UNIVERSIDAD CARLOS III DE MADRID

Máster en Ciencias Actuariales y Financieras

TRABAJO DE FIN DE MÁSTER

Uso de datos sintéticos provenientes de redes neuronales para la mejora de la modelización de la severidad de eventos infrecuentes

Alumno:
Luis Adrián Valdivia Ameller
100396048

Septiembre de 2020

ÍNDICE

ÍNDICE DE FIGURAS

A ellos, que me acompañaron
en la distancia.

1. INTRODUCCIÓN

1.1. BREVE DESCRIPCIÓN

El avance de la tecnología y su implementación en más esferas de la vida del ser humano contribuye a la disminución de eventos inesperados y/o limita la potencial pérdida que se pudiera experimentar a partir de ellos. No obstante, el mundo es una compleja estructura en el que la aleatoriedad juega un rol importante en el curso de la historia y constituye un componente inherente a la vida del ser humano. Si bien la tecnología puede ayudar en la mitigación del riesgo, no es capaz de eliminarlo, de lo contrario no sería un riesgo y no habría una pérdida económica asociada a él. En este sentido, se espera que en el futuro los siniestros sean menos frecuentes pero que, en contrapartida, sean eventos que acarreen elevadas sumas de dinero en concepto de indemnización para las empresas aseguradoras. El presente trabajo explora dos técnicas de modelización de la severidad de eventos infrecuentes, proponiendo como una de ellas la teoría de Valores Extremos (EVT) y por otro lado las redes generativas adversarias (GAN), propias del campo del machine learning. La composición de este documento comprende cinco apartados. En el primero se explora desde una perspectiva literaria las referidas metodologías en lo relacionado a sus características, aplicaciones y limitaciones. En el segundo se examina la metodología cualitativa que subyace detrás de ambas técnicas de modelización. En la tercera parte se cuenta con una descripción de los datos. El apartado cuatro contiene los resultados y, finalmente, en el apartado cinco se encuentran las conclusiones y recomendaciones.

1.2. RESUMEN DE RESULTADOS

Si bien la teoría de Valores Extremos es una metodología más fácil de aplicar, los resultados no son muy precisos, pues la misma falla al capturar la observación más extrema en la base de datos. Por otro lado, la red neuronal, que requiere de conocimiento especializado y una estructura computacional más potente, aprende notablemente la distribución real de los datos. No obstante, se limita a replicarla y es incapaz de generar datos fuera de ella. En consecuencia, los mejores resultados se obtienen al combinar ambas metodologías. La modelización mediante la EVT con la adición de los datos sintéticos generados por la red GAN, permite la selección de un umbral más alto y, subsecuentemente, proveer un ajuste incluso para las observaciones más extremas.

1.3. CONTRIBUCIONES DEL TRABAJO

Por un lado, la demostración de la aplicabilidad de redes neuronales en temas actuariales que podrían implementarse en el sector asegurador. Por otro lado, la constatación de que las técnicas más modernas pueden combinarse con otras más antiguas para generar mejores resultados.

1.4. MOTIVACIÓN

La implementación de una red neuronal para la modelización de la severidad en aquellos casos en el que se cuenta con información insuficiente y su combinación con técnicas estadísticas tradicionales es una propuesta nueva de la que no se tiene conocimiento que se haya realizado con anterioridad. Además, supone una metodología que podría aplicarse por aquellas compañías de seguro que incursionen en nuevas líneas de negocio o en nuevos mercados. En este sentido, supone una idea novedosa en la que se fusionan nociones actuariales con otras propias de la informática. Lo que implica, a nivel personal, el aprendizaje de un nuevo lenguaje de programación para la implementación de redes neuronales y, por lo tanto, un desafío.

1.5. OBJETIVOS

Objetivo general:

Determinar la efectividad del uso de una red neuronal generativa adversaria - GAN (por sus siglas en inglés) para la modelización de la severidad de eventos poco frecuentes, pero elevada cuantía, y la generación de datos sintéticos a partir de la misma.

Objetivos específicos:

- Explorar técnicas estadísticas tradicionales en la modelización de datos extremos y desarrollar una de ellas para compararla con la técnica de machine learning.
- Investigar la flexibilidad de la red neuronal frente a otras distribuciones teóricas cuando se ajustan a la totalidad de los datos sin una previa distinción entre siniestros masa y siniestros punta.
- Analizar si existe una mejora en el ajuste de los valores extremos cuando se complementan la teoría de valores extremos y las redes neuronales al incluir los datos sintéticos generados por este último.

2. REVISIÓN LITERARIA

Actualmente, uno de los desafíos que se plantea al mercado asegurador es la modelización de eventos infrecuentes, pues, el Internet de las Cosas (Internet of Things – IoT) y el avance de la tecnología en el procesamiento y análisis de datos contribuyen a la prevención de riesgos en sectores tan diversos como la manufactura (Yang et al., 2019) o la agricultura, mediante el seguimiento del estado de la tierra y, consecuentemente, el estado de los cultivos (Ayaz et al., 2019), o en mercados más convencionales para las aseguradoras como son los seguros de hogar a través de los hogares inteligentes que son capaces de advertir a tiempo un posible problema, previendo siniestros o disminuyendo las pérdidas de los mismos (Francis et al., 2015; Serrenho & Bertoldi, 2019; Sassani et al., 2020); como también en el seguro de vehículos automotor con la incursión del campo de la telemática y el seguimiento del comportamiento del conductor en el volante (Ayuso et al., 2017; Valandro, 2019), por lo que los siniestros punta ya no sólo se asociarán con los fenómenos físicos y las catástrofes naturales que ellos producen, sino que se amplía la generalización para todos aquellos siniestros que se caracterizan por tener una baja frecuencia pero alta severidad en caso de acaecimiento.

A diferencia de lo que sucede en una industria convencional, el precio de un producto de seguro no se lo determina al finalizar la cadena de producción, cuando se conocen todos los gastos en los que se incurrieron, sino al comienzo, incluso, antes de que se conozca el costo del mismo. Esto es lo que conoce como el ciclo inverso del seguro.

En este sentido, la determinación del precio de un producto de riesgo es un proceso harto complejo. Tradicionalmente, la asignación de un precio a un contrato de seguro consiste en el cálculo de la siniestralidad esperada y los costos asociados a ella, en base a una adecuada identificación del riesgo y la disponibilidad de una importante cantidad de registros históricos para el cálculo de la frecuencia y severidad esperadas (Outreville, 1998).

Considerando la importancia de los datos, incluso antes de iniciar el modelaje estadístico (Embrechets et al., 1996), surge, indudablemente, un problema evidente en la elaboración de un nuevo producto o en el caso de la modelización de sucesos atípicos, cuando la información es escasa o en el peor de los casos inexistente. Si bien en la práctica existen formas de sobrellevar esta dificultad, no se tratan de soluciones definitivas. En el primer caso, se pueden elaborar hipótesis a partir de productos similares, existiendo, pues, la

posibilidad de que las suposiciones resulten equivocadas y se establezca una tarifa que discrepa con la naturaleza del riesgo, resultando en tarifas elevadas poco competitivas en el mercado o tarifas muy bajas que ponen en peligro la solvencia de la compañía de seguros, como le sucedió a Penn Treaty Network America Insurance Company en el año 2009 con los seguros Long-Term Care (Mohey-Deen & Rosen, 2018). Por otro lado, en el caso de los denominados siniestros punta, se pueden aplicar técnicas de extrapolación, las cuales se basan fundamentalmente en la teoría de los eventos extremos (EVT por sus siglas en inglés). Sin embargo, una de las principales críticas que enfrenta esta metodología es que busca, a partir de un conjunto de registros, deducir valores nunca antes observados (Coles, 2001). Por ejemplo, la industria aseguradora americana en la década de los noventa no contemplaba la posibilidad de que en el año 1994 alcanzaría un ratio de siniestralidad (loss ratio) de 2.272,7% tras un terremoto de 6,6 en la escala de Richter en California, cuando en los últimos 20 años previos el loss ratio más alto del que tenían registro era tan sólo de 129,8% (Embrechets et al., 1999).

La teoría de valores extremos, basado en el teorema Fisher-Tippet-Gnedenko[1] y Pickands, Balkema and de Haan, tiene por objeto la "predicción de la probabilidad de acaecimiento de eventos raros que apenas o nunca suceden, fuera de la información de la que se tiene disponible" (Charras-Garrido y Lezaud, 2013; traducción propia). En efecto, su utilidad reside en la modelización de eventos de poca probabilidad, tanto para valores máximos como mínimos (Fraga Alves y Neves, 2011). La teoría señala que existen dos enfoques distintos para modelar eventos inusuales, estos son *block máxima* (BM) y *peaks-over-threshold* (POT). El primero de ellos consiste en la división del período de estudio en bloques no superpuestos de igual longitud en los que se identifican, de manera respectiva, las observaciones máximas. En el segundo se entiende por observaciones extremas todas aquellas que sobrepasan un umbral previamente establecido. Nejc Bezak et al. (2014) verificaron que este último método funciona mejor cuando se tienen series relativamente más cortas, pues BM necesita series extensas en tanto que sólo toma en cuenta una o n observaciones máximas por bloque. A pesar de este inconveniente, éste sortea con mayor destreza problemas de dependencia entre observaciones que el método POT, en el que se debe verificar que se cumpla esta condición cuando se comienza a modelizar.

1 Gnedenko (1943) unifica y formaliza las ideas de Fisher y Tippet.

La aplicabilidad de la teoría de valores extremos abarca distintas ramas del conocimiento y se ha demostrado su aplicabilidad en distintas áreas. Minkah (2016) estudia la probabilidad de niveles de agua muy bajos o muy altos en represas hidroeléctricas. Elvidge y Angling (2018) deducen la probabilidad de erupciones solares extremas. Chen et al. (2015) estudian el brote de una epidemia de influenza en China. En el campo de las finanzas, Gilli y Këllezi (2006) cuantifican el riesgo de mercado empleando un análisis de valores extremos (EVA, por sus siglas en inglés); Gençay y Selçuk (2002) concluyen que la referida metodología es la mejor alternativa para modelar sucesos de colas pesadas en economías altamente volátiles como las emergentes. En el ramo asegurador, la teoría de valores extremos prueba ser una herramienta útil para la modelización de siniestros de incendios industriales (Beirlant et al., 2001; McNeil y Saladin, 1997) o siniestros punta en la línea de vehículos automotor (Adesina et al., 2008). Sin duda, la literatura es extensa y abarca, con mayor o menor detalle, variados casos en los que se puede aplicar esta metodología.

Algunas de las críticas de la que es objeto esta teoría, además de la que señala Coles (2001) relacionada a la insuficiencia de observaciones extremas, recaen en la dificultad de la comprobación de las suposiciones que se realizan en la caracterización de las colas de las distribuciones o en la complejidad de su estimación en carteras de muchos activos cuando se desea utilizarla como herramienta de gestión del riesgo (Embrechets, 2000). Makkonen (2008) identifica, además, que la aplicación de esta técnica se la realiza aún a sabiendas que el ajuste no es el óptimo debido a la esencia asintótica que subyace a la teoría. Asimismo, McNeil (1997) demuestra que la incertidumbre es inherente a la calibración de los parámetros en el ajuste de estos modelos y que, independientemente a la calidad de la base de datos, es un factor que se debe tener en cuenta.

Al margen de contrariedades, McNeil (1997) señala que cuando se tienen pocos datos, como es habitual en líneas de negocio comerciales o industriales, especialmente en aquellos casos que cuentan con un deducible elevado (Fackler , s.f.), la alternativa a no poder tarificar y por lo tanto no suscribir un seguro por falta de una mayor experiencia en el negocio, es preferible utilizar la ETV, demostrando que es capaz de ajustar las colas de los siniestros de incendios en Dinamarca teniendo únicamente 36 valores extremos. Cai and Hames (2010) determinan que el tamaño mínimo al que puede ajustar una GEV sin dificultades es de 25 observaciones, aunque, dicho sea de paso, concluye que el tamaño mínimo óptimo depende de la naturaleza misma de cada conjunto de datos en concreto.

Makkonen y Tikanmäki (2019) proponen que se puede mejorar el ajuste que se realiza en el análisis EVA mediante la aplicación del método VWLS en la derivación de parámetros para muestras pequeñas.

Sin embargo, la EVT no es la única metodología que se puede adoptar, existen otros nuevos enfoques que buscan abordar la problemática con el uso de técnicas computaciones más intensivas. Por ejemplo, Ibn Musah et al. (2018) alcanzaron un nivel de precisión superior al noventa por cien llevando a cabo un estudio de valores extremos del índice de mercado de valores de Ghana al emplear redes neuronales. En este sentido, con los recursos del mundo actual se abren las posibilidades de la realización de estudios clásicos mediante técnicas innovadoras que presumen de mayor exactitud, aunque ésta sea a cambio de la parsimonia e interpretabilidad de los modelos tradicionales (Ciampi & Lechevallier, 2007).

Las redes neuronales artificiales o ANN (por sus siglas en inglés) son herramientas que, al igual que sus pares biológicos, capturan el comportamiento complejo de la realidad y las distintas interacciones que subyacen (Ibn Musah, 2018). Formalmente, Bsheer y Hajmerr (2000) las definen como "estructuras compuestas por elementos de procesamiento interconectados y adaptativos capaces de realizar cálculos paralelos masivos para el procesamiento de datos y representación del conocimiento." (traducción propia).

El elemento más sencillo de una ANN se denomina perceptrón, el cual al igual que una neurona del cerebro animal, recibe un estímulo y emite una señal si el primero es lo suficientemente fuerte. En otras palabras, un perceptrón recibe un input x_i con una determinada fuerza o peso w_i , a cuya combinación se aplica una función de activación y se evalúa si supera un umbral, si lo hace la neurona se activa y se trasmite el output y_i. Normalmente, una red se compone de varios cientos de estas unidades fundamentales, por lo que una ANN puede entenderse como una asociación de perceptrones, aglomeradas en distintas capas (Kohli et al. 2014). En la literatura a estas últimas se las conocen como Multilayer Perceptron o MLP (Gallo, 2015).

Las ANN poseen una naturaleza altamente adaptativa, la cual puede ser concebida como el proceso por el cual una red de neuronas cambia su estructura, adaptando los pesos, umbrales y conexiones en función a los estímulos que reciba. Es decir, las observaciones

que se utilizan durante el entrenamiento juegan un rol crucial en la determinación de la estructura final.

El entrenamiento de una red, de acuerdo a Kumar & Sharma (2014), es el proceso análogo estadístico por el cual se calibran los parámetros del modelo. Para ello, se utilizan algoritmos o reglas de aprendizaje como la denominada retro propagación (backpropagation), que sea dicho de paso es la norma en la gran mayoría de los casos.

Las ANN, que se incluyen dentro de la familia de la inteligencia artificial junto al Fuzzy logic, Expert Systems y Support Vector Machines, se dividen de manera general en modelos de aprendizaje supervisado o no supervisado (Kukreja et al., 2016). No obstante, esta clasificación puede ser más rigurosa si se toma en cuenta la función que cumplen, el grado de conectividad de las neuronas, la dirección del flujo de la información, el tipo del algoritmo de aprendizaje, la regla de aprendizaje y el grado de supervisión en el proceso de aprendizaje (Basheer y Hajmerr, 2000).

Estos sistemas, propios de la inteligencia artificial (AI, por sus siglas en inglés), no son un invento nuevo, sino que datan de la primera mitad del siglo XX cuando Warren McCulloch y Walter Pitts, nurofisiólogo y matemático respectivamente, sientan las bases de esta especialidad. Ahora bien, es gracias a la potencia de los ordenadores actuales y la cada vez mayor generación de datos (SINTEF, 2013; LeCun et al., 2015) que estas estructuras computacionales empiezan a atraer la atención de investigadores y científicos de diversas disciplinas como la biotecnología, medicina, finanzas (Collins et al., 1988; Wong et al., 1997) y seguros, entre otros.

En el ámbito asegurador, Vieria y Rodriguez Zaurin (2019) proponen la elaboración de modelos de riesgo a través de la generación datos sintéticos[2] a partir de modelos GANs (Generative Adversarial Networks) en aquellos casos en los que las observaciones sean insuficientes. También, Kevin Kuo (2019) combina las ciencias actuariales y la inteligencia artificial empleando modelos CTGAN a dos bases de datos públicas que contienen información de siniestros por daños a terceros y seguros de vida, respectivamente.

2 Se entiende por datos sintéticos aquellos valores que no se obtienen producto de la observación, sino que son resultado de un modelo. Las ventajas de utilizar este tipo de información es que resuelve problemas relacionados con la disponibilidad y promueve un espacio seguro de exploración para el investigador, en el que se respetan las normas de privacidad de la información y evita cualquier riesgo de filtración de datos sensibles (Xu y Veeramachaneni, 2018; Park et al. 2018).

Las GANs son modelos de aprendizaje profundo[3] con una estructura híbrida (Xu et al., 2015), que están compuestas por dos modelos que se entrenan de manera simultánea de manera no supervisada, uno generativo G y otro discriminativo D. El objetivo de G es determinar la distribución de la variable objetivo, proceso en el que D contribuye examinando los datos generados por el primero y comparándolos con la información real (Wang, 2017). En otras palabras, se trata de un proceso en el que se confrontan dos modelos, uno que trata de generar data falsa y otro que adopta un rol verificador procurando no ser engañado. G intenta capturar la distribución de la que proviene la información real, que le es dada como input a través del set de entrenamiento, y D estima la probabilidad de que los datos generados por G provengan del set original (Goodfellow et al. 2014).

Desde su concepción en el año 2014, las redes GAN han sido sujeto de modificaciones y/o mejoras que han permitido adaptarlas a las necesidades del proyecto en el que se las emplee, verbigracia, el ya referido CTGAN. Otra alternativa es el WGAN, Wang (2017) señala que es más eficiente al valerse de la métrica de Wasserstein en lugar de la entropía relativa. En el caso de la generación de datos tabulados, Xu y Veeramachaneni (2018) proponen un modelo denominado TGAN (tabular GAN) para la generación de tablas relacionales con datos continuos y discretos. De hecho, demostraron que el TGAN es superior en términos de desempeño que las redes bayesianas o las cópulas en la generación de datos sintéticos. Park et al. (2018) desarrollan el Table-GAN que incorpora una tercera red neuronal que se encarga de clasificar y mantener la "integridad semántica", es decir un nivel adicional que, aunque aumenta la complejidad del mismo, ayuda a mantener la congruencia de los nuevos datos.

Al igual que el resto de redes neuronales los principales inconvenientes de ésta se deben a la exigencia de tiempo en el proceso de entrenamiento, el cual conlleva un proceso de

3 La siguiente definición, extraída de Goodfellow et al. (2016, p. 8), sintetiza lo que se entiende por aprendizaje profundo (deep learning):

> *"Es un enfoque hacia la AI. Específicamente, es un tipo de machine learning; una técnica que permite a los sistemas computacionales mejorar a través de la experiencia y los datos. Afirmamos que el machine learning es la única solución viable para la construcción de sistemas de inteligencia artificial que puedan operar en situaciones complejas del mundo real. El aprendizaje profundo es un tipo particular del machine learning que logra con gran destreza y flexibilidad representar al mundo como una jerarquía anidada de conceptos, en el que cada uno es definido en relación a otros más sencillos, y mediante representaciones abstractas computadas en términos de otras menos abstractas."* (traducción propia).

prueba y error (Ciampi y Lechevallier, 2007). Además, su funcionamiento es usualmente referido como una caja negra.

3. METODOLOGÍA

Basheer y Hajmerr (2000) sugieren un aumento progresivo en la complejidad de las técnicas a utilizar, siendo aquellas más sencillas las que deberían evaluarse primero. En consecuencia, en la primera sección se realiza un estudio estadístico haciendo uso de la teoría de valores extremos (EVT); mientras que, en la segunda sección se aprovecha la potencia de los ordenadores actuales para entrenar una red neuronal.

3.1. PRIMERA SECCIÓN: TEORÍA DE VALORES EXTREMOS (EVT)

La teoría de valores extremos proporciona un marco desde el cual se pueden estudiar los extremos de fenómenos aleatorios. Por antonomasia, se hace uso de aquellas observaciones que presenten los valores más altos para explicar el comportamiento de las colas de la distribución de las mismas, en otras palabras, se analiza el comportamiento límite de una muestra de datos extremos. La EVT comprende a su vez dos metodologías distintas para estudiar estos eventos: *Block máxima* (BM) y *Peaks-over-threshold* (POT).

3.1.1. MÉTODO BM

Se considera una muestra $X_1, \dots, X_n$ de n variables aleatorias que son independientes unas de las otras y tienen una misma distribución F. Siendo la distribución de M_n la función de interés, el valor máximo $M_n = X_{(n)}$ se observa a partir de la muestra ordenada $X_{(1)}, X_{(2)}, \dots, X_{(n)}$ donde $X_{(1)} \leq X_{(2)} \leq \cdots \leq X_{(n)}$, luego de dividir los datos en bloques de igual longitud. La función de distribución de M_n se obtiene como:

$$P(M_n \leq x) = P\left(X_{(1)} \leq x, \dots, X_{(n)} \leq x\right) = F^n(x)$$

Sin embargo, en la mayoría de los casos F^n no es conocida, por lo que EVT propone que si existen un conjunto de valores $\{b_n, n \geq 1\}$ y $\{a_n : a_n > 0,\ n \geq 1\}$ entonces:

$$\lim_{n\to\infty} P\left(\frac{M_n - b_n}{a_n} \leq x\right) = F^n(a_n x + b_n) \to G(x)$$

Para todos los valores $x \in \mathbb{R}$, la probabilidad del máximo normalizado converge en distribución a una función de distribución no degenerada G.

