Descartad todas aquellas columnas que no sean numéricas y por lo tanto no pueden usarse con el algoritmo k-means

   La función de k-means no funciona con NAs, aseguraros que los datos que tienen NA son utilizados siempre que sea posible, por ejemplo imputando el número de bytes servidos si por ejemplo sabemos que en el caso de 404 se deniega el acceso al recurso (0 bytes servidos).

6. **Representad visualmente en gráficos de tipo scatter plot el resultado de vuestros clústering.**
   Interpretad el resultado obtenido (documentad las características de los distintos grupos) con los 2 valores distintos de k probados en el apartado anterior en función de los valores de las variables y el número de clúster asignado.
   Pista:
   Para hacer la interpretación, podéis probar generando gráficos con las distintas combinaciones de variables en los ejes x e y. Alternativamente, buscad X casos aleatorios para cada clúster e intentad descifrar la segmentación ofrecida por k-means.

Formato de Entrega

Generación del correspondiente al análisis de los registros del servidor mediante un documento de tipo *reproducible research*, utilizando el formato RMarkdown (.Rmd) visto en clase con el contenido de cada una de las preguntas anteriores con el enunciado, el código utilizado para la obtención de los resultados, acompañamiento de cualquier gráfico adicional contemplado en el enunciado de cada pregunta y la documentación requerida.

- Para la entrega final, basta con crear un documento RMarkdown, por ejemplo, haciendo uso de la opción disponible en RStudio para generar el esqueleto de un fichero *.Rmd*, y a continuación, añadir una sección para cada una de las preguntas anteriormente resueltas.

- **Finalmente, el repositorio de código de Github debe incluir el fichero RMarkdown creado y el resultado de renderizar el documento en formato HTML o PDF.** Adicionalmente, si se desea se puede añadir un nuevo documento de código con extensión *.R* con cualquier otra función que se desee incorporar.