Universidad de Santiago de Compostela

Trabajo Fin de Máster en Técnicas Estadísticas

Análisis de supervivencia para la detección y clasificación de anomalías en vehículos

Autor:
Pablo Menéndez Trillo

Tutores:
Fabián Otero Vázquez
Jacobo de Uña Álvarez

3 de junio de 2024

Índice general

Resumen

Resumen
Este trabajo ilustra el uso de técnicas de Análisis de Supervivencia para la detección y clasificación de anomalías en vehículos. Su objetivo principal es el desarrollo de algoritmos que permitan detectar intrusiones en vehículos con alta efectividad y con una baja tasa de falsas alertas, así como clasificarlas entre posibles ataques conocidos, todo con una alta rigurosidad estadística. Para ello se hará hincapié en cómo sus parámetros influyen en la distribución de las variables de interés y, por ende, en las precisiones de detección. Se tratará de modelizar el problema con parámetros que jueguen un papel conocido, permitiendo así un pleno entendimiento de la naturaleza de los resultados.

Abstract
This work illustrates the use of Survival Analysis techniques for the detection and classification of anomalies in vehicles. Its main objective is the development of algorithms that allow detecting intrusions in vehicles with high effectiveness and with a low rate of false alerts, as well as classifying them among possible known attacks, all with high statistical rigor. To do this, emphasis will be placed on how its parameters influence the distribution of the variables of interest and, therefore, the detection accuracies. An attempt will be made to model the problem with parameters that play a known role, thus allowing a full understanding of the nature of the results.

Introducción

Hoy en día la digitalización está presente en todos los ámbitos que nos podamos imaginar, y los vehículos no son una excepción. Hace menos de 50 años la proporción de componentes electrónicas frente a mecánicas en los vehículos era inferior al 1 %. Hoy esta cifra se ha elevado al 50 %, y seguirá aumentando con la proliferación de los vehículos eléctricos y la aparición de los vehículos autónomos. La comunicación del vehículo con el exterior aporta facilidades para el día a día del conductor (GPS, *bluetooth*, Wi-Fi...), pero también crea vulnerabilidades que pueden ser utilizadas en su contra en forma de ciberataques. En 2012, investigadores de la Universidad de Corea mostraron varios procedimientos para burlar la seguridad de diversos automóviles con los que consiguieron manipular el panel de control y el sistema de alarma introduciendo un *malware* en una aplicación de inspección de vehículos, [1]. Ese mismo año se estima que más de 300 BMWs fueron robados en Reino Unido eludiendo el sistema antirrobo mediante *smartphones* conectados al puerto On-Board Diagnostic (OBD), [2]. En 2015, un equipo del Keen Security Lab en China mostró cómo conectar un vehículo a una red Wi-Fi maliciosa, tomando el control del sistema de cierre y navegación, [3]. Estos son solo unos ejemplos de las múltiples ocasiones en las que en los últimos años se han dejado en evidencia las vulnerabilidades electrónicas de nuestros vehículos. Es por ello una necesidad el desarrollo de técnicas de ciberseguridad capaces de detectar, clasificar y mitigar estos ataques. En este trabajo propondremos modelos estadísticos de detección y clasificación de anomalías en vehículos que se distinguirán de los estudios previamente realizados por su rigurosidad estadística. Esto no solo nos permitirá detectar y clasificar ataques, sino que también nos hará adentrarnos en la estructura de los datos y nos hará descubrir ciertos aspectos sobre cómo han sido generados.

El trabajo se dividirá en cuatro capítulos. El Capítulo 1 servirá como una acercamiento al funcionamiento de la red interna vehicular, presentándose el protocolo CAN, sus ventajas y vulnerabilidades. También se introducirá el Análisis de Supervivencia, se hablará sobre las métricas utilizadas para eva-

luar un método de clasificación y se presentarán las bases de datos utilizadas durante el estudio. En el Capítulo 2 se desarrollarán diferentes algoritmos de detección de anomalías. Se compararán sus resultados y se seleccionará un algoritmo sobre el que se optimizarán las precisiones con respecto a sus parámetros. Además, también se propondrá un modelo de clasificación capaz de distinguir entre diferentes ataques. En el Capítulo 3 se profundizará en la estructura de los datos, que como veremos dará lugar a incoherencias con la Teoría de la Probabilidad. Finalmente, en el Capítulo 4 se concluirá el trabajo.

Capítulo 1

Preliminares

En las siguientes secciones se hará una breve introducción a la red de comunicación de los dispositivos en los vehículos, se explicará de donde proceden estas vulnerabilidades y se expondrá el estado del arte de su detección. También se explicarán los conceptos básicos del Análisis de Supervivencia necesarios para comprender este estudio, y se presentarán las métricas que utilizaremos para evaluar nuestros algoritmos de detección y clasificación. Por último, se mostrarán las bases de datos utilizadas durante el trabajo y las especificaciones del PC donde se han desarrollado los experimentos.

1.1. Red interna vehicular y sus vulnerabilidades

Los dispositivos electrónicos de los vehículos están organizados por unidades de control electrónico (ECUs). Para su correcto funcionamiento, cada ECU debe estar en constante comunicación con el resto. El protocolo estándar de comunicación entre ECUs en los vehículos es el Controller Area Network (CAN), barato, ligero, rápido y de sencilla instalación. Se trata de un protocolo basado en mensajes que conecta en paralelo todas las ECUs a través de un bus CAN, dando lugar a un sistema multimaestro en el que ninguna ECU controla a las demás. Cada ECU interacciona con el resto recibiendo y enviando mensajes, formándose una red interna de comunicación. Cada mensaje está estructurado como se muestra en la Figura 1.1, y sus campos se describen a continuación:

- *Start of frame* (SOF): indica el comienzo del mensaje.
- *Arbitration field* (CAN-ID): identificador del mensaje, indica su prioridad dentro de la red. Cuando dos mensajes intentan transmitirse simultánea-

mente, tendrá prioridad el mensaje con menor ID. Cada ECU tiene asignados varios IDs, pero cada ID solo puede ser utilizado por una ECU.

- *Control field* (DLC): indica la longitud del *data field*.
- *Data field*: contiene la información del mensaje, es decir, la información que la ECU emisora quiere transmitir a la ECU receptora.
- *Cyclic redundancy code field* (CRC): también conocido como campo de seguridad, comprueba la validez del mensaje.
- *Ackonwledge field* (ACK): comprueba que el mensaje llega a la ECU receptora.
- *End of frame* (EOF): indica el final del mensaje

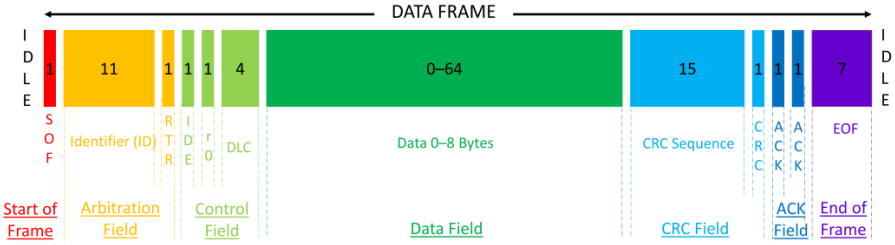

Figura 1.1: Figura tomada de [4]. Estructura de los mensajes del protocolo CAN.

A pesar de las ventajas con las que cuenta el protocolo CAN, también presenta vulnerabilidades que pueden ser utilizadas por los ciberdelincuentes para introducirse y tomar parte de la red interna del vehículo:

- **Dominio de difusión**: el bus CAN es un dominio de difusión, por lo que todas las ECUs tienen acceso a todos los mensajes de la red.
- **No encriptación**: la velocidad a la que son enviados y recibidos los mensajes no permite su encriptación. Junto al punto anterior, esto puede ser utilizado para acceder al sistema conectándose desde una ECU maliciosa.
- **No autenticación**: el bus CAN no cuenta con un sistema de autenticación de mensajes, por lo que nada impide a una ECU transmitir mensajes con IDs que no le pertenezcan. Esto puede ser utilizado por una ECU maliciosa para suplantar la identidad de las ECUs legítimas y enviar mensajes que engañen o colapsen el sistema.

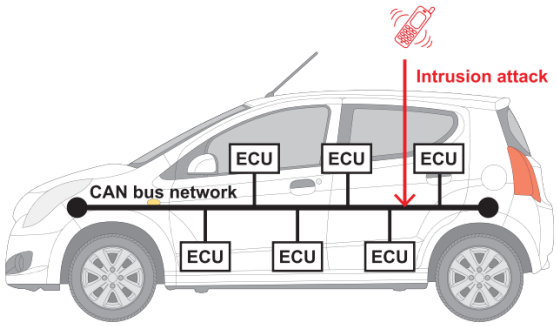

Figura 1.2: ECU maliciosa conectada a una red de comunicación bus CAN, [5].

Basados en estas vulnerabilidades, existen diferentes ataques posibles sobre la red interna de los vehículos. Todos ellos consisten en la conexión al sistema de una ECU maliciosa que toma ciertas acciones:

- ***Sniffing attack***: intercepción de la red de mensajes del vehículo.
- ***Flooding attack* (DDoS)**: inyección de mensajes de máxima prioridad que colapsan la red, impidiendo la transmisión del resto de mensajes. Estos ataques son muy populares y también son conocidos como *Distributed Denial of Service* (DDoS).
- ***Fuzzy attack***: inyección de mensajes con IDs, DLC y *data fields* aleatorios, pertenecientes y no pertenecientes a la red interna del vehículo, que se transmiten como si mensajes legítimos se trataran. Para generar estos mensajes los atacantes pueden basarse en la información obtenida a partir de un ataque *sniffing*, o pueden hacerlo sin información previa de la red interna.
- ***Malfunction attack***: inyección de mensajes con un ID concreto perteneciente a la red interna, con el *data field* modificado para contaminar el sistema.
- ***Replay attack***: inyección de mensajes reales extraídos previamente de la red de mensajes a través de un ataque *sniffing*.

1.2. Estado del arte de la detección de anomalías en vehículos

En los últimos años se han desarrollado muchos estudios con el objetivo de detectar estos ataques, basados la mayoría de ellos en técnicas de IA y Machine Learning, tanto aprendizaje supervisado como no supervisado. En [6] se hace un estado del arte muy completo de este tipo de estudios, diferenciando tres tipos de algoritmos de detección en función de su enfoque:

- **Modelos basados en los IDs**. Basándose en la ordenación o en la frecuencia de los mensajes según su ID, estos modelos diferencian entre una conducción libre o una conducción atacada. Se han utilizado una amplia gama de modelos de machine learning: en [7] y [8] redes neuronales convolutivas, en [9] y [10] modelos Long Short-Term Memory (LSTM), en [11] modelos secuenciales, en [12] One Class Support Vector Machines (OCSVM), en [5] análisis de componentes principales y árboles kd... En ellos se ha observado en mayor o menor medida que la secuencia de IDs en los mensajes es una variable útil para detectar ataques *flooding*, *fuzzy*, *malfunction* y *replay*.

- **Modelos basados en el *data field***. Algunos ataques, como el *fuzzy* o el *malfunction*, no solo inyectan mensajes, sino que también modifican su *data field*. Es posible detectar un ataque comparando el *data field* de los nuevos mensajes con el de los mensajes legítimos. En este enfoque se han aplicado muchos modelos de redes neuronales, como en [13] y [14] que se utilizan modelos LSTM, en [15] y [16] autoencoders y en [17] y [18] Gated Recurrent Units. También se han aplicado otros modelos, como en [19], por ejemplo, que se utilizan máquinas de soporte vectorial, o en [20] que se utilizan clasificadores k-nearest neighbors. Algunos de estos modelos han conseguido detectar con precisión varios ataques, aunque en general presentan más desventajas que los modelos de detección basados en los IDs, como por ejemplo un mayor tiempo de respuesta.

- **Modelos basados en el mensaje completo**. Además de modelos basados en IDs y en el *data field*, se han desarrollado modelos basados en ambos e incluso también en otros atributos, como el DLC. Destacan, entre otros, [21], donde se aplican modelos de redes neuronales, LSTM, SVM y OCSVM, y [22], donde se aplican modelos Random Forest, bagging, boosting, Naive Bayes y regresión logística.

Además de modelos de machine learning, también se han desarrollado otros modelos con un enfoque más cercano a la estadística clásica. Por ejemplo, en [23] se presenta un modelo que compara el ratio de aparición de cada ID entre una base libre y en las bases atacadas, detectando ataques cuando estos ratios se salen de ciertos límites. En [4] se hace un estudio similar, comparando también la desviación típica de los ratios de aparición. En ambos casos se obtienen precisiones muy buenas en la detección de ataques *flooding*, *fuzzy* y *malfunction*. No obstante, estos trabajos profundizan poco en los desarrollos estadísticos formales que son necesarios para una correcta comprensión del problema y estudio de soluciones óptimas. A diferencia de los modelos basados en técnicas de machine learning, que quedan a merced de los datos, nuestros algoritmos serán construidos con un conocimiento previo o sospecha de cómo se comportan las variables en diferentes situaciones, de manera que puedan detectar diferencias entre una conducción libre y una conducción con intrusiones. Además, la mayoría de los modelos de detección de anomalías en vehículos de la literatura están basados en el conteo o cálculo del ratio de aparición de los mensajes, variables discretas. Nosotros consideraremos el tiempo entre mensajes, una variable continua. Aunque están muy relacionadas, el tiempo entre mensajes contiene más información, además de permitirnos hacer uso de las técnicas del Análisis de Supervivencia.

1.3. Análisis de Supervivencia

El Análisis de Supervivencia, [24], estudia modelos y métodos en los que la variable respuesta representa el tiempo hasta un evento perfectamente especificado. Tiene aplicaciones en multitud de ramas: en biología y medicina la respuesta suele representar el tiempo desde el nacimiento de un individuo hasta su muerte, el tiempo desde la entrada de un enfermo en un tratamiento hasta su recuperación, o el tiempo desde el diagnóstico de una enfermedad hasta la muerte. También es frecuentemente utilizado en ingeniería, donde también es conocido como Análisis de Fiabilidad y suele medir el tiempo de duración de las máquinas desde su instalación, o en economía, donde suele medir el tiempo en paro o la duración de empleo. Es usual en Análisis de Supervivencia encontrarnos con datos censurados por la derecha, para los cuales no se observa el evento final debido a un evento previo que lo impide. Esto puede darse, por ejemplo, si un estudio termina antes del fallecimiento o dada de alta de algún individuo. Estos datos censurados contienen información, no completa pero valiosa, y se han desarrollado técnicas estadísticas para sacar el máximo partido de dicha información.

La función principal del Análisis de Supervivencia que caracteriza la variable temporal es la función de supervivencia, que representa la probabilidad de sobrevivir al tiempo t. Conocer esta función es equivalente a conocer la función de distribución F o la función de densidad f. Si Y denota la variable tiempo de fallo, la función de supervivencia S se define como:

$$S(t) = \mathbb{P}(Y > t) \qquad \text{y satisface:} \qquad S(t) = 1 - F(t) = \int_t^\infty f(u)du, \;\; t \geq 0, \tag{1.1}$$

El objetivo del Análisis de Supervivencia es modelizar, estimar y comparar curvas de supervivencia. Otra función importante es la función de riesgo λ, que se define como:

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{\mathbb{P}(t \leq Y < t + \Delta t | Y \geq t)}{\Delta t} \qquad \text{y satisface} \qquad \lambda(t) = \frac{f(t)}{S(t)}. \tag{1.2}$$

Representa la probabilidad de que un individuo, habiendo sobrevivido a tiempo t, experimente el fallo en el instante inmediato $[t, t + \Delta t)$. Conocer λ es equivalente a conocer f o a conocer S, todas ellas caracterizan la distribución.

Dada una muestra de tiempos de fallo, existen dos posibilidades para estimar la función de supervivencia. Por un lado, existen métodos paramétricos basados en la suposición de que la variable Y sigue una distribución conocida. Usualmente se suelen considerar las distribuciones de Weibull, exponencial, log-normal, log-logística... (cualquiera con soporte positivo) y se estiman sus parámetros mediante máxima verosimilitud. Por otro lado, existen métodos no paramétricos, siendo el estimador de Kaplan-Meier el más conocido. Este estimador se calcula como:

$$\hat{S}(t) = \prod_{t_i \leq t} \left(1 - \frac{d_i}{n_i}\right), \quad t \geq 0, \tag{1.3}$$

donde $t_1 < \ldots < t_k$ son los tiempos de fallo observados, d_i el número de fallos en t_i y n_i el número de individuos a riesgo en t_i. Cuando no hay datos censurados el estimador de Kaplan-Meier se reduce al estimador empírico usual de la función de distribución:

$$\hat{S}(t) = 1 - \hat{F}(t) = \frac{1}{n} \sum_{i=1}^{n} I\{Y_i > t\}, \tag{1.4}$$

donde $Y_1, \ldots, Y_n$ es una muestra aleatoria simple de Y.

A lo largo de este trabajo nos interesará comparar si dos muestras independientes proceden de la misma distribución. Las pruebas más utilizadas son la prueba t de Student bajo la suposición de normalidad e igualdad de varianzas, la prueba de Welch bajo normalidad y desigualdad de varianzas y el test

de Gehan-Wilcoxon si no se cumple la suposición de normalidad. En Análisis de Supervivencia las pruebas más conocidas son las pertenecientes a la familia log-rank, que aceptan la presencia de datos censurados. Estas pruebas de comparación de distribuciones se suelen formular como sigue:

$$\begin{array}{l} Y_1,\ldots,Y_m \text{ i.i.d. } Y\sim F_Y, F_Y \text{ continua,} \\ Z_1,\ldots,Z_n \text{ i.i.d. } Z\sim F_Z, F_Z \text{ continua,} \\ Y_1,\ldots,Y_m \text{ y } Z_1,\ldots,Z_n \text{ independientes,} \end{array} \quad \begin{cases} H_0 : F_Y(x) = F_Z(x) \text{ para todo } x, \\ H_1 : F_Y(x) \neq F_Z(x) \text{ para algún } x, \end{cases} \tag{1.5}$$

aunque en Análisis de Supervivencia es más frecuente formularlas mediante las funciones de riesgo:

$$\begin{cases} H_0 : \lambda_Y(t) = \lambda_Z(t) \text{ para todo } t \geq 0, \\ H_1 : \lambda_Y(t) \neq \lambda_Z(t) \text{ para algún } t \geq 0, \end{cases} \tag{1.6}$$

Las pruebas log-rank se basan en la comparación de las estimaciones de las funciones de riesgo de cada grupo, y sus estadísticos se construyen como sigue. Sean $t_1 < t_2 < \ldots < t_D$ los tiempos donde ocurren fallos en la muestra conjunta de Y y Z. En cada instante t_i medimos:

- n_{i1} y n_{i2}: número de individuos a riesgo en t_i en los grupos Y y Z respectivamente.
- $n_i = n_{i1} + n_{i2}$: número de individuos a riesgo en t_i en la muestra conjunta.
- d_{i1} y d_{i2}: número de fallos en t_i en los grupos Y y Z respectivamente.
- $d_i = d_{i1} + d_{i2}$: número de fallos en t_i en la muestra conjunta.

Bajo la hipótesis nula, para todos los tiempos se espera que los riesgos observados en cada grupo sean similares entre sí y similares al riesgo observado en la muestra conjunta. Es decir, se espera que $\frac{d_{i1}}{n_{i1}} \approx \frac{d_i}{n_i} \approx \frac{d_{i2}}{n_{i2}}$, $i = 1,\ldots,D$, lo que sugiere construir los estadísticos:

$$U_k = \sum_{i=1}^{D} w_k(t_i)\left(\frac{d_{ik}}{n_{ik}} - \frac{d_i}{n_i}\right), \qquad k = 1,2, \tag{1.7}$$

que miden las diferencias entre los riesgos de Y y Z y el riesgo conjunto ponderadas por las funciones peso $w_k(t_i)$. Estas funciones se suelen escribir como: $w_k(t_i) = n_{ik}w(t_i)$, donde $w(t_i)$ es una función peso común, de manera que:

$$U_k = \sum_{i=1}^{D} w(t_i)\left(d_{ik} - n_{ik}\frac{d_i}{n_i}\right) = \sum_{i=1}^{D} w(t_i)(o_{ik} - e_{ik}), \qquad k = 1,2. \tag{1.8}$$

Nótese que $U_1 + U_2 = 0$, por lo que U_1 y U_2 contienen la misma información. Así, los estadísticos U_k miden la diferencia entre los eventos observados (o_{ik}) y esperados (e_{ik}) en cada tiempo t_i ponderada por una función peso $w(t_i)$. La función peso determina a qué partes de la distribución se le quiere dar más importancia. Los dos casos con los que trataremos son:

- $w(t_i) = 1$. Es el test log-rank usual, de máxima potencia bajo el supuesto de funciones de riesgo proporcionales: $\lambda_Z(t) = \theta\lambda_Y(t)$. Cuando las funciones de riesgo no son proporcionales puede dar lugar a resultados que no se correspondan con la realidad.

- $w(t_i) = n_i$. Recibe el nombre de test de Gehan, generalización del test de Mann-Whitney-Wilcoxon para muestras censuradas. Es el test de máxima potencia bajo el supuesto de diferencias en localización: $\lambda_Z(t) = \lambda_Y(t-\theta)$. Cuando las diferencias no sean de localización puede dar lugar a resultados que no se correspondan con la realidad.

En ambos casos los estadísticos satisfacen:

$$\frac{U_k}{\sqrt{Var(U_k)}} \rightsquigarrow \mathcal{N}(0,1) \Leftrightarrow \frac{U_k^2}{\sqrt{Var(U_k^2)}} \rightsquigarrow \chi_1^2, \qquad k = 1, 2. \tag{1.9}$$

Habiendo fijado el nivel de significación α, el test se rechaza para valores de $\frac{U_1^2}{\sqrt{Var(U_1^2)}}$ superiores al percentil $(1-\alpha)$ de la χ_1^2.

1.4. Evaluación de un método de clasificación

Nuestros algoritmos tendrán como objetivo detectar anomalías, por lo que dado un conjunto de datos deberán ser capaces de distinguir si son lícitos o si han sido inyectados maliciosamente. En otras palabras, nuestros algoritmos serán métodos de clasificación que agruparán conjuntos de datos en dos clases: **alerta** y **no alerta**. Para evaluar un método de clasificación es necesario conocer las etiquetas reales de los datos. En nuestro caso, cada conjunto de datos sometido a la predicción de su clase es etiquetado previamente como **ataque** o **no ataque**. Estas etiquetas se asignan a ciencia cierta de acuerdo con lo observado en la recogida de los datos. Una vez aplicado el método de clasificación, a partir de las observaciones y predicciones se obtiene la matriz de confusión del Tabla 1.1. En nuestro contexto:

- TN (*true negative*): se predice **no alerta** para un conjunto etiquetado como **no ataque**.

Observado\Predicción	**No alerta**	**Alerta**
No ataque	TN	FP
Ataque	FN	TP

Tabla 1.1: Matriz de confusión en el contexto de predicción (ver texto).

- FN (*false negative*): se predice **no alerta** para un conjunto etiquetado como **ataque**.
- FP (*false positive*): se predice **alerta** para un conjunto etiquetado como **no ataque**.
- TP (*true positive*): se predice **alerta** para un conjunto etiquetado como **ataque**.

A partir de esta matriz de confusión, existen diferentes métricas que miden la precisión del método. Véase por ejemplo [25] para una introducción al contexto de la evaluación de métodos de clasificación y predicción. A continuación se presentan las métricas que utilizaremos en este trabajo junto a su significado en nuestro contexto.

- **Sensibilidad** (*sensibility*, *recall*, *true positive rate*; TPR): mide la proporción de ataques detectados con respecto a los ataques observados.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}. \tag{1.10}$$

  La sensibilidad valdrá 0 si ningún ataque es alertado y valdrá 1 si todos los ataques son alertados.

- **Especificidad** (*specificity*, *true negative rate*; TNR): mide la proporción de no ataques predichos con respecto a los no ataques observados.

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN+FP}}. \tag{1.11}$$

  La especificidad valdrá 0 si todos los no ataques son alertados y valdrá 1 si ningún no ataque es alertado. El valor FPR = 1-TNR es conocido como *false positive rate*.

- **Valor predictivo positivo** (*precision*, *positive predictive value*; PPV): mide la proporción de ataques detectados con respecto a las alertas dadas.

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \tag{1.12}$$

Puede ser utilizado como una estimación de la probabilidad de ataque cuando se da alerta.

- **Precisión** (*accuracy*; ACC): mide la tasa de aciertos global.

$$\text{ACC} = \frac{\text{TP+TN}}{\text{P+N}} = \frac{\text{TP+TN}}{\text{TP+FN+TN+FP}}. \tag{1.13}$$

La precisión valdrá 0 cuando no haya ningún acierto y 1 cuando no haya ningún fallo. Esta métrica no es la más adecuada para medir la precisión global del método cuando las clases están desbalanceadas o cuando es más importante detectar la clase positiva que la negativa. Para las bases de datos utilizadas en este estudio, ver Capítulo 2, se darán ambas circunstancias, ya que tendremos muchos más datos no atacados que atacados y será más grave no detectar un ataque que dar una falsa alarma. En estas situaciones resulta más conveniente utilizar la siguiente medida.

- F_1-score (**F-score**): media armónica de la sensibilidad y del valor predictivo positivo.

$$\text{F} = \frac{2}{\frac{\text{TP+FN}}{\text{TP}} + \frac{\text{TP+FP}}{\text{TP}}} = \frac{2\text{TP}}{2\text{TP+FN+FP}}. \tag{1.14}$$

Al igual que la sensibilidad, el F-score valdrá 0 si ningún ataque es alertado, y al igual que la precisión, el F-score valdrá 1 si no hay ningún fallo.

- **AUC** (*area under ROC curve*): como se muestra en la Figura 1.3, las curvas ROC se construyen representando el TPR frente al FPR (sensibilidad frente a 1-especificidad) en función de un parámetro del modelo. Dicho parámetro ha de hacerse variar en un intervalo para el cual se den las dos situaciones límite en las que todos los datos sean clasificados en la misma clase.

Por ejemplo, si el modelo de clasificación se basa en un test de hipótesis, dicho parámetro podría ser el nivel de significación $\alpha \in [0, 1]$, de manera que si $\alpha = 0$ la sensibilidad valdrá 0 y la especificidad 1, y si $\alpha = 1$ ocurrirá lo contrario.