Si existen constantes de normalización $a_n > 0$, $b_n \in \mathbb{R}$ y una distribución G a la que converge el valor máximo normalizado, entonces G pertenece a una de las siguientes distribuciones de valores extremos:

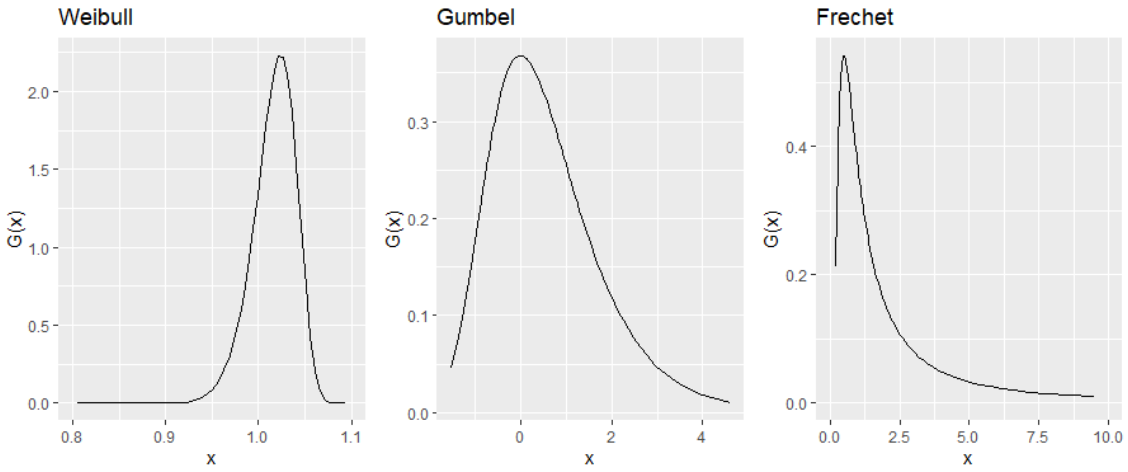
$$(i) \qquad G(x) = \exp\left(-exp\left(-(\frac{x-b}{a})\right)\right), \qquad x \in \mathbb{R}$$

$$(ii) \qquad G(x) = \begin{cases} 0, & x \leq b \\ \exp\left(-(\frac{x-b}{a})^{-\alpha}\right), & x > b \end{cases}$$

$$(iii) \qquad G(x) = \begin{cases} \exp\left(-(-(\frac{x-b}{a}))^{\alpha}\right), & x < b \\ 1, & x \geq b \end{cases}$$

Donde (i) Gumbel, (ii) Fréchet y (iii) Weibull. En las cuales, a y b representan los parámetros de escala y ubicación, respectivamente. Adicionalmente (I) y (III) cuentan con un parámetro de forma α. Esto supone que independientemente de la distribución de la población de X, la distribución límite de M_n, una vez normalizada, sólo puede adoptar una de las tres anteriores.

Figura 1. Función de densidad de las distribuciones de valores extremos



Fuente: Elaboración propia

Las distribuciones se caracterizan por el comportamiento de sus colas. La de la función de densidad de Gumbel decae exponencialmente, mientras que la de Fréchet lo hace polinomialmente. Weibull tiene un punto extremo finito.

La teoría BM se basa en la teoría de distribuciones Max-estable. Se dice que una distribución es Max-estable si existen sets $\{a_n > 0\}$ y $\{b_n \in \mathbb{R}\}$ tal que el máximo de una

secuencia de valores aleatorios sea igual en distribución a la de variable normalizada $\max(X_1, \dots, X_n) =^d a_n X + b_n$. La misma condición se puede expresar de manera que $F^n(x) = F\left(\frac{x-b_n}{a_n}\right)$ para todos los valores de $n \in \mathbb{N}$.

Estas tres familias de distribuciones, a las que se refiere el teorema de Fisher-Tipett-Gnedenko, pueden enmarcarse dentro de una única, denominada distribución extrema generalizada (GEV, por sus siglas en inglés).

Cuando a_n y b_n existen y se reexpresan las constantes de normalización se dice que $G(x)$ es miembro de la familia de la distribución GEV:

$$G(x) = G(x; \mu, \sigma, \xi) \coloneqq \exp\left(-\left(1 + \xi\left(\frac{x-\mu}{\sigma}\right)^{-\frac{1}{\xi}}\right), \qquad \forall x: 1 + \xi\left(\frac{x-\mu}{\sigma}\right) > 0$$

donde $\xi \in \mathbb{R}$ se denomina índice de valores extremos o tail index y determina el peso de la cola, $\mu \in \mathbb{R}$ es el parámetro de ubicación y $\sigma > 0$ el de escala. Si $\xi \to \infty$, entonces $G(x; \xi)$ se reduce a una distribución Gumbel para todos los valores de x. Por lo tanto, se dice que las tres distribuciones anteriores pertenecen al dominio de atracción $\mathcal{D}(G)$. De acuerdo a Minkah (2016) los parámetros pueden calibrarse mediante la estimación de los momentos ponderados, máxima verosimilitud o estimación bayesiana. Coles (2001), en contraste, recomienda el cálculo de los parámetros mediante el método de verosimilitud perfil (profile likelihood en inglés), el cual consiste de la estimación del intervalo de confianza de uno de los parámetros al fijar el mismo a un valor determinado (ej. $\xi = \xi_0$), maximizando la función de verosimilitud con los parámetros restantes.
La distribución inversa de la GEV que permite obtener cuantiles extremos es:

$$G^{-1}(x) = \begin{cases} \mu - \frac{\sigma}{\xi}\{1 - [-\log(1-p)]^{-\xi}\}, & si\ \xi \neq 0 \\ \mu - \sigma \log\left[-\log(1-p)\right], & si\ \xi = 0 \end{cases}$$

donde $p \in [0,1]$ es la probabilidad. Si se reemplaza $y_p = -\log(1-p)$, entonces:

$$G^{-1}(x) = \begin{cases} \mu - \frac{\sigma}{\xi}\{1 - (y_p)^{-\xi}\}, & si\ \xi \neq 0 \\ \mu - \sigma \log(y_p), & si\ \xi = 0 \end{cases}$$

3.1.1.1. MÉTODO MÁXIMA VEROSIMILITUD

La estimación de los parámetros (μ, σ, ξ) se puede obtener a través de la maximización de la siguiente función de verosimilitud:

$$L(\mu, \sigma, \xi) = -m\, log(\sigma) - \left(1 + \frac{1}{\xi}\right) \sum_{i=i}^{m} \log\left[1 + \xi\left(\frac{z_i - \mu}{\sigma}\right)\right] - \sum_{i=1}^{m} \left[1 + \xi\left(\frac{z_i - \mu}{\sigma}\right)\right]^{-\frac{1}{\xi}}$$

si $\xi \neq 0$, siempre y cuando se cumpla que:

$$1 + \xi\left(\frac{z_i - \mu}{\sigma}\right) > 0, i = 1, \dots, m$$

Si $\xi = 0$, entonces la función de verosimilitud se obtiene a partir de la distribución Gumbel:

$$L(\mu, \sigma) = -m \log(\sigma) - \sum\nolimits_{i=1}^{m} \left(\frac{z_i - \mu}{\sigma}\right) - \sum_{i=1}^{m} \exp\left[-\left(\frac{z_i - \mu}{\sigma}\right)\right]$$

En cualquier caso, z_i es el valor que toma Z_i dada una serie $Z_1, \dots, Z_m$ conformada por los valores máximos de cada uno de los m bloques en los que se haya dividido la muestra original.

La distribución de $(\hat{\mu}, \hat{\sigma}, \hat{\xi})$ sigue una normal multivariante con media (μ, σ, ξ) y matriz de covarianza igual a la inversa de la matriz de información.

3.1.2. MÉTODO POT

Dada una secuencia $X_1, \dots, X_n$ de variables aleatorias independientes e igualmente distribuidas de acuerdo a una distribución F, que no es conocida, se considera que una observación es extrema si sobrepasa un umbral $u < x^F$ previamente determinado, donde $x^F = \sup\{x: F(x) < 1\}$ es el punto extremo a la derecha de F. El excedente se lo modeliza utilizando probabilidades condicionadas:

$$F_y(x|u) = P(X - u < x | X > u) = \frac{F(x + u) - F(u)}{1 - F(u)}$$

Si u es lo suficientemente alto, entonces $F(x|u)$ puede aproximarse con la función generalizada de Pareto (GPD, por sus siglas en inglés).

$$G(x) = 1 - \left(1 + \frac{\xi(x-u)}{\sigma}\right)^{-\frac{1}{\xi}} \qquad si\ \xi \neq 0$$

$$G(x) = 1 - exp\left(-\frac{x-u}{\sigma}\right) \qquad si\ \xi = 0$$

donde $\sigma > 0$ y $\xi \in \mathbb{R}$ son los parámetros de escala y forma, respectivamente.
De acuerdo al teorema de Pickands-Balkema-de Haan la distribución de los excedentes $y = X - u$ se encuentra en el dominio de G, $F_y \in \mathcal{D}(G)$, si cumple que:

$$\lim_{u \to x^F} \left|F_y(x) - G(x|\sigma_u)\right| = 0$$

La elección del umbral es determinante, pues, si es muy bajo muchas observaciones se considerarán extremas y se pondrían en duda las suposiciones asintóticas sobre las que se sostiene el modelo, lo que conduciría a problemas de sesgo. Por otro lado, si el umbral es muy alto se pueden tener muy pocas observaciones, haciendo evidente la existencia de problemas de varianza en la estimación de los parámetros. Por lo tanto, la estimación del umbral es relevante en la determinación de los resultados. Se pueden adoptar distintas técnicas. Dos de ellas, propuestas por Coles (2001), son el análisis exploratorio o una evaluación de la estabilidad de los parámetros a diferentes valores del umbral.
La inversa de la distribución GDP es tal que:

$$G^{-1}(x) = u + \frac{\hat{\sigma}}{\hat{\xi}}\left(\left(\frac{n}{N_u(1-p)}\right)^{-\hat{\xi}} - 1\right)$$

donde $p \in [0, 1]$ es una probabilidad, $(\hat{\sigma}, \hat{\xi})$ son los parámetros estimados, n el número total de realizaciones y N_u el número de observaciones que exceden el umbral u.

3.1.2.1. MÉTODO MÁXIMA VEROSIMILITUD

La estimación de los parámetros (σ, ξ) se puede obtener a través de la maximización de la siguiente función de verosimilitud:

$$L(\sigma,\xi) = -m\,log(\sigma) - \left(1+\frac{1}{\xi}\right)\sum_{i=i}^{m}\log\left[1+\xi\left(\frac{x_i-u}{\sigma}\right)\right]$$

si $\xi \neq 0$, siempre y cuando se cumpla que:

$$1+\xi\left(\frac{x_i-u}{\sigma}\right) > 0, i = 1,\dots,m$$

Si $\xi = 0$, entonces la función de verosimilitud se obtiene a partir de la distribución Gumbel:

$$L(\mu,\sigma) = -m\log(\sigma) - \sigma^{-1}\sum_{i=1}^{m}(x_i-u)$$

En cualquier caso, $y_i = x_i - u$ es el excedente de las i-ésimas observaciones que sobrepasan el umbral u.

Las primeras y segundas derivadas parciales del logaritmo de la función de verosimilitud son las siguientes:

$$\frac{\partial L}{\partial \sigma} = \frac{1}{\sigma}\left[-m + (1+\xi)\sum_{i=1}^{m}\left(\frac{y_i}{\sigma+\xi y_i}\right)\right]$$

$$\frac{\partial L}{\partial \xi} = \frac{1}{\xi^2}\sum_{i=1}^{m}\ln\left(1+\frac{\xi y_i}{\sigma}\right) - \left(1+\frac{1}{\xi}\right)\sum_{i=1}^{m}\left(\frac{y_i}{\sigma+\xi y_i}\right)$$

$$\frac{\partial^2 L}{\partial \sigma^2} = \frac{1}{\sigma^2}\left[m - (1+\xi)\sum_{i=1}^{m}\frac{y_i(2\sigma+\xi y_i)}{(\sigma+\xi y_i)^2}\right]$$

$$\frac{\partial^2 L}{\partial \xi^2} = \frac{2}{\xi^3}\left[\xi\sum_{i=1}^{m}\frac{y_i}{\sigma+\xi y_i} - \sum_{i=1}^{m}\ln\left(1+\frac{\xi y_i}{\sigma}\right)\right] + \left(1+\frac{1}{\xi}\right)\sum_{i=1}^{m}\frac{y_i^2}{(\sigma+\xi y_i)^2}$$

$$\frac{\partial^2 L}{\partial \sigma \partial \xi} = \frac{1}{\xi}\left[(1+\xi)\sum_{i=1}^{m}\frac{y_i}{(\sigma+\xi y_i)^2} - \frac{1}{\sigma}\sum_{i=1}^{m}\frac{y_i}{\sigma+\xi y_i}\right]$$

3.1.3. MÉTODO DE NEWTON-RAPHSON

El método de Newton-Raphson es una técnica iterativa que permite la aproximación lineal de las raíces de una función continua y diferenciable a través del cálculo de la pendiente

de la función objetivo, o el gradiente cuando se tienen más de dos dimensiones, y la actualización de la estimación hasta que la distancia entre la tangente y la raíz verdadera sea considerablemente pequeña.

La fórmula que se aplica de manera reiterativa es la siguiente:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

donde x_n representa el valor aproximado de la raíz y $f'(x_n)$ la derivada de la función objetivo evaluada en el punto x_n.
Esta técnica de optimización puede ser utilizada para maximizar las funciones de verosimilitud presentes en la teoría de valores extremos para la estimación de los parámetros.

3.2. SEGUNDA PARTE: REDES GENERATIVAS ADVERSARIAS

Las redes generativas adversarias son complejas estructuras propias del campo del machine learning cuya configuración comprende dos redes neuronales antagónicas que compiten entre sí para obtener los mejores resultados. Las redes que la componen son la red generativa y la red discriminadora.

El objetivo de la red generativa es el de recrear la distribución de los datos reales x tal que la distribución $p_g = p_{data}$, a partir de datos $p_z(z)$ que básicamente constituyen ruido al inicio del proceso. Esta primera red neuronal se representa como una función $G(z; \theta_g)$ que depende de los parámetros θ_g. La segunda red, $D(x; \theta_d)$, es una función que determina la probabilidad de que el output de G provenga de los datos reales. Ambas funciones se constituyen como redes de múltiples capas (MLP).

Así, ambas funciones se enfrentan en un problema de minimax en la fase de entrenamiento del modelo. Por un lado, se busca maximizar la probabilidad de que D asigne la probabilidad correcta y, por otro, minimizar la probabilidad de que el output de G resulte en una incorrecta clasificación $\log(1 - D(G(z)))$:

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}\big[\log\big(D(x)\big)\big] + E_{z \sim p_z(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

donde $D\big(G(z)\big)$ representa la probabilidad de que el output de $G(z)$ venga de x y no de p_g. Para su resolución, se aplican técnicas numéricas y se alternan los distintos procesos de optimización entre un paso de G y k pasos de D.

En la práctica, el proceso de optimización puede quedar estancado si G no tiene suficiente gradiente. Como solución, proponen los especialistas maximizar $\log\left(D(G(z))\right)$ en el entrenamiento de G.

De acuerdo a Adigun y Kosko (2018) este tipo de red se entrena a través del uso de algún método de descenso por gradiente que contemple la técnica de retro propagación en el cálculo del gradiente. El algoritmo propuesto por Goodfellow et al. (2014) se compone de los siguientes pasos:

1. Se extrae una muestra $\{z^{(1)}, \dots, z^{(m)}\}$ de m observaciones de $p_g(z)$.
2. Se extrae una muestra $\{x^{(1)}, \dots, x^{(m)}\}$ de m observaciones de $p_{data}(x)$.
3. Se actualiza el discriminador empleando el método de descenso por gradiente:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [logD\big(x^{(i)}\big) + \log\left(1 - D(G(z^{(i)}))\right]$$

4. Se repiten 1,2 y k veces. k es un hiperparámetro que representa el número de pasos.
5. Se extrae una muestra $\{z^{(1)}, \dots, z^{(m)}\}$ de m observaciones de $p_g(z)$.
6. Se actualiza el discriminador empleando el método de descenso por gradiente:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D(G(z^{(i)}))\right)$$

7. Se repiten todos los pasos anteriores de acuerdo al número de veces que se desee iterar el proceso de entrenamiento.

3.2.1. ENTRENAMIENTO DE UNA RED ADVERSARIA GENERATIVA

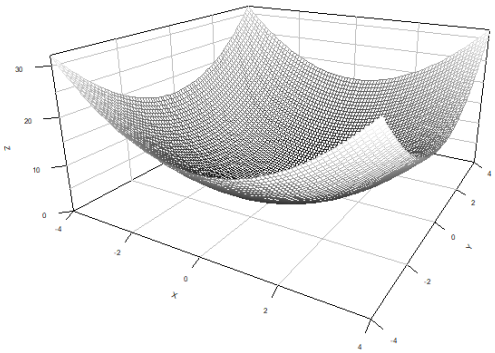
El entrenamiento de un modelo consiste en la minimización de la función de costo, que se alcanza al descender a lo largo del gradiente hasta alcanzar el punto mínimo. Dos conceptos estrechamente relacionados entran en juego, la retro propagación y descenso

por gradiente. A grandes rasgos, el primer término hace referencia a la técnica de cálculo de derivadas a lo largo de una ANN usando la regla de la cadena. El segundo, más general, hace alusión al ajuste de los parámetros de la función objetivo mediante la sustracción de las respectivas derivadas multiplicadas por una tasa de aprendizaje que determina la velocidad de movimiento en la dirección señalada por el gradiente.

3.2.1.1. GRADIENTE DESCENDIENTE

Se trata de un algoritmo de optimización en el que se encuentran los valores de los parámetros $\theta \in \mathbb{R}^d$ que minimizan la función objetivo $J(\theta)$. Para ello se actualizan los parámetros en la dirección contraria al gradiente $\nabla_\theta J(\theta)$, en otras palabras, se desciende a lo largo del gradiente de la función.

Figura 2. Función 3D con gradiente



Fuente: Elaboración propia

De acuerdo a la disponibilidad de datos, el algoritmo puede adoptar distintas configuraciones: gradiente descendiente por lote (batch gradient descent), gradiente descendiente minilote (mini-batch gradient descent) o gradiente descendiente estocástico. El primero de ellos, también denominado vainilla, computa el gradiente empleando la totalidad de los datos de entrenamiento disponibles para la actualización del parámetro:

$$\theta = \theta - \eta \nabla_\theta J(\theta)$$

En el caso del segundo, se extraen n muestras de i pares ordenados (x, y), donde $x^{(i)}$ son las observaciones e $y^{(i)}$ las etiquetas, cuyos elementos se seleccionan aleatoriamente en cada batch, acelerando el tiempo de computación considerablemente:

$$\theta = \theta - \eta \nabla_\theta J\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right)$$

Por último, el método estocástico es una simplificación del anterior en el que se considera una única muestra de i elementos:

$$\theta = \theta - \eta \nabla_\theta J\left(\theta; x^{(i)}; y^{(i)}\right)$$

Cualquiera sea la forma que adopte el algoritmo de gradiente descendiente η representa la tasa de aprendizaje (learning rate), es decir, el largo del paso que se recorrerá en la dirección del gradiente para alcanzar el mínimo.

3.2.1.2. RETRO PROPAGACIÓN

Conocida en el campo del machine learning por su nombre en inglés back propagation (BP), se trata de una de las reglas de aprendizaje más utilizadas en la ciencia de datos. El algoritmo de BP busca minimizar el valor de la función de costo E encontrando la combinación óptima de los pesos, enmarcándose para ello en el método del gradiente. Razón por la que es importante que la función objetivo sea diferenciable en cada una de los pasos del proceso de aprendizaje.

El funcionamiento del algoritmo de BP depende a su vez de dos componentes que operan secuencialmente una detrás de otra. La primera de ellas comprende el procedimiento por el cual se calcula el output de la red, es decir, un algoritmo que computa el valor de la función de activación en cada una de las capas de la ANN (este tipo de algoritmo se denomina algoritmo forward). La segunda componente ajusta gradualmente los pesos de la red de acuerdo a la influenza que ellos tengan sobre la función de error, que en su interpretación más sencilla representa la diferencia entre el output de la primera componente y el valor observado.

Dado un set de entrenamiento de k pares ordenados $\{(\boldsymbol{x}_1, \boldsymbol{t}_1), \ldots, (\boldsymbol{x}_k, \boldsymbol{t}_k)\}$ conformado por vectores de m y n dimensiones, respectivamente, y un vector inicial de pesos aleatorios $\boldsymbol{w} \in \mathbb{R}$**.** El objetivo del proceso de entrenamiento es la minimización de E corrigiendo los pesos iniciales. Si se asume que E es la función de error cuadrático medio:

$$E = \frac{1}{2}\sum_{i=1}^{k}\left|\left|\boldsymbol{y}_i - \boldsymbol{t}_i\right|\right|^2$$

el punto mínimo se alcanza cuando $\boldsymbol{y}_i = \boldsymbol{t}_i$ para $i = 1, \dots, k$. Siendo $\boldsymbol{y}_i$ el output que devuelve la red, dado el input $\boldsymbol{x}_i$. $\boldsymbol{t}_i$ representa el i-ésimo vector conformado por los valores observados (targets).