Como muestra la Figura 1.3, la curva ROC de un método de clasificación aleatorio se aproxima a la diagonal, y cuanto más se acerque la curva a la esquina superior izquierda mejor será el método. La curva ROC de un clasificador perfecto pasa por dicho punto, para el cual la precisión es del 100 %. Así, cuanto más se acerque a 1 el área bajo la curva, AUC, mejor será el método de clasificación.

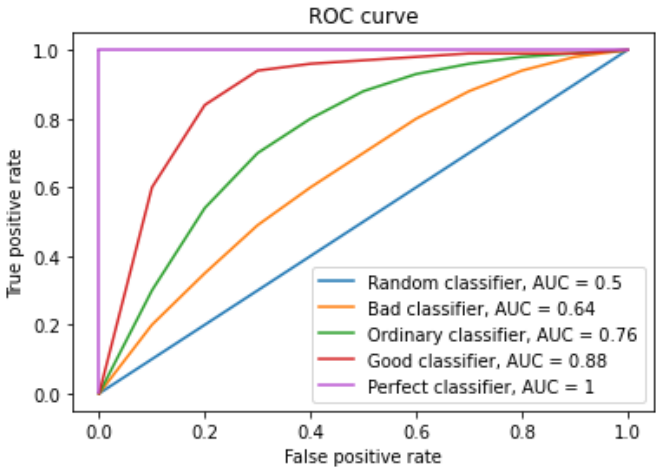

Figura 1.3: Ejemplos de curvas ROC.

1.5. Bases de datos utilizadas

Basaremos nuestro estudio en dos bases de datos. Por un lado, *Survival Analysis Dataset for Automobile IDS* (Survival), [26], utilizada en *2019 Information Security R&D Dataset Challenge* en Corea del Sur, que contiene datos sobre la red de mensajes de tres vehículos sometidos a ataques *flooding*, *fuzzy* y *malfunction*. Por otro lado, *Car Hacking Dataset* (Car-Hacking), [27], utilizada en *Car Hacking: Attack & Defense Challenge 2020* también en Corea del Sur y que recoge datos de un vehículo adicional sometido a ataques *flooding*, *fuzzy*, *malfunction* y *replay*.

1.5.1. Base de datos Survival

La base de datos Survival [26] reúne la comunicación entre los dispositivos de tres vehículos: KIA Soul, HYUNDAI Sonata y CHEVROLET Spark, para cada uno de los cuales contamos con cuatro conjuntos de datos: un conjunto de datos libre de ataques y tres conjuntos de datos recogidos bajo la inyección, respectivamente, de ataques *flooding*, *fuzzy* y *malfunction*. Por un lado, los conjuntos libres de ataques han sido recogidos bajo una conducción 'normal' real de cada vehículo. Por otro lado, los ataques han sido simulados previamente e inyectados periódicamente durante la conducción. Tanto la recogida como la inyección de los datos han sido realizadas a través de un puerto intravehicular OBD-II. Para los tres vehículos los ataques *flooding* se basaron en la inyección del mensaje de máxima prioridad con ID 0x000. Durante los ataques *fuzzy* se in-

yectaron mensajes aleatorios, tanto mensajes pertenecientes a la red de IDs del vehículo como mensajes externos. Por último, durante los ataques *malfunction* se inyectaron los mensajes con IDs 0x153, 0x316 y 0x18E para los vehículos Soul, Sonata y Spark respectivamente.

En el el Tabla 1.2 se muestra el número de mensajes contenidos en cada conjunto de datos, y en el Tabla 1.3 se muestra, como ejemplo, la estructura de los datos para el ataque *flooding* en el KIA Soul. Vemos como para cada mensaje disponemos del instante en el que es emitido, su ID, el *control field* (DLC), el *data field* y una etiqueta (Et.). Dicha etiqueta toma el valor R para mensajes legítimos y T para mensajes malignos, es decir, constituyentes de un ataque. En la Figura 1.4 se muestra la inyección de los ataques para cada vehículo y para cada conjunto de datos. Como ya se ha comentado, los mensajes malignos se introducen entre mensajes legítimos, lo que hace que mensajes con etiqueta R aparezcan en todo el rango de tiempos, incluso durante los propios ataques.

	Soul		**Sonata**		**Spark**	
Base	*Free*	*Attack*	*Free*	*Attack*	*Free*	*Attack*
Libre	192516	0	117173	0	136934	0
Flood.	139788	33141	109931	32422	52632	22587
Fuzzy	197539	39812	110332	18103	32092	5812
Malf.	155974	7401	109511	15974	38402	8047

Tabla 1.2: Número de mensajes en cada conjunto de datos. Base Survival.

Mens.	**Tiempo (s)**	**ID**	**DLC**	***Data field***	**Et.**
1	1513921815.350702	0x081	8	7F 84 69 00 00 00 00 69	R
2	1513921815.350959	0x165	8	08 E8 7F 00 00 00 01 9E	R
3	1513921815.351189	0x18F	8	00 2B 1F 00 00 3F 00 00	R
⋮	⋮	⋮	⋮	⋮	⋮
70949	1513921849.300626	0x081	8	7F 84 69 00 00 00 00 3C	R
70950	1513921849.301111	0x000	8	00 00 00 00 00 00 00 00	T
70951	1513921849.301353	0x260	8	05 22 00 30 FF 8F 6B 0E	R
⋮	⋮	⋮	⋮	⋮	⋮

Tabla 1.3: Conjunto de datos *flooding* para el KIA Soul. Base Survival.

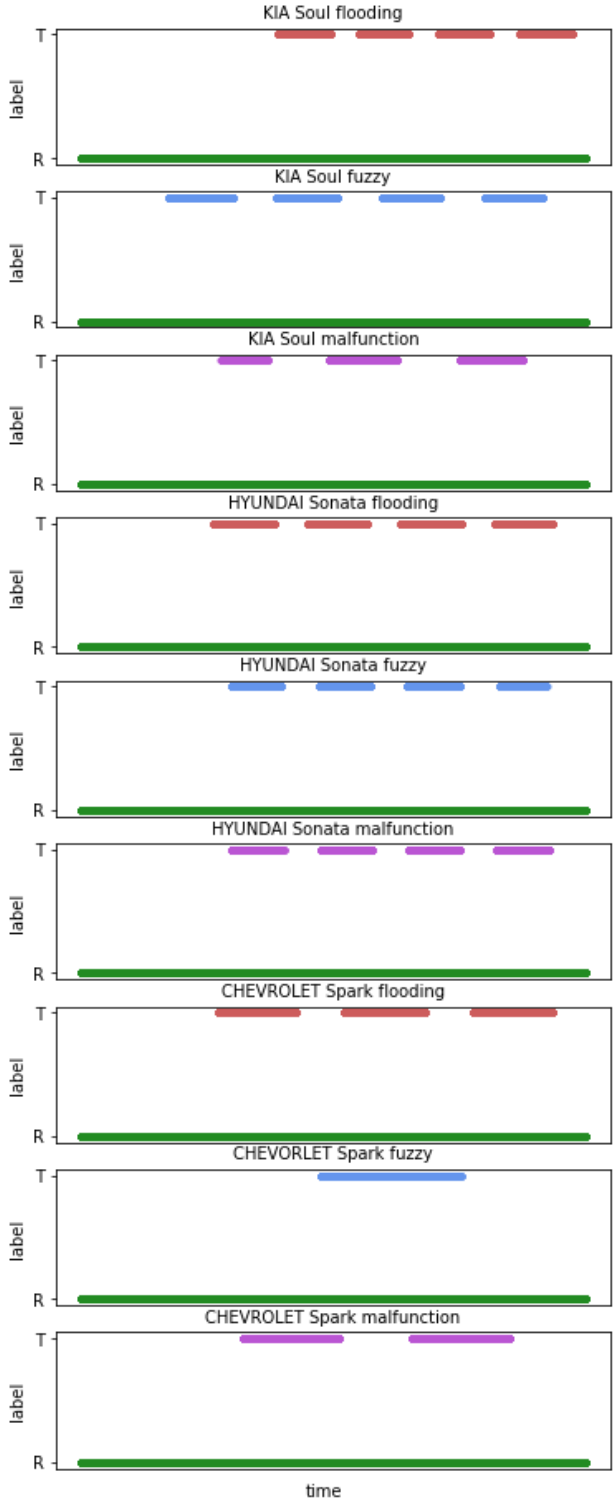

Figura 1.4: Inyección de los ataques para cada vehículo. Base Survival.

1.5.2. Base de datos Car-Hacking

La base de datos Car-Hacking [27] cuenta con una base libre y dos bases atacadas, cada una con ataques *flooding*, *fuzzy*, *malfunction* y *replay*, recogidas para un vehículo HYUNDAI Avante CN7. A diferencia de la base Survival, en este caso en cada una de las bases atacadas contamos con los cuatro ataques mencionados, tal y como se muestra en la Figura 1.5. El proceso de recogida, simulación e inyección de los mensajes ha sido similar al de la base Survival. De nuevo los ataques *flooding* consistieron en la inyección del mensaje de máxima prioridad ID 0x000 y los *fuzzy* en la inyección de mensajes aleatorios, pertenecientes y no pertenecientes a la red del vehículo. Los ataques *malfunction* en este caso se centraron simultáneamente en dos IDs: 0x553 y 0x366. Por último, los ataques *replay* copiaron e inyectaron mensajes de un 90 % de los IDs de la red vehicular. En el Tabla 1.4 se muestra el número de mensajes contenidos en cada conjunto de datos.

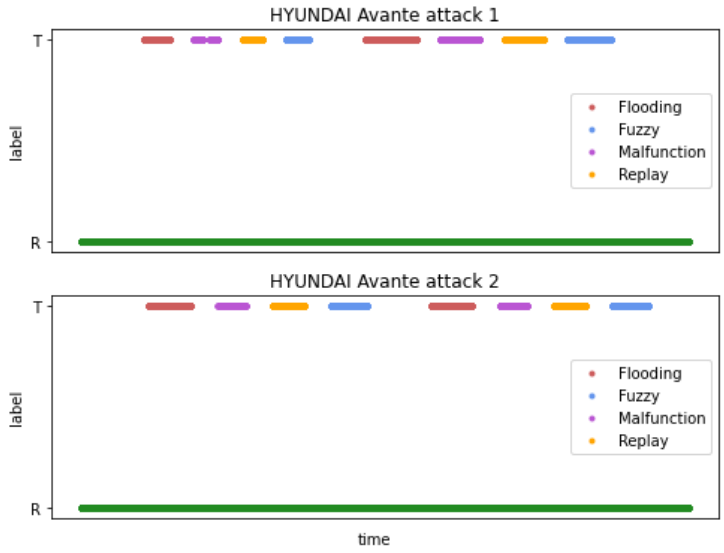

Figura 1.5: Inyección de los ataques. Base Car-Hacking.

1.6. *Setup* experimental

Todos los experimentos llevados a cabo en este trabajo han sido realizados en un PC con las especificaciones mostradas en la Tabla 1.5.

Avante	*Free*	*Flood.*	*Fuzzy*	*Malf.*	*Replay*
Base libre	179 346	0	0	0	0
Base atacada 1	733 752	38 521	22 620	988	10 509
Base atacada 2	811 532	38 852	22 854	2891	13 266

Tabla 1.4: Número de mensajes en cada conjunto de datos. Base Car-Hacking.

Especificaciones del PC	
Modelo de PC	Lenovo ThinkPad L13 Yoga Gen 2
CPU	11th Gen Intel Core i7-1165G7
Memoria	16 GB
Unidad gráfica integrada	Intel Iris Xe Graphics
Disco	476 GB
Sistema Operativo	Windows 10

Tabla 1.5: Especificaciones del PC utilizado en la realización de los experimentos en este estudio.

Capítulo 2

Algoritmos de detección y clasificación de anomalías en vehículos

En las siguientes secciones se desarrollarán los algoritmos de detección y clasificación de anomalías en vehículos. En primer lugar se presentarán dos algoritmos de detección basados en el tiempo entre mensajes de cada ID. El primero hará uso de un ID concreto de la red vehicular, mientras que el segundo hará uso de todos ellos. A continuación se presentarán otros tres algoritmos de detección basados en el tiempo entre mensajes sin distinguir su ID, teniendo como diferencia la base libre utilizada. Veremos cómo los resultados mejoran, lo que nos permitirá seleccionar uno de ellos para optimizar con respecto a sus parámetros. Estos algoritmos de detección se desarrollarán en base al conjunto de datos Survival. Finalizaremos aplicando el algoritmo seleccionado sobre la base Car-Hacking y presentando el algoritmo de clasificación.

2.1. Algoritmos de detección basados en el tiempo entre mensajes de cada ID

Nuestros primeros algoritmos se inspiran en los buenos resultados de la literatura al utilizar como variable de interés la frecuencia de los mensajes en función de su ID. En particular, nos basaremos en el estudio realizado en [23], en el que la detección de ataques se basa en la comparación del ratio de aparición de cada ID en la base libre frente al de las bases atacadas. El procedimiento seguido en dicho estudio es el siguiente: se divide tanto la base libre como la

base atacada en paquetes disjuntos de N mensajes. En la base libre, se cuenta el número de mensajes correspondientes a cada ID en cada uno de sus paquetes, y para cada ID, se construye un intervalo 'base' a partir del mínimo y máximo de apariciones en dichos paquetes. A continuación, ahora en la base atacada, se cuenta el número de mensajes correspondientes a cada ID en cada uno de sus paquetes. Un paquete es clasificado como no atacado si el recuento se encuentra dentro sus correspondientes intervalos, y atacado en caso contrario.

La filosofía de los dos algoritmos que se muestran a continuación es similar, siendo en nuestro caso la variable de interés el tiempo entre mensajes y no su conteo. Estas variables son inversamente proporcionales y poseen una estrecha relación, y es que la información proporcionada por el conteo está contenida en la proporcionada por el tiempo entre mensajes. La idea consiste en comparar la distribución de tiempos entre mensajes con un ID determinado en la base libre y en la base atacada, y detectar un ataque cuando las diferencias sean significativas. Resulta conveniente definir las variables aleatorias:

$$Y_i^x = \text{tiempo entre } x \text{ mensajes con ID} = i \text{ en la base libre}, \quad i \in I, \tag{2.1}$$

$$Z_i^x = \text{tiempo entre } x \text{ mensajes con ID} = i \text{ en la base atacada}, \quad i \in I, \tag{2.2}$$

donde I es el conjunto de IDs del vehículo en cuestión y el parámetro x determina el número de mensajes con ID $= i$ entre los que se calcula el tiempo. Por ejemplo, en el conjunto de datos mostrado en el Tabla 2.1, las variables Y_i^x tomarían los valores que se muestran a la derecha de la tabla.

Mens.	**Tiempo (s)**	**ID**
1	0.1	A
2	0.4	B
3	0.6	C
4	0.7	A
5	1.1	C
6	1.2	A
7	1.5	B
$\vdots$	$\vdots$	$\vdots$

$$Y_A^1 = \{t_4 - t_1 = 0.6, t_6 - t_4 = 0.5, \ldots\},$$

$$Y_A^2 = \{t_6 - t_1 = 1.1, \ldots\},$$

$$Y_B^1 = \{t_7 - t_2 = 1.1, \ldots\},$$

$$Y_C^1 = \{t_5 - t_3 = 0.5, \ldots\},$$

$$\ldots$$

Tabla 2.1: Conjunto de datos.

Para garantizar que los elementos de las muestras no contienen información repetida, los tiempos se calcularán entre posiciones disjuntas. Es decir, si por ejemplo $i = A$ y $x = 5$, el primer valor que tomaría la variable Y_A^5 se calcularía

como la diferencia entre el tiempo en el que aparece por sexta vez A y el tiempo en el que aparece por primera vez A, y el segundo valor se calcularía como la diferencia entre el tiempo en el que aparece por undécima vez A y el tiempo en el que aparece por sexta vez A (y no diferencia entre séptima vez y segunda vez). Aunque perdemos parte de la muestra, si no lo hiciéramos los datos no serían independientes y se perdería una suposición importante de los contrastes de hipótesis.

2.1.1. Algoritmo 1: tiempo entre mensajes para el ID = i

Comenzamos considerando un algoritmo que se aplica sobre cierto ID del vehículo en cuestión. Para un vehículo y un ataque dados, el algoritmo de detección se muestra en el Tabla 2.2.

Algoritmo de detección 1

1. Para la muestra de ataque, se crean N paquetes de tamaño L con saltos de tamaño j.
2. Los paquetes que cuenten con más de un porcentaje p de mensajes malignos son etiquetados como **ataque**. Los que no, son etiquetados como **no ataque**.
3. Se fija el parámetro x y se determina el conjunto I^* de IDs de interés en base al criterio de aparición de más de $10 \cdot x$ veces. Se fija el ID sobre el que se aplica el algoritmo ID $= i \in I^*$.
4. Se calculan los valores que toma la variable Y_i^x en la base libre.
5. Se calculan los valores que toman las variables $Z_i^{x,n}$ para cada uno de los paquetes $n \in \{1, \ldots, N\}$.
6. Se realiza un test de comparación a nivel de significación α entre las distribuciones de Y_i^x y $Z_i^{x,n}$ para cada $n \in \{1, \ldots, N\}$

$$\begin{cases} H_0: & Y_i^x =^d Z_i^{x,n} \\ H_1: & Y_i^x \neq^d Z_i^{x,n}, \end{cases} \qquad n \in \{1, \ldots, N\}.$$

Los paquetes para los que H_0 se rechaza son clasificados como **alerta**. Los que no, son clasificados como **no alerta**.
7. Se obtiene la matriz de confusión entre alertas y ataques y, a partir de ella, se calculan las medidas de precisión de interés: sensibilidad, especificidad y F-score.

Tabla 2.2: Algoritmo de detección 1.

Los parámetros del algoritmo, mostrados en el Tabla 2.3, merecen algunos comentarios:

Parámetro	**Función**
i	ID sobre el que se aplica el algoritmo.
x	Número de mensajes con ID = i entre los que se calcula el tiempo.
L	Número de mensajes que componen cada paquete.
j	Salto entre paquetes.
N	Número de paquetes.
p	Porcentaje de mensajes malignos en un paquete a partir del cual es considerado como ataque.
test y α	Contraste de hipótesis utilizado y su nivel de significación.

Tabla 2.3: Parámetros del algoritmo de detección 1.

- Los parámetros L y j determinan la agrupación de los mensajes en paquetes. Por ejemplo si $L = 5000$ y $j = 1000$, los paquetes serían:

$$\{1, \dots, 5000\}, \{1001, \dots, 6000\}, \{2001, \dots, 7000\} \dots$$

- Habiendo fijado el ID i, los parámetros x y L determinan el tamaño de las muestras de tiempos $Z_i^{x,n}$. Si x es demasiado grande o L demasiado pequeño entonces el tamaño de las muestras no será suficiente para estimar con precisión sus funciones de densidad/supervivencia. Aunque lo más intuitivo sería fijar $x = 1$, el esfuerzo computacional de este algoritmo no nos lo permite, aumentando demasiado los tiempos de ejecución. Para este primer algoritmo fijaremos $\boldsymbol{x = 5}$.

  Por otro lado, el parámetro L ha de ser lo suficientemente pequeño para que entre ataques haya paquetes con todos sus mensajes legítimos (en la Figura 1.4, entre los intervalos rojos deben caber paquetes de tamaño L). Para este primer algoritmo fijaremos $\boldsymbol{L = 5000}$ y trabajaremos con los IDs que nos aporten un tamaño muestral en la base de datos libre de ataques superior a 10. Estos IDs, a cuyo conjunto denotaremos por I^*, serán aquellos que tomando un paquete de tamaño 5000 de la base libre aparezcan de media al menos $10 \cdot x$ veces. Así, en el punto 3 del algoritmo es necesario que ID = $i \in I^*$.

- El parámetro j es equivalente al tiempo (ideal) de reacción entre el inicio del ataque y la detección del mismo (a lo que después habrá que añadir el tiempo de ejecución del programa). Lo ideal sería fijar $j = 1$, para lo cual en una situación de máxima precisión la detección sería instantánea. No obstante, esto implicaría realizar tantos contrastes de hipótesis como

mensajes en las bases de datos, lo cual conlleva un tiempo de ejecución inaccesible para realizar un estudio exhaustivo del problema. Inicialmente fijaremos $\boldsymbol{j = 1000}$, lo que permite un tiempo de reacción de en torno a 0.5 s. Más adelante reduciremos este parámetro tanto como nos permita la computación.

- El número de paquetes N viene determinado por el tamaño de la base de datos M y los parámetros L y j:

$$N = \Big\lfloor \frac{M - L}{j} \Big\rfloor. \tag{2.3}$$

Este parámetro está fuertemente relacionado con el salto j: cuanto menor sea j, mayor será el número de paquetes y más valores tendremos para obtener las medidas de precisión. Para $L = 5000$ y $j = 1000$, N toma los valores que se muestran en el Tabla 2.4 para la base Survival.

	Soul	**Sonata**	**Spark**
Flooding	176	144	115
Fuzzy	244	130	60
Malfunction	168	127	74

Tabla 2.4: Número de paquetes en cada conjunto de datos de ataque. Base Survival.

- El parámetro p representa el porcentaje de mensajes malignos en un paquete a partir del cual es etiquetado como ataque. En una situación ideal de máxima precisión, un paquete con tan solo un mensaje maligno sería clasificado como alerta. Esto es demasiado ambicioso en la realidad, ya que son necesarios decenas de mensajes malignos para que la distribución de tiempos cambie con respecto a la distribución base. Aunque en el algoritmo final profundizaremos más en este aspecto y reduciremos este parámetro a 0, en este primer algoritmo fijaremos $\boldsymbol{p = 0.06}$.

- Es conveniente utilizar contrastes de hipótesis que acepten datos censurados, ya que las muestras $Z_i^{x,n}$ no son demasiado grandes y en cada paquete todos los IDs salvo a lo sumo uno tendrán su último dato incompleto (para cada paquete puede darse el caso de que el último mensaje sea el mensaje n-ésimo para un ID concreto, con n múltiplo de x. En ese caso, para dicho ID no habría datos censurados en dicho paquete). Así, utilizaremos las pruebas **log-rank** y de **Gehan**. Con respecto al nivel de significación, la

teoría de la probabilidad nos dice que, bajo H_0, los p-valores se distribuyen como una $\mathcal{U}(0,1)$. Así, fijado α, se espera una tasa de error del tipo I del $100\alpha\,\%$, o lo que es lo mismo, una especificidad de $1-\alpha$. Por otro lado, cuanto mayor sea α más fácil se rechazará H_0 y mayor será la sensibilidad. Así, el parámetro α mide la tensión entre ambas métricas. Además, el hecho de fijar $p > 0$ puede provocar una disminución de la especificidad con respecto a $1-\alpha$, ya que puede darse el caso de que paquetes atacados que hayan sido clasificados como no ataque sean detectados acertadamente como alerta por nuestros algoritmos. Inicialmente fijaremos el nivel de significación estándar $\boldsymbol{\alpha = 0.05}$.

Habiendo fijado los parámetros, procedemos a aplicar el primer algoritmo sobre los tres vehículos y sus tres ataques. Como ya se ha comentado, para cada vehículo se aplica el algoritmo sobre los IDs que aparecen de media al menos $10 \cdot x = 50$ veces en paquetes de tamaño $L = 5000$ de la base libre correspondiente. En el Tabla 2.5 se muestra un resumen de los IDs seleccionados para cada vehículo. En las Figuras 2.1-2.9 se muestra, para cada vehículo, cada ataque y cada ID, la sensbilidad, especificidad y F-score obtenidos aplicando el algoritmo de detección 1 mediante los tests log-rank y de Gehan. Los resultados generales se comentan a continuación:

- **Sensibilidad**. Exceptuando algunos casos, la sensibilidad utilizando el test de Gehan es muy mala, siendo 0 en muchas ocasiones. Las excepciones se dan para algún ID en los ataques *flooding* y *malfunction*, siendo más destacables en este último, para el cual vemos como en los tres vehículos se consigue una sensibilidad de 1 para el ID introducido en dicho ataque (0x153, 0x316 y 0x18E respectivamente).

  Con respecto al test log-rank, los ataques *flooding* y *malfunction* presentan en los tres vehículos una alta variabilidad en función del ID, siendo muy buena en algunos casos y muy mala en otros. De nuevo, para el ataque *malfunction* se consigue una sensibilidad de 1 para el ID inyectado. Para el ataque *fuzzy*, las sensibilidades son muy malas exceptuando algunos IDs del Sonata para los cuales rondan el 0.8.

- **Especificidad**. A la inversa de lo que ocurre para la sensibilidad, el test de Gehan obtiene especificidades muy altas, rondando al 1 en todos los casos excepto para algún ID del Soul en el ataque *malfunction*.

  El test log-rank presenta una mayor variabilidad, sobre todo para los ataques *flooding* y *malfunction*, disminuyendo en algunos casos por debajo de

0.5. Las especificidades del ataque *fuzzy* son bastante buenas salvo para algunos IDs del Sonata.

Sorprende el hecho de que en la mayoría de los casos las especificidades no se correspondan con el nivel de significación fijado. Esto es algo que se repetirá a lo largo de todo el estudio y en cuyas causas profundizaremos en el Capítulo 3.

- **F-score**. El F-score crece y disminuye de acuerdo con la sensibilidad y la especificidad, por lo que los resultados obtenidos no nos sorprenden habiendo observado las otras gráficas. Dado que sensibilidad nula implica F-score nulo, existen muchos casos para los cuales el F-score vale 0, sobre todo para el test de Gehan.

	num IDs selec.	**ID más numeroso**	**ID menos numeroso**
Soul	23	0x164 (241)	0x350 (119)
Sonata	22	0x2A0 (257)	0x5F0 (51)
Spark	33	0x1E5 (218)	0x1F3 (73)

Tabla 2.5: Para cada vehículo, número de IDs seleccionados según nuestro criterio de aparición más de 50 veces y, dentro de estos IDs, el más y el menos numeroso. Entre paréntesis, el número medio de apariciones de dicho ID en un paquete de tamaño 5000 en la correspondiente base libre. Base Survival.

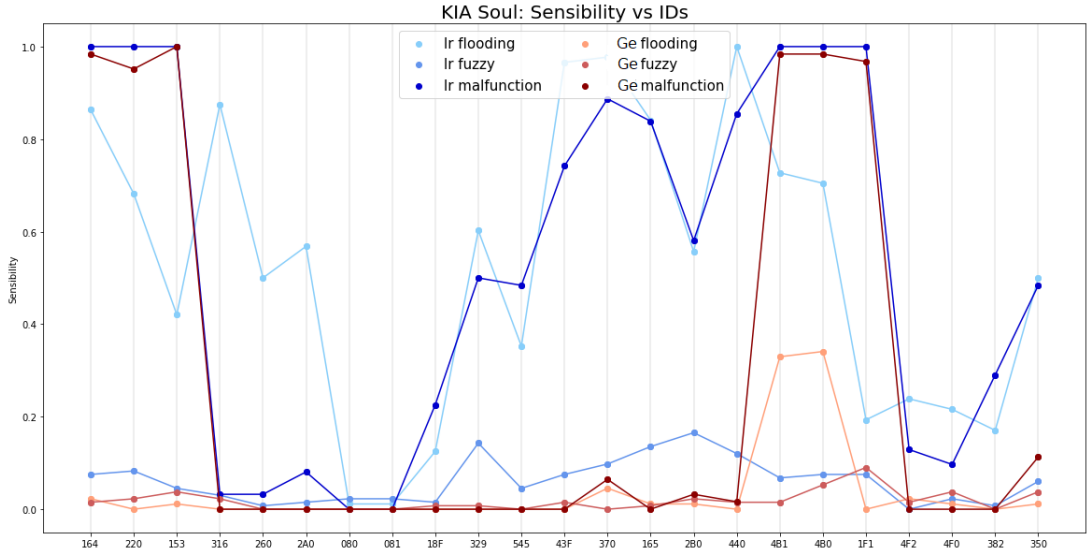

Figura 2.1: Sensibilidad para cada ID y para cada ataque en el KIA Soul. Los IDs están ordenados de mayor a menor por frecuencia de aparición en la base libre. El ID 0x153 es el tercero empezando por la izquierda. Base Survival.

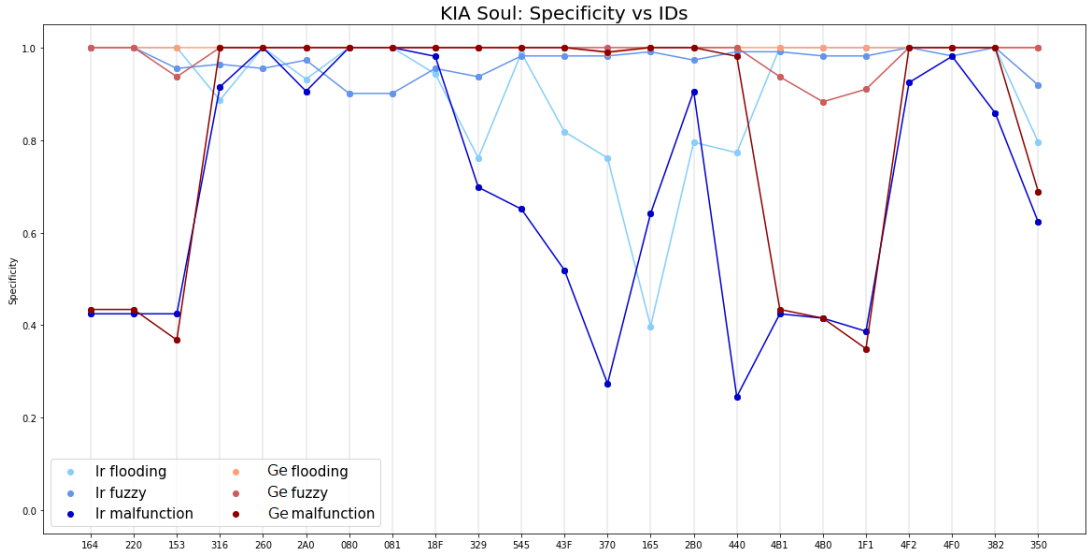

Figura 2.2: Especificidad para cada ID y para cada ataque en el KIA Soul. Los IDs están ordenados de mayor a menor por frecuencia de aparición en los paquetes. El ID 0x153 es el tercero empezando por la izquierda. Base Survival.

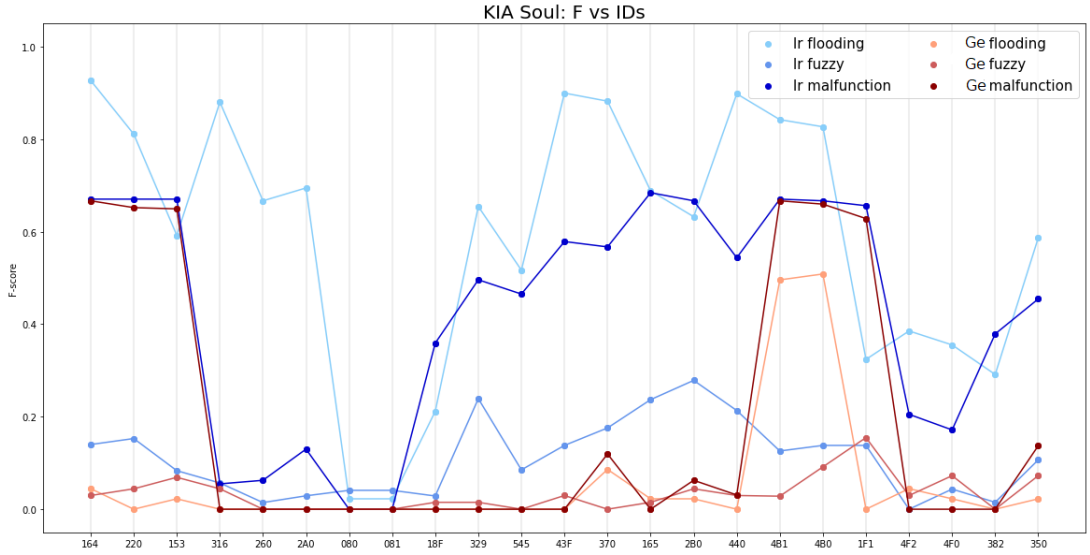

Figura 2.3: F-score para cada ID y para cada ataque en el KIA Soul. Los IDs están ordenados de mayor a menor por frecuencia de aparición en los paquetes. Base Survival.

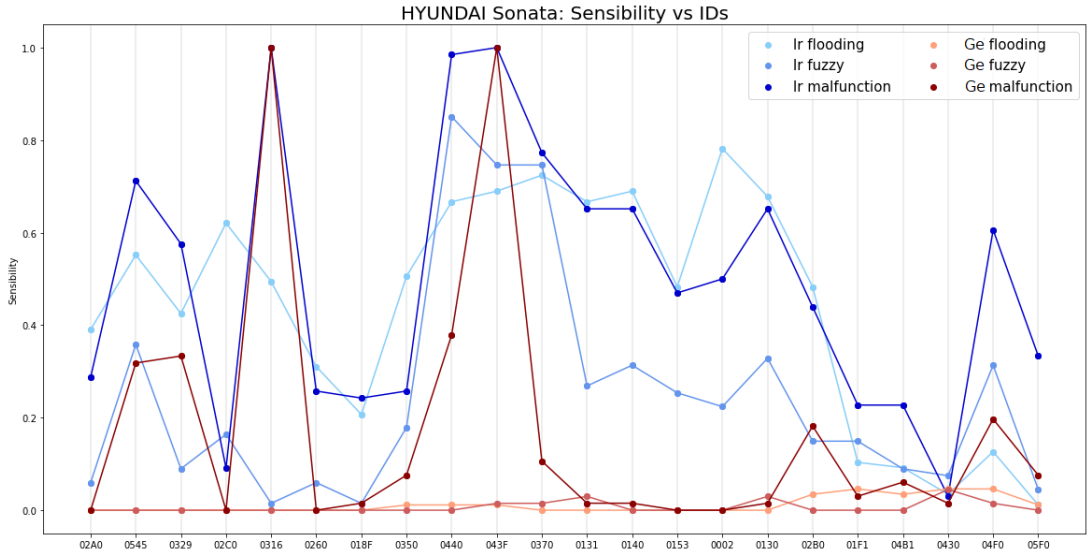

Figura 2.4: Sensibilidad para cada ID y para cada ataque en el HYUNDAI Sonata. Los IDs están ordenados de mayor a menor por frecuencia de aparición en los paquetes. El ID 0x316 es el tercero empezando por la izquierda. Base Survival.

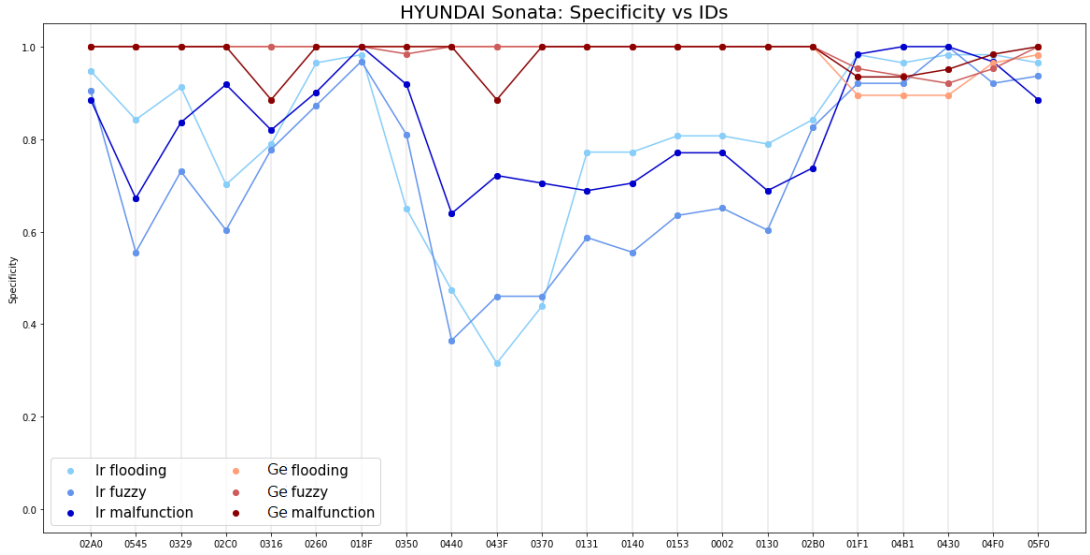

Figura 2.5: Especificidad para cada ID y para cada ataque en el HYUNDAI Sonata. Los IDs están ordenados de mayor a menor por frecuencia de aparición en los paquetes. El ID 0x316 es el tercero empezando por la izquierda. Base Survival.

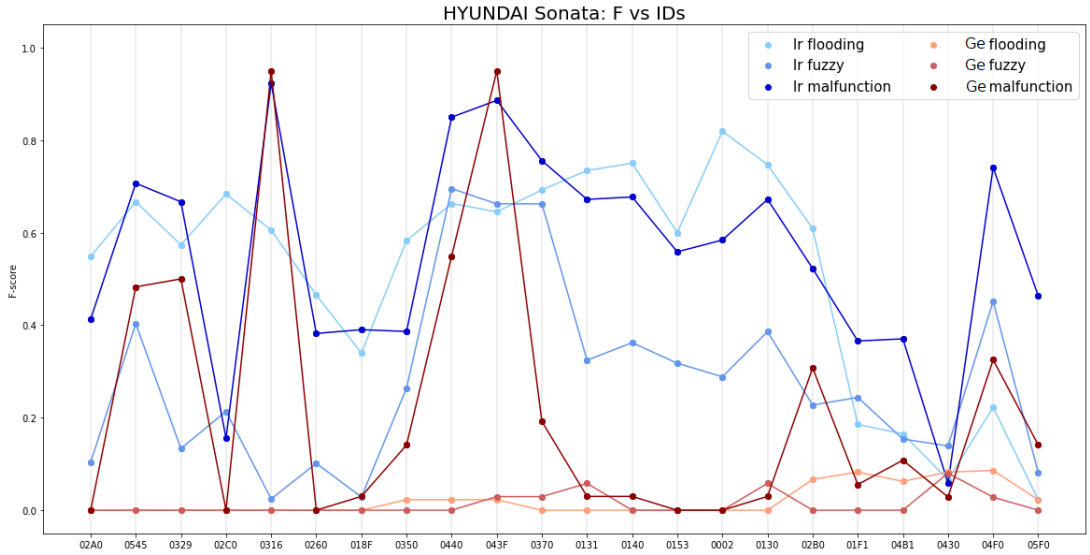

Figura 2.6: F-score para cada ID y para cada ataque en el HYUNDAI Sonata. Los IDs están ordenados de mayor a menor por frecuencia de aparición en los paquetes. Base Survival.

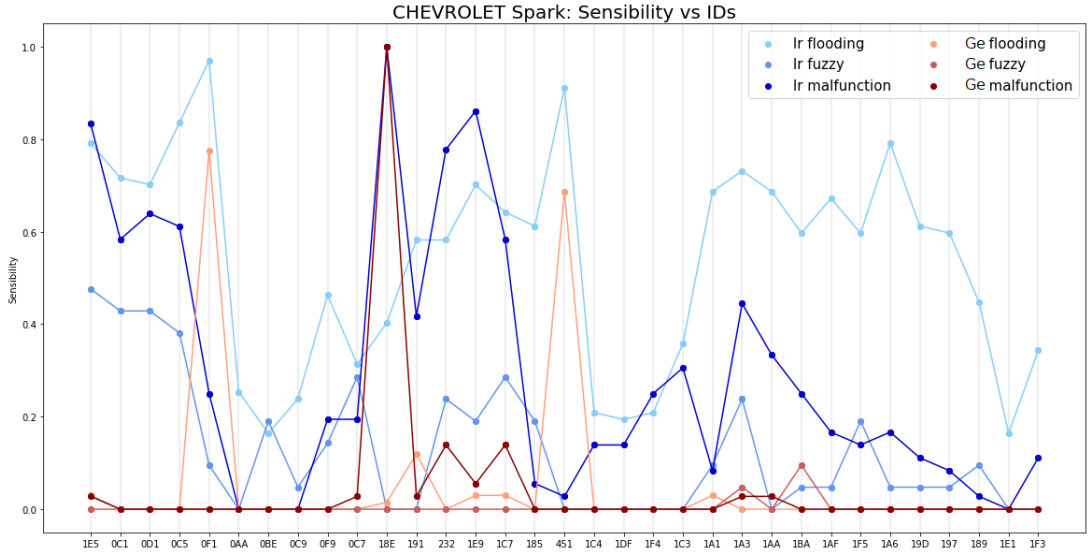

Figura 2.7: Sensibilidad para cada ID y para cada ataque en el CHEVROLET Spark. Los IDs están ordenados de mayor a menor por frecuencia de aparición en los paquetes. El ID 0x18E es el tercero empezando por la izquierda. Base Survival.

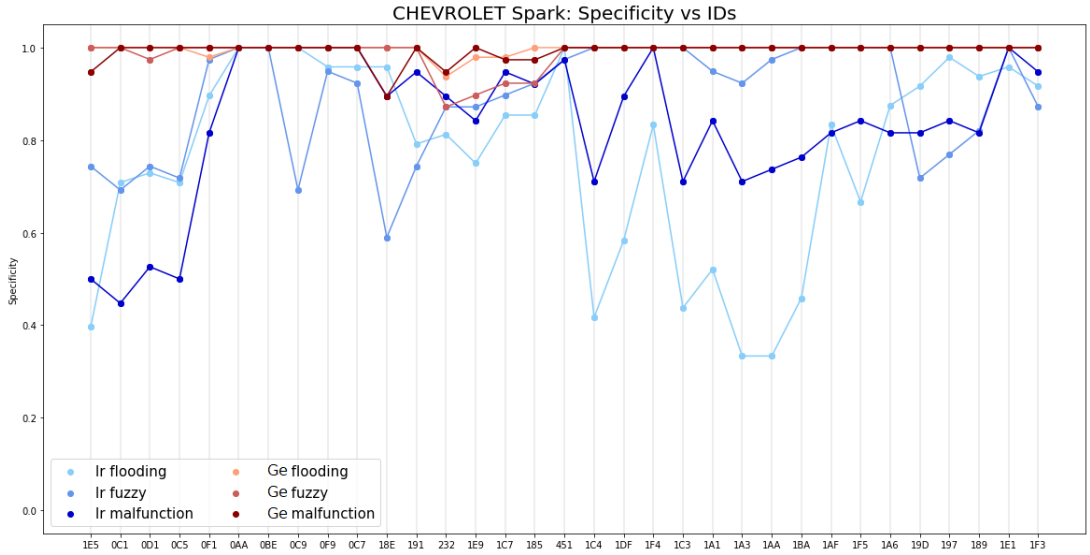

Figura 2.8: Especificidad para cada ID y para cada ataque en el CHEVROLET Spark. Los IDs están ordenados de mayor a menor por frecuencia de aparición en los paquetes. Base Survival.

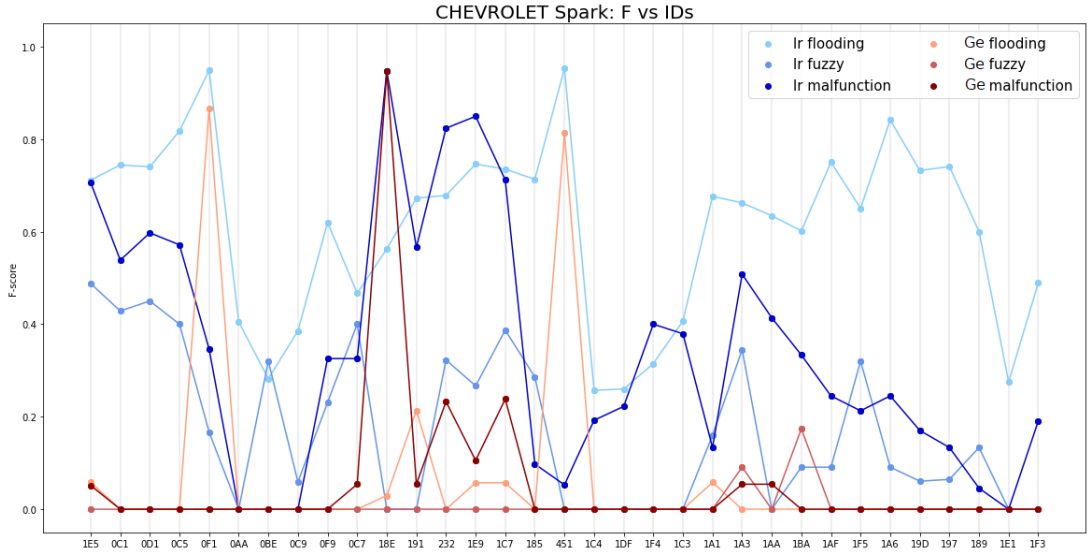

Figura 2.9: F-score para cada ID y para cada ataque en el CHEVROLET Spark. Los IDs están ordenados de mayor a menor por frecuencia de aparición en los paquetes. Base Survival.

- **Tiempo de ejecución**. En el Tabla 2.6 se muestran los tiempos medios de ejecución por paquete para el ID más y menos numeroso de cada vehículo. Por un lado y como era de esperar, vemos como el tiempo de ejecución aumenta cuanto mayor es el número de apariciones del ID en los paquetes. Por otro lado, la diferencia entre los tiempos para el test log-rank y para el test de Gehan es claramente notable, siendo dos o incluso tres veces mayor para el test log-rank. Esto puede deberse a que el test de Gehan esté implementado de forma más eficiente en Python que el test log-rank. Cabe recalcar que estos tiempos de ejecución son por paquete, es decir, se refieren a los tiempos que tardaría el programa en detectar un ataque al aplicar el algoritmo en tiempo real. En este estudio los tiempos de ejecución son mucho mayores, ya que para obtener resultados es necesario ejecutar el programa para todos los paquetes, para todos los IDs, para todos los ataques y para todos los vehículos. Por tanto, aunque la diferencia por paquete entre los dos test sea de unas milésimas, en nuestro estudio esas milésimas se convierten en segundos e incluso minutos.

	ID más num.	Tiempo (s)	ID menos num.	Tiempo (s)
Soul	0x164 (241)	lr: 0.024 Ge: 0.012	0x350 (119)	lr: 0.018 Ge: 0.006
Sonata	0x2A0 (257)	lr: 0.024 Ge: 0.011	0x5F0 (51)	lr: 0.014 Ge: 0.004
Spark	0x1E5 (218)	lr: 0.022 Ge: 0.011	0x1F3 (73)	lr: 0.015 Ge: 0.005

Tabla 2.6: Para cada vehículo, tiempo medio de ejecución por paquete para el ID más y menos numeroso y para los tests log-rank (lr) y de Gehan (Ge). Entre paréntesis el número medio de apariciones del ID en paquetes de tamaño 5000 en la correspondiente base libre. Base Survival.

Observando más detenidamente las gráficas, para cada vehículo, cada ataque y cada test de hipótesis podemos clasificar los IDs en cuatro categorías en función de su sensibilidad y especificidad:

1. Buena sensibilidad, buena especificidad. Ejemplo: 0x0316 para el ataque *malfunction* del Sonata con ambos tests. En estos casos el algoritmo es capaz de discernir correctamente cuándo hay ataque y cuándo no lo hay, con pocos falsos positivos y falsos negativos, obteniendo un F-score alto.

2. Buena sensibilidad, mala especificidad. Ejemplo: 0x153 para el ataque *malfunction* del Soul con ambos tests. En estos casos el algoritmo detecta los ataques, pero también da un número relevante de falsas alarmas, reduciendo así el F-score con respecto a la sensibilidad.

3. (Muy) mala sensibilidad, (muy) buena especificidad. A este grupo pertenecen la mayoría de los IDs para el test de Gehan. El algoritmo no es capaz de detectar los ataques y tampoco da falsas alarmas. Es un algoritmo inútil, no hace nada, y por ello tiene un F-score igual o próximo a 0.

4. Mala sensibilidad, mala especificidad. A este último grupo pertenecen el resto de IDs que cuentan con una sensibilidad y una especificidad por debajo de 0.7-0.8. Ejemplo: 0x140 para el ataque *fuzzy* del Sonata con el test log-rank. El algoritmo en ocasiones da alarma cuando no hay ataque, y en ocasiones no da alarma cuando sí lo hay. El algoritmo es malo y se obtienen F-scores bajos.

Recordemos que nuestro objetivo consiste en obtener un algoritmo aplicable sobre cualquier tipo de ataque e, idealmente, sobre cualquier vehículo. Si nos limitamos a construir un algoritmo para cada vehículo, en cada caso debemos seleccionar un ID capaz de detectar con precisión los tres ataques, pero ya hemos visto que ninguno de los IDs para ninguno de los tres vehículos cumple esto, incluso ignorando el ataque *fuzzy*. Suponiendo que sí hubiera algún ID que lo cumpliera, si fuéramos más allá y pretendiéramos obtener un algoritmo general, entonces para cada vehículo tendríamos que seleccionar de forma automática el ID de interés. Una posibilidad sería utilizar el ID con mayor frecuencia en la base libre. Otra, elegir un ID aleatorio. En cualquier caso, la alta variabilidad de los F-scores en función del ID haría muy probable que la selección no fuera óptima y que la clasificación no fuera buena. Dicho todo esto, el Algoritmo 1 no resulta conveniente para la detección de anomalías en vehículos.

2.1.2. Algoritmo 2: tiempo entre mensajes para todos los IDs

Una alternativa al Algoritmo 1 consiste en basar la detección de un ataque en la comparación de los tiempos entre mensajes de todos los IDs, y no solamente de uno. Para un vehículo y un ataque dados, el algoritmo de detección se muestra en el Tabla 2.7. Vemos como este algoritmo no basa la detección de ataques en un ID en particular, sino que se aplica sobre todos los IDs (que aportan muestras de tamaño suficiente) y da señal de alerta si hay diferencias significativas para k o más de ellos. Es decir, el parámetro i es sustituido por el parámetro k. Existe claramente una relación cercana entre el Algoritmo 1 y el Algoritmo 2, por lo que esperamos obtener resultados acordes a los del algoritmo anterior. Fijando los parámetros como en el Algoritmo 1:

$$x = 5, \qquad L = 5000, \qquad j = 1000, \qquad p = 0.06, \qquad \alpha = 0.05,$$

en las Figuras 2.10, 2.11 y 2.12 se muestran las curvas ROC obtenidas variando $k \in \{0, 1, \ldots, |I^*| + 1\}$ para cada vehículo, cada ataque y cada contraste de hipótesis, Además, en los Tablas 2.8, 2.9 y 2.10 se muestran los AUCs, y el k óptimo obtenido en cada caso con su correspondiente F-score. Los resultados se comentan a continuación:

- El algoritmo no es capaz de detectar el ataque *fuzzy*. Esto era de esperar: si el Algoritmo 1 no es capaz de detectar un ataque para ninguno de los IDs, tampoco lo será el Algoritmo 2.

Algoritmo de detección 2

1. Para la muestra de ataque, se crean N paquetes de tamaño L con saltos de tamaño j.
2. Los paquetes que cuenten con al menos un porcentaje p de mensajes malignos son etiquetados como **ataque**. Los que no, son etiquetados como **no ataque**.
3. Se fija el parámetro x y se determina el conjunto I^* de IDs de interés en base al criterio de aparición de más de $10 \cdot x$ veces.
4. Se calculan los valores que toma la variable Y_i^x en la base libre para cada $i \in I^*$.
5. Se calculan los valores que toman las variables $Z_I^{x,n}$ para cada uno de los paquetes $n \in \{1, \ldots, N\}$ y para cada $i \in I^*$.
6. Se realiza un test de comparación a nivel de significación α entre las distribuciones de Y_i^x y $Z_i^{x,n}$ para cada $i \in I^*$ y cada $n \in \{1, \ldots, N\}$

$$\begin{cases} H_0: & Y_i^x = Z_i^{x,n} \\ H_1: & Y_i^x \neq Z_i^{x,n}, \end{cases} \qquad n \in \{1, \ldots, N\},\ i \in I^*. \tag{2.4}$$

Para cada paquete se realizan $|I^*|$ contrastes. Los paquetes para los que se rechace H_0 en k o más de estos contrastes son clasificados como **alerta**. Los que no, son clasificados como **no alerta**. **7.** Se obtiene la matriz de confusión entre alertas y ataques y, a partir de ella, se calculan las medidas de precisión de interés: sensibilidad, especificidad y F-score.