El error total de la red $E = E_1 + \cdots + E_k$ se obtiene a partir de la suma del error E_i de cada uno de los pares ordenados, que a su vez resulta de la sumatoria de $E_{i1}, E_{i2}, \dots, E_{in}$ de cada uno de los j elementos de $\boldsymbol{y}_i$ y $\boldsymbol{t}_i$

$$E_{ij} = \frac{1}{2}\left(y_{ij} - t_{ij}\right)^2$$

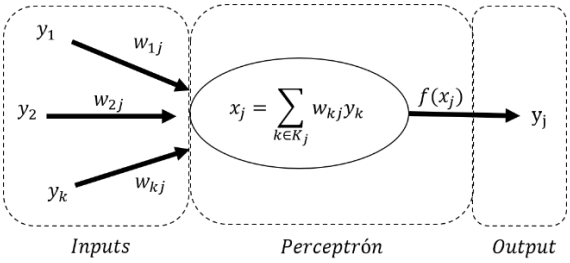
Los pesos son los parámetros que se deben calibrar para disminuir el error del proceso. En tanto que E es resultado de la composición de funciones (funciones de activación) a lo largo de los diferentes nodos, es una función diferenciable de ℓ pesos $w_1, \dots, w_\ell$. Por lo que se puede minimizar E mediante un proceso iterativo. Donde

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_\ell}\right)$$

es el gradiente que se emplea para determinar la variación en la que los pesos deben ser actualizados.

Por ejemplo, si se considera una red sencilla como el de la figura 3, en el que x_j es la suma ponderado de los pesos e y_j el output de la función de activación aplicada a x_j

Figura 3. Estructura de un perceptrón



Fuente: Elaboración propia basado en la ilustración de Makin (2006)

la variación que corresponde al peso w_{kj}, donde j es un nodo que pertenece a la capa output y k a uno de las capas ocultas, se obtiene a partir de:

$$\Delta w_{kj} = -\gamma \frac{\partial E}{\partial w_{kj}}$$

donde γ es un hyperparámetro que determina la tasa de aprendizaje (learning rate en inglés), y el signo negativo denota que la variación va en dirección decreciente del error. Asimismo, si aplica la regla de derivación de la cadena, entonces:

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{y_j} \frac{\partial y_j}{x_i} \frac{\partial x_i}{\partial w_{kj}}$$

A su vez,

$$\frac{\partial x_i}{\partial w_{kj}} = y_k$$

Por otra parte, el producto de las dos primeras derivadas parciales se lo conoce como el término de error, normalmente se lo representa con la letra griega delta:

$$\delta_j \coloneqq -\frac{\partial E}{y_j} \frac{\partial y_j}{x_i}$$

Si la función de activación es Sigmoide, se tiene que:

$$\frac{\partial y_j}{x_i} = y_j(1 - y_j)$$

Cuando se tiene en cuenta que el j-ésimo nodo pertenece a la última capa de la red, la derivada parcial de$E$ respecto a y_j es la derivada de la función de error, en este caso la función cuadrática:

$$\frac{\partial E}{y_j} = -(t_j - y_j)$$

Finalmente,

$$\frac{\partial E}{\partial w_{kj}} = -\big(t_j - y_j\big) y_j \big(1 - y_j\big) y_k$$

Sin embargo, esta ecuación es válida siempre y cuando j sea una neurona en la capa del output. En el caso que j sea un nodo de las capas ocultas, servirá determinar el error de y_j que se propagó a las i capas posteriores, es decir, $\frac{\partial E}{\partial y_j}$. Si se vuelve a aplicar la regla de la cadena:

$$\frac{\partial E}{\partial y_j} = \sum_i \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial x_i}\frac{\partial x_i}{\partial y_j}$$

Nuevamente, se puede identificar el término de error:

$$\delta_i = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial x_i}$$

Mientras que el último término se puede resolver de la siguiente manera:

$$\frac{\partial x_i}{\partial y_j} = w_{ji}$$

Reemplazando se tiene que:

$$\frac{\partial E}{\partial y_j} = -\sum_i \delta_i w_{ji}$$

Por lo tanto, si j es una neurona perteneciente a una capa oculta es necesario hacer los siguientes ajustes:

$$\frac{\partial E}{\partial w_{kj}} = -\sum_{\boldsymbol{i}} (\boldsymbol{\delta_i w_{ji}})\, y_j(1 - y_j)y_k$$

Finalmente, independientemente de la posición del nodo en la red, se puede generalizar de manera que:

$$\frac{\partial E}{\partial w_{kj}} = -\delta_j y_k$$

si j está en una capa oculta:

$$\delta_j = \sum_i (\delta_i w_{ji})\, y_j(1 - y_j)$$

si j se encuentra al final de la red:

$$\delta_j = (t_j - y_j) y_j (1 - y_j)$$

Por último, cabe mencionar que en el caso de las redes adversarias – GANs, la función de costo que se suele emplear es la de entropía cruzada binaria:

$$E = -tlog(o) - (1 - t)\log(1 - o)$$

Misma que se expresa de la siguiente manera cuando existen N instancias en el método de descenso por gradiente estocástico:

$$E = -\frac{1}{N}\sum_{i=1}^{N}[t_i \log(o_i) + (1 - t_i)\log(1 - o_i)]$$

Asimismo, señalar que una de las funciones de activación más utilizadas es la función Sigmoide, dado que la derivada de la misma es siempre positiva y asegura la continuidad de E. El uso de esta función de activación permite que siempre exista una dirección y una fuerza de descenso. En la tabla 1 se muestran algunas de las funciones más usadas en el campo de las redes neuronales, así como sus respectivas derivadas.

Tabla 1. Funciones de activación

Función de activación	Función	Derivada
Sigmoide	$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)\big(1 - f(x)\big)$
TanH	$f(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Rectified Linear Unit (ReLU)	$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$
Leaky ReLU	$f(x) = \begin{cases} x, & x < 0 \\ 0.1x, & x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 1, & x < 0 \\ 0.1, & x \geq 0 \end{cases}$

4. DESCRIPCIÓN DE LA MUESTRA

La hipótesis de la que se parte es que la generación de observaciones sintéticas, a través de las redes generativas adversariales, pueden asistir en la modelización de la severidad de eventos infrecuentes en los que se cuenta con poca información.

Para comprobar la veracidad de esta afirmación se utiliza la base de datos "fremarine", disponible en el paquete "CASdatasets" de R. Es a partir de estos datos que se pone en práctica el marco metodológico desarrollado en la sección anterior. Por un lado, se identifican y modelizan los siniestros extremos presentes en esta base de datos usando GDP y, por otro lado, la misma proporciona los datos de entrada para la generación de observaciones sintéticas utilizando redes neuronales. Disponiendo de la misma información en ambos casos, se pueden comparar las ventajas y desventajas de la técnica propuesta frente a aquella más convencional que se utiliza normalmente en el ámbito de la modelización de valores extremos en el mercado asegurador (Ghosh & Resnick, 2010).

La referida base de datos fue recolectada por una empresa privada de seguros francesa. Refleja el pago indemnizatorio de siniestros marítimos ocurridos entre los años 2003 y 2006[4], además de otra información relacionada a las características de los navíos. La misma consiste de 1.274 observaciones y veinte variables en total, las cuales se resumen en las tablas 2 y 3.

En la tabla 2 se observa claramente que el costo de las contingencias presenta valores extremos, pues los momentos de la serie se alejan mucho de aquellos propios de una distribución normal. Con un coeficiente de asimetría mayor a cero, además de una curtosis muy por encima de tres, se puede inferir que la severidad tiene un sesgo positivo y, consecuentemente, una pesada cola a la derecha. La figura 4 confirma la intuición al observarse la presencia de outliers. Aún después de la transformación, empleando una escala logarítmica, en la figura 5 se evidencia la existencia de valores extremos en la base de datos.

4 En virtud al reducido horizonte temporal se desaconseja la implementación del método BM y en su lugar el método POT sobresale como la mejor alternativa. En este sentido, es esta última la que se desarrolla en el presente documento para la modelización de los valores extremos.

Tabla 2. Momentos de las variables numéricas

Variable	Media	Desv. Est	Asimetría	Curtosis	NA's
ShipPower	434,96	294,94	1,93	8,60	208
ShipLength	7,92	3,00	1,30	4,39	208
ShipTonnage	30,90	48,35	2,80	12,01	208
InsuredValue	1.184,14	3.858,73	5,98	41,37	0
ClaimPaid	21,61	138,22	18,33	423,15	0
ClaimCharge	25,56	159,23	16,29	321,96	0
ClaimRecourse	0,37	4,26	18,56	409,57	0
Deductible	2,08	6,49	5,86	41,43	1093

ShipPower es la potencia de la embarcación asegurada; ShipLength es el largo de la embarcación asegurada; ShipTonnage es el tonelaje de la embarcación asegurada; InsuredValue es la suma asegurada; ClaimPaid es el monto pagado en euros por concepto de indemnización (reescalada y alterado); ClaimCharge es el importe del siniestro en euros (reescalado y alterado); ClaimRecourse es la cantidad recuperada en euros por la aseguradora (reescalada y alterado); Deductible es el valor de la franquicia (reescalado y alterado).

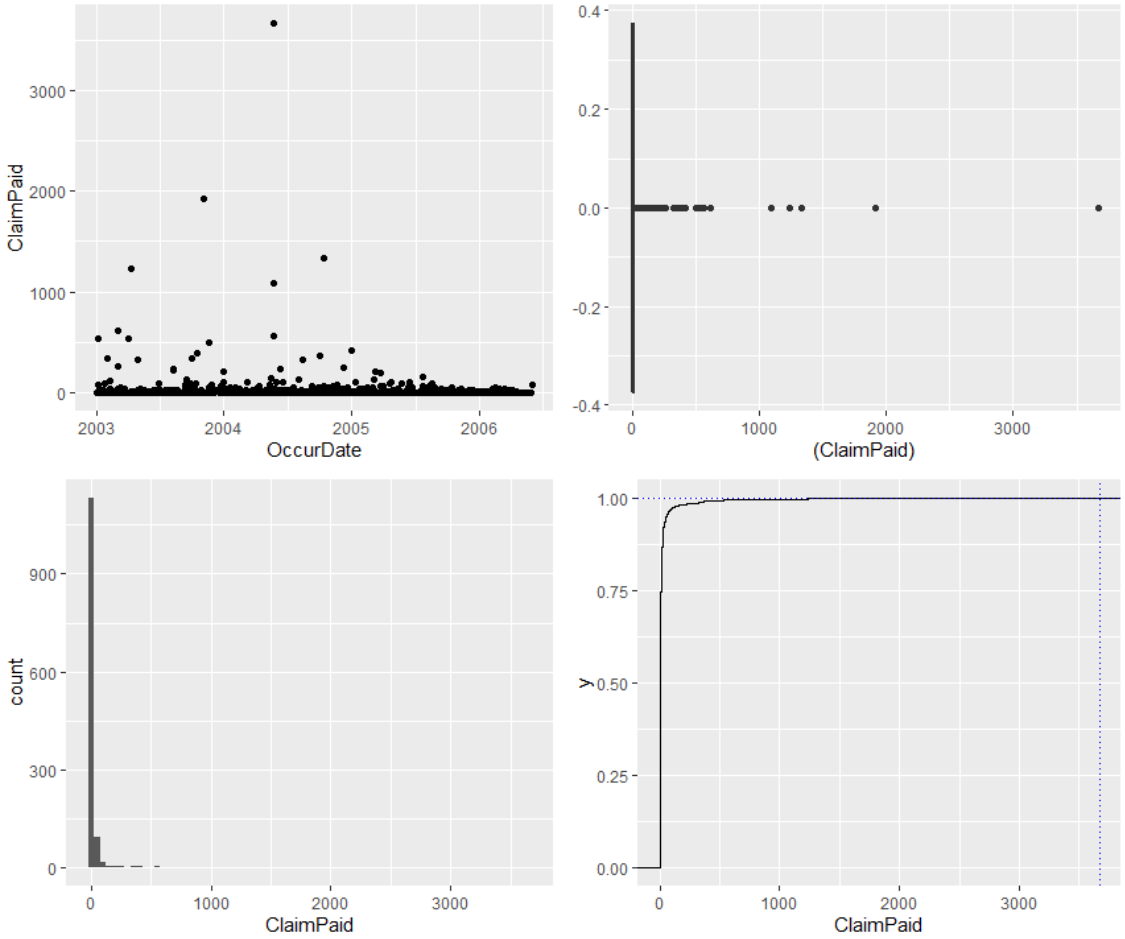
Tabla 3. Características de las variables nominales

Variable	Niveles	Moda	Veces	NA's
ShipCateg	20	Trawler	380	0
ShipBrand	32	BAUDOUIN	213	208
ShipEngNb	2	1	991	208
ShipEngYear	34	1995	82	208
ShipBuildYear	57	1978	105	208
ShipHull	7	Polyester	511	208
ClaimCateg	4	C2	1093	0
ClaimStatus	2	settled	1101	0
HeadQuarter	125	Boulougne sur mer	87	229
Departement	19	Finistere	434	0

ShipCateg es la categoría de la embarcación asegurada; ShipBrand es la marca del barco (remuestreado); ShipEngNb es el número de motores del barco; ShipEngYear es el año de los motores (remuestreado); ShipBuildYear es el año de construcción de la embarcación (remuestreado); ShipHull es el casco del barco (remuestreado); ClaimCateg es la categoría del siniestro (remuestreado); ClaimStatus es el estado del siniestro (remuestreado); HeadQurter es el nombre de la ciudad en la que se encuentran las oficinas centrales del barco (remuestreado); Departement es el departamento francés correspondiente al nombre de la ciudad.

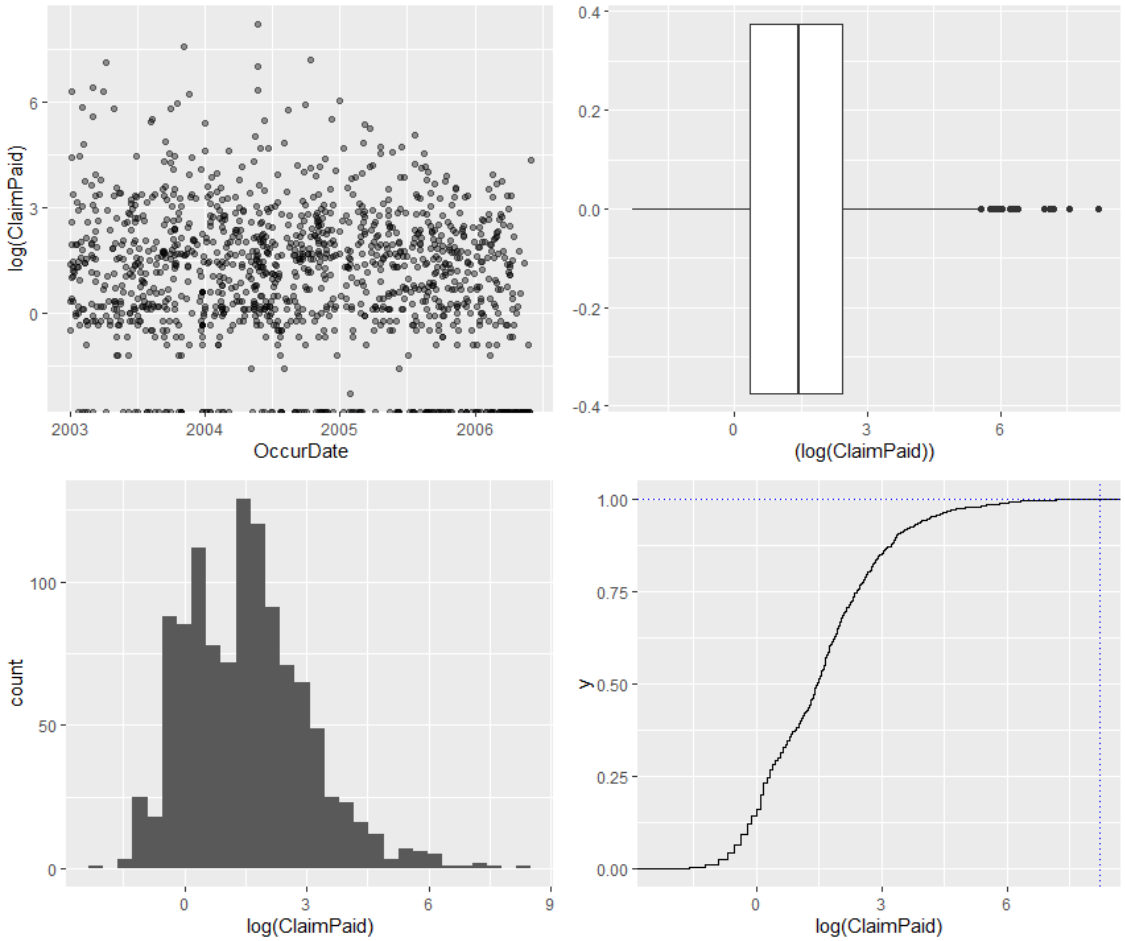
Cabe mencionar que, para el caso de la red neuronal, se explora la posibilidad de incluir las variables categóricas una vez reexpresadas en valores numéricos en un intervalo [0,1]. De acuerdo a Patki (2016), el proceso consiste en el cálculo de la proporción de cada categoría en el total de observaciones, estableciendo un intervalo $[a, b]$, a partir del cual se genera un número aleatorio distribuido $N \sim \left(\mu = \frac{a+b}{2}, \sigma = \frac{b-a}{6}\right)$.

Figura 4. Cuantía de los siniestros (escala lineal)



Fuente: Elaboración propia

Figura 5. Cuantía de los siniestros (escala logarítmica)



Fuente: Elaboración propia

Otra manera que se tiene para detectar la presencia de observaciones punta en un conjunto de datos es la función de exceso de la media:

$$m(x) = E(X - x|X > x)$$

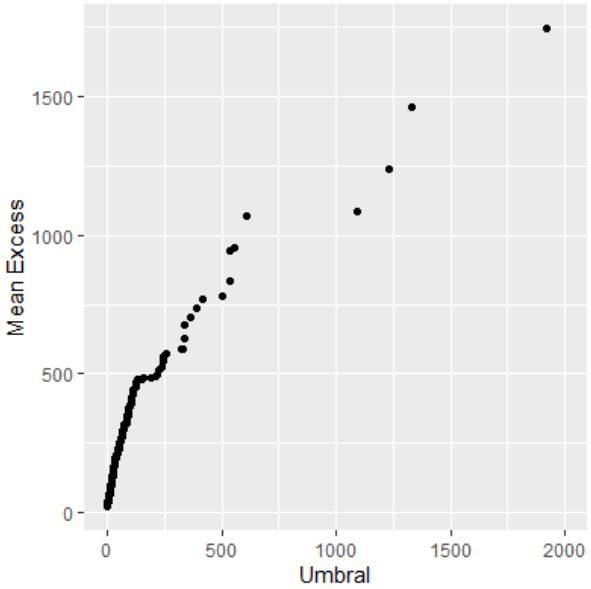
donde x representa una realización de la variable aleatoria X; la cual se puede interpretar como el exceso esperado de la severidad. La tendencia positiva en la figura 6 confirma una vez más que se trata de una distribución de cola pesada, al dibujarse una tendencia positiva que recorre el gráfico de un extremo a otro.

Finalmente, para concluir la verificación del tipo de distribución que siguen los datos de estudio, en la figura 7 se confirma que la cola de la distribución no decrece de manera exponencial, a través de un diagrama de cuantiles, cuyos pares ordenados se obtienen con la fórmula:

$$(-\log\left(\frac{j}{n+1}\right), X_{n-j+1,n})$$

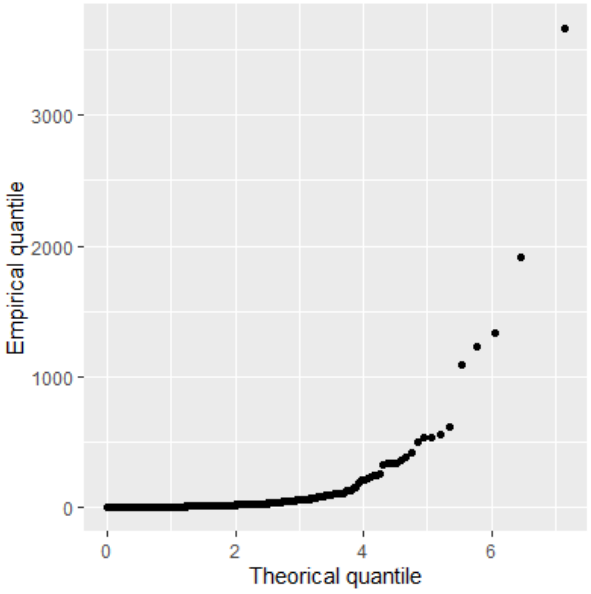
donde $X_{1,n}, X_{2,n}, \ldots, X_{n,n}$ representa los ordenados y $j = 1, \ldots, n-1$. El gráfico presenta una curvatura que se aleja de un ajuste lineal que representaría un modelo exponencial.