Tabla 2.7: Algoritmo de detección 2.

- El algoritmo es capaz de detectar de forma aceptable o incluso buena los ataques *flooding* y *malfunction* para el test log-rank, y en algún caso también para el test de Gehan.
- Los picos obtenidos para el F-score en el Algoritmo 1 mediante el test de Gehan explican alguno de los valores obtenidos para este algoritmo. Por ejemplo, para el Sonata, en la Figura 2.6 vemos como el ataque *malfunction* presenta dos picos con alto F-score para dos IDs, lo que se traduce en que en el Tabla 2.9 se obtenga $k_{opt} = 2$ con un alto F-score. Lo mismo ocurre para los ataques *flooding* y *malfunction* del Spark.
- Los tiempos medios de ejecución por paquete se muestran en el Tabla 2.11, donde vemos como los tiempos para el Spark son mayores, ya que se utiliza un mayor número de IDs que para el Soul y el Sonata. Además, la diferencia entre ambos tests se magnifica, siendo ahora de unas décimas de segundo, lo que podría resultar un inconveniente en su implementación.

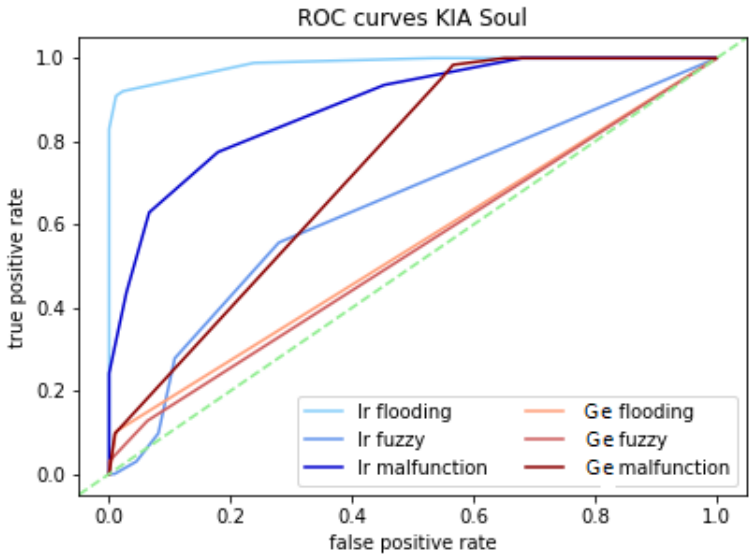

Figura 2.10: Curvas ROC para el KIA Soul para cada ataque y cada contraste. Base Survival.

Soul	***Flooding***		***Fuzzy***		***Malfunction***	
	AUC	k_{opt} (F)	AUC	k_{opt} (F)	AUC	k_{opt} (F)
log-rank	0.99	4 (0.95)	0.64	1 (0.62)	0.88	6 (0.74)
Gehan	0.55	1 (0.18)	0.53	1 (0.22)	0.73	2 (0.67)

Tabla 2.8: AUC y k óptimo para el KIA Soul para cada ataque. Entre paréntesis el F-score correspondiente. Base Survival.

Sonata	***Flooding***		***Fuzzy***		***Malfunction***	
	AUC	k_{opt} (F)	AUC	k_{opt} (F)	AUC	k_{opt} (F)
log-rank	0.77	4 (0.76)	0.47	1 (0.67)	0.88	6 (0.82)
Gehan	0.51	1 (0.05)	0.52	1 (0.11)	0.97	2 (0.95)

Tabla 2.9: AUC y k óptimo para el HYUNDAI Sonata para cada ataque. Entre paréntesis el F-score correspondiente. Base Survival.

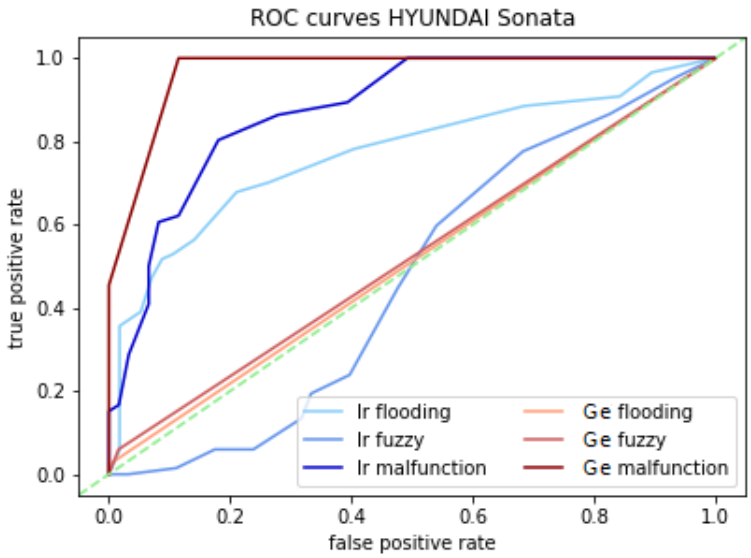

Figura 2.11: Curvas ROC para el HYUNDAI Sonata para cada ataque y cada contraste. Base Survival.

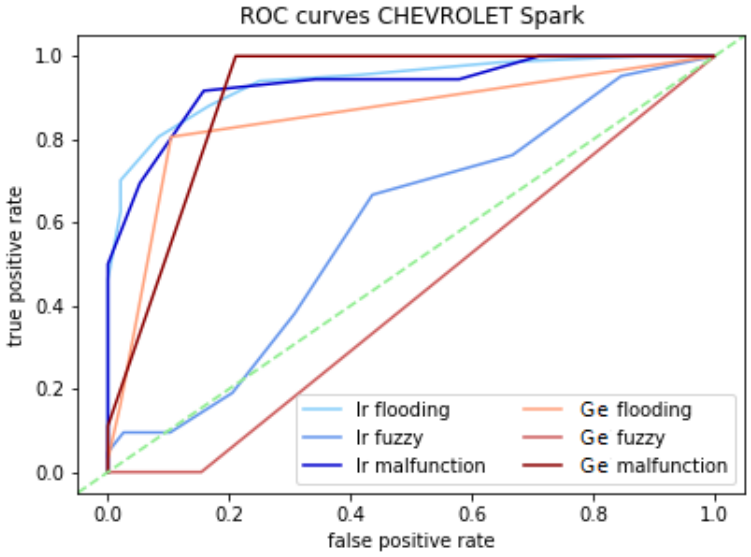

Figura 2.12: Curvas ROC para el CHEVROLET Spark para cada ataque y cada contraste. Base Survival.

Spark	***Flooding***		***Fuzzy***		***Malfunction***	
	AUC	k_{opt} (F)	AUC	k_{opt} (F)	AUC	k_{opt} (F)
log-rank	0.94	4 (0.89)	0.59	1 (0.54)	0.93	5 (0.88)
Gehan	0.85	2 (0.86)	0.42	- -	0.91	1 (0.90)

Tabla 2.10: AUC y k óptimo para el CHEVROLET Spark para cada ataque. Entre paréntesis el F-score correspondiente. Para el ataque *fuzzy* con el test Gehan todos los F-scores obtenidos son 0. Base Survival.

	Tiempo (s) log-rank	**Tiempo (s) Gehan**
Soul (23)	0.60	0.25
Sonata (22)	0.58	0.24
Spark (33)	0.75	0.32

Tabla 2.11: Para cada vehículo, tiempo medio de ejecución por paquete para los tests log-rank y de Gehan. Entre paréntesis el número de ID utilizados para cada vehículo. Base Survival.

Volviendo a lo comentado para el Algoritmo 1, queremos obtener un algoritmo unificado aplicable sobre cualquier ataque, lo que implica seleccionar un parámetro k para cada vehículo o, yendo más allá, un parámetro k global. Vemos como ni una cosa ni la otra son posibles si queremos obtener buenas precisiones, por lo que concluimos que el Algoritmo 2 tampoco resulta conveniente para la detección de anomalías en vehículos.

Si deseamos obtener un algoritmo que mejore los resultados es necesario preguntarnos por qué los algoritmos basados en el tiempo entre mensajes para cada ID no son convenientes, y por qué para estos algoritmos el test de Gehan funciona peor que el test log-rank. Fijémonos por ejemplo en los resultados obtenidos para el ID 0x140 del Sonata para el ataque *fuzzy*, mostrados en las Figuras 2.4, 2.5 y 2.6. Para el test log-rank tanto la sensibilidad como la especificidad son malas, es decir, se obtienen numerosos falsos negativos y falsos positivos y, por tanto, el F-score es bajo. Para el test de Gehan la especificidad es máxima pero la sensibilidad nula, es decir, nunca da señal de alerta y, por tanto, el F-score es nulo. Veamos por qué ocurre esto. En la Figura 2.13 se muestran las densidades y funciones de riesgo de los tiempos entre 5 mensajes para el ID 0x140 del Sonata en la base libre y la base fuzzy en dos situaciones diferentes.

A la izquierda se muestran las densidades y funciones de riesgo para un

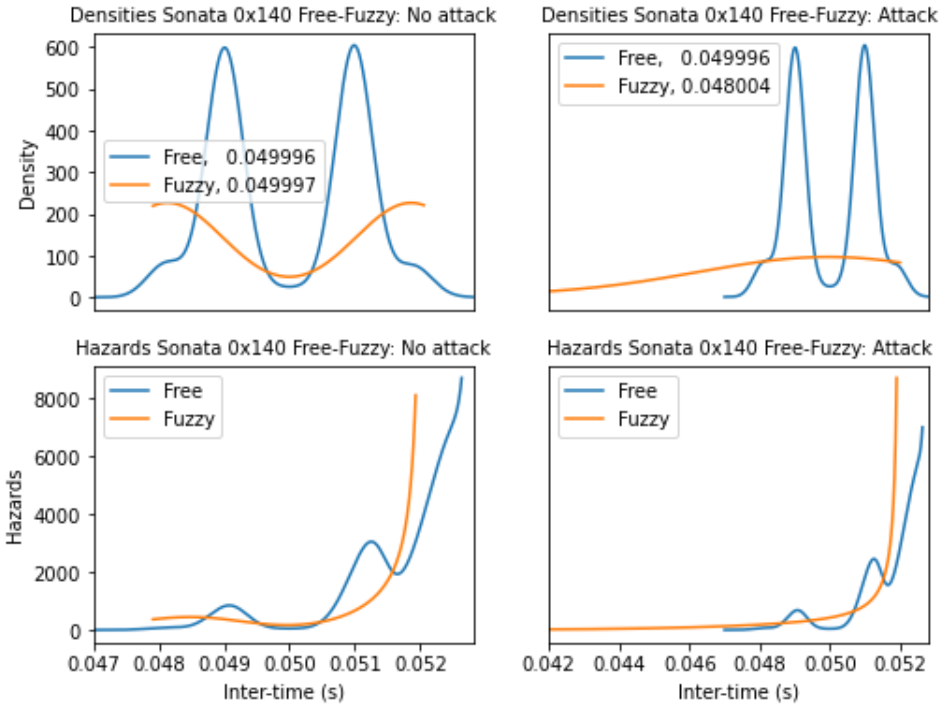

Figura 2.13: Para el Sonata, a la izquierda arriba funciones de densidad de las variables Y^{5}_{0x140} y $Z^{5,21}_{0x140}$ (paquete no atacado) con sus respectivas medias, y abajo funciones de riesgo. A la derecha lo mismo para las variables Y^{5}_{0x140} y $Z^{5,33}_{0x140}$ (paquete atacado).

paquete etiquetado como **no ataque** y clasificado como **alerta** para el test log-rank (FP, p-valor = 0) y **no alerta** (TN, pvalor = 0.945) para el test de Gehan. Aunque las medias son prácticamente idénticas, a simple vista existen diferencias de forma entre ambas distribuciones. El test log-rank es capaz de captar dichas diferencias a pesar de que las funciones de riesgo no son proporcionales (se cruzan). Por otro lado, el test de Gehan, más centrado en diferencias de localización, no da señal de alerta.

A la derecha se muestran las densidades y funciones de riesgo para un paquete etiquetado como **ataque** y clasificado como **no alerta** para el test log-rank (FN, p-valor = 0.107) y **no alerta** (FN, pvalor = 0.418) para el test de Gehan. La diferencia entre las medias no llega a ser significativa, por lo que el test de Gehan no rechaza la hipótesis nula. Por otro lado, el test log-rank tampoco es capaz de hallar diferencias bajo un nivel de significación $\alpha = 0.05$. Es posible que esto se deba a que, de nuevo, las funciones de riesgo no son proporcionales. Como ya hemos comentado, en estos casos el test log-rank puede devolver resul-

tados que no se corresponden con la realidad. Estas representaciones también nos sirven para ver que las distribuciones no son normales, descartando así la idea de usar la prueba t de Student.

2.1.3. Conclusiones

En esta sección hemos considerado dos algoritmos basados en el tiempo entre mensajes para cada ID, siendo el segundo una extensión del primero. Estos algoritmos nos aportan las siguientes conclusiones:

- De forma general, las variables Y_i^x no representan fielmente las diferencias entre una conducción libre y una conducción con anomalías. Existen excepciones para ciertos IDs y para los ataques *flooding* y *malfunction*, pero en ningún caso nos permiten crear un algoritmo general que detecte ataques con garantías de obtener una buena precisión. El ataque *fuzzy* no es detectado con este enfoque.

- El Algoritmo 2 permite unificar los resultados del Algoritmo 1, obteniendo en algunos ocasiones buenas precisiones. Sin embargo, la necesidad de seleccionar un k global hace que, en nuestro caso, en el que los k's óptimos son diferentes para cada ataque y además las precisiones no son demasiado buenas, tampoco sea aplicable.

- En general, la media de tiempos entre mensajes de cada ID no cambia significativamente entre la base libre y las bases atacadas, por lo que el test de Gehan no es conveniente para este enfoque. Esto tiene sentido si nos detenemos a pensar cómo se ejecutan los ataques: se inyectan mensajes malignos entre los mensajes libres. Así, si los mensajes malignos no pertenecen a la red de IDs del vehículo, como es el caso del ataque *flooding* y de la mayoría del ataque *fuzzy*, los tiempos medios de los IDs de la red no cambian. Existe la excepción del ID inyectado en el ataque *malfunction*, cuyo tiempo medio entre mensajes disminuye significativamente (hemos visto que el test de Gehan funciona bien para estos IDs en este ataque).

  Existen casos en los que la distribución de tiempos cambia entre la base libre y las partes libres de las bases atacadas, generando para el test log-rank un número elevado de FP. Además, las funciones de riesgo no son en general proporcionales, por lo que en ocasiones el test log-rank no se comporta como esperaríamos y da lugar a FN.

- El test log-rank requiere de un tiempo de ejecución considerablemente mayor que el test de Gehan. Para el Algoritmo 1 la diferencia es tan solo de unas milésimas, pero para el Algoritmo 2 se convierte en varias décimas, retrasando así la detección de los ataques. Además, a estos tiempos de ejecución hay que añadirles el tiempo de reacción de 0.5 s que conlleva tomar $j = 1000$. Así, para el test log-rank en el Algoritmo 2 pasaría más de 1 segundo desde que se inicia el ataque hasta su detección, lo cual obviamente no nos interesa.

2.2. Algoritmos de detección basados en el tiempo entre mensajes

En los anteriores algoritmos hemos visto que los ataques considerados cambian la distribución de los tiempos entre mensajes de cada ID, pero no lo suficiente como para que las diferencias sean detectadas con precisión ni por el test log-rank ni, sobre todo, por el test de Gehan. Además, hemos visto como los tiempos de ejecución son considerablemente mayores para el test log-rank, retrasando así la detección de los ataques. Así, lo ideal sería encontrar un algoritmo en el que el test de Gehan fuera capaz de detectar con precisión las diferencias entre una conducción libre y una conducción anómala. Dado que que este test es de máxima potencia bajo diferencias de localización, buscamos un algoritmo en el que existan diferencias significativas en la media de los tiempos entre mensajes de las dos categorías.

Como ya hemos comentado, los ataques considerados se basan en la inyección de mensajes malignos entre los mensajes libres. Esto nos hace pensar que, aunque los tiempos medios entre mensajes de cada ID del vehículo no cambien significativamente, sí lo harán los tiempos medios entre mensajes, ignorando su ID y sí pertenecen o no a la red del vehículo en cuestión. Esto podemos comprobarlo en la Figura 2.14, donde se representa el tiempo medio entre mensajes calculado en paquetes de 5000 mensajes para los ataques del KIA Soul. Vemos como el tiempo medio disminuye apreciablemente durante los ataques, siendo mayor la diferencia para el *flooding*, seguido del *fuzzy*. Vemos también como existen periodos de transición con un ratio de ataque menor en los que se comienza o se finaliza de inyectar los ataques. Además, parece que el *malfunction* es más inestable durante las inyecciones del ataque, produciéndose picos con mayor y menor ratio de inyección.

Habiendo observado que el tiempo entre mensajes (sin distinguir su ID) es

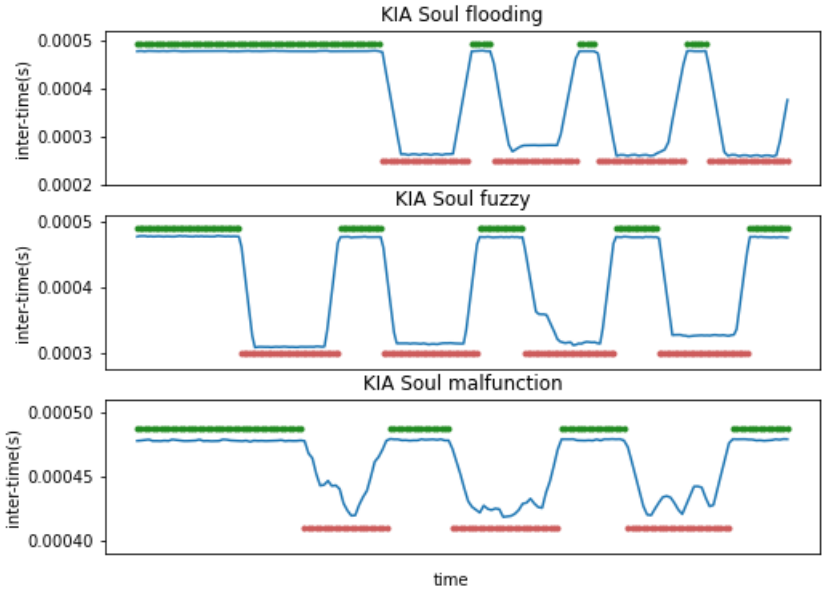

Figura 2.14: Tiempo medio entre mensajes en paquetes de tamaño 5000 para los ataques del KIA Soul. En rojo los paquetes con algún mensaje maligno (atacados) y en verde el resto (no atacados).

una cantidad que a priori diferencia correctamente entre una conducción libre y una conducción anómala, definimos las siguientes variables aleatorias:

$$Y^x = \text{tiempo entre } x \text{ mensajes en la base libre,} \tag{2.5}$$

$$Z^x = \text{tiempo entre } x \text{ mensajes en la base atacada,} \tag{2.6}$$

donde x denota el número de mensajes entre los que se calcula el tiempo. Al igual que para los algoritmos anteriores, las diferencias de tiempos se calcularán entre posiciones disjuntas, por ejemplo: $Z^5 = \{t_6 - t_1, t_{11} - t_6, \ldots\}$.

Consideraremos tres algoritmos en los cuales se compararán los tiempos entre mensajes de las bases atacadas con respecto a los de una base libre. En [5] se ha llevado a cabo un estudio con un enfoque similar, siendo su variable de interés el número de mensajes en un intervalo de tiempo dado en lugar del tiempo entre mensajes. Nuestro enfoque tiene la ventaja de que nuestra variable de interés contiene más información que la suya, ya que el número de mensajes en un intervalo de tiempo solo aporta el tiempo medio entre mensajes. La diferencia entre los tres algoritmos con los que trabajaremos será la base libre a considerar. Comenzaremos por el más simple: para cada vehículo tomamos como base libre el conjunto de datos libre original.

2.2.1. Algoritmo 3: tiempo entre mensajes con la base libre original

Para un vehículo y un ataque dados, el algoritmo de detección es el siguiente:

Algoritmo de detección 3

1. Para la muestra de ataque, se crean N paquetes de tamaño L con saltos de tamaño j.
2. Los paquetes que cuenten con al menos un porcentaje p de mensajes malignos son etiquetados como **ataque**. Los que no, son etiquetados como **no ataque**.
3. Se fija el parámetro x.
4. Se calculan los valores que toma la variable Y^x en la base libre.
5. Se calculan los valores que toman las variables Z_n^x para cada uno de los paquetes $n \in \{1, \ldots, N\}$.
6. Se realiza un test de comparación a nivel de significación α entre las distribuciones de Y^x y Z_n^x para cada $n \in \{1, \ldots, N\}$

$$\begin{cases} H_0: & Y^x =^d Z^{x,n} \\ H_1: & Y^x \neq^d Z^{x,n}, \end{cases} \qquad n \in \{1, \ldots, N\}.$$

Los paquetes para los que H_0 se rechaza son clasificados como **alerta**. Los que no, son clasificados como **no alerta**.
7. Se obtiene la matriz de confusión entre alertas y ataques y, a partir de ella, se calculan las medidas de precisión de interés: sensibilidad, especificidad y F-score.

El hecho de considerar todos los mensajes por igual y no diferenciarlos según su ID nos permite ahorrarnos los parámetros i y k de los algoritmos 1 y 2 respectivamente. El resto de parámetros tienen las mismas funciones y características que en dichos algoritmos. Como veremos, no diferenciar entre IDs reduce considerablemente el tiempo de ejecución, lo que nos permite mover los parámetros y ver como se ve afectada la precisión. Tras un estudio preliminar descubrimos que el parámetro de mayor interés es x, por lo que será en el que profundizaremos a lo largo de este apartado. Así, hacemos variar $\boldsymbol{x \in \{1, 5, 10, 15, \ldots, 200\}}$ y fijamos el resto de parámetros:

$$L = 5000, \qquad j = 1000, \qquad p = 0.06, \qquad \alpha = 0.05.$$

En las Figuras 2.15-2.23 se muestran las sensibilidades, especificidades y F-scores obtenidos para cada vehículo y cada ataque con el test log-rank y con el test de Gehan. Los resultados se comentan a continuación:

- Para valores pequeños de x se obtienen tanto malas sensibilidades como

malas especificidades. Esto nos dice que el tiempo entre mensajes individuales no es una variable que detecte correctamente anomalías debidas a la inyección de mensajes.

- Salvo para el ataque *fuzzy*, el test log-rank presenta inestabilidad en las sensibilidades y, sobre todo, en la especificidades. Se observan patrones para los ataques *flooding* y *malfunction*, lo que nos indica que la inestabilidad es originada por la base libre. Por otro lado, el test de Gehan presenta muy buenas sensibilidades, y buenas especificidades a partir de $x = 40$. Existe inestabilidad, aunque mucho menor que para el test log-rank. En cualquier caso, habiendo fijado $\alpha = 0.05$ esperaríamos especificidades de 0.95 (o ligeramente inferiores por haber fijado $p = 0.06$). Sin embargo, esto no ocurre en casi ninguna ocasión. Las posibles razones se comentan más abajo.

- Para cuantificar estos resultados, en el Tabla 2.12 se muestran las medias y desviaciones típicas de los F-scores obtenidos para cada vehículo, cada ataque y cada test, en el rango $x \in \{40, \ldots, 200\}$. Vemos como las desviaciones típicas son considerablemente mayores para el test log-rank. Los resultados para el test de Gehan son muy buenos, tanto en media como en desviación típica. Podría mejorarse el resultado del ataque *malfunction* del Soul. A pesar de ser el ataque con menor impacto sobre los tiempos entre mensajes (Figura 2.14), vemos cómo la sensibilidad es muy alta y la disminución del F-score es debida a la especificidad, que nunca llega a ser completamente buena. Es decir, con esta combinación de parámetros, para dicha base de datos se da un número considerable de falsas alertas para todos los x considerados.

- **Tiempo de ejecución**. En la Figura 2.24 se muestra el tiempo de ejecución por paquete en función de x para el ataque *flooding* del KIA Soul (para el resto de ataques y vehículos los tiempos de ejecución son muy similares). Vemos como para el test de Gehan los tiempos de ejecución rápidamente convergen a unas pocas milésimas de segundo, aunque para el test log-rank no logran bajar de 0.015 segundos. Sorprende el elevado tiempo de ejecución del test de Gehan para x bajos, aunque esto no nos preocupa ya que no trabajaremos con ellos debido a su inestabilidad. En comparación con el Algoritmo 1, el test log-rank se mantiene en el mismo orden de magnitud, mientras que el test de Gehan disminuye entre 5 y 10 veces su tiempo de ejecución.

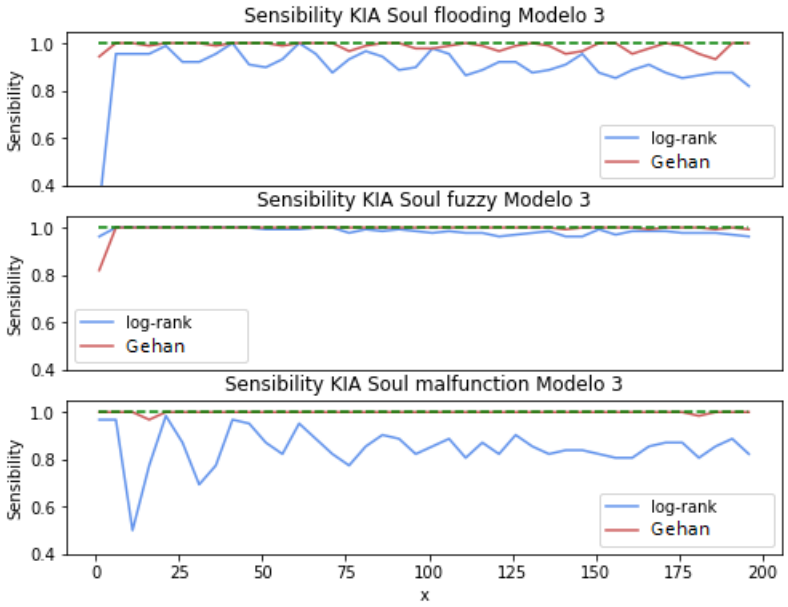

Figura 2.15: Sensibilidad del Algoritmo 3 en función del parámetro x para los tres ataques del KIA Soul. Base Survival.

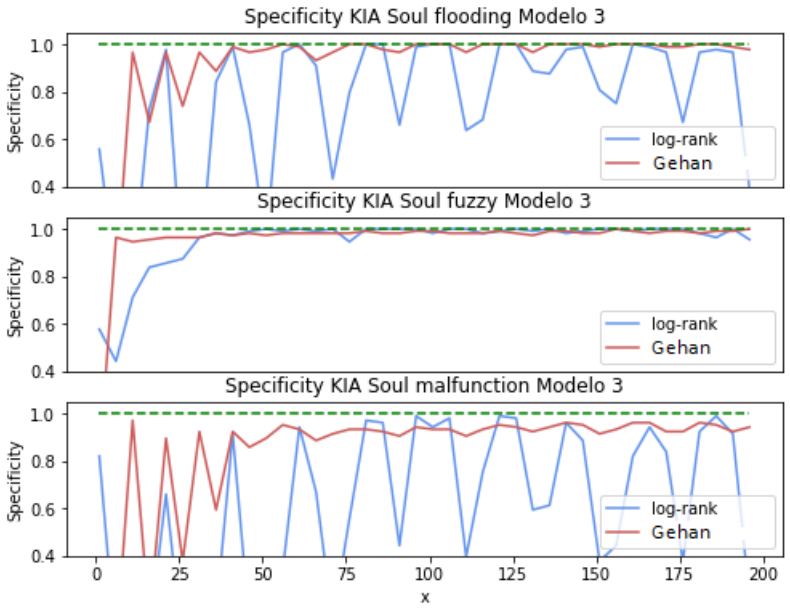

Figura 2.16: Especificidad del Algoritmo 3 en función del parámetro x para los tres ataques del KIA Soul. Base Survival.

Figura 2.17: F-score del Algoritmo 3 en función del parámetro x para los tres ataques del KIA Soul. Base Survival.

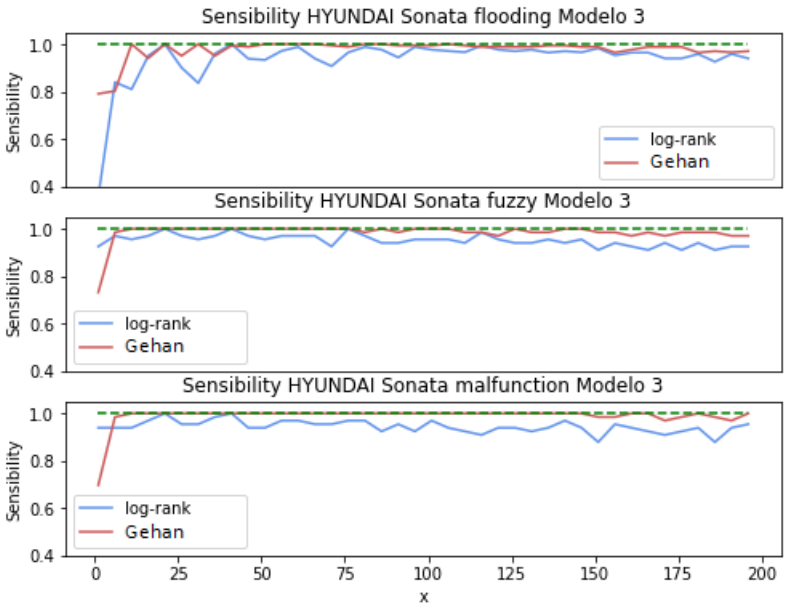

Figura 2.18: Sensibilidad del Algoritmo 3 en función del parámetro x para los tres ataques del HYUNDAI Sonata. Base Survival.

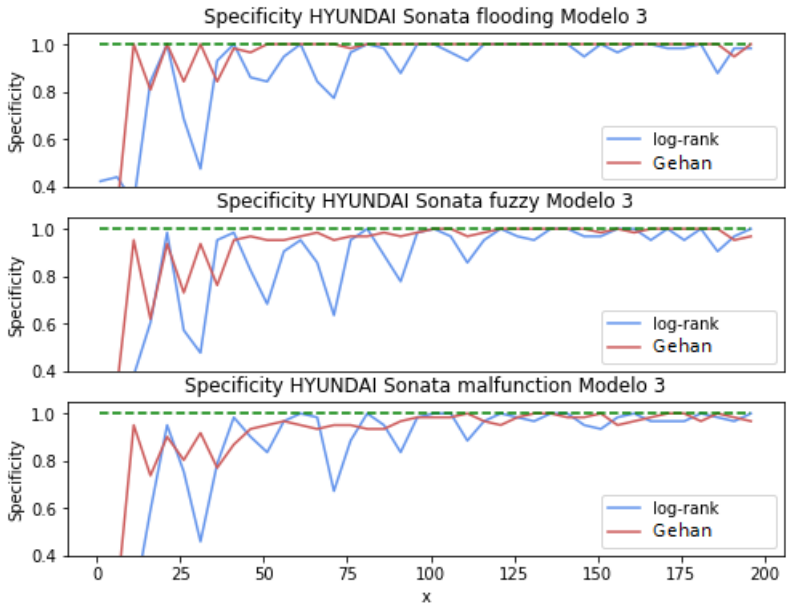

Figura 2.19: Especificidad del Algoritmo 3 en función del parámetro x para los tres ataques del HYUNDAI Sonata. Base Survival.

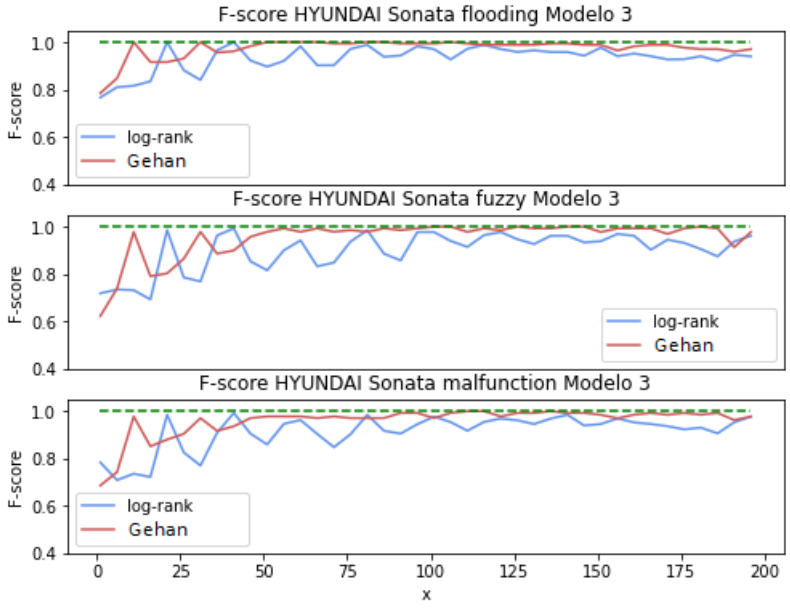

Figura 2.20: F-score del Algoritmo 3 en función del parámetro x para los tres ataques del HYUNDAI Sonata. Base Survival.

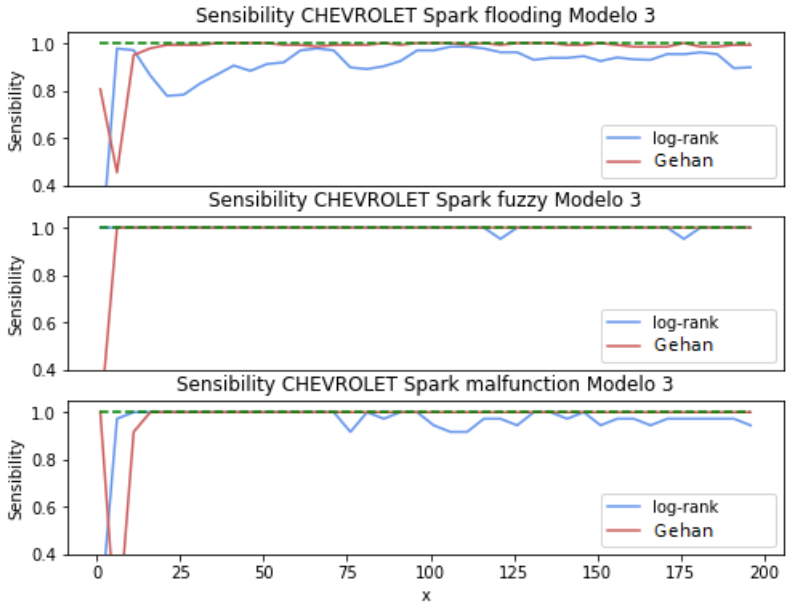

Figura 2.21: Sensibilidad del Algoritmo 3 en función del parámetro x para los tres ataques del CHEVROLET Spark. Base Survival.

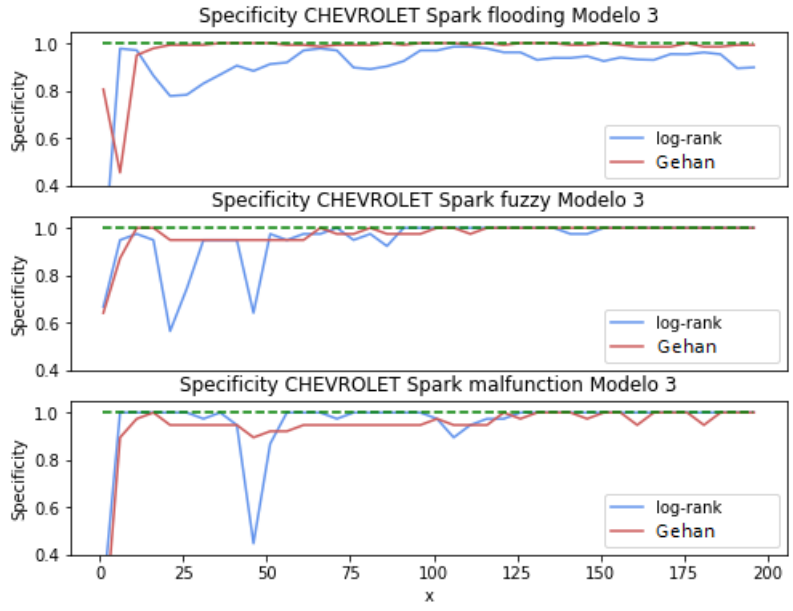

Figura 2.22: Especificidad del Algoritmo 3 en función del parámetro x para los tres ataques del CHEVROLET Spark. Base Survival.

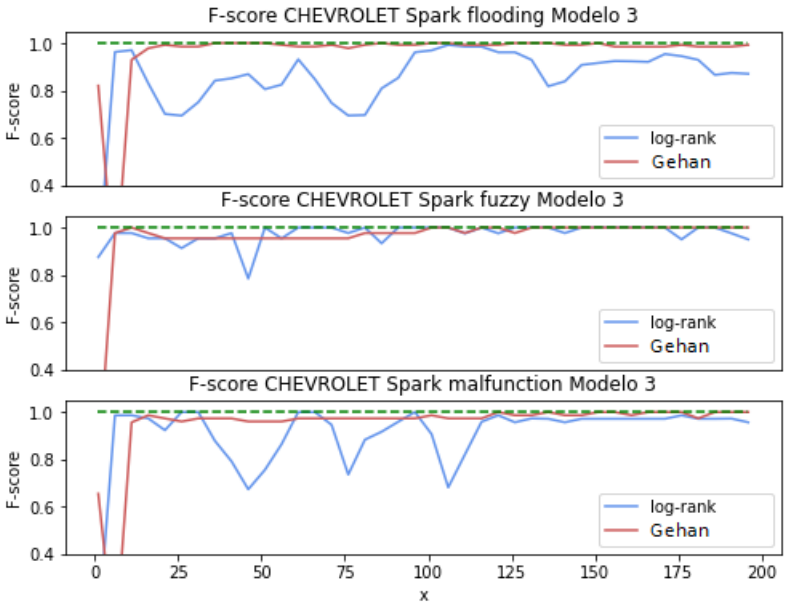

Figura 2.23: F-score del Algoritmo 3 en función del parámetro x para los tres ataques del CHEVROLET Spark. Base Survival.

Algoritmo 3		***Flooding***		***Fuzzy***		***Malfunction***	
		μ	σ	μ	σ	μ	σ
Soul	lr	0.89	0.09	0.99	0.01	0.75	0.15
	Ge	0.99	0.01	0.994	0.003	0.95	0.01
Sonata	lr	0.95	0.03	0.93	0.04	0.94	0.03
	Ge	0.99	0.01	0.99	0.02	0.98	0.01
Spark	lr	0.89	0.08	0.99	0.02	0.93	0.08
	Ge	0.99	0.01	0.99	0.02	0.98	0.01

Tabla 2.12: Para el Algoritmo 3, medias y desviaciones típicas de los F-scores para $x \in \{40, 45, \ldots, 200\}$ para ambos tests. Base Survival.

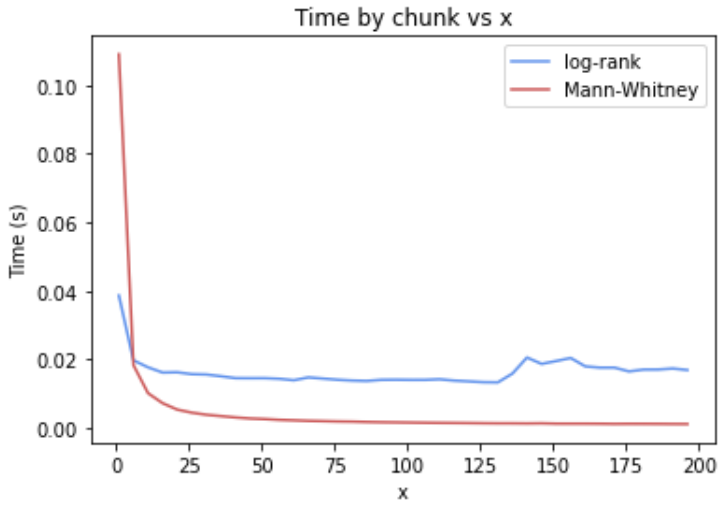

Figura 2.24: Tiempo medio de ejecución por paquete en función de x para el ataque *flooding* del KIA Soul. Base Survival.

¿Por qué el test log-rank presenta tanta inestabilidad en la especificidad con el parámetro x? ¿Por qué las especificidades no se corresponden con el nivel de significación? Hallamos las respuestas en la Figura 2.25, donde se muestran las densidades de las variables Y^x y Z^x para $x \in \{10, \ldots, 200\}$ para el KIA Soul. Las variables Z^x son calculadas a partir de los primeros 60 000 mensajes de la base *flooding*, en los que no hay ninguna inyección maligna. Dado un x, cuando más alta sea la densidad de sus tiempos entre mensajes menor será su varianza. Vemos como dependiendo del x la densidad alcanza mayor o menor altura, habiendo picos y valles que se repiten periódicamente. De esto deducimos que los mensajes no son generados de forma individual e independiente, sino que forman parte de lotes de cierto número de mensajes que se generan de forma periódica. Dado que las Z^x son calculadas sobre una conducción libre, esperaríamos que se distribuyeran de igual forma a las Y^x. Sin embargo, vemos cómo existen x para los cuales las densidades de Y^x presentan picos y las de Z^x presentan valles. Para dichos valores de x el test log-rank halla diferencias entre ambas distribuciones y H_0 es frecuentemente rechazada. Esto ocurre por ejemplo para $x = 50$, para el cual en la Figura 2.16 vemos como se obtiene una mala especificidad. Para otros x, como $x = 105$, ambas densidades presentan picos, no rechazándose generalmente H_0 y obteniendo muy buena especificidad. Sin embargo, aunque la forma de la distribución cambie, la media permanece invariante para todo el rango de x, por lo que el test de Gehan no halla diferencias significativas. Es

por esto que la especificidad presenta una fuerte dependencia con x para el test log-rank y no para el test de Gehan.

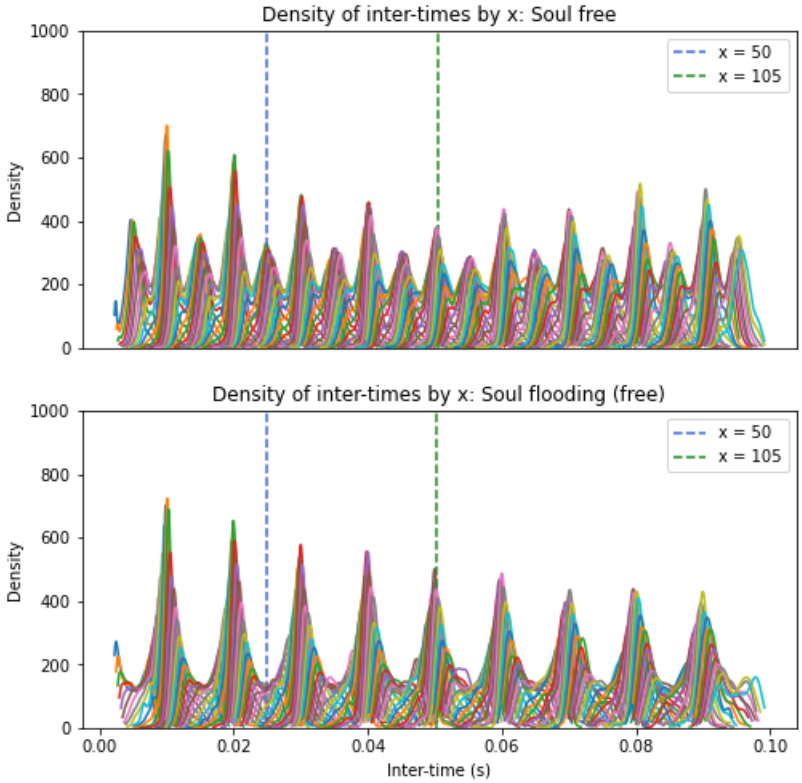

Figura 2.25: Para el KIA Soul y para $x \in \{10, \ldots, 200\}$, arriba: densidades de las variables Y^x calculadas a partir de la base libre y, abajo: densidades de las variables Z^x calculadas a partir de los primeros 60 000 mensajes de la base *flooding*. Base Survival.

De todo esto concluimos tres cosas. Por un lado, el proceso de generación de mensajes legítimos por lotes hace que los tiempos entre mensajes presenten una fuerte correlación y no se cumpla la hipótesis de independencia requerida en los contrastes log-rank y Gehan. Esto podría explicar que las especificidades obtenidas no se correspondan con el nivel de significación fijado. Por otro lado, dicho proceso de generación de mensajes evoluciona en el tiempo, dando lugar a una alta variabilidad de los resultados para el test log-rank. Sin embargo, la media de los tiempos entre mensajes es una variable que caracteriza correctamente una conducción libre de ataques. De hecho, bajo H_0 los tiempos medios son prácticamente idénticos, otra posible razón por la que las especificidades del test de Gehan sean tan altas (por encima de $1 - \alpha = 0.95$). Este diferente com-

portamiento de los tests log-rank y Gehan merece, en cualquier caso, un mayor estudio en el futuro, teniendo también presente que, en otras situaciones prácticas, el patrón de ataques podría diferir del existente en los datos analizados. Además, en situaciones en las que las funciones de riesgo de los tiempos entre mensajes libres de ataque y contaminados se crucen, los citados tests podrían tener baja potencia estadística, requiriéndose tests alternativos.

En resumen, aunque quedan ciertos aspectos por discutir acerca de la correlación de los datos y el no cumplimiento de la hipótesis de independencia requerida en los contrastes utilizados, es un hecho que los resultados obtenidos mediante el test de Gehan son realmente buenos. Esto, unido a que los tiempos de ejecución son considerablemente menores que los del test log-rank, nos anima a utilizar el algoritmo de detección 3 con el test de Gehan para realizar una optimización de sus parámetros y obtener los algoritmos finales. No obstante, la evolución temporal de la generación de mensajes nos ha dado que pensar. Una posibilidad es que el modo de conducción (carretera, ciudad, autovía,...) influya en el proceso de generación de los mensajes, y la base libre se haya tomado en un modo de conducción diferente a las bases atacadas. Otra posibilidad es que el modo de generación de los mensajes cambie en el tiempo independientemente de factores externos. Así, antes de pasar a los algoritmos finales, nos gustaría proponer dos algoritmos adicionales que, aunque de dudosa aplicabilidad real, son menos sensibles a este hecho. La diferencia de estos algoritmos con el Algoritmo 3 radica en la base libre a considerar.

2.2.2. Algoritmo 4: tiempo entre mensajes con base libre conjunta

La primera de las alternativas consiste en considerar una base libre que sea capaz de englobar todos los modos de generación de mensajes. Si es cierta nuestra hipótesis acerca de que la generación de mensajes cambia en función del modo de conducción, habría que considerar una base libre obtenida a partir de todos los modos de conducción posibles, cuanto más general mejor. En nuestro caso, al conjunto libre original de cada vehículo le añadimos fragmentos libres de sus bases atacadas. Por ejemplo, para el KIA Soul se añaden los fragmentos que se muestran en color violeta en la Figura 2.26. Análogamente se procede para los otros dos vehículos. En el Tabla 2.13 se muestra, para cada vehículo, el tamaño de las bases libres conjuntas y el número de datos libres que aporta cada conjunto de datos a dichas bases.

Una vez obtenidas las bases libres conjuntas, para un vehículo y un ataque dados, el algoritmo de detección es el que se muestra en el Tabla 2.14. Co-

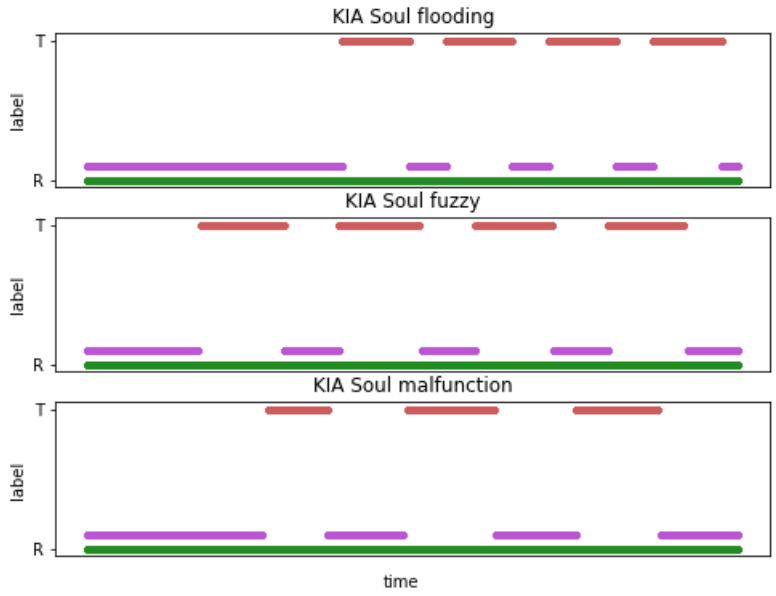

Figura 2.26: Para el KIA Soul y para cada uno de sus ataques, en verde los mensajes libres, en rojo los mensajes inyectados y en violeta los mensajes añadidos a la base libre original. Base Survival.

	Libre	*Flooding*	*Fuzzy*	*Malfunction*	**Total**
Soul	192 516	96 453	123 890	108 236	**521 095**
Sonata	117 173	77 869	81 466	78 371	**354 879**
Spark	136 934	64 690	47 425	48 837	**297 886**

Tabla 2.13: Para cada vehículo, datos libres que proporciona cada conjunto de datos a la base libre conjunta. En negrita número de mensajes que forman las bases libres conjuntas. Base Survival.

Algoritmo de detección 4

1. Para la muestra de ataque, se crean N paquetes de tamaño L con saltos de tamaño j.
2. Los paquetes que cuenten con al menos un porcentaje p de mensajes malignos son etiquetados como **ataque**. Los que no, son etiquetados como **no ataque**.
3. Se fija el parámetro x.
4. Se calculan los valores que toma la variable Y^x en la base libre.
5. Se calculan los valores que toman las variables $Z^{x,n}$ para cada uno de los paquetes $n \in \{1, \ldots, N\}$.
6. Se realiza un test de comparación a nivel de significación α entre las distribuciones de Y^x y Z^x_n para cada $n \in \{1, \ldots, N\}$, habiendo eliminado previamente de la muestra de Y^x los valores que pertenezcan al paquete n.

$$\begin{cases} H_0: & Y^x =^d Z^{x,n} \\ H_1: & Y^x \neq^d Z^{x,n}, \end{cases} \qquad n \in \{1, \ldots, N\}.$$

Los paquetes para los que H_0 se rechaza son clasificados como **alerta**. Los que no, son clasificados como **no alerta**.
7. Se obtiene la matriz de confusión entre alertas y ataques y, a partir de ella, se calculan las medidas de precisión de interés: sensibilidad, especificidad y F-score.

Tabla 2.14: Algoritmo de detección 4.

mo vemos, el algoritmo es muy similar al Algoritmo 3, salvo por dos factores importantes que hay que tener en consideración:

- Dado que en este algoritmo las bases libres están divididas en fragmentos, cada uno con tiempos diferentes, los valores de Y^x se calculan por separado en cada fragmento y luego se agrupan formando una muestra conjunta. Por ejemplo, para el KIA Soul se calcula la variable Y^x en cada fragmento violeta de la Figura 2.26 y se añadirían dichos valores a la muestra dada por la base libre.

- Para un n dado, la muestra de Y^x utilizada en el test de comparación de distribuciones será la muestra original de Y^x habiendo eliminado los valores que pertenezcan al intervalo n. Esto lo hacemos basándonos en la técnica de validación cruzada con el objetivo de garantizar independencia entre las muestras de Y^x y $Z^{x,n}$.

Fijando los parámetros habituales:

$$L = 5000, \qquad j = 1000, \qquad p = 0.06, \qquad \alpha = 0.05$$

y tomando de nuevo $x \in \{1, 5, 10, 15, \ldots, 200\}$, en las Figuras 2.27-2.29 y en el Tabla 2.15 se muestran los F-scores, con sus respectivas medias y desviaciones típicas para $x \in \{40, 45, \ldots, 200\}$. Los resultados para el test de Gehan son similares a los del algoritmo anterior.