Figura 6. Función de exceso de la media



Fuente: Elaboración propia

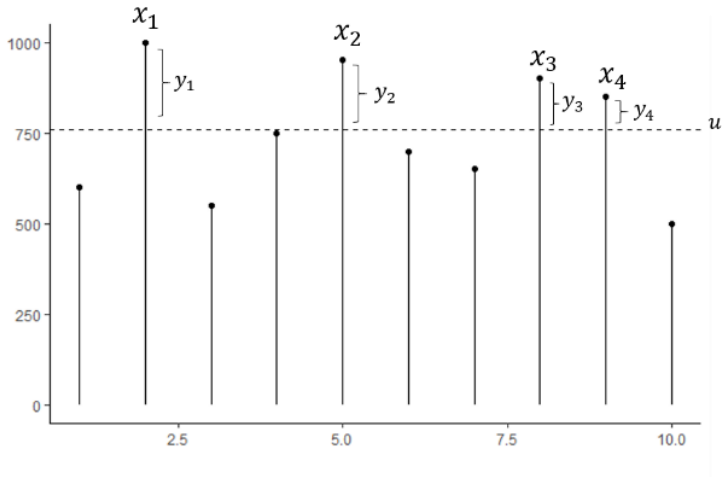
Figura 7. Gráfico QQ exponencial



Fuente: Elaboración propia

Por otro lado, en el método POT se deben identificar los valores extremos. Para distinguir los mismos en una serie $x_1, \dots, x_n$ i.i.d., se debe determinar el umbral u más bajo posible que asegure un balance entre sesgo y varianza. Los valores extremos $x_1, \dots, x_k$ se componen de aquellas observaciones que sobrepasen el umbral previamente determinado $\{x_i : x_i > u\}$. En la modelización se toma en cuenta los excedentes $y_j = x_j - u$, para $j = 1, \dots, k$.

Figura 8. Método Peaks-over-threshold (POT)



Fuente: Elaboración propia

Seleccionar el umbral correcto es una tarea de suma importancia que se puede realizar mediante distintos métodos. Por ejemplo, a través del cálculo iterativo de la curtosis (Chen et al., 2005), mediante la exploración o la evaluación de la estabilidad de los parámetros tras realizar el ajuste estadístico con distintos umbrales (Coles, 2001).

Las bases del primer método propuesto por Coles (2001) se sostienen sobre el hecho de que si $X_1, \dots, X_n$ se distribuyen de acuerdo a una GPD para un umbral u_0, entonces para umbrales $u > u_0$ continúan distribuyéndose como una GDP, tras un ajuste del parámetro de escala inicial σ_{u_0}.

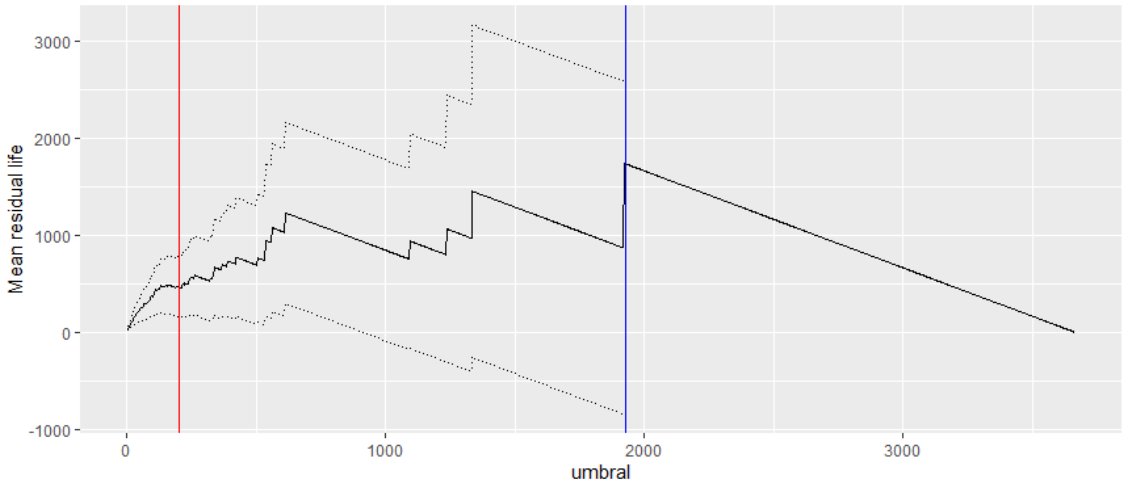
$$E(X - u | X > u) = \frac{\sigma_u}{1 - \xi} = \frac{\sigma_{u_0} + \xi u}{1 - \xi}$$

En la figura 9 se grafica la vida residual media, misma que se calcula a partir de la siguiente fórmula:

$$\left\{\left(u, \frac{1}{n_u}\sum_{i=1}^{n_u}\left(x_{(i)}-u\right)\right) : u < x_{max}\right\}$$

donde $x_{(i)}, \ldots, x_{(n_u)}$ contiene las n_u observaciones ordenadas que exceden u y x_{max} es la observación más extrema. A priori, se observa que partir de $u = 1925$ se identifica una relación lineal entre el umbral y el valor residual de vida medio; no obstante, se tiene una única observación que sobrepasa dicho umbral. Por lo tanto, el método exploratorio sugiere seleccionar $u = 200$ en tanto que este es el punto en el que parece desaparecer la concavidad, además, si bien es cierto que la tendencia cambia de pendiente posteriormente, es partir de este punto que existe cierta evidencia de linealidad.

Figura 9. Gráfico de la vida residual media

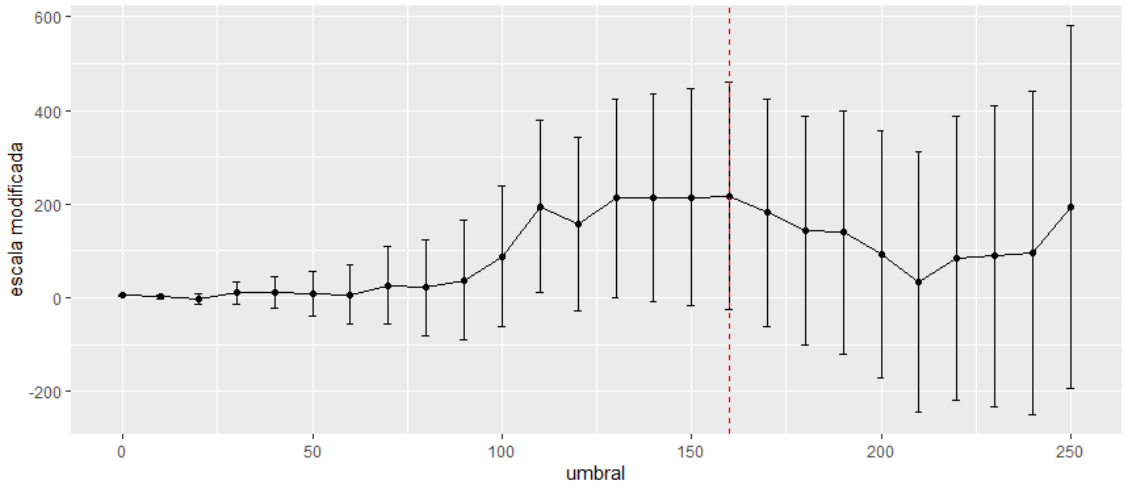


Fuente: Elaboración propia

5. RESULTADOS

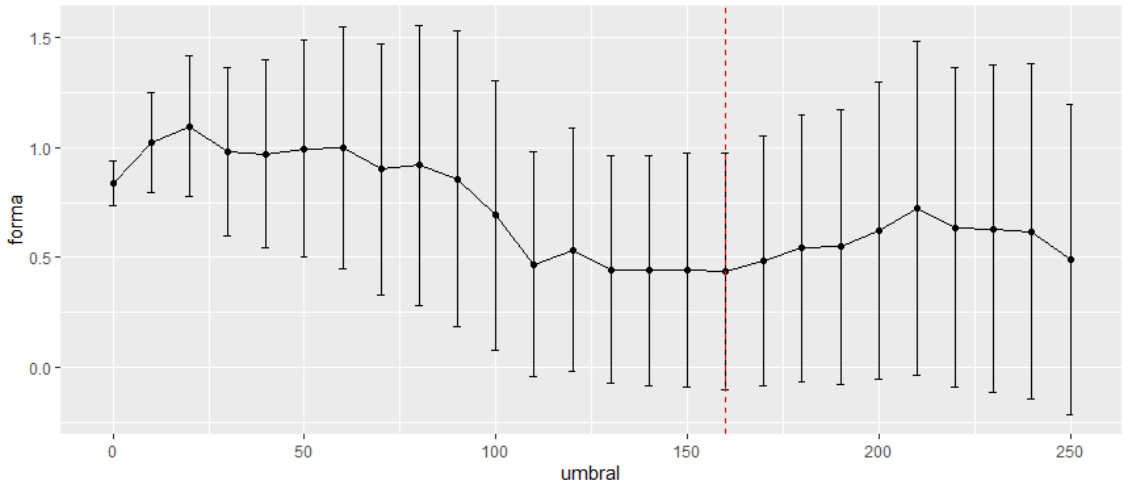
En primera instancia, en la sección anterior surgía $u = 200$ como una posible opción, el cual resulta una buena aproximación si no se cuenta con mayor información. No obstante, en virtud a los resultados de las figuras 10 y 11, en los que se evalúa la estabilidad de los parámetros con distintos umbrales, se advierte que es alrededor de $u = 160$ donde el comportamiento de los parámetros ξ y σ^* (forma y escala modificada, respectivamente) se mantiene invariable. Ergo, este punto constituye el umbral a partir del cual se delimitan las observaciones extremas.

Figura 10. Estimación parámetro σ* a diferentes niveles del umbral



Fuente: Elaboración propia

Figura 11. Estimación parámetro ξ a diferentes niveles del umbral



Fuente: Elaboración propia

Tras la delimitación del umbral, se tienen 25 observaciones con las que la estimación de los parámetros por el método de la máxima verosimilitud resulta en los siguientes valores:

$$\left(\hat{\sigma}, \hat{\xi}\right) = (287.522\,, 0.438)$$

alcanzando un valor máximo del log-verosimilitud de -177.477. Asimismo, a partir de la inversa de la matriz de información observada $J\left(\hat{\sigma}, \hat{\xi}\right)^{-1}$, se obtiene la matriz de varianzas y covarianzas:

$$\begin{bmatrix} 8{,}910 & -14.453 \\ -14.453 & 0.075 \end{bmatrix}$$

La desviación estándar es 94.392 y 0.075 para $\hat{\sigma}$ y $\hat{\xi}$, respectivamente. El correspondiente intervalo de confianza al 95% para σ es $[102.515, 472.528]$ y $[-0.101,\ 0.976]$ para ξ.

Tabla 4. Estimación de parámetros máxima verosimilitud

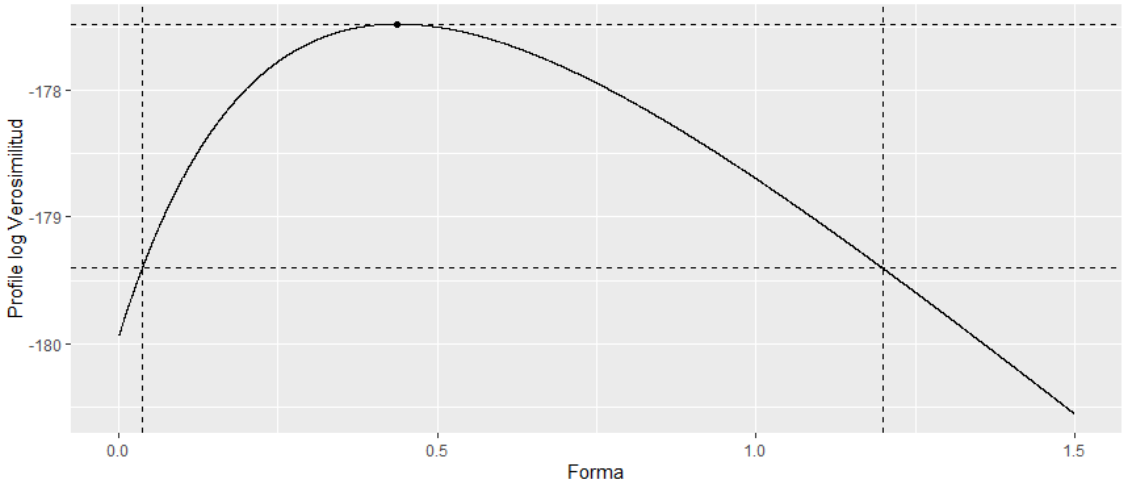
Umbral (u)	**160**	**200**
Número observaciones	25	24
$\hat{\sigma}$	287.522	216.73
$Var(\widehat{\sigma})$	8,980	6,763.84
$IC_{95\%}$	[102.515, 472.528]	[31.723, 401.74]
$\hat{\xi}$	0.438	0.622
$Var(\hat{\xi})$	0.075	0.118
$IC_{95\%}$	[-0.101, 0.976]	[0.083, 1.16]

Por otro lado, como método de verificación, se calcula el parámetro que maximiza la función de verosimilitud a través del método de log-verosimilitud perfil, asumiendo que ξ es constante. Las figuras 12 y 13 muestran que los parámetros estimados anteriormente son congruentes. Asimismo, el intervalo de confianza se calcula como $argmax\left(L(\hat{\sigma})\right) - 0.5\chi_1^2(0.05)$.

Tabla 5. Estimación log-verosimilitud perfil

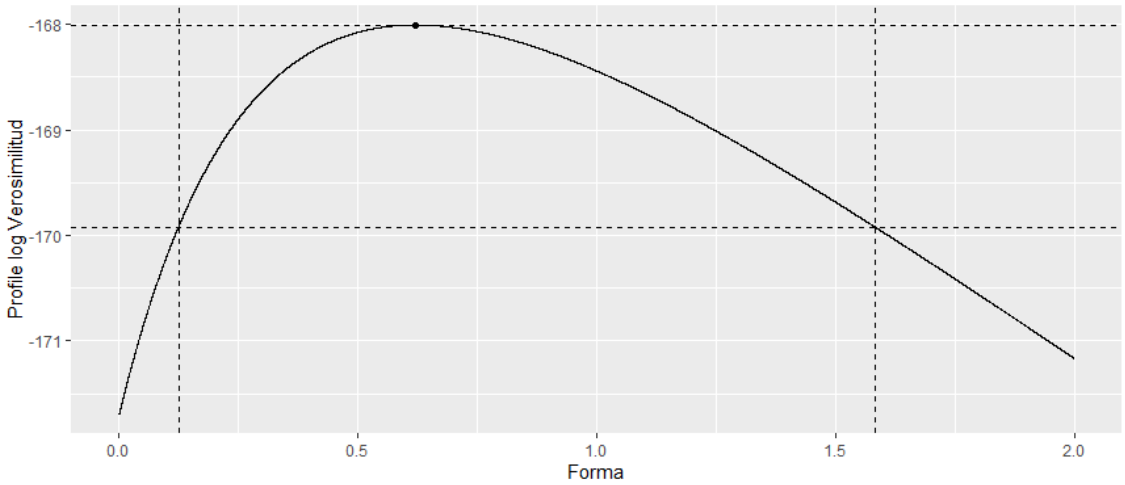
Umbral	160	200
MLE	-177.48	-168.00
$MLE - 0.5\chi_1^2(0.05)$	-179.40	-169.92
$\hat{\xi}$	0.438	0.622
$IC_{95\%}$	[0.037, 1.199]	[0.125, 1.584]

Figura 12. Log-verosimilitud perfil (u=160)



Fuente: Elaboración propia

Figura 13. Log-verosimilitud perfil (u=200)

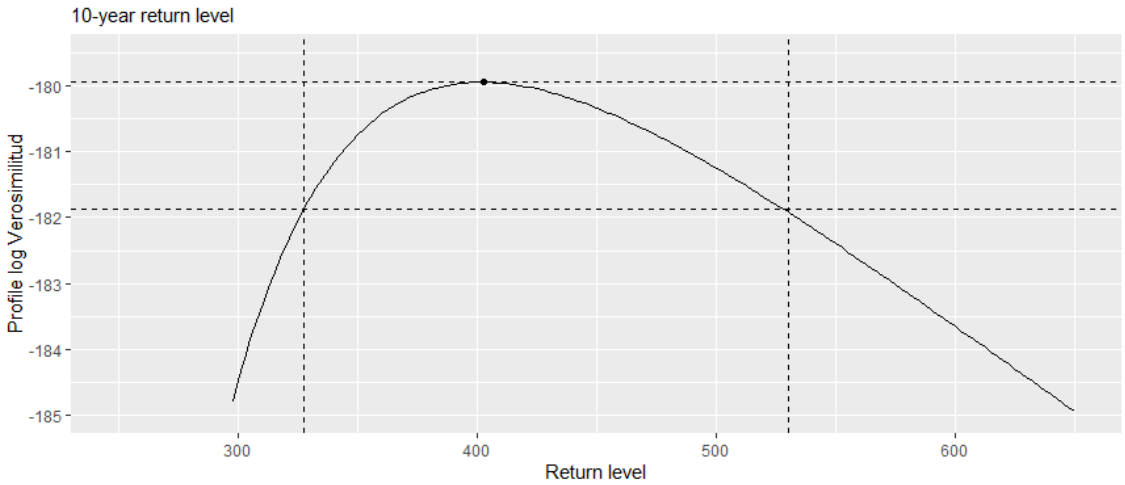


Fuente: Elaboración propia

Las figuras 14 y 15 muestran que el ajuste realizado tiene un carácter conservador. En la primera de ellas se exploran las observaciones extremas que se esperan en el curso de 10

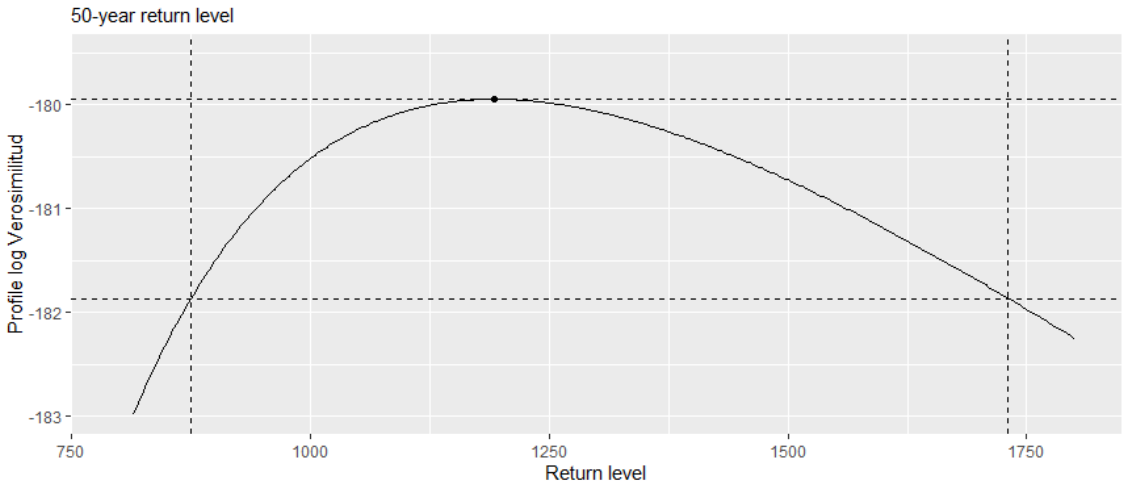
años a un nivel de confianza del 95%, mientras que en el segundo se extrapola el mismo análisis para a un período de 50 años, en ambos casos se advierte que la observación empírica más extrema (3.667) no se contempla en ninguno de los escenarios. Pues, en el primero se esperan valores extremos entre 350 y 550 y en el segundo entre 875 y 1.750.