La diferencia es mucho más notable para el test log-rank, con un aumento significativo en las precisiones y, sobre todo, unos resultados mucho más estables en función de x. Vemos como con este algoritmo, al menos con esta elección de parámetros, el ataque *malfunction* del KIA Soul sigue siendo el peor detectado. Aunque no se incluyen las gráficas de sensibilidad y especificidad, para el test de Gehan el motivo de no alcanzar una precisión tan buena como en el resto de casos sigue siendo la especificidad, es decir, sigue habiendo más falsos positivos de los que nos gustaría. Esto podría arreglarse, por ejemplo, disminuyendo el nivel de significación α de manera que al algoritmo le cueste más detectar una alerta, aunque esto podría afectar también a las precisiones del resto de ataques. Las conclusiones de este algoritmo se presentan a continuación:

- Este algoritmo iguala los resultados del Algoritmo 3 con el test de Gehan y los mejora con el test log-rank, tanto en la media de las precisiones como en su desviación típica. Que los resultados sean estables con x es algo muy importante ya que queremos que nuestro algoritmo sea lo menos sensible posible a pequeñas variaciones de sus parámetros.

Algoritmo 4		***Flooding***		***Fuzzy***		***Malfunction***	
		μ	σ	μ	σ	μ	σ
Soul	lr	0.96	0.02	0.99	0.01	0.88	0.07
	Ge	0.99	0.01	0.995	0.002	0.95	0.01
Sonata	lr	0.96	0.02	0.94	0.04	0.95	0.03
	Ge	0.99	0.01	0.99	0.01	0.99	0.01
Spark	lr	0.94	0.03	0.99	0.02	0.98	0.02
	Ge	0.99	0.01	0.99	0.02	0.98	0.01

Tabla 2.15: Medias y desviaciones típicas de los F-scores para $x \in \{40, 45, \ldots, 200\}$ para ambos tests aplicando el Algoritmo 4. Base Survival.

Figura 2.27: F-score del Algoritmo 4 en función del parámetro x para los tres ataques del KIA Soul. Base Survival.

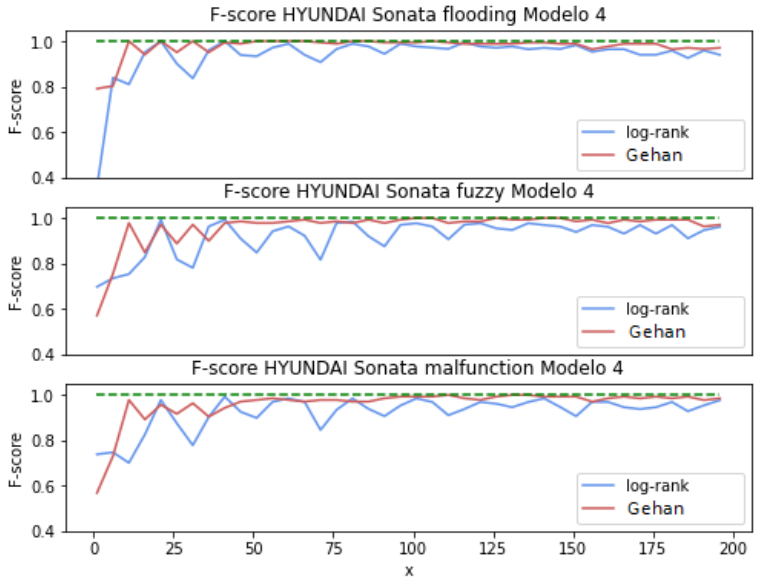

Figura 2.28: F-score del Algoritmo 4 en función del parámetro x para los tres ataques del HYUNDAI Sonata. Base Survival.

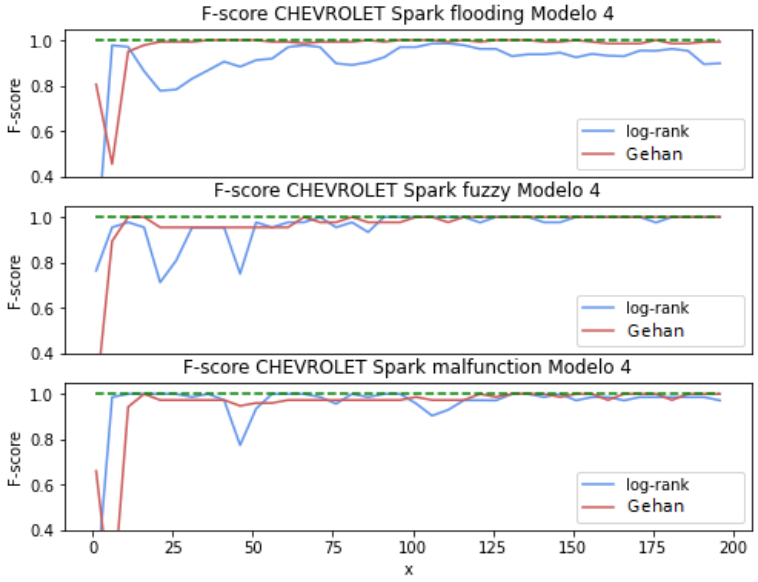

Figura 2.29: F-score del Algoritmo 4 en función del parámetro x para los tres ataques del CHEVROLET Spark. Base Survival.

- Este algoritmo es una extensión del Algoritmo 3, y lo hemos desarrollado para estudiar como cambian los resultados al considerar una base libre más general. No es aplicable a tiempo real ya que para construir la base libre hemos hecho uso de partes de las bases atacadas. En un caso real se podría tratar de conseguir una base libre que abarque varios modos de generación de mensajes, midiéndola durante intervalos de tiempo separados o, si nuestra hipótesis fuera cierta, con diferentes modos de conducción. En cualquier caso, en la realidad se aplicaría el Algoritmo 3 y no este algoritmo.

A continuación se comenta otra alternativa interesante que, aunque de dudosa aplicación real, también abarca el problema de los diferentes modos de generación de mensajes.

2.2.3. Algoritmo 5: tiempo entre mensajes con base libre dinámica

Este algoritmo se diferencia de los anteriores en que la base libre utilizada para la comparación no está fija, si no que se actualiza a lo largo del tiempo. Partiremos como base libre de los primeros M mensajes del ataque en cuestión e iremos actualizándola con los mensajes que creamos estar seguros de que

son libres. Obviamente, para cada ataque debemos cerciorarnos previamente de que estos M mensajes iniciales son libres. Para un vehículo y ataque dados, el algoritmo de detección es el mostrado en el Tabla 2.16. Con respecto a los algoritmos anteriores contamos con dos nuevos parámetros:

- M: es el tamaño de la base libre, que como ya hemos comentado permanece constante. Otro posible algoritmo consistiría en no eliminar los primeros L mensajes y simplemente ir añadiendo los paquetes libres, haciendo la base libre cada vez más grande. Esto no nos interesa actualmente ya que buscamos darle una continuidad a los datos, eliminando así el posible efecto de diferentes modos de conducción. Tras asegurarnos de que en todas las bases atacadas los primeros 25 000 mensajes son libres, decidimos fijar un valor unificado $\boldsymbol{M = 25\,000}$. Al igual que ocurría en el Algoritmo 4, la base libre puede estar fraccionada, por lo que la variable Y^x se calcula en cada fragmento por separado y luego se unifica en una muestra conjunta.

- β: este parámetro, que ha de ser mayor o igual a α, determina si la base libre es actualizada o no. Cuando más grande sea, mayor seguridad tendremos de que la actualización se produzca con mensajes libres. Se podría fijar $\beta = \alpha$, aunque correríamos un elevado riesgo de tomar como parte de la base libre mensajes malignos, lo que arrastraría el problema. Decidimos fijar $\boldsymbol{\beta = 0.50}$.

De nuevo fijando los parámetros:

$$L = 5000, \qquad j = 1000, \qquad p = 0.06, \qquad \alpha = 0.05$$

y tomando $x \in \{1, 6, 11, 16, \ldots, 196\}$, en las Figuras 2.27, 2.31 y 2.32 y en el Tabla 2.17, se muestran los F-scores, con sus respectivas medias y desviaciones típicas para $x \in \{51, 56, \ldots, 196\}$.

Los resultados para el test de Gehan son muy similares a los del Algoritmo 4, presentando muy buena precisión y poca inestabilidad, y siendo el ataque *malfunction* del KIA Soul el único detectado con un F-score por debajo de 0.98. Para el test log-rank los resultados también son parecidos, aumentando notablemente el F-score del ataque que acabamos de mencionar y disminuyendo ligeramente la variabilidad.

Algoritmo de detección 5

1. Se toman como base libre los primeros M mensajes de la base de ataque. Todos ellos deben ser libres, es decir, estar dotados de la etiqueta R. Para la muestra de ataque restante, se crean N paquetes de tamaño L con saltos de tamaño j.
2. Los paquetes que cuenten con al menos un porcentaje p de mensajes malignos son etiquetados como **ataque**. Los que no, son etiquetados como **no ataque**.
3. Se fija el parámetro x.
4. Se inicializa: $n = 1$.
5. Se calculan los valores que toma la variable Y^x en la base libre.
6. Se calculan los valores que toman las variables Z_n^x para cada el paquete n.
7. Se realiza un test de comparación a nivel de significación α entre las distribuciones de Y^x y Z_n^x:

$$\begin{cases} H_0: & Y^x =^d Z^{x,n} \\ H_1: & Y^x \neq^d Z_{x,n}, \end{cases} \qquad n \in \{1, \ldots, N\}.$$

a) Si se rechaza H_0 entonces el paquete n es clasificado como **alerta** y la base libre no es actualizada.

b) Si no se rechaza H_0 pero el pvalor es inferior a β entonces el paquete n es clasificado como **no alerta** pero la base libre no es actualizada.

c) Si no se rechaza H_0 y además el pvalor es superior a β entonces el paquete n es clasificado como **no alerta** y la base libre es actualizada. La actualización consiste en eliminarle los primeros L mensajes y añadirle los L mensajes pertenecientes al paquete n, de manera que la base libre siga teniendo un tamaño M.

8. Si $n < N$, se hace $n = n + 1$ y se vuelve al paso 5. Si $n = N$ se avanza al paso 9.
9. Se obtiene la matriz de confusión entre alertas y ataques y, a partir de ella, se calculan las medidas de precisión de interés: sensibilidad, especificidad y F-score.

Tabla 2.16: Algoritmo de detección 5.

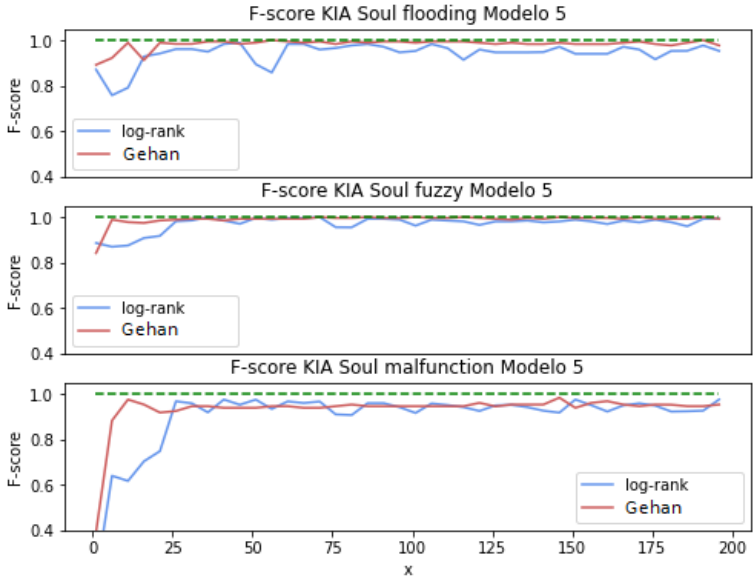

Figura 2.30: F-score del Algoritmo 45en función del parámetro x para los tres ataques del KIA Soul. Base Survival.

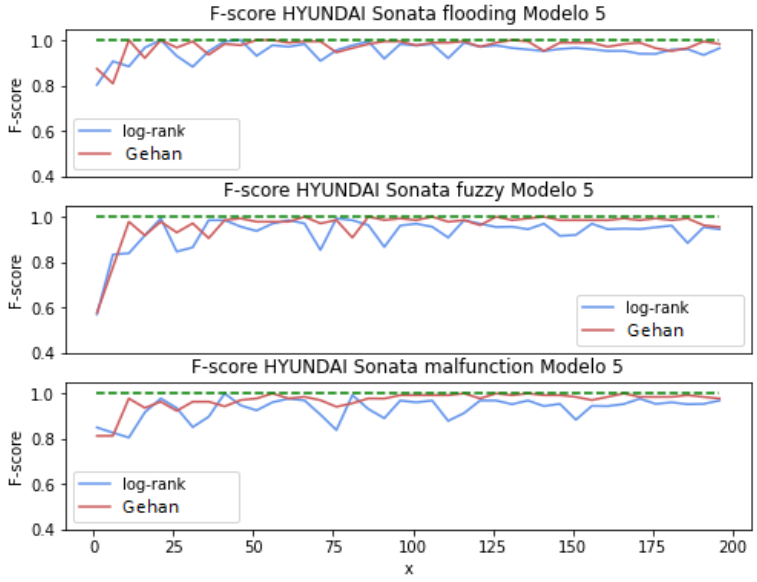

Figura 2.31: F-score del Algoritmo 5 en función del parámetro x para los tres ataques del HYUNDAI Sonata. Base Survival.

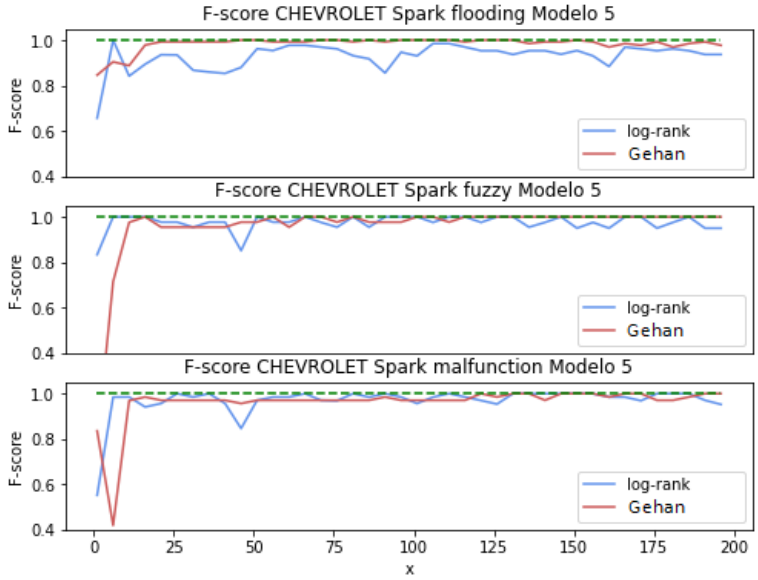

Figura 2.32: F-score del Algoritmo 5 en función del parámetro x para los tres ataques del CHEVROLET Spark. Base Survival.

Algoritmo 5		***Flooding***		***Fuzzy***		***Malfunction***	
		μ	σ	μ	σ	μ	σ
Soul	lr	0.95	0.03	0.98	0.01	0.94	0.02
	Ge	0.99	0.01	0.996	0.003	0.95	0.01
Sonata	lr	0.96	0.02	0.95	0.03	0.94	0.03
	Ge	0.98	0.01	0.98	0.02	0.98	0.01
Spark	lr	0.95	0.03	0.98	0.02	0.99	0.02
	Ge	0.99	0.01	0.99	0.01	0.98	0.01

Tabla 2.17: Medias y desviaciones típicas de los F-scores para $x \in \{40, 45, \ldots, 200\}$ para ambos tests aplicando el Algoritmo 5. Base Survival.

Las conclusiones de este algoritmo se presentan a continuación:

- Para ambos tests las precisiones y su estabilidad son muy buenas. El test de Gehan se comporta de forma similar que para el Algoritmo 3, y el test log-rank mejora notablemente, tanto en media como en varianza, sobre todo para el ataque *flooding* del KIA Soul.
- Aunque a diferencia del Algoritmo 4 este algoritmo si puede ser aplicado

a tiempo real, el modo de construcción de la base libre hace que este algoritmo no sea fiable a la larga en una implementación real. En los otros algoritmos se parte de una base libre fija en la que su datos son recogidos con certeza de que no están sufriendo un ataque. Sin embargo, en este algoritmo la base libre es dinámica y sus datos dependen del propio algoritmo, existiendo una probabilidad alta de arrastre de error: si un paquete atacado es incorporado a la base libre es más probable que esto vuelva a ocurrir, dando lugar a una base libre contaminada. Una forma de mitigar este hecho sería reiniciar el algoritmo periódicamente. Sin embargo, para ello habría que estar seguros de que esos 25 000 datos iniciales son libres, algo que no es factible en un uso habitual del vehículo. Por esto, aunque los resultados que nos arroja en este estudio son buenos, decidimos quedarnos con el Algoritmo 3 como algoritmo final.

2.2.4. Selección de algoritmo y optimización de sus parámetros

Como ya hemos comentado, por la no aplicabilidad a tiempo real del Algoritmo 4 y la alta probabilidad de arrastre de error de Algoritmo 5, decidimos basarnos en el Algoritmo 3 con el test de Gehan para realizar la optimización de parámetros. Para la base Survival optimizaremos con respecto al F-score tanto para cada vehículo por separado como para los tres vehículos en conjunto, obteniendo una combinación de parámetros que, idealmente, aportará buenas precisiones para los tres ataques de los tres vehículos. Finalmente, aplicaremos esta combinación de parámetros sobre la base Car-Hacking, comprobando así la capacidad predictiva del algoritmo sobre un conjunto de datos diferente al del ajuste. Nuestro algoritmo tiene demasiados parámetros, por lo que tomaremos ciertas consideraciones sobre ellos y únicamente optimizaremos con respecto a x, el parámetro de mayor influencia en los resultados, y α, el parámetro que mide la tensión entre la sensibilidad y la especificidad.

- Los parámetros L y x determinan el tamaño de las muestras de las variables $Z^{x,n}$ de la forma: $m = \lfloor L/x \rfloor$. Para realizar la optimización fijaremos un tamaño aceptable, digamos $m = 50$, y L quedará determinado por x de la forma: $\boldsymbol{L = 50 \cdot x}$.

- Por un lado, el parámetro j determina el tiempo de reacción ante un ataque. Por otro lado, j determina el número de contrastes de hipótesis en la ejecución del programa, por lo que también influye directamente en el tiempo de ejecución. En cuanto al tiempo de reacción, lo ideal sería fijar

$j = 1$, de manera que la detección fuera instantánea. Sin embargo, esto implicaría realizar tantos contrastes de hipótesis como mensajes en la base de datos, lo que supondría un tiempo de ejecución inaccesible para hacer un estudio de parámetros. El test de Gehan nos permite fijar $\boldsymbol{j = 50}$ con tiempos de ejecución razonables. Para $j = 50$ el tiempo de reacción es de aproximadamente 0.025 s.

- Hasta ahora hemos fijado $p = 0.06$, con lo cual los paquetes con un porcentaje de mensajes malignos inferior al 6 % no eran considerados como atacados. Haciendo esto no estamos siendo fieles a la realidad, ya que puede haber ataques que causen problemas con un porcentaje menor de intrusiones. Así, fijaremos $\boldsymbol{p = 0}$ de manera que un paquete con tan solo un mensaje maligno sea considerado como ataque. Esto puede dar lugar a un incremento en el número de falsos negativos, ya que paquetes con pocos mensajes malignos son difícilmente detectables.

Habiendo fijado L, j y p, haremos variar $x \in \{40, \ldots, 200\}$ y $\alpha \in [0, 1]$. Para cada vehículo y cada ataque tomamos una muestra de entrenamiento y una muestra de test. Aunque podría tomarse un porcentaje de entrenamiento y test diferente para cada caso, por simplicidad decidimos tomar para todos el 65 % inicial de los datos como muestra de entrenamiento y el 35 % restante como muestra de test. En la Figura 2.33 se muestra como en todos los casos ambas muestras contienen tanto partes libres como partes atacadas.

Como ya hemos comentado, optimizamos tanto para cada vehículo por separado como para los tres vehículos en conjunto.

Optimización individual para cada vehículo

En el Tabla 2.18 se muestran los valores óptimos de x y α y las sensibilidades, especificidades y F-scores obtenidos en cada caso para la muestra de test, además de los F-scores de las muestras de entrenamiento. A pesar de haber fijado $p = 0$ vemos cómo las sensibilidades obtenidas son muy buenas. Esto se debe a los altos niveles de significación seleccionados, que detectan diferencias muy fácilmente. Esto es un arma de doble filo, ya que puede dar lugar a un aumento en el número de falsos positivos. Por ejemplo, para el ataque *flooding* del Spark, a pesar de obtener muy buena precisión en la muestra de entrenamiento, la especificidad es realmente mala en la muestra de test. Por otro lado, destacan los resultados del Sonata, para el cual no se dan falsas alarmas, y los ataques *fuzzy* y *malfunction* del Spark, para el cual todos los paquetes atacados

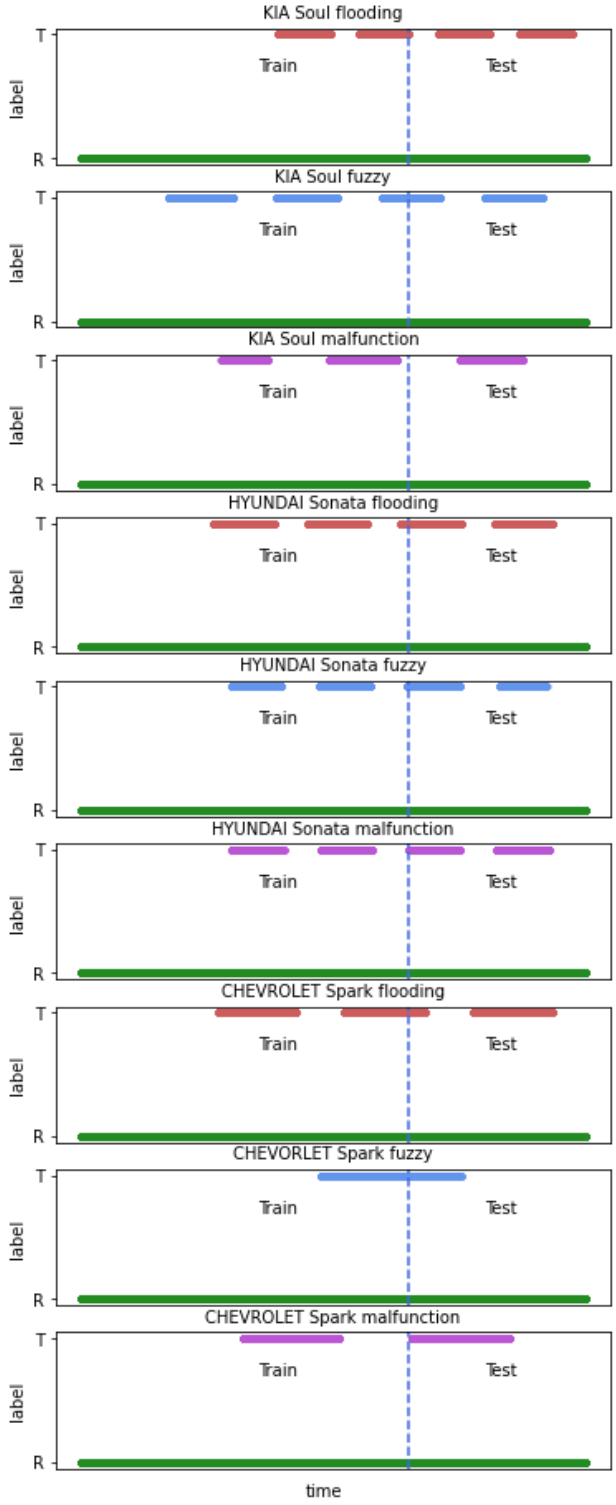

Figura 2.33: Separación en muestras de entrenamiento y test. Base Survival.

son alertados. Las especificidades obtenidas de nuevo no se corresponden con los niveles de significación fijados, siendo mucho mejores de lo esperado. Como ya hemos comentado, esto puede tener dos razones. Por un lado, que los tiempos medios comparados sean casi idénticos y el test de Gehan devuelva p-valores muy altos. Por otro lado, que no se cumpla la hipótesis de independencia entre los tiempos entre mensajes y el test de Gehan no sea totalmente válido.

(x_{opt}, α_{opt})		**Sens.**	**Esp.**	**F-score**	**Train F**
Soul (51, 0.33)	*Flooding*	0.989	0.992	0.993	0.996
	Fuzzy	0.992	0.987	0.992	0.993
	Malfunction	0.992	0.999	0.992	0.992
	Media	**0.991**	**0.993**	**0.992**	**0.994**
Sonata (49, 0.49)	*Flooding*	0.990	1	0.995	0.992
	Fuzzy	0.988	1	0.994	0.987
	Malfunction	0.985	1	0.992	0.991
	Media	**0.988**	**1**	**0.994**	**0.990**
Spark (80, 0.44)	*Flooding*	0.993	0.724	0.941	0.993
	Fuzzy	1	0.992	0.993	0.991
	Malfunction	1	0.917	0.981	0.988
	Media	**0.998**	**0.878**	**0.972**	**0.991**

Tabla 2.18: Parámetros óptimos para cada vehículo y sus correspondientes sensibilidades, especificidades y F-scores obtenidos en cada caso para la muestra de test, además de los F-scores de las muestras de entrenamiento. Base Survival.

Optimización global

Optimizando para los tres vehículos en conjunto obtenemos:

$$(x_{opt}, \alpha_{opt}) = (\mathbf{50}, \mathbf{0.27}),$$

y las sensibilidades, especificidades y F-scores para la muestra de test que se recogen en el Tabla 2.19. Como es lógico, los resultados del F-score empeoran con respecto a las optimizaciones individuales. Para los ataques *flooding* del Soul y del Spark se dan un número considerable de falsas alertas. Las sensibilidades del Soul son muy buenas. Se obtiene un F-score promedio de 0.978, buen resultado teniendo en cuenta la generalidad del algoritmo.