Figura 14. Return level plot (u=160)



Fuente: Elaboración propia

Figura 15. Return level plot (u=160)

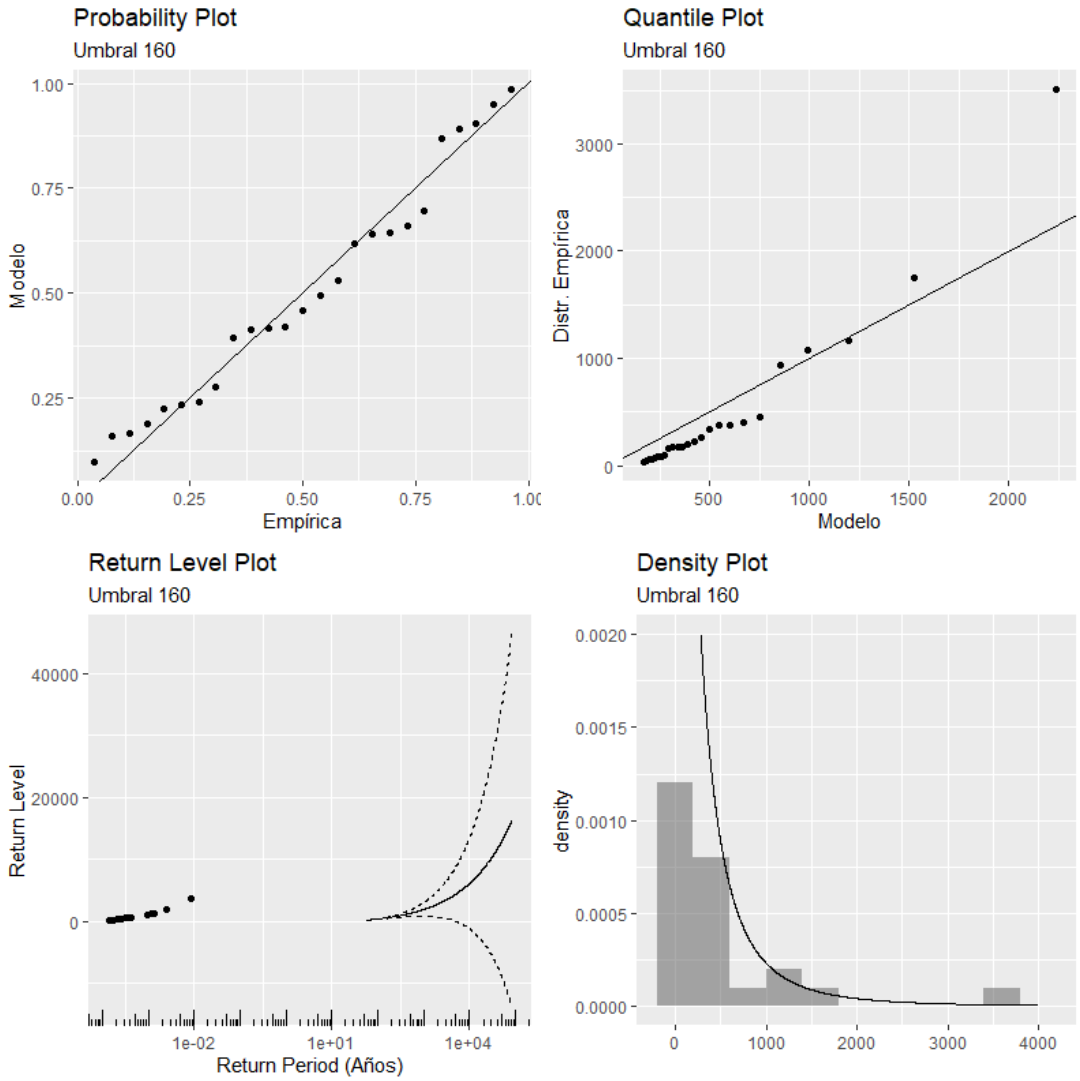


Fuente: Elaboración propia

Por último, a fin de verificar el ajuste del modelo se evalúan los gráficos de diagnóstico de la figura 16. El primero de ellos, el gráfico de probabilidad (Probability Plot), compara la función de distribución empírica y la del modelo, en él se observa un comportamiento lineal sobre la diagonal $x = y$ muy similar; sin embargo, esta es una representación poco indicativa del grado de ajuste en tanto que el valor máximo que puede adoptar la función

de distribución es uno en ambos casos. El segundo, el gráfico de quantil-quantil (Quantile Plot) confronta la inversa de la función de distribución acumulada de los datos observados y del modelo; en el mismo se distingue un comportamiento convexo que se aleja de la ansiada linealidad para las observaciones más extremas. Situación similar se advierte en el tercer gráfico (Return Level Plot) en el que se muestran los periodos de retorno en el eje de las abscisas y los niveles de retorno[5] en el eje de las ordenadas, una vez que las observaciones empíricas quedan fuera de los intervalos de confianza al 95%. Finalmente, en el último gráfico (Density Plot) se percibe que la función de densidad no se ajusta apropiadamente al histograma de los datos reales.

Figura 16. Gráficos de diagnóstico



Fuente: Elaboración propia

5 El objetivo de este gráfico es determinar la observación más extrema que se espera en un horizonte de tiempo determinado.

Una vez examinados los resultados de la distribución GPD, toca el turno de la ANN. Compuesta por una red discriminadora con cuatro capas totalmente conectadas con 16, 32, 128 y 2 nodos respectivamente, y una red generativa conformada por tres capas totalmente conectadas con 16, 32 y 128 nodos, la red neuronal aproxima notablemente la distribución subyacente después de haber sido entrenada durante 80.000 repeticiones y haberse considerado la totalidad de las observaciones[6]. En la figura 17 se muestran en amarillo los datos generados por la ANN y en color negro los datos reales. Se observa que a medida que aumenta el número de veces que la red es entrenada, los resultados mejoran hasta el punto en el que los primeros prácticamente terminan sobreponiéndose a los últimos. En definitiva, la red infiere la distribución real y se ajusta a la misma después de un intensivo proceso de entrenamiento en el que se ajustan los pesos de las neuronas para imitar el comportamiento de la información que le es dada como entrada.

A efectos de valorar la calidad del ajuste de la red neuronal y evitar caer en la subjetividad, se calcula la medida de divergencia Kullback-Leibler (KL) a partir de una muestra de valores aleatorios de la misma. Adicionalmente, se realiza el mismo ejercicio para muestras provenientes de distintas distribuciones teóricas[7] para inferir la eficacia de la red neuronal frente a métodos más convencionales.

El estadístico KL es un indicador de entropía relativa que permite comparar cuán diferente es una distribución respecto a otra de referencia. En la teoría de la información, es utilizado para determinar la pérdida de información que se produce por el uso de una distribución en reemplazo de otra.

Las distribuciones que se ajustaron a los datos reales se muestran en el anexo 1, en la que también se observa el p-valor de la prueba χ^2 de Pearson para determinar la bondad de ajuste, a un nivel de significancia $\alpha = 0.05$.

Recordando que la prueba de χ^2 se define de la siguiente manera:

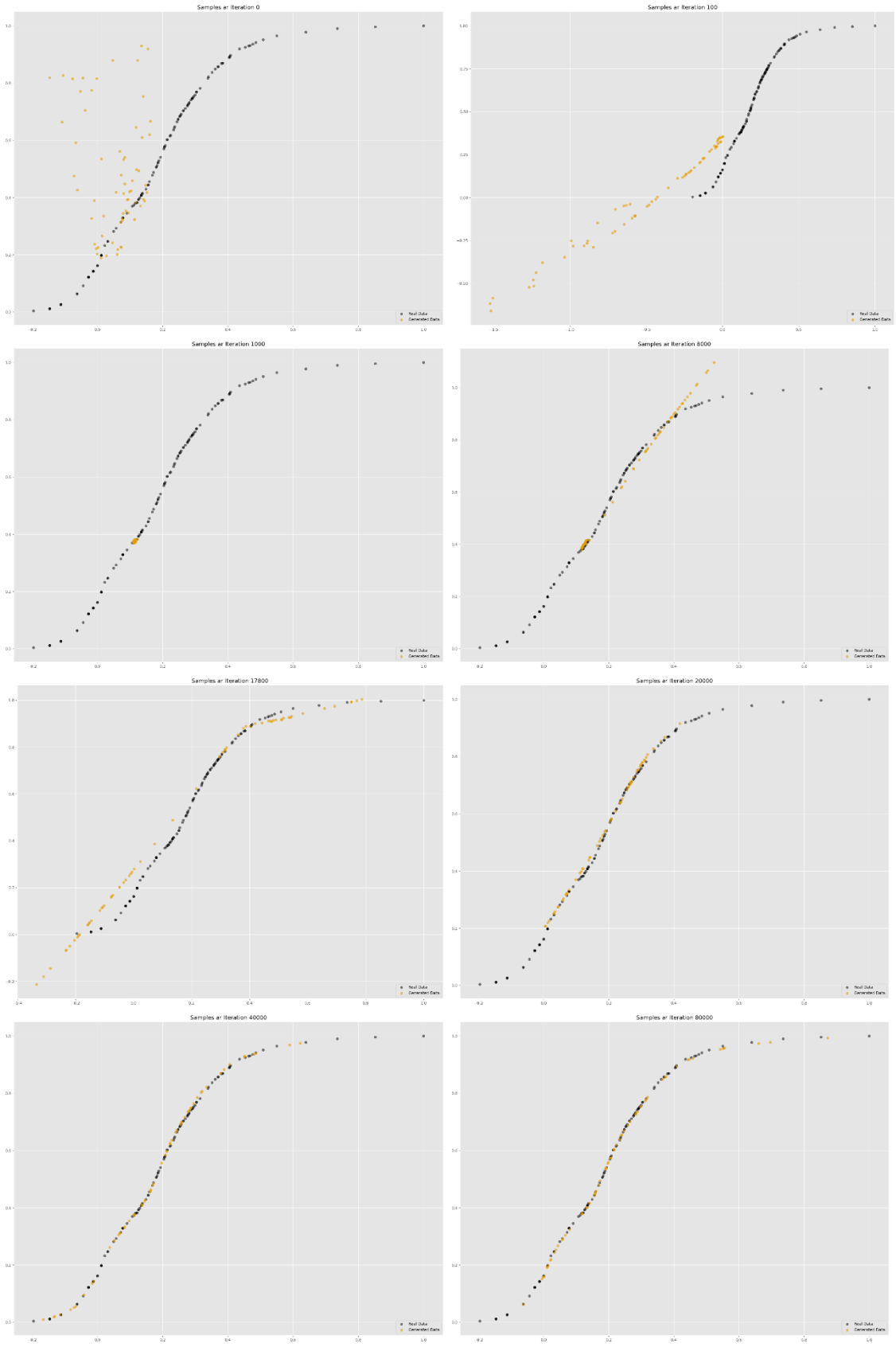
$$H_0: Los\ datos\ siguen\ la\ distribución\ especificada$$

$$H_1: Los\ datos\ no\ siguen\ la\ distribución\ especificada$$

6 En primera instancia, para evaluar la versatilidad de la red neuronal se omite la distinción entre siniestros masa y siniestros punta.

7 Se emplean todas las distribuciones continuas disponibles en el módulo de funciones estadísticas del paquete SciPy en Python para modelar la serie de datos original.

Figura 17. Entrenamiento de la red neuronal (todas las observaciones)



Fuente: Elaboración propia

La hipótesis nula se rechaza cuando el estadístico es mayor que el nivel crítico:

$$\chi^2 > \chi^2_{1-\alpha,k-c-1}$$

donde k es el número de intervalos de clase[8] en el que se dividen los datos que se analizan y c el número de parámetros que requiere la distribución teorética para ser totalmente definida. Asimismo, χ^2 se calcula a partir de las frecuencias observadas O y las frecuencias esperadas E, de manera que:

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

Habiendo realizado el ajuste de un total de 84 distribuciones, solamente las distribuciones de la tabla 6 presentan un p-valor mayor al 5% y, consecuentemente, no se rechaza la hipótesis nula para las mismas. Para verificar el grado de ajuste entre estas distribuciones teóricas y la aproximada por la red neuronal, se generaron 10.000 observaciones de cada una y se compararon con la distribución de los datos reales a través de la medida de divergencia de KL:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

donde P representa la distribución real y Q la aproximada. En otras palabras, se evalúa cuanta información, expresada en unidades naturales de infomación o nats (por sus siglas en inglés), se pierde por usar la distribución Q en lugar de P. En la tabla 7 se advierte que la distribución que más se asemeja a la real es la red neuronal con una pérdida de 0,17, que es la menor entre todas las alternativas; seguida por la distribución Mielke y Weibull inversa con 2,856892 y 3,012258, respectivamente.

8 Es necesario realizar este ajuste porque se trata de una variable continua.

Tabla 6. Distribuciones teóricas con p-valor mayor a 5%

Distribución	Estadístico	Valor Crítico	P-valor
Gamma inversa	198,92	229,66	0,4284
Weibull inversa	202,28	229,66	0,3642
Levy	117,99	230,75	1,0000
Mielke	204,95	228,58	0,2983
F no central (Ncf)	216,76	227,50	0,1257
t no central (Nct)	167,25	228,58	0,9256
Pareto	135,17	229,66	0,9997

En resumen, los valores del indicador señalan que la técnica de machine learning es capaz de aproximar la distribución subyacente con mayor precisión. No obstante, podría existir un sobreajuste que impida la extrapolación de los resultados. Para solventar esta limitación se podrían emplear variantes de redes GAN con estructuras más complejas.

Tabla 7. Resultados Kullback-Leibler

Distribution	KL
Gamma inversa	3,493459
Weibull inversa	3,012258
Levy	3034,451
Mielke	2,856892
F no central (Ncf)	8,68762
t no central (Nct)	13,91228
Pareto	17,27363
GAN	0,171424

Por otro lado, si se consideran únicamente aquellas observaciones que se etiquetaron como extremas en el método GPD, vale decir, los siniestros cuyo coste sobrepasa el umbral de 160 unidades monetarias, entonces la configuración de la red neuronal cambia para adaptarse a la cantidad de datos disponibles. Bajo esta consideración, la misma queda conformada por una red generativa con dos capas totalmente conectadas con 10 y 20 nodos respectivamente y una red discriminadora compuesto por tres capas totalmente conectadas entre sí con 10, 20 y 2 nodos. Para entrenar esta red se selecciona una muestra

de 10 observaciones reales, que son escogidas de manera aleatoria en cada una de las 100.000 repeticiones de entrenamiento. Los resultados se muestran en la figura 18.

A pesar de la menor cantidad de observaciones, los datos generados se aproximan notoriamente a los datos reales. En comparación con la modelización a través de GPD, la técnica machine learning denota una mejor aproximación de la distribución original. Esta aseveración se puede verificar en la tabla 8 usando la medida de divergencia KL después de seguir el mismo el procedimiento empleado anteriormente, generándose 10.000 valores aleatorios con ambos ajustes (en el caso de la ANN se consideran los pesos de la repetición 99.000 que es aquella que más cerca estuvo de los datos verdaderos).

Tabla 8. Resultados Kullback-Leibler

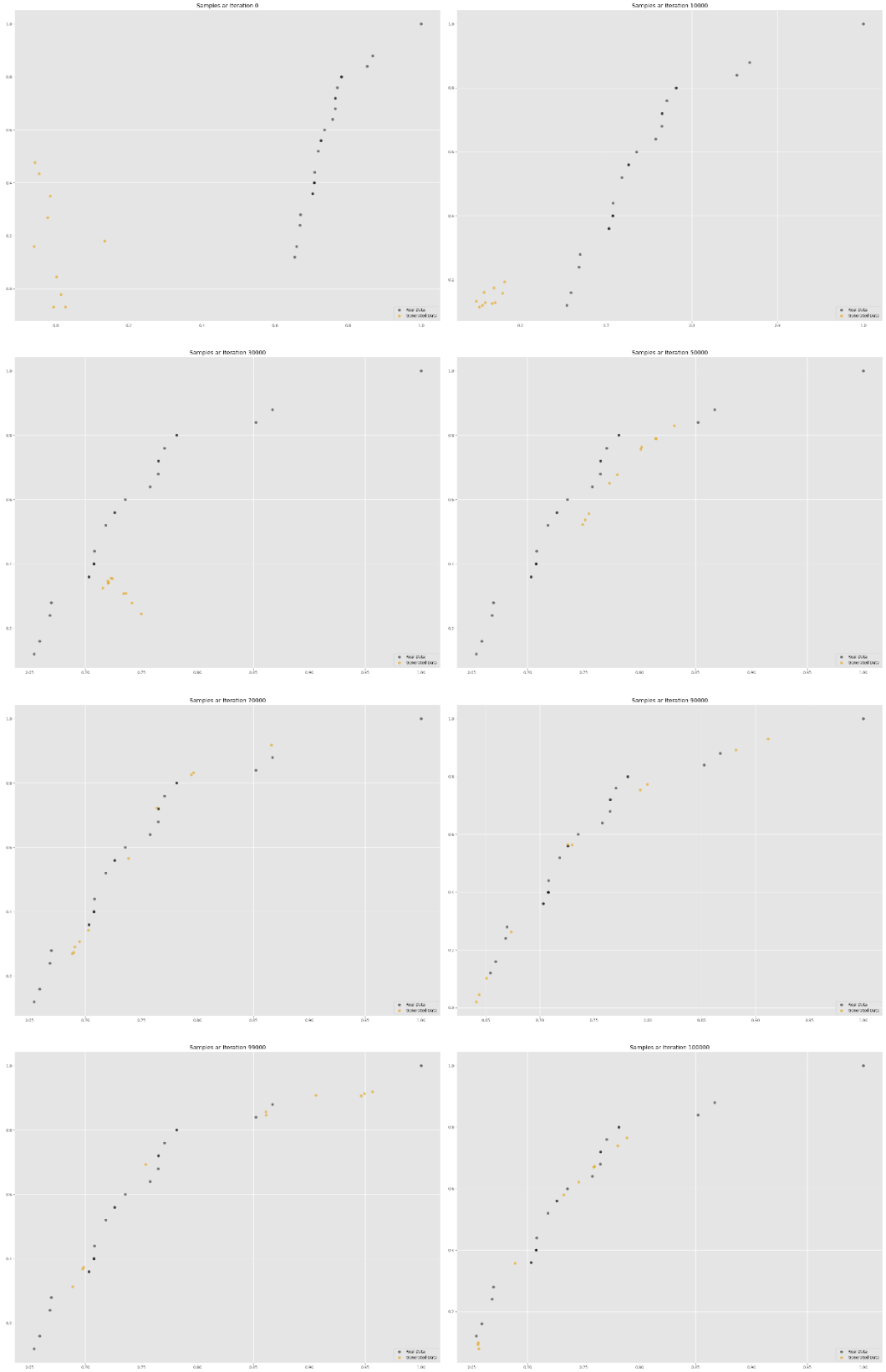
Distribution	KL
GPD	1,831682
GAN	0,032361

En virtud al alto grado de semejanza de los datos generados con el método GAN, se explora la posibilidad de complementar las dos técnicas analizadas en este documento, a efectos de mejorar la modelización de la severidad. En este sentido, las 10.000 observaciones sintéticamente generadas se adicionan a las 25 originales para alcanzar un total de 10.025. La disponibilidad de datos para realizar el ajuste mediante la distribución GPD se incrementa 400 veces.

Los resultados que se muestran en la tabla 9 incorporan los datos sintéticamente generados y se observan diferencias en la estimación de los parámetros. En el caso de la escala, $\hat{\sigma}_g$ es mayor que el cálculo inicial $\hat{\sigma}$, mientras que $\hat{\xi}_g < \hat{\xi}$. En ambos casos, se reduce la varianza y, consecuentemente, la amplitud de los intervalos de confianza.

Considerando esta calibración, la calidad del ajuste se evalúa a través de los gráficos de diagnóstico de la figura 19. La curvatura presente en los tres primeros gráficos indica la existencia de asimetría positiva. Puntualmente, el gráfico quantil-quantil expone que los valores extremos no se ajustan correctamente. Esto sugiere que se debe escoger un nuevo umbral.

Figura 18. Entrenamiento de la red neuronal (sólo observaciones extremas)

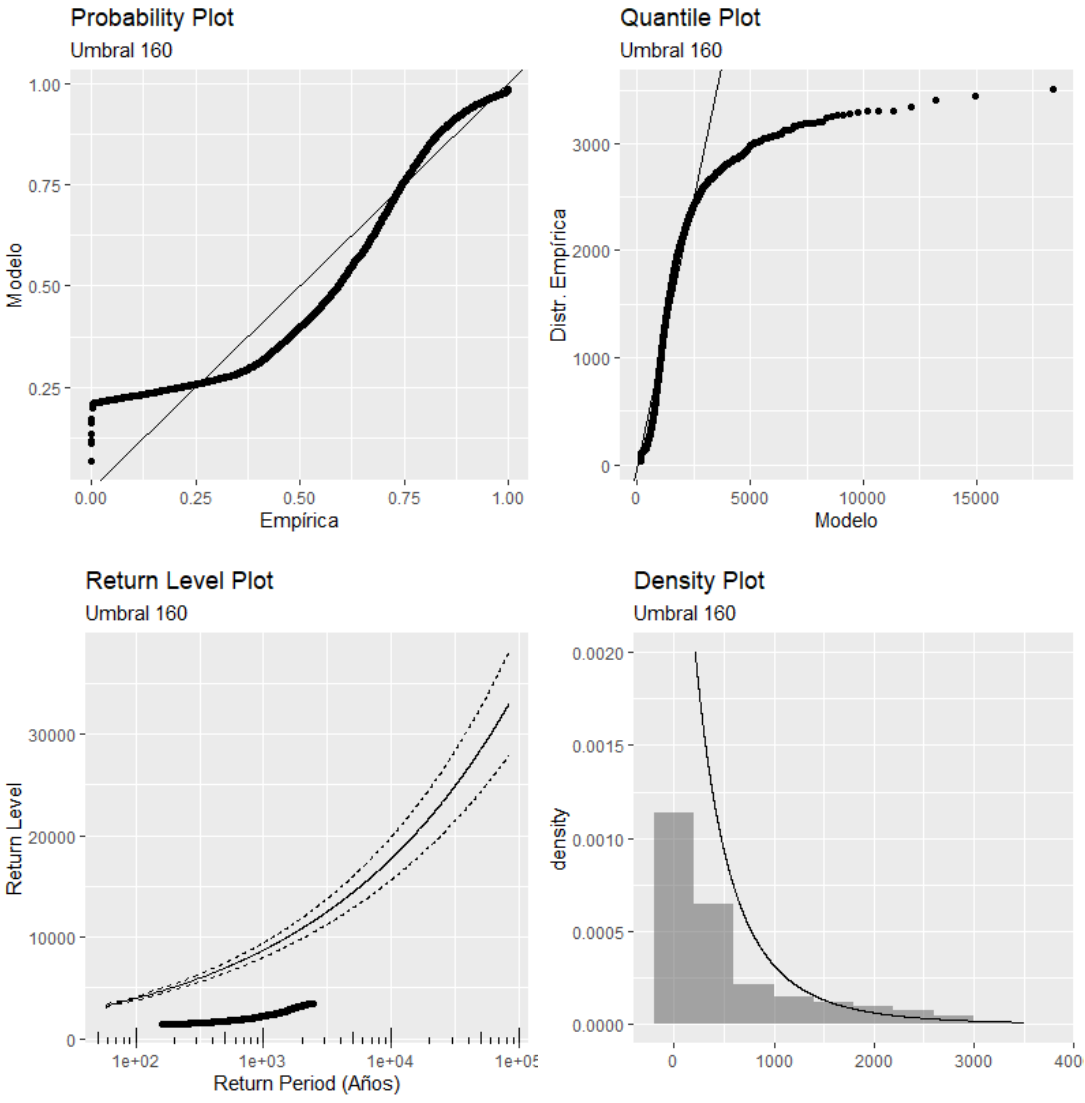


Fuente: Elaboración propia

Tabla 9. Estimación de parámetros por máxima verosimilitud

Umbral (u)	**160**
Número observaciones	10.025
$\hat{\sigma}_g$	430.530
$Var(\hat{\sigma}_g)$	51.753
$IC_{95\%}$	[416.430, 444.630]
$\hat{\xi}_g$	0.276
$Var(\hat{\xi}_g)$	0.00019
$IC_{95\%}$	[0.249, 0.303]

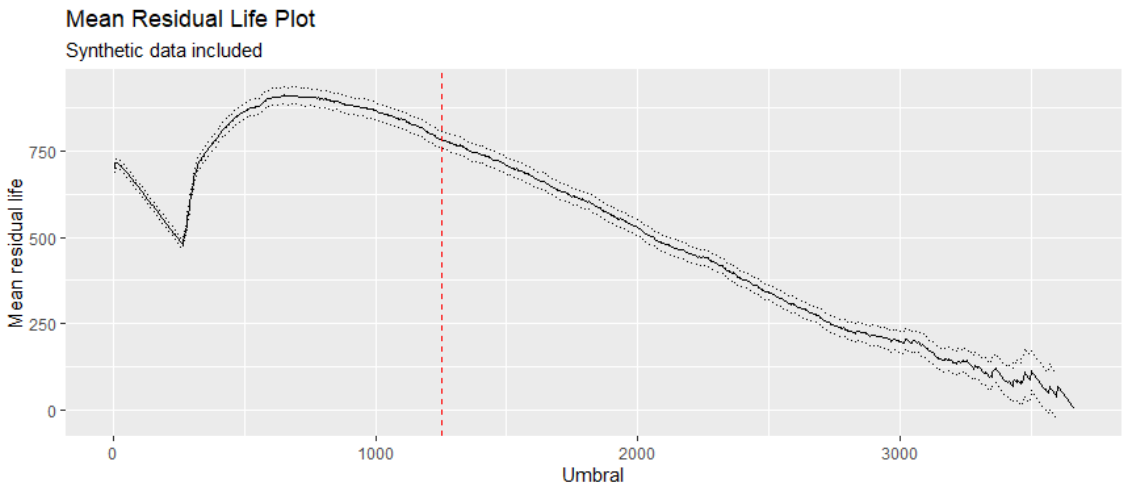
Figura 19. Gráficos de diagnóstico



Fuente: Elaboración propia

El nuevo umbral se aproxima empleando el gráfico de vida residual media de la serie de datos completa, esto es, combinando las observaciones reales con las generadas. En la figura 20 se observa que es alrededor del punto 1.250 en el que parece desaparecer la curvatura, la relación entre el umbral y la vida media residual se vuelve lineal, pudiendo observarse una clara tendencia negativa.

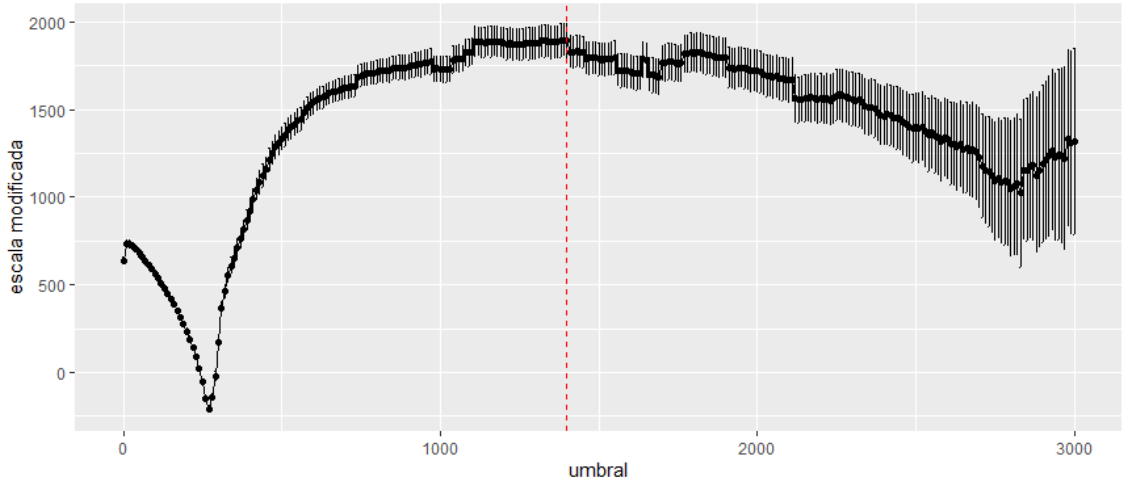
Figura 20. Gráfico de la vida residual media



Fuente: Elaboración propia

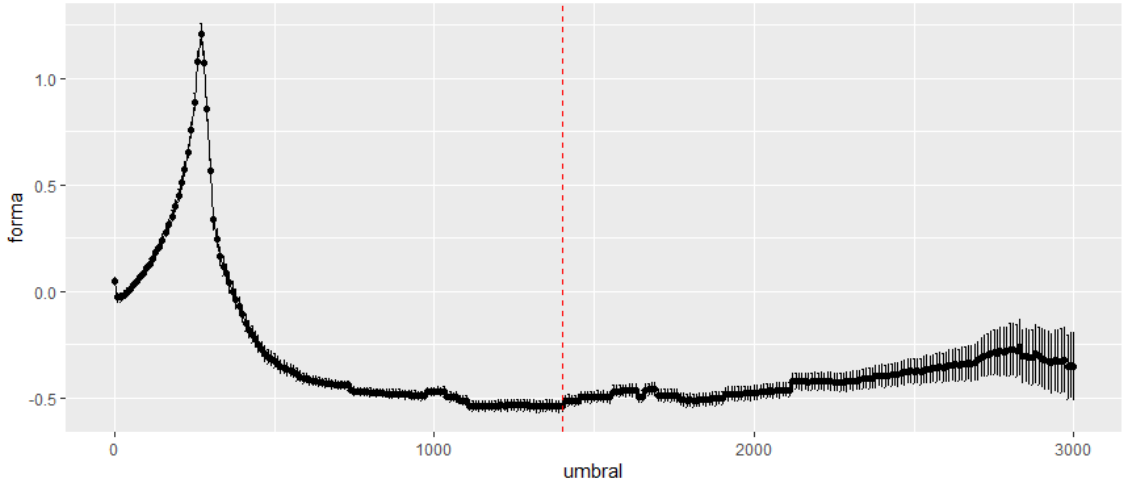
Para verificar este nivel con mayor precisión, las figuras 21 y 22 muestran la estabilidad de la estimación de los parámetros de escala σ y forma ξ, a diferentes valores de u. En ellos, el último punto en el que se aprecia una aparente consistencia es en el punto 1.400. Razón por la cual éste es el punto a partir del cual se consideran las observaciones como extremas.

Figura 21. Variación del comportamiento del parámetro σ*



Fuente: Elaboración propia

Figura 22. Variación del comportamiento del parámetro ξ



Fuente: Elaboración propia

Considerando un umbral de 1.400 los resultados del método GPD se observan en la tabla 10. Nuevamente, debe llamar la atención que la varianza se reduce en ambos casos si se la compara con el primer caso en el que el umbral era 160 y sólo se contaba con 25 observaciones.

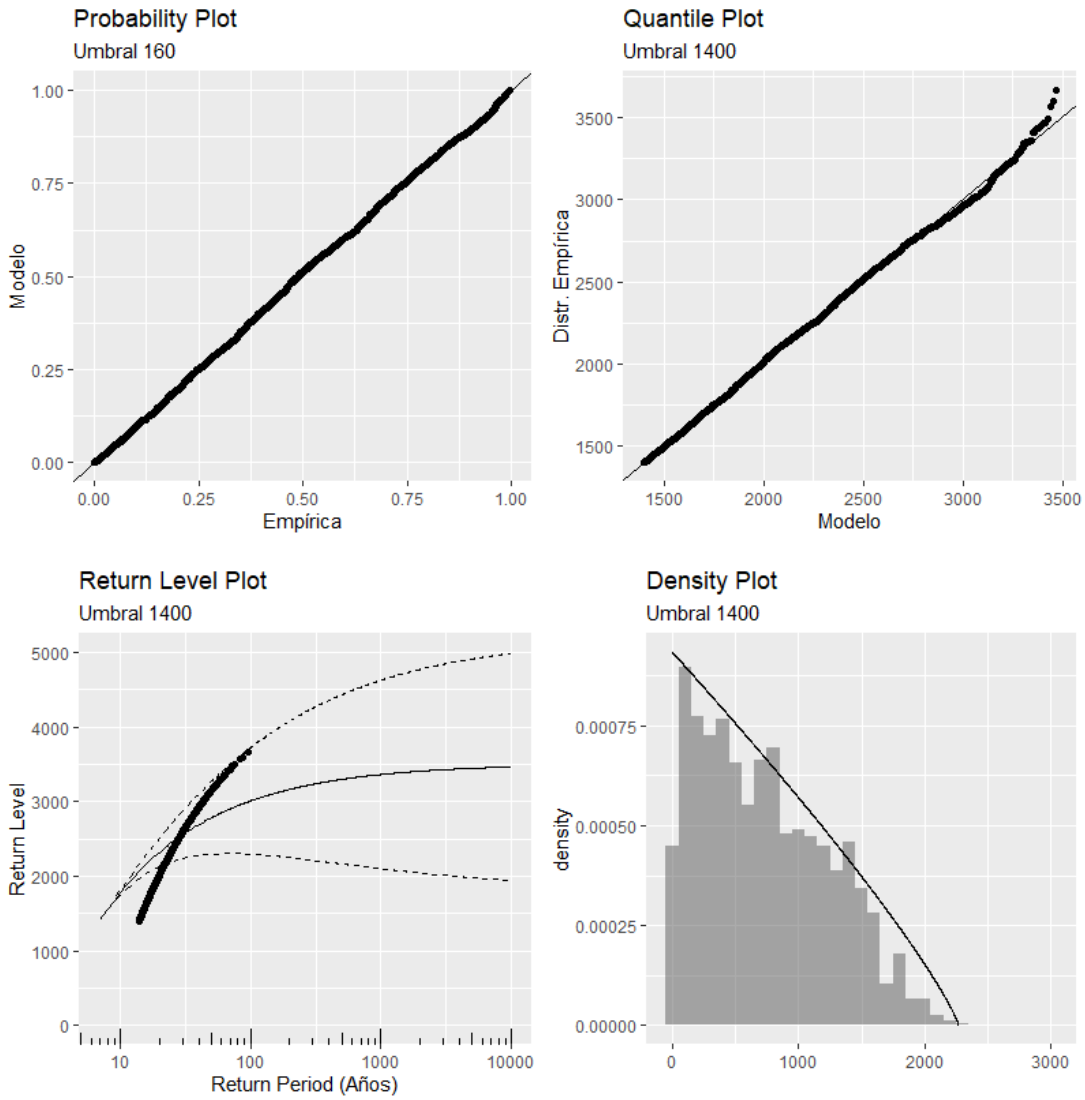
Tabla 10. Estimación de parámetros por máxima verosimilitud

Umbral (u)	**1400**
Número observaciones	1629
$\hat{\sigma}_g$	1139.7572
$Var(\hat{\sigma}_g)$	848.1689
$IC_{95\%}$	[1082.672 , 1196.838]
$\hat{\xi}_g$	-0.5406
$Var(\hat{\xi}_g)$	0.0002
$IC_{95\%}$	[-0.5696 , -0.5116]

Por último, la evaluación del ajuste de esta última modelización de la severidad se realiza a través de gráficos de diagnóstico de la figura 23. Como era de esperarse, lo primero que se observa son gráficos con una mayor concentración de observaciones. El gráfico de probabilidad presenta una correspondencia cuasi perfecta entre ambas funciones de distribución. De manera similar, la inversa de la función de distribución acumulada presenta un ajuste superior a los escenarios anteriores, aunque es cierto que se aleja de la

diagonal en la última sección de los datos más extremos. En el tercer gráfico se observa que la calibración actual sí contempla la ocurrencia de la observación real más extrema, quedando enmarcada dentro de los intervalos de confianza. Asimismo, se advierte un comportamiento asintótico alrededor de un valor finito, consecuencia del signo del parámetro de forma; mismo que, también, se puede constatar mirando la función de densidad.

Figura 23. Gráficos de diagnóstico



Fuente: Elaboración propia

6. CONCLUSIONES

Es evidente la presencia de observaciones extremas en la base de datos de estudio, aunque no lo es tanto la delimitación entre aquellas que pertenecen o no a esta categoría, menos aun cuando la extensión temporal no excede los tres años. Razón por la cual la modelización de los mismos se realizó a través de la distribución de Pareto generalizada y se descartó el método de Block Máxima.

En consecuencia, se modelaron 25 observaciones extremas con la distribución GPD, una vez seleccionado el umbral de 160, a través de la maximización de la función de verosimilitud. Esta metodología, aunque devuelve resultados interesantes, no recoge las observaciones más extremas dentro de los intervalos de extrapolación, siendo aquellas las que más daño pueden causar a una empresa aseguradora si se las excluye. No obstante, es una técnica cuya simplicidad sobresale por encima de sus debilidades, por la facilidad con la que puede ser puesta en práctica.

Por otro lado, al emplear técnicas machine learning como las redes neuronales, específicamente las redes generativas adversarias o vainilla GAN, se obtienen resultados importantes en cuanto al grado de precisión en el que la ANN aprende la distribución que subyace a los datos reales, inclusive cuando los datos le son dados sin clasificación previa entre siniestros masa y siniestros punta (escenario en el que fallan otras distribuciones teóricas si se compara el valor de KL). Sin embargo, la fragilidad de este método radica en la falta de libertad con la que cuenta la aproximación del mismo, puesto que se limita a capturar la distribución de los datos observados, omitiendo la posibilidad de que estos últimos no reflejen el comportamiento de observaciones futuras.

En este sentido, los mejores resultados se obtienen cuando ambas técnicas, GPD y GAN, trabajan de manera conjunta. Pues, la generación de datos sintéticos, muy similares a los reales, proveen de masa a la distribución GPD para determinar un umbral más alto y concentrarse en las observaciones menos frecuentes. La complementación de los datos de la red neuronal y la teoría de valores extremos deriva en una alternativa de modelización que fue capaz de capturar la observación real más inusual dejando, a su vez, espacio para la extrapolación.

Finalmente, cabe resaltar que los resultados de la red se pueden mejorar si se la entrena durante más tiempo, aunque dependerá de la capacidad del ordenador, así como también, si se emplean variantes de las redes adversarias (como el CGAN o WGAN) más eficientes

y más robustas (aunque más complejas). Por último, invitar al lector a continuar con la exploración y divulgación de esta área de conocimiento sobre la aplicación de nuevas tecnologías en el campo asegurador, ya que actualmente el material es limitado y poco difundido, especialmente si se refiere a las redes adversarias cuyo fuerte se centra en la generación de imágenes.

7. REFERENCIAS

ADESINA, O., ADELEKE, I., & OLADEJI, T. (2018). Using extreme value theory to model insurance risk of Nigeria's motor industrial class of business.

ALVES, F., & NEVES, C. (2011). Extreme Value Distributions. *International Encyclopedia of Statistical Science,* 493–496. https://doi.org/doi:10.1007/978-3-642-04898-2_246

AYAZ, M., AMMAD-UDDIN, M., SHARIF, Z., MANSOUR, A. & AGGOUNE, E. (2019). Internet-of-Things (IoT) based smart agriculture: Towards making the fields talk. *IEEE Access PP*, *99*. https://doi.org/10.1109/ACCESS.2019.

AYUSO, M., GUILLEN, M., & PERCH, J. (2017). Improving automobile insurance ratemaking using telematics: incorporating mileage and driver behavior data. *Transportation, 46*(3), 735-752. https://doi.org/10.1007/s11116-018-9890-7

BEIRLANT, J., MATTHYS, G., & DIERCKX, G. (2001) Heavy Tailed Distributions and Rating. *ASTIN Bulletin, 31*(1), 41-62. https://doi.org/10.2143/AST.31.1.993

CAI, Y., & HAMES, D. (2011). Minimum sample size determination for generalized extreme value distribution. *Communications in Statistics - Simulation and Computation, 40*(1), 87-98. https://doi.org/10.1080/03610918.2010.530368

CHARRAS-GUIDO, M., & LEZAUD, P. (2013). Extreme value analysis: An introduction. *Journal de la Société Française de Statistique, 154*(2). https://hal-enac.archives-ouvertes.fr/hal-00917995

CHEN, J., LEI, X., ZHANG, L., & PENG, B. (2015) Using Extreme Value Theory Approaches to Forecast the Probability of Outbreak of Highly Pathogenic Influenza in Zhejiang, China. *Plos One, 10*(2). https://doi.org/10.1371/journal.pone.0118521

CHI-SQUARE GOODNESS-OF-FIT TEST. (s.f.). Engineering statistics handbook. Recuperado de https://www.itl.nist.gov/div898/handbook/eda/section3/eda35f.htm

CIAMPI A., & LECHEVALLIER Y. (2007) Statistical Models and Artificial Neural Networks: Supervised Classification and Prediction Via Soft Trees. In: Auget JL., Balakrishnan N., Mesbah M., & Molenberghs G. (eds). Advances in Statistical Methods for the Health Sciences. Statistics for Industry and Technology. Birkhäuser Boston.

COLES, S. (2001). *An introduction to statistical modeling of extreme values*. London, Springer-Verlang.

COLES, S., CHU, H., & GREENLAND, S. (2014). Maximum likelihood, profile likelihood, and penalized likelihood: a primer. *American journal of epidemiology, 179*(2), 252–260. https://doi.org/10.1093/aje/kwt245

DUTANG, C. AND CHARPENTIER, A. (2019). Package "CASdatasets". Paquete de R versión 1.0-10.

ELVIDGE, S., & ANGLING, M. J. (2018). Using extreme value theory for determining the probability of Carrington-like solar flares. *Space Weather, 16*, 417–421. https://doi.org/10.1002/2017SW001727

EMBRECHETS, P., KLÜPPELBERG, C., & MIKOSCH, T. (1996). *Modelling extremal events for insurance and finance*. Springer-Verlag Berlin Heidelberg.

EMBRECHETS, P. (2000). Extreme value theory: Potential and limitations as an integrated risk management tool. *Derivatives Use, Trading & Regulation, 6*(1), 449-456. https://people.math.ethz.ch/~embrecht/ftp/evtpot.pdf

FACKLER, M. (s.f.). Rating without data – how to estimate the loss frequency of loss-free risks. https://www.actuaries.org/ASTIN/Colloquia/Helsinki/Papers/S7_8_Fackler.pdf

FRANCIS, A., KRISHNAMURTHY, G., KUMAR, M., KUTE, A., & KAUL, S. (2015). Next-generation insurance: Tapping into the intelligence of smart homes.

Cognizant. https://www.cognizant.com/InsightsWhitepapers/next-generation-insurance-tapping-into-the-intelligence-of-smart-homes-codex1411.pdf

GALLO, C. (2015). Artificial neural networks tutorial. Encyclopedia of information science and technology. USA, IGI Global.

GENÇAY, R., & SELÇUK, F. (2004). Extreme value theory and value-at-risk: Relative performance in emerging markets. *International Journal of Forecasting, 20*(2). https://doi.org/10.1016/j.ijforecast.2003.09.005

GHOSH, S. AND RESNICK, S. (2010). A discussion on mean excess plots. *Stochastics Processes and their Applications, 120*(8), 1492 – 1517. https://doi.org/10.1016/j.spa.2010.04.002

GILES, D., & FENG, H. (2009). Bias-corrected maximum likelihood estimation of the parameters of the Generalized Pareto Distribution. *Communications in Statistics-Theory and Methods, 45*(902). https://doi.org/10.1080/03610926.2014.887104

GILLI, M., & KËLLEZI, E. (2006). An application of extreme value theory for measuring financial risk. *Computational Economics, 21*(1) 1-23. https://doi.org/10.1007/s10614-006-9025-7

GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A., & BENGIO, Y. (2014). Generative adversarial nets. In Proceedings of the 27th International Conference on Neural Information Processing Systems, 2. USA, MIT Press. https://arxiv.org/pdf/1406.2661.pdf

GOODFELLOW, I., BENGIO, Y., & AARON COURVILLE. (2016). *Deep Learning*. The MIT Press.