Para estudiar la capacidad de detección, en la Figura 2.34 se muestran, para esta combinación de parámetros, los F-scores en función del ratio de ataque

de los paquetes, es decir, de la proporción de mensajes malignos. Vemos como basta un ratio de ataque del 6 % o incluso inferior para que todos los ataques sean detectados. Si vamos ahora a la Figura 2.35, en la que para cada caso se representa el ratio de ataque a lo largo del tiempo, vemos como los ratios suben rápidamente hasta colocarse entorno a un 45, 35 y 25 % para los ataques *flooding*, *fuzzy* y *malfunction* respectivamente. Para cada muestra de test, todos los paquetes cuyo ratio de ataque supera el valor dado por la línea gris han sido alertados. Dependiendo del efecto que puedan producir los ataques sobre el vehículo puede interesarnos disminuir α para reducir las falsas alertas a cambio de no ser capaces de detectar los ataques con un ratio de ataque pequeño, o aumentar α para detectar cualquier intrusión por pequeña que sea a cambio de dar falsas alertas. Un equilibrio entre estos dos aspectos lo da el F-score, que es lo que nosotros hemos maximizado.

(50, 0.27)		**Sens.**	**Esp.**	**F-score**	**Train F**
Soul	*Flooding*	0.994	0.841	0.966	0.994
	Fuzzy	0.991	1	0.995	0.993
	Malfunction	0.994	0.961	0.970	0.988
	Media	**0.993**	**0.934**	**0.977**	**0.992**
Sonata	*Flooding*	0.980	1	0.990	0.987
	Fuzzy	0.978	1	0.989	0.979
	Malfunction	0.973	1	0.986	0.983
	Media	**0.977**	**1**	**0.988**	**0.983**
Spark	*Flooding*	0.961	0.801	0.928	0.987
	Fuzzy	1	0.996	0.997	0.991
	Malfunction	0.985	1	0.992	0.980
	Media	**0.982**	**0.932**	**0.972**	**0.986**
	Media total	**0.955**	**0.979**	**0.979**	**0.987**

Tabla 2.19: Sensibilidades, especificidades y F-scores obtenidos en cada caso para la muestra de test, además de los F-scores de las muestras de entrenamiento, todo ello utilizando la combinación de parámetros óptima global. Base Survival.

Hablemos ahora del **tiempo de detección** de los ataques. Por un lado ya hemos comentado que fijando $j = 50$ el tiempo de reacción, es decir, el tiempo que se tarda en tomar cada paquete, es aproximadamente de 0.025 s. Por otro lado, con $x = 50$ el tiempo medio de ejecución es aproximadamente de 0.0035 s (ver Figura 2.24). Además, para los tres vehículos y los tres ataques pasan de

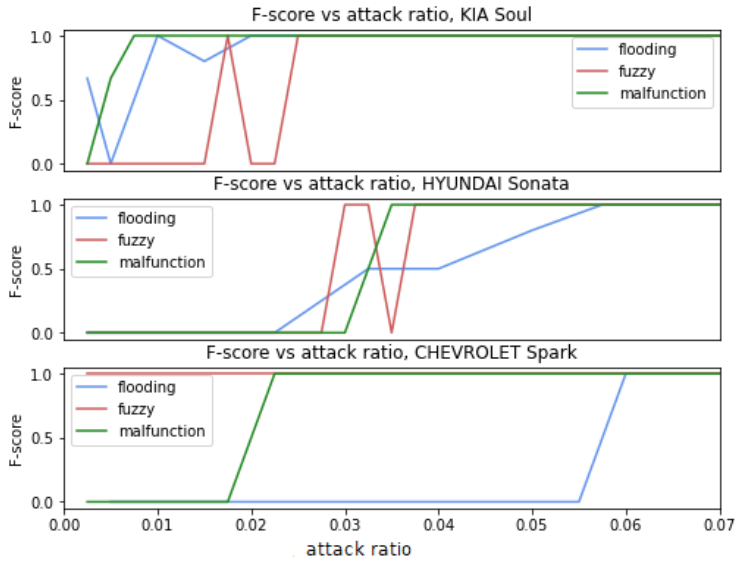

Figura 2.34: F-score en función del ratio de ataque para las muestras de test de los tres ataques de los tres vehículos. Base Survival.

media 3 paquetes desde el primer mensaje maligno hasta que la proporción de mensajes malignos en los paquetes es detectable por el algoritmo. Así, concluimos que el tiempo medio de detección de este algoritmo es $3 \times 0.0035 + 0.025 =$ **0.0355 s**, buen resultado si quisiéramos aplicarlo a tiempo real.

Finalizaremos esta sección aplicando el Algoritmo 3 sobre la base de datos Car-Hacking, comprobarndo así si nuestro algoritmo es útil para detectar anomalías en la red interna de otros vehículos.

2.2.5. Resultados del algoritmo seleccionado sobre la base de datos Car-Hacking

En primer lugar utilizaremos la combinación global óptima, lo que nos permitirá comprobar si es factible contar con un algoritmo único que detecte intrusiones con precisión sobre cualquier vehículo. Así, aplicando la combinación de parámetros:

$$x = 50, \quad \alpha = 0.27, \quad L = 2500, \quad j = 50, \quad p = 0$$

sobre la primera de las bases atacada de la base Car-Hacking obtenemos las métricas:

$$\textbf{Sens.} = \textbf{0.845}, \qquad \textbf{Esp.} = \textbf{0.988}, \qquad \textbf{F-score} = \textbf{0.906}\ .$$

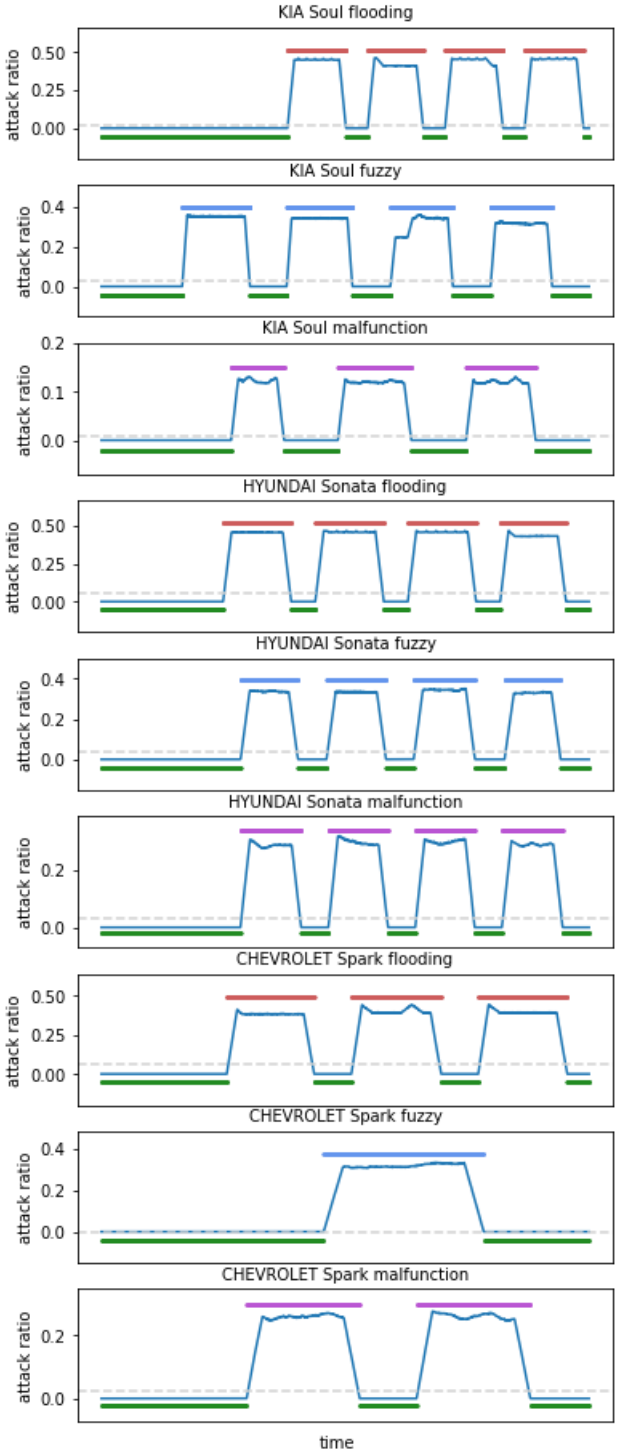

Figura 2.35: Ratio de ataque a lo largo del tiempo. En gris, para cada caso, el ratio de ataque a partir del cual todos los ataques de la muestra de test han sido detectados. Base Survival.

La especificidad es alta, de nuevo no correspondiéndose con el nivel de significación. Sin embargo, la detección de los ataques no es buena, dando lugar a una baja sensibilidad y por ende a un bajo F-score. En la Figura 2.36 podemos hallar el motivo, y es que los ratios de ataque de los ataques *replay malfunction* son más bajos que los tratados en la base Survival. De hecho, los ratios de ataque del *malfuncion* se encuentran al límite de los detectables en la base Survival con el nivel de significación $\alpha = 0.27$ (ver Figura 2.34). Además, podemos preguntarnos si para este vehículo existe también variabilidad con respecto al parámetro x, y la respuesta es afirmativa, tal y como podemos ver en la Figura 2.37. De hecho, teniendo en cuenta que estamos utilizando el test de Gehan, las desviaciones típicas son notablemente mayores que las obtenidas para nuestros tres vehículos iniciales, que rondaban el 0.01.

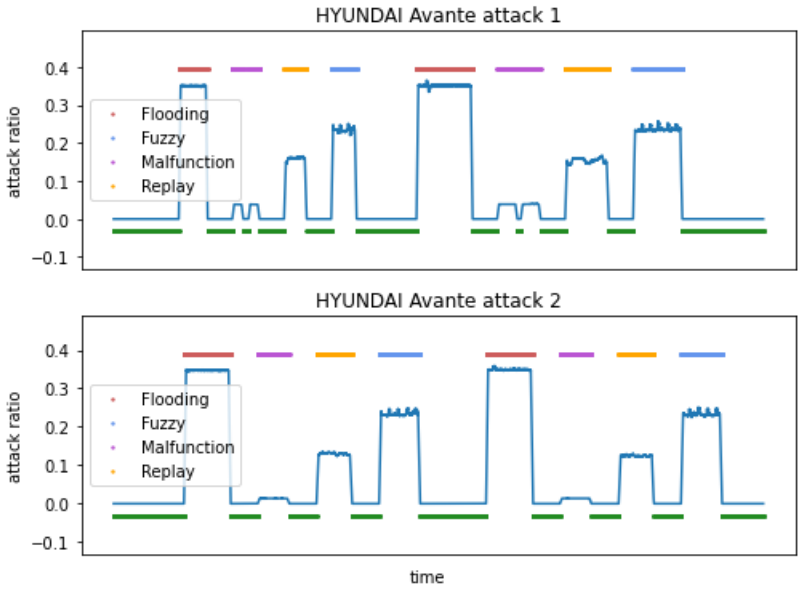

Figura 2.36: Ratio de ataque a lo largo del tiempo en los ataques. Base Car-Hacking.

Visto que $(x, \alpha) = (50, 0.27)$ no aporta buenas precisiones, empleamos de nuevo el Algoritmo 3 con el test de Gehan para optimizar con respecto a estos parámetros para este vehículo por separado. Con respecto al resto de parámetros, tomamos las mismas consideraciones que anteriormente ($L = 50 \cdot x, j = 50, p = 0$). Utilizamos la segunda base atacada como muestra de entrenamiento, y la primera como muestra de test, obteniendo la combinación $\boldsymbol{(x_{opt}, \alpha_{opt}) = (70, 0.45)}$ y las métricas:

$$\textbf{Sens.} = \mathbf{0.981}, \qquad \textbf{Esp.} = \mathbf{0.993}, \qquad \textbf{F-score} = \mathbf{0.987}\ .$$

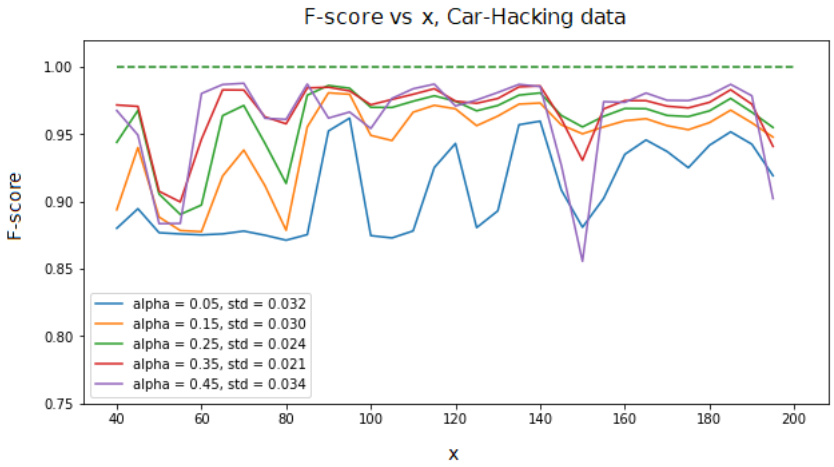

Figura 2.37: F-score en función del parámetro x para la primera base atacada. En la leyenda se muestra las desviación típica para cada nivel de significación. Base Car-Hacking.

Vemos cómo el hecho de aumentar el nivel de significación facilita la detección de los ataques, y el hecho de tomar $x = 70$ hace que la especificidad se mantenga alta. En la Figura 2.38 vemos como para un porcentaje de ataque superior al 3 % siempre se produce la detección, y para un porcentaje del 1.5 %, correspondiente al ataque *malfunction*, el F-score es superior a 0.9.

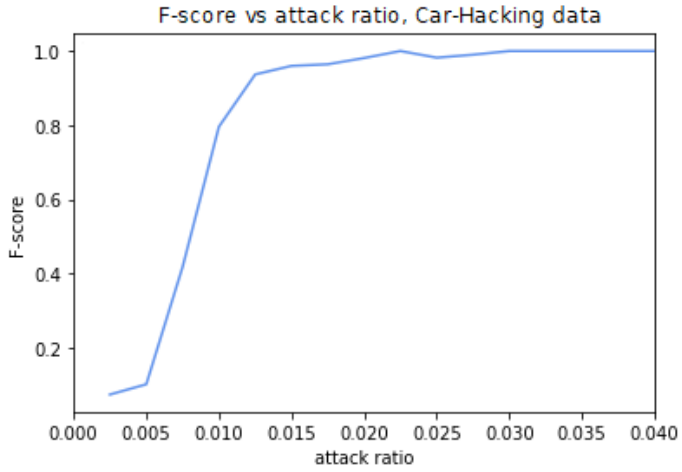

Figura 2.38: F-score en función del ratio de ataque para la muestra de test. Base Car-Hacking.

De los resultados para las bases Survival y Car-Hacking sacamos dos conclusiones principales. Nuestro algoritmo es capaz de detectar con alta precisión intrusiones en las redes vehiculares, quedando probado para cuatro vehículos distintos. Sin embargo, no es viable disponer de una combinación de parámetros global que se aplique con precisión sobre todos los vehículos, siendo necesaria una combinación de parámetros propia para cada uno de ellos.

2.3. Algoritmo de clasificación de anomalías en vehículos

Una vez detectado un ataque resulta altamente interesante estudiar si sus características coinciden con alguno de los ataques conocidos y, en ese caso, clasificarlo. Así, en este apartado propondremos un Algoritmo de clasificación en el que entrarán los paquetes alertados del algoritmo de detección. Para ello volveremos a hacer uso del ID de los mensajes, que aunque ya hemos visto que dentro de nuestro enfoque este atributo no resulta muy útil para detectar ataques, sí que lo es para clasificarlos una vez detectados. Como ya hemos comentado, cada ataque de los considerados en este estudio cumple con ciertas características a la hora de inyectar mensajes (*flooding* inyecta mensajes de máxima prioridad, *fuzzy* inyecta mensajes aleatorios...), por lo que cada uno de ellos influye de manera diferente sobre los tiempos entre mensajes de cada uno de los IDs. Estudiando cada ID por separado para después combinar la información nos permitirá clasificar los ataques según sus características. Así, en este algoritmo volveremos a hacer uso de las variables aleatorias:

$$Y_i^y = \text{tiempo entre } y \text{ mensajes con ID} = i \text{ en la base libre},$$
$$Z_i^y = \text{tiempo entre } y \text{ mensajes con ID} = i \text{ en la base atacada}.$$

Una vez un paquete es alertado como anomalía, el Algoritmo de clasificación se muestra en la Tabla 2.20. Este algoritmo clasifica los ataques en función del resultado de los tests de comparación de distribuciones que se realizan sobre las variables Y_i^y y Z_i^y. El nivel de significación γ ha de fijarse pequeño, de manera que cuando el test rechace H_0 haya cierta seguridad sobre ello. En el ataque *flooding*, la inyección de mensajes de máxima prioridad impide la comunicación correcta del resto de mensajes y provoca un cambio en la distribución de sus IDs. Es fácilmente identificable estudiando los IDs de máxima prioridad: 1) se detecta algún ID de máxima prioridad no perteneciente a la red vehicular, o 2) se rechaza H_0 para algún ID de máxima prioridad perteneciente a la red vehicular. Durante el ataque *fuzzy* se inyectan mensajes con IDs aleatorios, tanto de la red vehicular como mensajes externos. Aunque el tiempo entre mensajes cambia, el

impacto sobre el tiempo entre mensajes de cada ID de la red vehicular es muy pequeño, haciendo que no se rechacen rotundamente los contrastes de hipótesis. Así, el ataque *fuzzy* se identifica si: 1) se detecta algún ID no perteneciente a la red vehicular y que no es de máxima prioridad, o 2) no se rechaza H_0 para ninguno de los IDs de la red vehicular que no son de máxima prioridad. El ataque *malfunction* se centra en uno o unos pocos IDs de la red vehicular, por lo que la distribución de tiempos entre mensajes de dicho ID se ve fuertemente modificada. Sin embargo, no tiene el impacto del ataque *flooding* sobre el resto de IDs, por lo que sus distribuciones rara vez se ven afectadas. Es por eso por lo que clasificamos un ataque como *malfunction* cuando el número de test con rechazo de H_0 es n o menor. Por su parte, el ataque *replay* suele imitar mensajes de varios o incluso de la mayoría de los IDs de la red vehicular, rechazando la hipótesis nula en más ocasiones.

Recordemos que en el Algoritmo de clasificación contamos con tres parámetros adicionales: y, que marca el número de mensajes con cada ID entre los que se calcula el tiempo, γ, el nivel de significación de los tests de hipótesis para cada ID y n, que indica el número de test rechazados que diferencian a los ataques *malfunction* y *replay*. Tras un estudio preliminar del comportamiento de los IDs en cada ataque, nos parece razonable fijar $\boldsymbol{y = 3}$, $\boldsymbol{\gamma = 0.01}$ y $\boldsymbol{n = 6}$. Con respecto al resto de parámetros (L, j, x, α y p), para cada vehículo utilizamos la combinación óptima obtenida anteriormente. En la Tabla 2.21 se muestran los F-scores para cada muestra de test. El algoritmo es capaz de distinguir perfectamente entre los ataques *flooding*, *fuzzy* y *malfunction*, y las falsas alarmas son asignadas en su mayoría al ataque *fuzzy*. Es por esto por lo que para el Soul y el Spark los F-scores aumentan con respecto a los de la Tabla 2.18 para los ataques *flooding* y *malfunction* y dismininuyen para el ataque *fuzzy*, y para el Sonata los F-scores no cambian ya que no se dan falsas alarmas. Con respecto a la base Car-Hacking, de nuevo las falsas alarmas son clasificadas como *fuzzy* y en algún caso se confunden los ataques *malfunction* con ataques *replay*. Para visualizarlo, en la Figura 2.39 se muestra la clasificación para la base Car-Hacking. Al igual que para el algoritmo de detección, estos resultados podrían mejorarse optimizando con respecto a los parámetros y, γ y n.

Algoritmo de clasificación

1. Se listan los IDs I pertenecientes a la red del vehículo. Se listan los IDs de máxima prioridad J (no necesariamente $J \subset I$). Se listan los IDs K hallados en el paquete.
2. Se fija el parámetro y.
3. Para cada $i \in I$ se calculan los valores que toma la variable Y_i^y en la base libre.
4. Para cada $i \in I$ se calculan los valores que toma la variable Z_i^y en el paquete alertado.
5. Para cada $i \in I$ se realiza un test de comparación a nivel de significación γ entre las distribuciones de Y_i^y y Z_i^y

$$\begin{cases} H_0: & Y_i^y =^d Z_i^y \\ H_1: & Y_i^y \neq^d Z_i^y, \end{cases} \qquad n \in \{1, \ldots, N\}.$$

El ataque es clasificado como ***flooding*** si:

1. Existe algún $i \in K$ tal que $i \in J$ pero $i \notin I$.
2. Se rechaza H_0 para algún $i \in J$ e $i \in I$.

En otro caso, el ataque es clasificado como ***fuzzy*** si:

1. Existe algún $i \in K$ tal que $i \notin J$ e $i \notin I$,
2. No se rechaza H_0 para ningún $i \in (I \backslash J)$,

como ***malfunction*** si se rechaza H_0 para n o menos $i \in (I \backslash J)$, y como ***replay*** si se rechaza H_0 para más de n $i \in (I \backslash J)$. En otro caso, el paquete es clasificado como **anomalía desconocida**.

Tabla 2.20: Algoritmo de clasificación.

F-score	***Flooding***	***Fuzzy***	***Malfunction***	***Replay***	**Media**
Soul	0.995	0.984	0.994		**0.991**
Sonata	0.995	0.994	0.992		**0.994**
Spark	0.996	0.931	1		**0.976**
Avante	0.997	0.971	0.962	0.972	**0.975**

Tabla 2.21: F-scores del Algoritmo de clasificación.

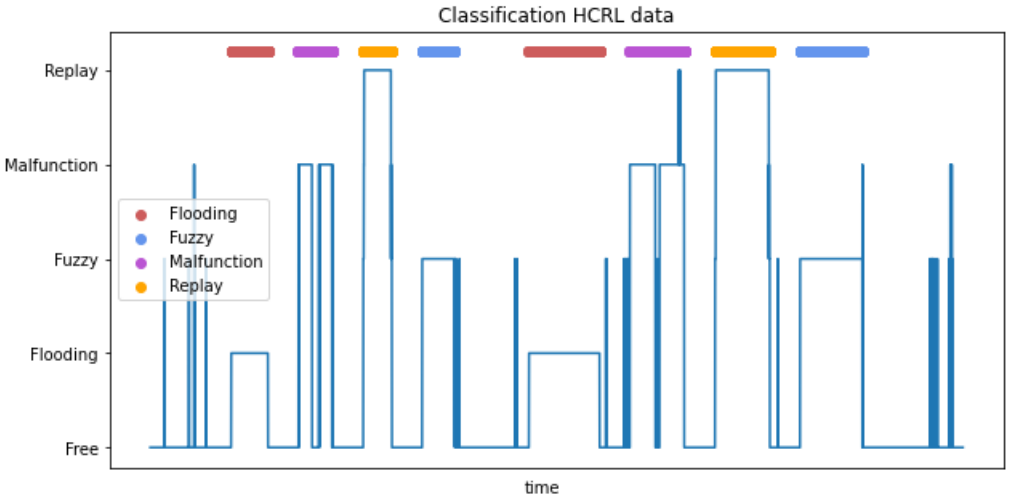

Figura 2.39: Clasificación. Base Car-Hacking

Capítulo 3

Estudio de los p-valores

Si recordamos lo comentado en la Sección 2.1.1 acerca del nivel de significación: *la teoría de la probabilidad nos dice que, bajo H_0, los p-valores se distribuyen como una $\mathcal{U}(0,1)$. Así, fijado α se espera una tasa de error del tipo I del $100\alpha\,\%$, o lo que es lo mismo, una especificidad de $1-\alpha$. Por otro lado, cuanto mayor sea α más fácil se rechazará H_0 y mayor será la sensibilidad. Así, el parámetro α mide la tensión entre ambas métricas.* Sin embargo, esto contradice los resultados obtenidos en este trabajo, y es que en ninguno de los algoritmos las especificidades se corresponden con los niveles de significación fijados. Esto ha jugado mayoritariamente a nuestro favor, ya que nos ha permitido obtener simultáneamente muy buenas sensibilidades y especificidades. No obstante, nos toca ser críticos y estudiar los motivos de este fenómeno.

Como ejemplo, en la Figura 3.1 se muestran las funciones de distribución empíricas de los p-valores resultantes de comparar las bases libres con los paquetes libres de las muestras de test de la base Survival, para cada ataque de cada vehículo, utilizando la combinación de parámetros óptima en cada caso. Estos p-valores, junto al nivel de significación fijado para cada vehículo, dan lugar a las especificidades mostradas en el Tabla 2.18. Los p-valores claramente no siguen una distribución $\mathcal{U}(0,1)$, sino que se concentran para valores superiores a 0.4. Destacan los p-valores obtenidos para el Sonata, todos muy próximos a 1, lo que explica las especificidades obtenidas para dicho vehículo. Como ya hemos anticipado, existen dos explicaciones por las que los p-valores no sigan una distribución $\mathcal{U}(0,1)$. Por un lado, el no cumplimiento de la hipótesis de independencia de los datos requerida en las pruebas log-rank y de Gehan. Por otro lado, con respecto al test de Gehan, que las medias sean prácticamente iguales y nunca halle diferencias significativas.

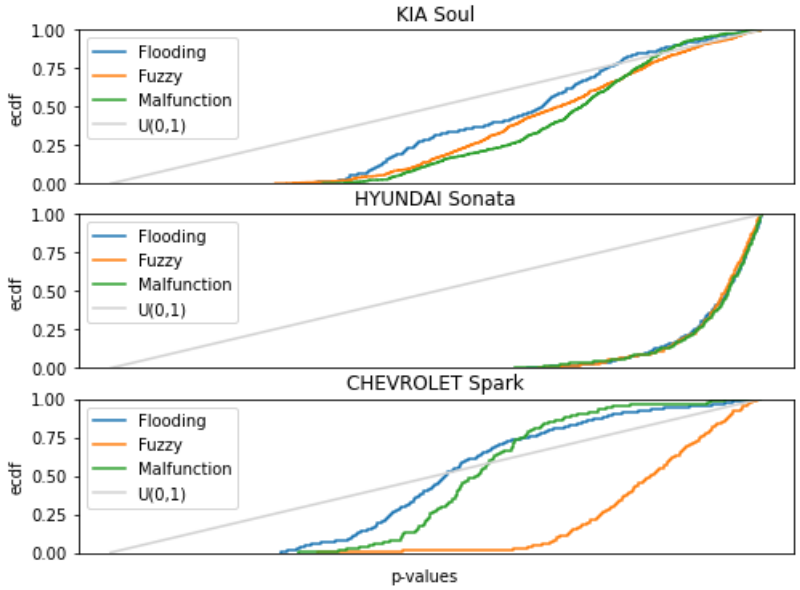

Figura 3.1: Funciones de distribución empíricas de los p-valores para cada vehículo y cada ataque para las muestras de test de la base Survival utilizando las combinaciones de parámetros óptimas individuales. En gris la función de distribución de una $\mathcal{U}(0,1)$.