HAYAT, A., & DILLON. (2018). Building a simple Generative Adversarial Network (GAN) using TensorFlow. Paperspace. Recuperado de https://blog.paperspace.com/ implementing-gans-in-tensorflow/

IBN MUSAH, A., DU, J., SALAH, H., & ABDUL-RASHEED, A. (2018). The asymptotic decision scenarios of an emerging stock exchange market: Extreme Value Theory and Artificial Neural Network. *Risks, 6*(132). https://doi.org/10.3390/risks6040132

KOHLI, S., MIGLANI, S., & RAPARIYA, R. (2014). Basics of artificial neural network. *International journal of computer science and mobile computing, 3*(9), 745 – 751. https://ijcsmc.com/docs/papers/September2014/V3I9201465.pdf

KUKREJA, H., BHARATH, N., SIDDESH, C. AND KULDEEP, S. (2016). An Introduction to artificial neural network. *International journal of advance research and innovative ideas in education, 1*(5). https://www.researchgate.net/publication/ 319903816_AN_INTRODUCTION_ TO_ARTIFICIAL_NEURAL_NETWORK

KUMAR, P, & SHARMA, P. (2014). Artificial neural networks-A study. *International journal of emerging engineering research and technology, 2*(2), 143 – 148. http://www.ijeert.org/pdf/v2-i2/24.pdf

KUO, K. (2019). Generative Synthesis of Insurance Datasets. arxiv:1912.02423v1

LECUN, Y., BENGIO, Y. & HINTON, G. (2015). Deep Learning. *Nature, 521*, 436 - 444. https://doi.org/10.1038/nature14539

MAKIN, J. (2006). Backpropagation. http://www.cs.cornell.edu/courses/cs5740/ 2016sp/resources/backprop.pdf

MAKKONEN, L. (2008). Problems in the extreme value analysis. *Structural safety, 30*(5), 405 – 419. https://doi.org/10.1016/j.strusafe.2006.12.001

MAKKONEN, L., & TIKANMÄKI, M. (2019). An improved method of extreme value analysis. Journal of Hydrology X, 2. https://doi.org/10.1016/j.hydroa.2018.100012

MCNEIL, A. (1997). Estimating the tails of loss severity distributions using Extreme Value Theory. *ASTIN Bulletin, 27*(1), 117-137. https://doi.org/10.2143/AST.27.1.563210

MCNEIL, A., & SALADIN, T. (1997). The Peaks over Thresholds Method for Estimating High Quantiles of Loss Distributions. Proceedings of XXVIITH International ASTIN colloquium. https://pdfs.semanticscholar.org/1fa8/aa18445a33b0d10da8868f668b73ed18e93a.pdf

MINKAH, R. (2016). An application of extreme value theory to the management of a hydroelectric dam. *SpringerPlus, 5,* 96. https://doi.org/10.1186/s40064-016-1719-2

MOHEY-DEEN, Z., & ROSEN, R. (2018). The risks of pricing new insurance products: The case of long-term care. *Chicago Fed Letter, 397*. https://doi.org/10.21033/cfl-2018-397

NASH, C. (s.f.). Create data from random noise with Generative Adversarial Networks. Toptal. https://www.toptal.com/machine-learning/generative-adversarial-networks

PATKI, N., WEDGE,R., & VEERAMACHANENI, K. (2016). The synthetic data vault. In Proc. IEEE International Conference on Data Science and Advanced Analytics (DSAA), 399 - 410. https://doi.org/10.1109/DSAA.2016.49

PARK, N., MOHAMMADI, M., GORDE, K., JAJODIA, S., PARK, H., & KIM, Y. (2018). Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment, 11*(10), 1071–1083. https://doi.org/10.14778/3231751.3231757

OUTREVILLE, J. (1998). Theory and practice of insurance. Springer.

ROJAS, R. (1996). The Backpropagation Algorithm. *Neural Networks*. Berlin, Springer-Verlag.

RUDER, S. (2017). An overview of gradient descent optimization algorithms. Insight Centre for Data Analytics. arXiv:1609.04747v2.

SARLE, W. (1994). Neural networks and statistical models.

SASSANI, B., JAMIL, N., VILLAPOL, M., MAILK, A., & SREMATH, S. (2020). FireNot – An IoT based fire alerting system: design and implementation. Bajwa, I., Sibalija, T., & Jawawi, D. Intelligent Technologies and Applications: Second International Conference, INTAP 2019. Springer Nature.

SERRENHO, T., & BERTOLDI, P. (2019). Smart home and appliances: State of the art. JRC Technical Reports. Luxemburgo: Publications Office of the European Union.

SINTEF. (2013). Big Data, for better or worse: 90% of world's data generated over last two years. *ScienceDaily*. Recuperado de https://www.sciencedaily.com/releases/ 2013/05/ 130522085217.htm

STORVIK, G. (2011). Numerical optimization of likelihoods: Additional literature for STK2120. University of Oslo. https://www.mn.uio.no/math/tjenester/kunnskap/ kompendier/ num_opti_likelihoods.pdf

VALANDRO, I. (2019). Future Advanced system for an on-demand Insurance Reliable product based on driving behaviour analysis. Project FAIR: FAIR. https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5c0216eba&appId=PPGMS

VIERIA, A. AND RODRIGUEZ ZAURIN, J. (2019). Generative Adversarial Networks (GANs) for Claim Frequency Prediction Models. https://insurancedatascience.org /downloads/Zurich2019/Abstracts_Insurance_Data_Science_Conference_2019.pdf

YANG, H., KUMARA, S., BUKKAPATNAM, S., & TSUNG, F. (2019): The internet of things for smart manufacturing: A review. *IISE Transactions, 51*(11), 1190-1216. https://doi.org/10.1080/24725854.2018.1555383

WANG, S. (2017). Generative adversarial networks (GAN): A gentle introduction. https://www.researchgate.net/publication/316382604_Generative_Adversarial_Networks_GAN_A_Gentle_Introduction_UPDATED

XU, J., LI, H., & ZHOU, S. (2015). An Overview of Deep Generative Models. *IETE technical review*, *32*(2), 131 – 139. https://doi.org/10.1080/02564602.2014.987328

XU, L. & VEERAMACHANENI, K. (2018). Synthesizing Tabular Data using Generative Adversarial Networks. arxiv:1811.11264

8. ANEXOS

ANEXO 1

Distribución	*Estadístico*	*Valor Crítico*	*P-valor*
Alpha	5,03E+02	2,30E+02	0,0000
Anglit	8,92E+04	2,31E+02	0,0000
Arcsine	1,70E+04	2,31E+02	0,0000
Beta	1,63E+10	2,29E+02	0,0000
Betaprime	2,39E+02	2,29E+02	0,0178
Bradford	7,84E+04	2,30E+02	0,0000
Burr	2,30E+02	2,29E+02	0,0446
Cauchy	1,54E+03	2,31E+02	0,0000
Chi	1,84E+07	2,30E+02	0,0000
Chi2	1,09E+03	2,30E+02	0,0000
Cosine	7,58E+04	2,31E+02	0,0000
Dgamma	7,38E+11	2,30E+02	0,0000
Dweibull	2,36E+10	2,30E+02	0,0000
Erlang	4,06E+12	2,30E+02	0,0000
Expon	9,91E+07	2,31E+02	0,0000
Exponnorm	1,04E+08	2,30E+02	0,0000
Exponweib	5,57E+08	2,29E+02	0,0000
Exponpow	7,60E+02	2,30E+02	0,0000
F	2,39E+02	2,29E+02	0,0178
Fatiguelife	2,53E+08	2,30E+02	0,0000
Fisk	6,28E+02	2,30E+02	0,0000
Foldcauchy	5,01E+02	2,30E+02	0,0000
Foldnorm	4,06E+12	2,30E+02	0,0000
Frechet_R	2,08E+09	2,30E+02	0,0000
Frechet_L	6,27E+06	2,30E+02	0,0000
Genlogistic	2,31E+09	2,30E+02	0,0000
Genpareto	5,22E+02	2,30E+02	0,0000
Gennorm	6,34E+06	2,30E+02	0,0000
Genexpon	9,91E+07	2,27E+02	0,0000
Genextreme	2,38E+03	2,30E+02	0,0000
Gausshyper	6,72E+02	2,26E+02	0,0000
Gamma	2,10E+13	2,30E+02	0,0000
Gengamma	1,17E+07	2,29E+02	0,0000
Genhalflogistic	1,02E+10	2,30E+02	0,0000
Gilbrat	5,67E+08	2,31E+02	0,0000
Gompertz	1,62E+12	2,30E+02	0,0000
Gumbel_R	1,48E+10	2,31E+02	0,0000
Gumbel_L	8,09E+04	2,31E+02	0,0000
Halfcauchy	4,73E+02	2,31E+02	0,0000
Halflogistic	6,12E+09	2,31E+02	0,0000
Halfnorm	4,06E+12	2,31E+02	0,0000
Halfgennorm	6,21E+05	2,30E+02	0,0000

Distribución	*Estadístico*	*Valor Crítico*	*P-valor*
Hypsecant	1,52E+10	2,31E+02	0,0000
Invgamma	1,99E+02	2,30E+02	0,4284
Invgauss	1,21E+04	2,30E+02	0,0000
Invweibull	2,02E+02	2,30E+02	0,3642
Johnsonsb	1,03E+06	2,29E+02	0,0000
Johnsonsu	2,01E+03	2,29E+02	0,0000
Ksone	1,62E+05	2,30E+02	0,0000
Kstwobign	1,66E+11	2,31E+02	0,0000
Laplace	8,89E+08	2,31E+02	0,0000
Levy	1,18E+02	2,31E+02	1,0000
Levy_L	2,35E+06	2,31E+02	0,0000
Logistic	4,86E+10	2,31E+02	0,0000
Loggamma	8,32E+12	2,30E+02	0,0000
Loglaplace	2,42E+02	2,30E+02	0,0145
Lognorm	9,55E+03	2,30E+02	0,0000
Lomax	5,21E+02	2,30E+02	0,0000
Maxwell	7,73E+10	2,31E+02	0,0000
Mielke	2,05E+02	2,29E+02	0,2983
Nakagami	1,17E+09	2,30E+02	0,0000
Ncx2	2,36E+05	2,29E+02	0,0000
Ncf	2,17E+02	2,27E+02	0,1257
Nct	1,67E+02	2,29E+02	0,9256
Norm	8,12E+12	2,31E+02	0,0000
Pareto	1,35E+02	2,30E+02	0,9997
Pearson3	2,33E+11	2,30E+02	0,0000
Powerlaw	7,94E+02	2,30E+02	0,0000
Powerlognorm	1,33E+03	2,29E+02	0,0000
Rdist	8,12E+12	2,30E+02	0,0000
Rayleigh	3,94E+10	2,31E+02	0,0000
Rice	3,94E+10	2,30E+02	0,0000
Recipinvgauss	6,80E+10	2,30E+02	0,0000
Semicircular	9,83E+04	2,31E+02	0,0000
T	6,81E+02	2,30E+02	0,0000
Triang	7,83E+04	2,30E+02	0,0000
Truncexpon	1,07E+05	2,30E+02	0,0000
Tukeylambda	7,39E+02	2,30E+02	0,0000
Uniform	1,57E+05	2,31E+02	0,0000
Vonmises	1,97E+32	2,30E+02	0,0000
Vonmises_Line	4,38E+08	2,30E+02	0,0000
Wald	4,06E+12	2,31E+02	0,0000
Weibull_Min	2,08E+09	2,30E+02	0,0000
Weibull_Max	6,27E+06	2,30E+02	0,0000

ANEXO 2

GPD - Código R: Descripción de los datos

```
library(CASdatasets)
library(ggplot2)
library(moments)
library(gridExtra)

data(fremarine)
x<-fremarine$ClaimPaid

##### Excess Loss Function ####
#Array que almacena MEAN EXCESS LOSS
me <- vector()
xaxis <- vector()
#Número de observaciones
n<-length(x)
#Ordena serie en orden descendiente
x_ord <- x[order(x)]
#
for (k in 1:(n-1)) {
  add <- 0
  idx <- x_ord[n-k]
  for (j in 1:k) {
    add <- add + x_ord[n-j+1] - idx
  }
  xaxis[k]<-idx
  me[k]<-add/k
}
#Ordena para graficar
xaxis_ord <- xaxis[order(xaxis, decreasing = T)]
me_ord <- me[order(me, decreasing = T)]
#Plot
ggplot()+
  geom_point(aes(x=xaxis_ord,y=me_ord)) +
  labs(x = "Umbral",y="Mean Excess") + xlim(c(0,2000))

#####Exponential quantile plot#####
n<-length(x)
x_ord <- x[order(x)]
x_coord <- vector()
y_coord <- vector()
for (j in 1:(n-1)) {
  x_coord[j]<--log(j/(n+1))
  y_coord[j]<-x_ord[n-j+1]
}
#Ordena para graficar
x_coord_ord<-x_coord[order(x_coord, decreasing = T)]
y_coord_ord<-y_coord[order(y_coord, decreasing = T)]

#Plot
ggplot()+
  geom_point(aes(x=x_coord_ord,y=y_coord_ord)) +
  labs(y = "Empirical quantile",x="Theorical quantile")


#####Mean Residual Life Plot#####

range<-max(x)-min(x)
ticks <-1000
width <- range/ticks
residual <- vector()
var_residual <- vector()
xaxis <- vector()
for(i in cumsum(rep(width,ticks-1))){
  add <- sum(x[x>i]-i)
  nu <- length(x[x>i])
  residual<- c(residual, add/nu)
  var_residual <- c(var_residual, qnorm(0.975)*sd(x[x>i]-i)/sqrt(nu))
  xaxis <- c(xaxis,i)
}

ggplot()+
  geom_line(aes(x = xaxis, y = residual)) +
  geom_line(aes(x = xaxis, y = residual + var_residual), linetype =3) +
  geom_line(aes(x = xaxis, y = residual - var_residual), linetype = 3) +
  geom_vline(xintercept=200, color = "red") + #200
  geom_vline(xintercept=1925.175, color = "blue") +#1925.175
  labs(y="Mean residual life", x = "Umbral")
```

ANEXO 3

GPD - Código R: Método Newton-Raphson

```
fnc_gpd_v3 <- function(y, eps = 0.00001){
  k<-length(y)
  sigma <- mean(y)**2/var(y) #SCALE
  xi <- mean(y)/var(y) #SHAPE
  theta <- c(sigma, xi)
  eps = 10**-6/mean(y)
  diff<-1
  while (diff>eps) {
    theta.old <- theta
    #Derivada parcial respecto a sigma
    d_sigma <- sigma**-1*(-k+(1+xi)*sum(y/(sigma+xi*y)))
    #Derivada parcial respecto a xi
    d_xi <- xi**-2*sum(log(abs(1+xi*y/sigma)))-(1+xi**-1)*sum(y/(sigma+xi*y))

    #Segunda derivada parcial respecto a sigma
    d_sigma2 <- sigma**-2*(k-(1+xi)*sum(y*(2*sigma+xi*y)/((sigma+xi*y)**2)))
    #Segunda derivada parcial respecto a xi
    d_xi2<- 2*xi**-3*(xi*sum(y/(sigma+xi*y))-sum(log(abs(1+xi*y/sigma))))+(1+xi**-1)*sum(y**2/(sigma+xi*y)**2)
    #Segunda reivada parcial respecto a sigma y xi
    d2_sigmaxi <- xi**-1*((1+xi)*sum(y/(sigma+xi*y)**2)-sigma**-1*sum(y/(sigma+xi*y)))

    #Vector con las primeras derivadas parciales (score function)
    s<-c(d_sigma,d_xi)
    #Vector con las segundas derivadas parciales (observed information matrix)
    J<--1*matrix(c(d_sigma2,d2_sigmaxi,d2_sigmaxi,d_xi2),ncol = 2)

    theta <- theta + solve(J,s)
    sigma <- theta[1]
    xi <- theta[2]
    diff <- sum(abs(theta - theta.old))
    print(theta)
    print(diff)
  }
  #Maximum log likelihood
  l <- -k*log(sigma)-(1+1/xi)*sum(log(abs(1+xi*y/sigma)))
  return(list(theta = theta, Jbar = J, mle = l))
}
```

ANEXO 4

GPD - Código R: Verificación parámetros y estimación umbral

```
#Shape/Scale vs threshold plots
shape <- vector()
shape_var <- vector()
scale <- vector()
scale_var <- vector()
mle <- vector()


for (i in seq(from = 0, to = 250, by = 10)) {
  y <- x[x>i]-i
  f<-fnc_gpd_v3(y)
  scale_star <- f$theta[1] - f$theta[2]*i
  shape <- c(shape, f$theta[2])
  scale <- c(scale, scale_star)
  mle <- c(mle, f$mle)
  nabla <- matrix(c(1,-i))
  shape_var<-c(shape_var, inv(f$Jbar)[2,2])
  scale_var <- c(scale_var, (t(nabla)%*%inv(f$Jbar))%*%nabla)
}


df_scale <- data.frame(x = seq(0,250,10),
                       V = scale,
                       L = scale-qnorm(0.975)*sqrt(scale_var),
                       U = scale+qnorm(0.975)*sqrt(scale_var))

ggplot(df_scale, aes(x=x,y=V)) +
  geom_point() + geom_line() +
  geom_errorbar(aes(ymax = U, ymin = L), width = 2) +
  geom_vline(xintercept = 160, colour = "red", linetype = 2) +

  #ylim(c(-2000,2000)) +
  labs(x="umbral", y = "escala modificada")

df_shape <- data.frame(x = seq(0,250,10),
                       V = shape,
                       L = shape-qnorm(0.975)*sqrt(shape_var),
                       U = shape+qnorm(0.975)*sqrt(shape_var))

ggplot(df_shape, aes(x=x,y=V)) +
  geom_point() + geom_line() +
  geom_errorbar(aes(ymax = U, ymin = L), width = 2)  +
  geom_vline(xintercept = 160, colour = "red", linetype = 2) +

  labs(x="umbral", y = "forma")


#####Profile log likelihood#####

fnc_profile_gpd <- function(y, shape, eps = 0.00001){
  k <- length(y)
  sigma <- mean(y)**2/var(y)
  eps = 10**-6/mean(y)
  diff <- 1
  shape.i <- shape
  while(diff > eps){
    sigma.old <- sigma
    d_sigma <- sigma**-1*(-k+(1+shape.i)*sum(y/(sigma+shape.i*y)))
    s <- d_sigma
    d_sigma2 <- sigma**-2*(k-(1+shape.i)*sum(y*(2*sigma+shape.i*y)/((sigma+shape.i*y)**2)))
    J <- -1*d_sigma2
    sigma <- sigma + s/J
    diff <- abs(sigma - sigma.old)
    l <- -k*log(sigma)-(1+1/shape.i)*sum(log(1+shape.i*y/sigma))
  }

  return(list(mle = l, theta = c(sigma, shape.i), Jbar = J))
}

pShape <- vector()
pSigma <- vector()
pMLE <- vector()
ic <- vector()
for (i in seq(0.001, 2, by = 0.0001)) {
  f <- fnc_profile_gpd(y, i)
  pShape <- c(pShape, i)
  pSigma <- c(pSigma, f$theta[1])
  pMLE <- c(pMLE, f$mle)
  if (round(f$mle,2) == round(-179.3974,2)) { #u=200 -> -169.9245
    ic <- c(ic, i)
  }
}

ggplot()+
  geom_line(aes(x = pShape, y= pMLE))+
  geom_point(aes(x= pShape[pMLE == max(pMLE)], y = max(pMLE))) +
  geom_vline(xintercept = round(ic[1],3), color = "black", linetype = 2) +
  geom_vline(xintercept = round(ic[length(ic)],3), color = "black", linetype = 2) +
  geom_hline(yintercept =  max(pMLE), color = "black", linetype = 2 ) +
  geom_hline(yintercept =  max(pMLE) - 0.5*qchisq(0.95,1), color = "black", linetype = 2 ) +
  labs(x = "Forma", y = "Profile log Verosimilitud")

print(pShape[pMLE == max(pMLE)]);print(round(ic[1],3));print(round(ic[length(ic)],3));print(max(pMLE) -
0.5*qchisq(0.95,1))
```

```r
#Return level plot (return level vs likelihood)
fnc_profile_return <- function(data, u, x_m, N=10, eps = 0.001){
  x <- data$ClaimPaid
  y <- x[x>u]-u
  k <- length(y)
  N <- N
  n_y <- data %>%
    mutate(year= format(ReporDate,"%Y")) %>%
    filter(ClaimPaid>u) %>% group_by(year) %>%
    summarise(counts = n()) %>%
    summarise(average = mean(counts)) %>%
    as.numeric(.)
  m <- N*n_y
  xi <- mean(y)/var(y)
  zeta <- k/length(x)
  xm <- x_m
  eps = 10**-6/mean(y)
  diff <- 1
  while(diff > eps){
    xi.old <- xi
    d_xi <- -k*(xi**-1-((m*zeta)**xi*log(m*zeta))/((m*zeta)**xi-1))-(-xi**-2*sum(log(1+(y*((m*zeta)**xi-1))/(xm-
u)))+(1+xi**-1)*(sum((y*(m*zeta)**xi*log(m*zeta))/((xm-u)+y*((m*zeta)**xi-1)))))
    s <- d_xi
    d_xi2 <- k*(xi**-2+(((log(m*zeta))**2*(m*zeta)**xi)/((m*zeta)**xi-1))*(1-1/((m*zeta)**xi-1)))-xi**-
3*sum(log(1+(y*((m*zeta)**xi-1))/(xm-u)))+xi**-2*sum((y*(m*zeta)**xi*log(m*zeta))/((xm-u)+y*((m*zeta)**xi-
1)))+xi**-2*sum((y*(m*zeta)**xi*log(m*zeta))/((xm-u)+y*((m*zeta)**xi-1)))-(1+xi**-
1)*sum(((y*(log(m*zeta))**2*(m*zeta)**xi)/((xm-u)+y*((m*zeta)**xi-1)))*(1-(y*(m*zeta)**xi)/((xm-
u)+y*((m*zeta)**xi-1))))
    J <- -1*d_xi2
    xi <- xi + s/J
    diff <- abs(xi - xi.old)
    l <- -k*log((xi*(x_m-u))/((m*zeta)**xi-1))-(1+1/xi)*sum(log(1+(xi*y)/((xi*(x_m-u))/((m*zeta)**xi-1))))

  }

  return(list(mle = l, theta = c(xi, x_m), Jbar = J))
}
```

ANEXO 5

GPD - Código R: Gráficos de diagnóstico

```
##### MODEL CHECKING ######

# PROBABILITY PLOT

pp_plot <- function(y, scale, shape){
  y_ord <- sort(y)
  k <- length(y)
  x_axis <<- (1:k)/(k+1)
  H <- 1 - (1+shape*y_ord/scale)**(-1/shape)
  ggplot() +
    geom_point(aes(x = x_axis, y = H)) +
    geom_abline(slope = 1, intercept = 0) +
    labs(title = "Probability Plot", subtitle = "Umbral 160", x = "Empírica", y = "Modelo")
}

pp_plot(y, try$theta[1], try$theta[2])

# QUANTILE PLOT

qq_plot <- function(y, u, scale, shape){
  y_ord <<- sort(y)
  k <- length(y)
  p <- (1:k)/(k+1)
  H_inv <<- u + scale/shape*(p**-shape-1)
  ggplot() +
    geom_point(aes(x = sort(H_inv), y = y_ord)) +
    geom_abline(slope = 1, intercept = 0) +
    labs(title = "Quantile Plot", subtitle = "Umbral 160", x = "Modelo", y = "Distr. Empírica")
}

qq_plot(y,u,try$theta[1], try$theta[2])


# RETURN LEVEL PLOT
returnLevel_plot <- function(data, u, scale, shape, mle_obj){
  x <- data$ClaimPaid
  y <- x[x>u]-u
  k <- length(y)
  zeta <- k/length(x)
  m_emp_o<<-1/(zeta*(1+shape*((y)/scale)))**(-1/shape)
  n_y <- data %>%
    mutate(year= format(ReporDate,"%Y")) %>%
    filter(ClaimPaid>u) %>% group_by(year) %>%
    summarise(counts = n()) %>%
    summarise(average = mean(counts)) %>%
    as.numeric(.)
  #n_y <- 1
  J <- mle_obj$Jbar
  V <- J %>% cbind(0,.) %>% rbind(0,.)
  V[1,1]<-zeta*(1-zeta)/length(x)
  m <- seq(7,10000,by=1)*n_y #Start from 7, xm > u
  xm <- u + scale/shape*((m*zeta)**shape-1)
  d_zeta <- scale*m**shape*zeta**(shape-1)
  d_scale <- shape**-1*((m*zeta)**shape-1)
  d_shape <- -scale*shape**-2*((m*zeta)**shape-1)+scale*shape**-1*(m*zeta)**shape*log(m*zeta)
  nabla <- t(matrix(c(d_zeta, d_scale, d_shape), ncol = 3))
  xm_var <- diag(t(nabla) %*% (V %*% nabla))
  ggplot() +
    geom_line(aes(x=m, y = xm)) +
    geom_line(aes(x = m, y = xm + qnorm(.975)*sqrt(xm_var)/k), linetype = 2)+
    geom_line(aes(x = m, y = xm - qnorm(.975)*sqrt(xm_var)/k), linetype = 2)+
    geom_point(aes(x = sort(m_emp_o), y = sort(y)+u)) +
    scale_x_continuous(trans = "log10")+
    annotation_logticks(sides = "b") +
    labs(title = "Return Level Plot", subtitle = "Umbral 160", x = "Return Period (Años)", y = "Return Level")

}

returnLevel_plot(fremarine, 160,try$theta[1], try$theta[2], try)

# DENSITY PLOT
axis <- 0:4000
densv2<-(shape*((axis-u)/scale)+1)**(-(shape+1)/shape)*scale**-1
densv2<-scale**-1*(1+shape*((axis-u)/scale))**(-1-1/shape)

ggplot(mapping = aes(x = y))+
  geom_histogram(aes(x = y, y = ..density..), binwidth = 400, alpha = 0.5) +
  geom_line(aes(x=axis, y = densv2)) + ylim(c(0,0.0020)) +
  labs(title = "Density Plot", subtitle = "Umbral 160", x = "")
```

ANEXO 6

GAN - Código Python: Red generativa adversarial

```
import tensorflow as tf
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt

tf.reset_default_graph()


def sample_Z(m,n):
    return np.random.uniform(-1., 1., size = (m,n))


def generator(Z, hsize =  [16, 32, 128], reuse = False): #hsize: number of units in the hidden layers; reuse:
reuse same layers
    with tf.variable_scope("GAN/Generator", reuse = reuse):
        h1 = tf.layers.dense(Z, hsize[0], activation=tf.nn.leaky_relu) #layer structure (input nodes, output
nodes)
        h2 = tf.layers.dense(h1, hsize[1], activation=tf.nn.leaky_relu)
        h3 = tf.layers.dense(h2, hsize[2], activation=tf.nn.leaky_relu)
        out = tf.layers.dense(h3, 2) #2-dimensional vector, which corresponds to dimensions of real dataset
    return out


def discriminator(X, hsize=[16,32, 128], reuse = False):
    with tf.variable_scope("GAN/Discriminator", reuse = reuse):
        h1 = tf.layers.dense(X, hsize[0], activation=tf.nn.leaky_relu)
        h2 = tf.layers.dense(h1, hsize[1], activation=tf.nn.leaky_relu)
        h3 = tf.layers.dense(h2, hsize[2], activation=tf.nn.leaky_relu)
        h4 = tf.layers.dense(h3, 2) #2-dimensional vector to visualize trasnformed feature space in 2D
        out = tf.layers.dense(h4, 1) #Output is logit prediction. Logit function is the inverse of the sigmoid
function which is used to represent the logarithm of the odds (ratio of the probability of variable being 1 to
that of it being 0)
    return out, h4


X = tf.placeholder(tf.float32, [None,2])
Z = tf.placeholder(tf.float32, [None, 2])


G_sample = generator(Z)
r_logits, r_rep = discriminator(X)
f_logits, g_rep = discriminator(G_sample, reuse= True)

# Loss functions for generator and discriminator
disc_loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits=r_logits, labels =
tf.ones_like(r_logits)) + tf.nn.sigmoid_cross_entropy_with_logits(logits=f_logits, labels=
tf.zeros_like(f_logits)))
gen_loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits= f_logits, labels=
tf.ones_like(f_logits)))

# Define optimizers
gen_vars = tf.get_collection(tf.GraphKeys.GLOBAL_VARIABLES, scope = "GAN/Generator")
disc_vars = tf.get_collection(tf.GraphKeys.GLOBAL_VARIABLES, scope = "GAN/Discriminator")

gen_step = tf.train.AdamOptimizer(learning_rate = 0.001).minimize(gen_loss, var_list = gen_vars) #G Train step
disc_step = tf.train.AdamOptimizer(learning_rate = 0.001).minimize(disc_loss, var_list = disc_vars) #D Train step



def get_empirical_batch(train, m, seed = 0, log = False, transform = None):

    if log:
        train = train[train!=0]
        train = np.log(train)

    if transform == "maxAbsScaler":
        train = train/np.abs(train).max()

    if transform == "simpleScaler":
        train = (train - np.mean(train))/np.std(train)

    n = len(train)
    x_list = list(train)
    x_list.sort()
    Fx = []

    for x in x_list:
        i = sum(train <= x)
        Fx.append([x, i/n])

    np.random.seed(seed)
    out = [Fx[i] for i in np.random.choice(n, m)]

    return out


#####################################################################
#                              TEST                                 #
#                         WHOLE DATASET                             #
#####################################################################

saver = tf.train.Saver()
```

```
plt.style.use('ggplot')

sess = tf.Session()
tf.global_variables_initializer().run(session=sess)



data_train = data.ClaimPaid
batch_size = 64
nd_steps = 10
ng_steps = 10
data_dir = "C:/Users/adriv/OneDrive/Documents/uc3m/Segundo/Trabajo fin de máster/Output/"

x_plot = get_empirical_batch(data_train, 128, 8, log=True, transform="maxAbsScaler") #Seed = 8 captures most
extreme value

f = open(data_dir+"loss_logs.csv","w")
f.write("Iteration, Discriminator Loss, Generator Loss\n")


for i in range(1001):
    X_batch = get_empirical_batch(data_train, batch_size, i, log=True, transform="maxAbsScaler")
    Z_batch = sample_Z(batch_size, 2)

    for _ in range(nd_steps):
        _, dloss = sess.run([disc_step, disc_loss], feed_dict={X: X_batch, Z: Z_batch})
    rrep_dstep, grep_dstep = sess.run([r_rep, g_rep], feed_dict={X:X_batch, Z: Z_batch})

    for _ in range(ng_steps):
        _, gloss = sess.run([gen_step, gen_loss], feed_dict={Z: Z_batch})

    rrep_gstep, grep_gstep = sess.run([r_rep, g_rep], feed_dict={X: X_batch, Z: Z_batch})

    print("Iterations: %d\t Discriminator loss: %.4f\t Generator loss: %.4f"%(i, dloss, gloss))



    if i%10 == 0:
        f.write("%d,%f,%f\n"%(i,dloss, gloss))


    if i %100 ==0:
        plt.figure()
        g_plot = sess.run(G_sample, feed_dict={Z: Z_batch})
        xax = plt.scatter(list(zip(*x_plot))[0], list(zip(*x_plot))[1], s = 40, c = "black", alpha = 0.5 )
        gax = plt.scatter(g_plot[:,0], g_plot[:,1], s=40, c = "#e69f00", alpha = .6)

        plt.legend((xax, gax), ("Real Data", "Generated Data"), loc = "lower right")
        plt.title("Samples ar Iteration %d"%i)
        plt.tight_layout()
        plt.savefig(data_dir+"plots/iterations/raw/iteration_%d.png"%i)
        plt.close()


        plt.figure()
        g_plot = sess.run(G_sample, feed_dict={Z: Z_batch})
        xax =  plt.scatter(list(zip(*x_plot))[0], list(zip(*x_plot))[1], marker="o")
        gax =  plt.scatter(g_plot[:,0], g_plot[:,1], marker="o")

        plt.legend((xax, gax), ("Datos reales", "Datos generados"), loc = "lower right")
        plt.title("Muestras en la Iteración %d"%i)
        plt.tight_layout()
        plt.xlim(-0.002, 0.025)
        plt.ylim(0,1.)
        plt.savefig(data_dir+"plots/iterations/clean/iteration_%d.png"%i)
        plt.close()

        plt.figure()
        rrd = plt.scatter(rrep_dstep[:,0], rrep_dstep[:,1], alpha = 0.5)
        rrg = plt.scatter(rrep_gstep[:,0], rrep_gstep[:,1], alpha=0.5)
        grd = plt.scatter(grep_dstep[:,0], grep_dstep[:,1], alpha=0.5)
        grg = plt.scatter(grep_gstep[:,0], grep_gstep[:,1], alpha = 0.5)

        plt.legend((rrd, rrg, grd, grg), ("Real Data Before G step", "Real Data After G step",
                                          "Generated Data Before G step", "Generated Data After G step"))
        plt.title("Transformed Features ar Iteration %d"%i)
        plt.savefig(data_dir+"plots/features/transform/feature_transform_%d.png"%i)
        plt.close()

        plt.figure()

        rrdc = plt.scatter(np.mean(rrep_dstep[:,0]), np.mean(rrep_dstep[:,1]),s=100, alpha=0.5)
        rrgc = plt.scatter(np.mean(rrep_gstep[:,0]), np.mean(rrep_gstep[:,1]),s=100, alpha=0.5)
        grdc = plt.scatter(np.mean(grep_dstep[:,0]), np.mean(grep_dstep[:,1]),s=100, alpha=0.5)
        grgc = plt.scatter(np.mean(grep_gstep[:,0]), np.mean(grep_gstep[:,1]),s=100, alpha=0.5)

        plt.legend((rrdc, rrgc, grdc, grgc), ("Real Data Before G step","Real Data After G step",
                                "Generated Data Before G step","Generated Data After G step"))

        plt.title('Centroid of Transformed Features at Iteration %d'%i)
        plt.tight_layout()
        plt.savefig(data_dir+"plots/features/centroid/feature_transform_centroid_%d.png"%i)
        plt.close()

    if i % 1000 ==0:
        save_path = saver.save(sess, data_dir + "checkpoint/GAN_weights_step_%d.ckpt"%i)

f.close()
```

ANEXO 7

GAN - Código Python: Generación datos sintéticos

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()



from Vanilla_GAN2 import sample_Z, generator


# Generar muestas del modelo entrenado
tf.reset_default_graph()


gen_vars = tf.get_collection(tf.GraphKeys.GLOBAL_VARIABLES, scope = "GAN/Generator")
disc_vars = tf.get_collection(tf.GraphKeys.GLOBAL_VARIABLES, scope = "GAN/Discriminator")

Z_test = tf.placeholder(tf.float32, [None, 2])

G_sample_test = generator(Z_test)

saver = tf.train.Saver()
batch_size = 10_000
data_dir = "C:/Users/adriv/OneDrive/Documents/uc3m/Segundo/Trabajo fin de máster/Output/"

with tf.Session() as sess:
    saver.restore(sess, data_dir + "checkpoint/GAN_weights_step_%d.ckpt"%80_000)
    np.random.seed(0)
    Z_batch_test = sample_Z(batch_size, 2)
    generated_data = sess.run(G_sample_test, feed_dict={Z_test: Z_batch_test})

# Reverse maxabsscaler transformation. Transform generated data to original scale
reshape_data = generated_data[:,0].reshape(-1,1)
generated_data_back = reshape_data * np.abs(np.log(data.ClaimPaid[data.ClaimPaid!=0])).max()
generated_data_back = np.exp(generated_data_back)
```

ANEXO 8

GAN - Código Python: KL

```
# Comprobar la calidad de los datos sintéticos
import pandas as pd
from scipy.stats import norm
from scipy.interpolate import interp1d

# Same range for generated and real data
def pre_kernel(p,q):
    p_array = np.asarray(p)
    q_array = np.asarray(q)

    p_min = p_array.min()
    q_min = q_array.min()
    p_max = p_array.max()
    q_max = q_array.max()

    minimum = min(p_min, q_min)
    maximum = max(p_max, q_max)
    # Add minimmum
    if minimum == p_min:
        q_array = np.append(minimum, q_array)
    else:
        p_array = np.append(minimum, p_array)
    # Add maximum
    if maximum == p_max:
        q_array = np.append(q_array, maximum)
    else:
        p_array = np.append(p_array, maximum)

    return p_array, q_array


# Calcula bandwidth
def thumb_bandwidth(x):
    n = len(x)
    sigma = np.std(x)
    IQR = np.quantile(x, 0.75) - np.quantile(x, 0.25)
    h = 0.9 * min(sigma, IQR)* n**(-1/5)
    return h

# Aproxima la distribucion de probabilidad
def kernel(x):
    x_array = np.asarray(x)
    x_array.sort()
    n = len(x)
    h = thumb_bandwidth(x)
    density = []

    for element_0 in x_array:
        sumation = sum(norm.pdf((element_0 - x_array)/h))/(n*h)
        density.append([element_0, sumation])

    return np.asarray(density)


# Calcula Kullback-Leibler divergence
def KL(p,q, umbral=0):
    # Kernel calculation
    p_pre, q_pre = pre_kernel(p,q)
    p_density = kernel(p_pre)
    q_density = kernel(q_pre)
    # Linear interpolation
    p_x = p_density[:,0].squeeze()
    p_y =  p_density[:,1].squeeze()
    q_x = q_density[:,0].squeeze()
    q_y =  q_density[:,1].squeeze()
    p_f = interp1d(p_x, p_y)
    q_f = interp1d(q_x, q_y)

    minimum = min(p_pre.min(), q_pre.min())
    maximum = p_pre.max()
    rango = list(range(int(minimum)+ 1, int(maximum), 10))
    divergence = np.sum(p_f(rango) * (np.log(p_f(rango)) - np.log(q_f(rango))))
    return divergence


#Comparación GAN vs real
a = KL(data.ClaimPaid[data.ClaimPaid>160], generated_data_back) #neural_network
print("KL: %f"%a)


#Exportar datos sintéticos para ser utilizados en el método GPD (R)
f = open(data_dir+"Synthetic_data.csv","w")
f.write("Synthetic values\n")
for rv in generated_data_back:
    f.write("%s\n" %rv)
f.close()


df = pd.DataFrame(generated_data_back)
df.applymap(str).replace(r"\.",",",regex=True)
df.to_csv(data_dir+"Synthetic_data.csv", sep = ";", index= False)

file = data_dir+"Synthetic_data.csv"
np.savetxt(file, generated_data_back, fmt= "%s")
```

ANEXO 9

GAN - Código Python: Ajuste distribuciones teóricas

```
import warnings
import numpy as np
import pandas as pd
import scipy.stats as st
import statsmodels as sm
import matplotlib
import matplotlib.pyplot as plt

################################################################################
#                                   FIT                                        #
################################################################################

datas = data.ClaimPaid[data.ClaimPaid!=0]

data_dir = "C:/Users/adriv/OneDrive/Documents/uc3m/Segundo/Trabajo fin de máster/Output/"

significance = 0.05

f = open(data_dir+"theoretical_fit.csv","w")
f.write("Distribution,Test statistic,  P-value, Critic Value\n")

breaks = 200
O_counts, bins = np.histogram(datas, bins=breaks)  # y has counts and x the bins
y, x = np.histogram(datas, bins=breaks)
x = (x + np.roll(x, -1))[:-1] / 2.0  # Encuentra el punto medio de los histogramas
n = len(datas)

DISTRIBUTIONS = [
    st.alpha, st.anglit, st.arcsine, st.beta, st.betaprime, st.bradford, st.burr, st.cauchy, st.chi, st.chi2,
st.cosine,
    st.dgamma, st.dweibull, st.erlang, st.expon, st.exponnorm, st.exponweib, st.exponpow, st.f, st.fatiguelife,
st.fisk,
    st.foldcauchy, st.foldnorm, st.frechet_r, st.frechet_l, st.genlogistic, st.genpareto, st.gennorm, st.genexpon,
    st.genextreme, st.gausshyper, st.gamma, st.gengamma, st.genhalflogistic, st.gilbrat, st.gompertz, st.gumbel_r,
    st.gumbel_l, st.halfcauchy, st.halflogistic, st.halfnorm, st.halfgennorm, st.hypsecant, st.invgamma,
st.invgauss,
    st.invweibull, st.johnsonsb, st.johnsonsu, st.ksone, st.kstwobign, st.laplace, st.levy, st.levy_l,
    st.logistic, st.loggamma, st.loglaplace, st.lognorm, st.lomax, st.maxwell, st.mielke, st.nakagami, st.ncx2,
st.ncf,
    st.nct, st.norm, st.pareto, st.pearson3, st.powerlaw, st.powerlognorm, st.powernorm, st.rdist, st.reciprocal,
    st.rayleigh, st.rice, st.recipinvgauss, st.semicircular, st.t, st.triang, st.truncexpon, st.truncnorm,
    st.tukeylambda,
    st.uniform, st.vonmises, st.vonmises_line, st.wald, st.weibull_min, st.weibull_max
]

parametros = {}
distribuciones = []
p_value = []
t_stadistico  = []
RMSE = []

distribution = st.alpha
for distribution in DISTRIBUTIONS:

    fitted = distribution.fit(datas)
    loc = fitted[-2]
    scale = fitted[-1]
    arg = fitted[:-2]

    dof = breaks - len(fitted) - 1

    print(distribution)

    with warnings.catch_warnings():
        warnings.filterwarnings('ignore')
        # Theoretical probabilities (Expected probabilities)
        E_prob = distribution.cdf(bins, loc = loc, scale = scale, *arg)
        E_prob = (np.roll(E_prob,-1) - E_prob)[:-1]
        # Theoretical counts (Expected counts)
        E_counts = E_prob * n
        # Chi square value
        chi2_crit = np.power(O_counts-E_counts,2)/E_counts
        chi2_crit = chi2_crit[chi2_crit!=np.inf]
        chi2_crit = chi2_crit[~np.isnan(chi2_crit)]
        t_stat = sum(chi2_crit)
        chi2_crit =  1 - st.chi2.cdf(sum(chi2_crit), dof)
        critic_value  = st.chi2.ppf(1 - significance, dof)

        # RMSE
        pdf = distribution.pdf(x, loc=loc, scale=scale, *arg)
        rmse = np.sqrt(np.sum(np.power(y - pdf, 2.0))/n)

        f.write("%s,%f, %f,%f\n"%(distribution.name, t_stat, chi2_crit, critic_value))

        distribuciones.append(distribution.name)
        parametros[distribution.name] = fitted
        print(parametros)
        p_value.append(np.round(chi2_crit,4))
        RMSE.append(np.round(rmse,5))
        t_stadistico.append(np.round(t_stat,4))

f.close()

for entry in zip(distribuciones, t_stadistico, p_value, RMSE):
```

```
    print(entry)


#########################################################################
#                     COMPARACIÓN DISTRIBUCIONES                        #
#########################################################################

f = open(data_dir+"KL_divergence.csv","w")
f.write("Distribution,KL\n")


DISTRIBUTIONS = [st.invgamma, st.invweibull, st.levy, st.mielke,st.ncf, st.nct,
                 st.pareto]

samples = dict()
KL_divergence = dict()


for distribution in DISTRIBUTIONS:
    sample_size = 10_000
    loc = parametros[distribution.name][-2]
    scale = parametros[distribution.name][-1]
    arg = parametros[distribution.name][:-2]
    np.random.seed(1)
    sample = distribution.rvs(*arg, loc = loc, scale = scale, size = sample_size )
    samples[distribution.name] = sample
    KL_entry = KL(datas, sample)
    KL_divergence[distribution.name] = KL_entry
    f.write("%s,%f\n"%(distribution.name, KL_entry))

    print(distribution.name)

f.close()


#Comparación GPD vs real
Real_extreme = data.ClaimPaid[data.ClaimPaid>160]
RV_GPD = pd.read_csv("C:/Users/adriv/OneDrive/Documents/uc3m/Segundo/Trabajo fin de
máster/Data/Valores_aleatorios_GPD.csv")
KL_entry = KL(Real_extreme, RV_GPD)
print("KL extreme GPD: %f"%KL_entry)
```