3.1. Correlación de los datos

Comenzamos abordando la primera posibilidad. Como ya hemos comentado (ver Figura 2.25), la evolución temporal de la estructura de los mensajes nos indica que estos no han sido generados de forma individual e independiente, sino que forman parte de lotes de cierto número de mensajes que se generan periódicamente. Esto probablemente tenga como consecuencia que los tiempos entre mensajes legítimos presenten una fuerte correlación, anulando la hipótesis de independencia requerida en los contrastes usuales, incluidos los utilizados en este trabajo.

Efectivamente, la correlación entre los tiempos queda totalmente evidenciada en las gráficas de la Figura 3.2, realizadas todas ellas para la base libre del KIA Soul tomando $x = 51$, valor óptimo para dicho vehículo. Arriba a la izquierda se muestran los primeros 100 valores de la variable Y^{51}. La serie de tiempos es altamente oscilante, con valores grandes seguidos por valores pequeños y viceversa. Esta correlación negativa se aprecia aún mejor en la gráfica de arriba a la derecha, donde se muestran los valores de Y^{51} frente a sí mismos con un lag de diferencia (es decir, y_1^{51} vs y_2^{51}, y_2^{51} vs y_3^{51}, ...). Si no hubiera correlación, la

nube de puntos no mostraría ningún patrón, y claramente se observa un patrón con tendencia negativa. Por último, en las gráficas de abajo se muestran las funciones de autocorrelaciones simples y de autocorrelaciones parciales. Vemos como las autocorrelaciones simples tienden a oscilar debido al carácter oscilatorio de la serie, y no desaparecen por muchos lags que consideremos. Por otro lado, las autocorrelaciones parciales, en las que se elimina el efecto de los tiempos intermedios, son mayoritariamente negativas. Esto se debe a la presencia periódica de picos que sobresalen del resto, como se puede ver en la gráfica superior izquierda.

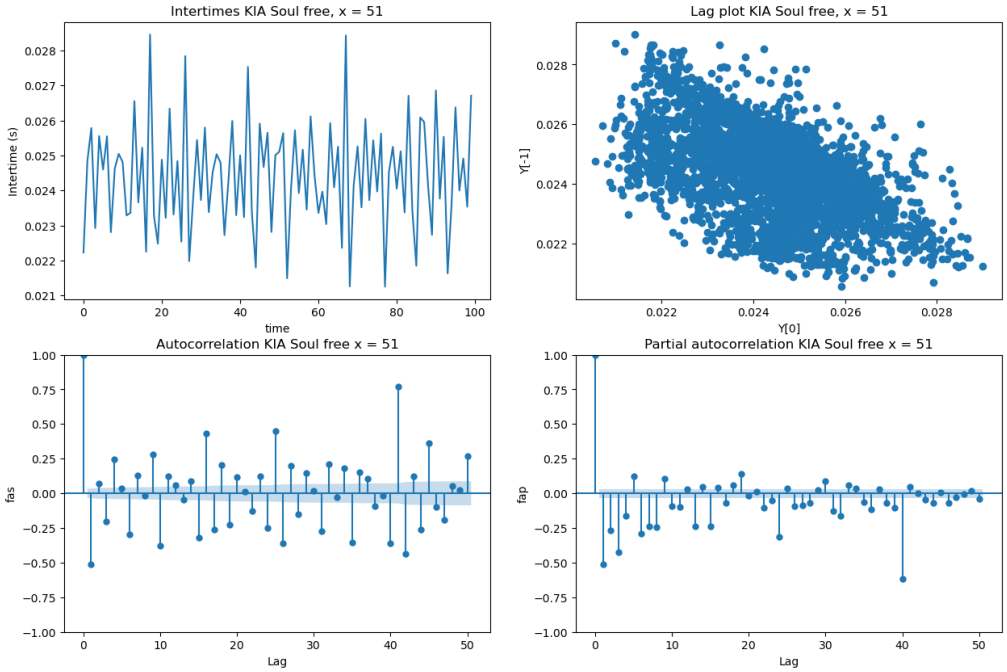

Figura 3.2: Para la base libre del KIA Soul y $x = 51$: arriba a la izquierda: 100 primeros valores de la variable Y^{51}. Arriba a la derecha: Y^{51} vs Y^{51} con un lag de diferencia. Abajo a la izquierda: función de autocorrelaciones simples. Abajo a la derecha: función de autocorrelaciones parciales.

La correlación existe en mayor o menor medida para todos x, y evoluciona de unos a otros de forma continua. En la Figura 3.3 se muestran algunos casos, como $x = 104$, para el cual la correlación negativa es aún más evidente, $x = 67$, para el cual ocurre lo contrario, o $x = 42$ y $x = 125$, para los cuales se forman figuras un tanto curiosas.

La correlación de los tiempos entre mensajes legítimos se da de igual manera

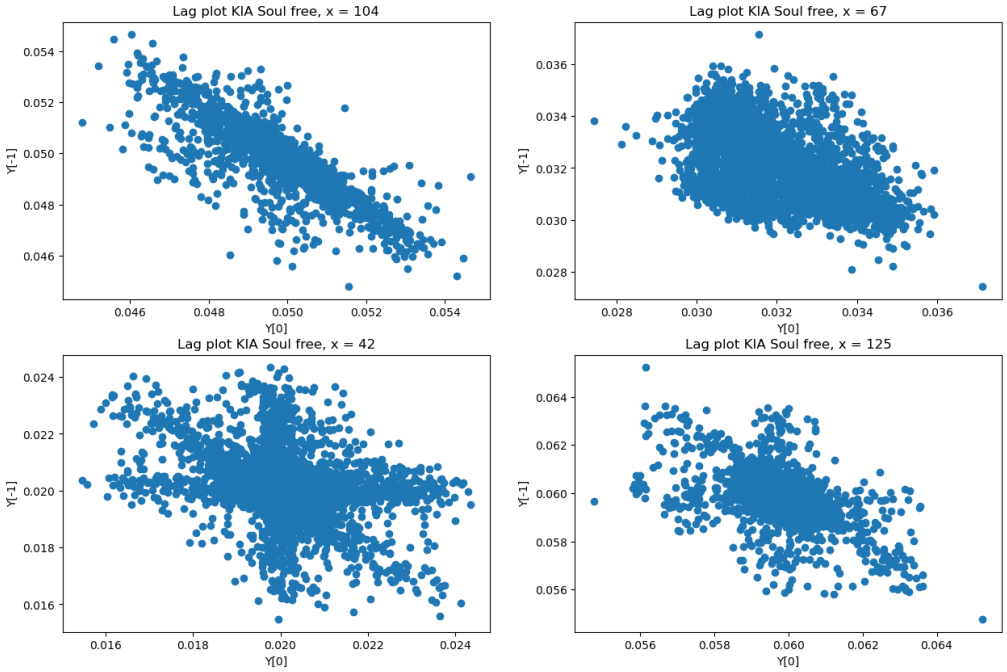

Figura 3.3: Para la base libre del KIA Soul y $x = 51$: Y^{51} vs Y^{51} con un lag de diferencia para $x = 104$, $x = 67$, $x = 42$ y $x = 125$.

para el resto de vehículos. Así, aunque el contraste de Gehan nos haya sido de gran utilidad para detectar y clasificar anomalías, el incumplimiento de la hipótesis de independencia de los datos hace que no resulte válido para sacar conclusiones estadísticamente significativas. Además, la conclusión general de que el Algoritmo 3 con Gehan lo haga bien pero que el control del error tipo I con α sea irrelevante desde el punto de vista de la significación estadística (incumplimiento de la hipótesis de independencia, especificidades alejadas de $1-\alpha$ sugiere que un algoritmo que utilice exclusivamente el tiempo medio entre mensajes podría ser ideal (buena discriminación, mínimo tiempo de ejecución).

3.2. Medias prácticamente idénticas

La correlación de los datos puede dar lugar a que los p-valores no representen fielmente la realidad. No obstante, esto sigue sin explicar al 100 % por qué se concentran en valores tan altos para el test de Gehan. Recordemos que el test de Gehan es de máxima potencia bajo diferencias de localización, por lo que está especializado en descubrir diferencias en las medias de las distribuciones.

Cuando las medias son muy parecidas pero las diferencias se dan en otros aspectos de la distribución, como su desviación típica o su forma, el test de Gehan puede dar lugar a resultados extraños.

Para ver esto, fijémonos en las Figuras 3.4, 3.5 y 3.6. En la primera de ellas, a la izquierda, se representa en azul la media de la variable Y^{49} para el Sonata ($x = 49$ fue el valor óptimo para este vehículo), en gris la media de las variables $Z^{49,n}$ para los paquetes libres del ataque *fuzzy*, y en naranja la media de las variables $Z^{49,n}$ más menos su desviación típica. Vemos como, aunque las desviaciones típicas fluctúan bastante debido a la evolución de la generación de mensajes, las medias se mantienen siempre muy cercanas al valor medio de la base libre. A la derecha se muestra la función de distribución empírica de los p-valores para estos contrastes, que como ya vimos antes se concentra entorno a 0.9. En la Figura 3.5 se representa lo mismo, pero en este caso comparando variables $\mathcal{N}(0, 1)$. Para ello simulamos una muestra de 100 000 valores de una $\mathcal{N}(0, 1)$ (*Normal1*) y 200 muestras de 500 valores de otra $\mathcal{N}(0, 1)$ (*Normal2*). Vemos como tanto las medias como las desviaciones típicas oscilan de forma similar. A la derecha se muestra la función de distribución empírica de los p-valores mediante el test de Gehan al comparar cada una de las muestras de la *Normal2* con la muestra de la *Normal1*. Como era de esperar, sigue una distribución $\mathcal{U}(0, 1)$. Por último, en la Figura 3.6 realizamos lo mismo pero tomando una $\mathcal{N}(0, 0.1)$ como *Normal2*, de manera que sus medias están mucho más concentradas sobre la media de la Normal1. Efectivamente, los p-valores dados por el test de Gehan en este caso no siguen una $\mathcal{U}(0, 1)$, sino que se asemejan mucho a los del ataque *fuzzy* del Sonata.

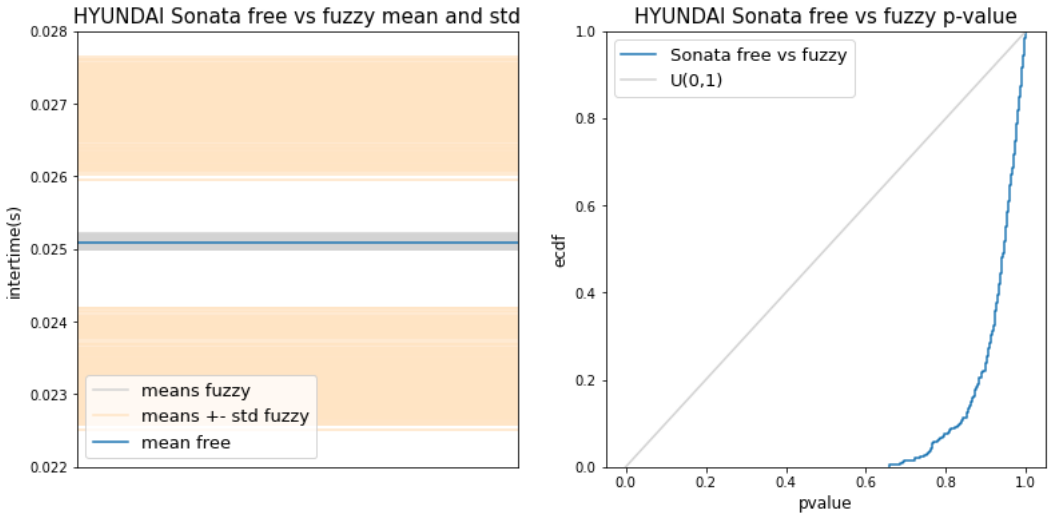

Figura 3.4: A la izquierda, en azul, la media muestral de la variable Y^{51} del Sonata; en gris, la media muestral de las variables Z^{51} para el ataque *fuzzy* del Sonata; en naranja, la media muestral más menos la desviación típica de las variables Z^{51} para el ataque *fuzzy* del Sonata. A la derecha, la función de distribución empírica de los p-valores al comparar la base libre del Sonata con los paquetes libres de la parte de test del ataque *fuzzy* del Sonata utilizando el test de Gehan; en gris, la función de distribución de una $\mathcal{U}(0,1)$.

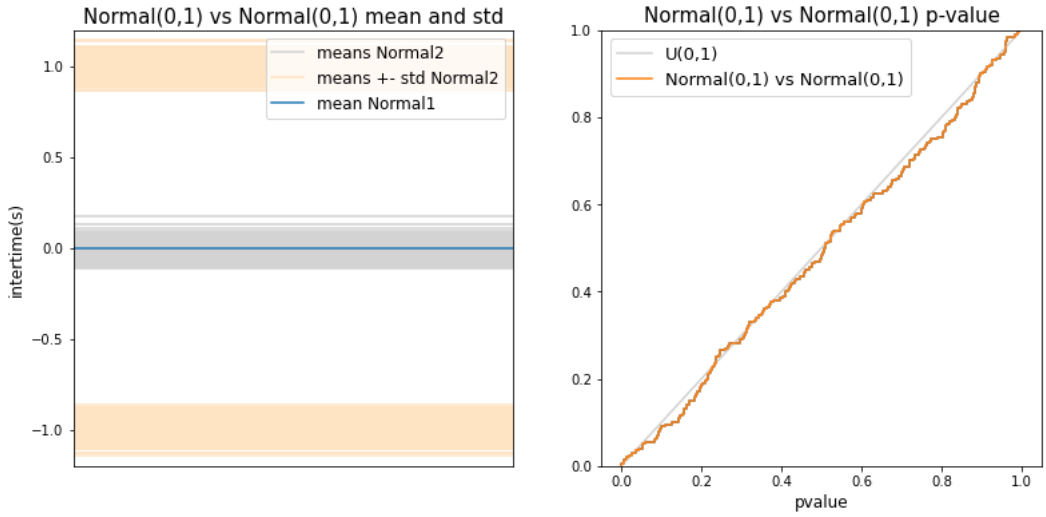

Figura 3.5: A la izquierda, en azul, la media muestral de una variable $\mathcal{N}(0,1)$; en gris, la media muestral de 200 muestras de otra variable $\mathcal{N}(0,1)$; en naranja, la media muestral más menos la desviación típica de estas muestras. A la derecha, la función de distribución empírica de los p-valores al comparar la muestra de la primera normal con las muestras de la segunda normal utilizando el test de Gehan; en gris, la función de distribución de una $\mathcal{U}(0,1)$.

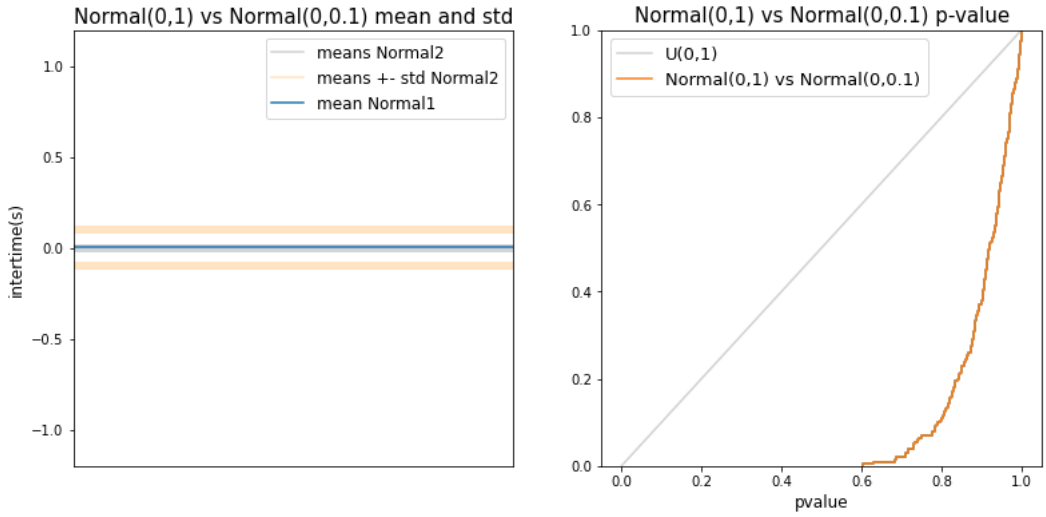

Figura 3.6: A la izquierda, en azul, la media muestral de una variable $\mathcal{N}(0,1)$; en gris, la media muestral de 200 muestras de una variable $\mathcal{N}(0,0.1)$; en naranja, la media muestral más menos la desviación típica de estas muestras. A la derecha, en azul, la función de distribución empírica de los p-valores al comparar la muestra de la primera normal con las muestras de la segunda normal utilizando el test de Gehan; en gris, la función de distribución de una $\mathcal{U}(0,1)$.

Capítulo 4

Conclusiones y trabajo futuro

A lo largo de este trabajo hemos profundizado en la detección y clasificación de anomalías en vehículos utilizando técnicas de Análisis de Supervivencia. En el Capítulo 1 hemos introducido la red interna vehicular, con sus ventajas, vulnerabilidades y posibles ataques asociados conocidos hoy en día. Nos hemos centrado en la detección de ataques basados en la inyección de mensajes, concretamente en los ataques *flooding*, *fuzzy*, *malfunction* y *replay*, para lo cual en el Capítulo 2 hemos desarrollado varios algoritmos. Los dos primeros, presentados en la Sección 2.1, se basan en la comparación de tiempos entre mensajes de cada ID. A pesar de que existe literatura que, basándose en esta variable, detecta con precisión anomalías utilizando algoritmos de Machine Learning, desde nuestro enfoque hemos visto que el tiempo entre mensajes para cada ID no diferencia correctamente entre una conducción libre y una conducción atacada. Por dicha razón, en la Sección 2.2 hemos descartado la consideración del tiempo entre mensajes diferenciando su ID para basarnos simplemente en el tiempo entre mensajes. Esto no solo ha disminuido los tiempos de ejecución, sino que también nos ha permitido desarrollar algoritmos que detectan con alta precisión los ataques considerados con una tasa muy baja de falsas alertas. No obstante, haber obtenido especificidades que no se correspondieran con los niveles de significación nos ha hecho adentrarnos en los datos para buscar una posible explicación. Hemos visto que el proceso de generación de mensajes libres evoluciona con el tiempo, lo que explica las malas especificidades del test log-rank y nos hace intuir una posible correlación entre los datos, comprobada en la Sección 3.1. Esta correlación hace que se incumpla la hipótesis de independencia de los contrastes utilizados, haciendo que sus resultados no representen fielmente la realidad. En la Sección 3.2 hemos visto que a pesar de la evolución en la generación de los datos, la media de los tiempos entre mensajes permanece

prácticamente invariante, lo que explica que se obtengan p-valores tan altos bajo la hipótesis nula con el test de Gehan. En conclusión, aunque la correlación entre los datos hace que no se verifiquen las hipótesis requeridas en los contrastes, haberlos considerado como discriminante entre una conducción libre y una atacada no solo ha aportado un buen algoritmo de detección y clasificación de anomalías, sino que también nos ha hecho adentrarnos en su estructura y nos ha permitido descubrir su peculiar modo de generación de mensajes.

Dicho todo esto, existen varios aspectos que sería interesante considerar en el futuro. Por un lado, que la media entre mensajes permanezca prácticamente idéntica sugiere que un método de clasificación que se base exclusivamente en el tiempo medio podría ser ideal, obteniendo tan buenos resultados como los de este trabajo y disminuyendo los tiempos de ejecución al no utilizar contrastes de hipótesis. Por otro lado, la correlación entre los datos también sugiere la utilización de técnicas de Series de Tiempo que acepten esta correlación. Por último, sería interesante la aplicación de estos algoritmos para la detección de anomalías en otros ámbitos en los que la comunicación entre dispositivos también sea en forma de mensajes, por ejemplo, los entornos industriales.

Bibliografía

[1] Crypto World. Smart Car Hacking. (22 de enero de 2013). Accedido el 18 de marzo de 2024. [Video en línea]. Disponible: https://www.youtube.com/watch?v=katVec_SwA4.

[2] J. Torchinsky. "Hackers steal BMWs in 3 minutes using security loophole". NBC News. Accedido el 18 de marzo de 2024. [En línea]. Disponible: https://www.nbcnews.com/tech/tech-news/hackers-steal-bmws-3-minutes-using-security-loophole-flna868400.

[3] Black Hat — Home. Accedido el 18 de marzo de 2024. [En línea]. Disponible: https://www.blackhat.com/docs/us-17/thursday/us-17-Nie-Free-Fall-Hacking-Tesla-From-Wireless-To-CAN-Bus-wp.pdf.

[4] Khan, J., Lim, D. W., & Kim, Y. S. (2023). Intrusion Detection System CAN-Bus In-Vehicle Networks Based on the Statistical Characteristics of Attacks. Sensors, 23(7), 3554.

[5] Kuwahara, T., Baba, Y., Kashima, H., Kishikawa, T., Tsurumi, J., Haga, T., ... & Matsushima, H. (2018). Supervised and unsupervised intrusion detection based on CAN message frequencies for in-vehicle network. Journal of Information Processing, 26, 306-313.

[6] Rajapaksha, S., Kalutarage, H., Al-Kadri, M. O., Petrovski, A., Madzudzo, G., & Cheah, M. (2023). Ai-based intrusion detection systems for in-vehicle networks: A survey. ACM Computing Surveys, 55(11), 1-40.

[7] Seo, E., Song, H. M., & Kim, H. K. (2018, August). GIDS: GAN based intrusion detection system for in-vehicle network. In 2018 16th Annual Conference on Privacy, Security and Trust (PST) (pp. 1-6). IEEE.

[8] Song, H. M., Woo, J., & Kim, H. K. (2020). In-vehicle network intrusion detection using deep convolutional neural network. Vehicular Communications, 21, 100198.

[9] Desta, A. K., Ohira, S., Arai, I., & Fujikawa, K. (2020, March). ID sequence analysis for intrusion detection in the CAN bus using long short term memory networks. In 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops) (pp. 1-6). IEEE.

[10] Jedh, M., Othmane, L. B., Ahmed, N., & Bhargava, B. (2021). Detection of message injection attacks onto the can bus using similarities of successive messages-sequence graphs. IEEE Transactions on Information Forensics and Security, 16, 4133-4146.

[11] Kalutarage, H. K., Al-Kadri, M. O., Cheah, M., & Madzudzo, G. (2019, October). Context-aware anomaly detector for monitoring cyber attacks on automotive CAN bus. In Proceedings of the 3rd ACM Computer Science in Cars Symposium (pp. 1-8).

[12] Avatefipour, O., Al-Sumaiti, A. S., El-Sherbeeny, A. M., Awwad, E. M., Elmeligy, M. A., Mohamed, M. A., & Malik, H. (2019). An intelligent secured framework for cyberattack detection in electric vehicles' CAN bus using machine learning. IEEE Access, 7, 127580-127592.

[13] Taylor, A., Leblanc, S., & Japkowicz, N. (2016, October). Anomaly detection in automobile control network data with long short-term memory networks. In 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA) (pp. 130-139). IEEE.

[14] Wang, Y., Chia, D. W. M., & Ha, Y. (2020, August). Vulnerability of deep learning model based anomaly detection in vehicle network. In 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS) (pp. 293-296). IEEE.

[15] Novikova, E., Le, V., Yutin, M., Weber, M., & Anderson, C. (2020). Autoencoder anomaly detection on large CAN bus data. Proceedings of DLP-KDD.

[16] Narasimhan, H., Ravi, V., & Mohammad, N. (2021). Unsupervised deep learning approach for in-vehicle intrusion detection system. IEEE Consumer Electronics Magazine, 12(1), 103-108.

[17] Kukkala, V. K., Thiruloga, S. V., & Pasricha, S. (2020). INDRA: Intrusion detection using recurrent autoencoders in automotive embedded systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 39(11), 3698-3710.

[18] Kukkala, V. K., Thiruloga, S. V., & Pasricha, S. (2021). Latte: L stm self-att ention based anomaly detection in e mbedded automotive platforms. ACM Transactions on Embedded Computing Systems (TECS), 20(5s), 1-23.

[19] Chockalingam, V., Larson, I., Lin, D., & Nofzinger, S. (2016). Detecting attacks on the CAN protocol with machine learning. Annu EECS, 558(7).

[20] Martinelli, F., Mercaldo, F., Nardone, V.,& Santone, A. (2017, July). Car hacking identification through fuzzy logic algorithms. In 2017 IEEE international conference on fuzzy systems (FUZZ-IEEE) (pp. 1-7). IEEE.

[21] Berger, I., Rieke, R., Kolomeets, M., Chechulin, A., & Kotenko, I. (2018, September). Comparative study of machine learning methods for in-vehicle intrusion detection. In International Workshop on Security and Privacy Requirements Engineering (pp. 85-101). Cham: Springer International Publishing.

[22] Kalkan, S. C., & Sahingoz, O. K. (2020, July). In-vehicle intrusion detection system on controller area network with machine learning models. In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

[23] Han, M. L., Kwak, B. I., & Kim, H. K. (2018). Anomaly intrusion detection method for vehicular networks based on survival analysis. Vehicular Communications, 14, 52–63.

[24] J. P. Klein y y M.L. Moeschberger, "Survival analysis: Techniques for censored and truncated data", 2a ed. *New York: Springer*, 2003.

[25] Pepe, M. S. The Statistical Evaluation of Medical Tests for Classification and Prediction. Oxford University Press, 2003.

[26] Hacking and Countermeasure Research Lab. 2020. Survival Analysis Dataset for Automobile IDS. Retrieved August 1, 2021 from https://ocslab.hksecurity.net/Datasets/survival-ids.

[27] Hacking and Countermeasure Research Lab. 2020. Car-Hacking Dataset for the Intrusion Detection. Retrieved August 1, 2021 from https://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset.