UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

USO DE SERIES DIFUSAS PARA EL DESARROLLO DE MODELOS PREDICTIVOS DE LA DEMANDA DE ENERGÍA ELÉCTRICA

MEMORIA PARA OPTAR AL TITULO DE INGENIERO CIVIL ELÉCTRICO

JOSÉ MANUEL RUBIO CIENFUEGOS

PROFESOR GUÍA:
José Miguel Rubio León

MIEMBROS DE LA COMISIÓN:
Francisco Rivera Serrano
Santiago Bradford Vicuña

SANTIAGO DE CHILE
2022

USO DE SERIES DIFUSAS PARA EL DESARROLLO DE MODELOS PREDICTIVOS DE LA DEMANDA DE ENERGÍA ELÉCTRICA

El siguiente trabajo tiene como finalidad el desarrollo de un modelo de predicción de demanda de energía eléctrica mediante el uso de series difusas, donde se explorarán diferentes métodos que utilizan esta técnica para encontrar el sistema más preciso para realizar estas predicciones. Para ello se utiliza una base de datos pública que contiene información sobre el consumo de energía de España durante 4 años. Hay que destacar que ademas de la información sobre el consumo de energía, esta base, también, tiene datos sobre datos relacionados al consumo, como la generación de diversas fuentes de energía y el precio del mercado eléctrico.

Si bien la finalidad principal es utilizar la lógica difusa para la realización de modelos de predicción de la demanda de la energía eléctrica, para poder analizar su eficacia se debió comparar el rendimiento de este sistema con otros algoritmos que realizan este tipo de trabajo. Para ello se programaron modelos de *machine learning* los cuales han sido utilizados en trabajos previos a este, lo que constituye los modelos del estado del arte de esta investigación desarrollada.

Para poder realizar las predicciones por los modelos mencionados anteriormente, se tuvo que convertir los datos de la demanda energética en una serie temporal. Tras este punto, para aplicar los modelos de series difusas, esta información se tuvo que llevar al dominio de la lógica difusa, lo que se conoce como **universo de discurso**, donde las funciones que realizan este proceso se denominan **particiones**, con lo que se crean conjuntos de asociación lógica, para que de esta forma la información se convierta en valores de asociación a estos conjuntos generados. Una vez que la información se encuentra como su universo de discurso, los modelos aprenden como se comporta cada área (conjunto) y extraen reglas de estas a través de los patrones de las series temporales, donde estas reglas indican como se realiza la asociación de los datos a los del universo de discurso. El proceso de llevar la información al dominio difuso se llama **fuzzyficación**, y una vez que se ha aplicado la predicción a los datos deseados, entonces se debe llevar al dominio original (tiempo) de la información, es decir se debe realizar una **defuzzyficación**. Por otro lado, el uso de los modelos de *machine learning* no requirió los procesos mencionados anteriormente, simplemente, se procesaron las series temporales en el mismo dominio, convirtiendo la data en diversas series temporales que representaron diferentes períodos de consumo.

Para realizar el entrenamiento de los modelos mencionados, se utilizaron librerías de *machine learning* y de series difusas en python, donde la evaluación de los modelos se hizo a través de la comparación con otros sistemas ya implementados. En la literatura relacionada al tema, este fue el método base para analizar si el modelo propuesto realmente era efectivo, por lo que en este trabajo de memoria también se desarrollaron comparaciones con respecto al estado del arte. Todos estos resultados generaron utilizando un notebook de *Google Colaboratory*, utilizando el lenguaje *Python*. El código completo puede encontrarse en el siguiente enlace[1].

[1] Código utilizado en este trabajo: https://github.com/JossRubio/Prediccion_demanda_energetica.git

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

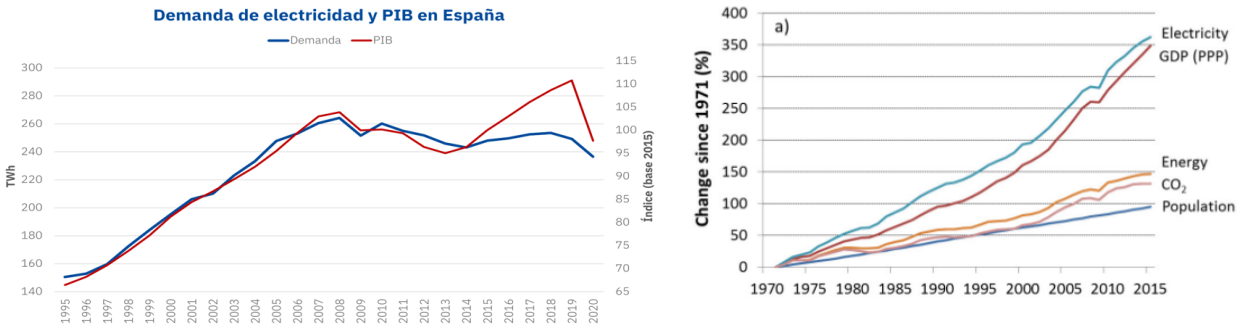
Introducción

La energía eléctrica es uno de los grandes motores de la humanidad desde finales del siglo XVIII. Dado esto, comprender todo lo relacionado a ella se convierte en algo primordial. En la actualidad no existe una industria que no requiera el uso de electricidad, lo que genera una gran necesidad de administrar la energía generada, esto con la finalidad de mantener y desarrollar todas las actividades humanas que requieran el uso de esta. Debido a la gran importancia de la energía eléctrica, surgen los mercados eléctricos, donde estos analizan económicamente todo lo relacionado a la generación, transmisión y distribución de la energía, cuya finalidad es poder satisfacer todas las necesidades energéticas que requiera cada sector e industria de cada país. Una de las variables más importantes para ver el comportamiento de este corresponde a la **demanda de energía eléctrica**. Esta información indica cuanto es la energía requerida para que una entidad o serie de consumidores puedan abastecer sus necesidades[1].

Dado que las actividades humanas están sujetas a una gran cantidad de factores externos, la demanda de la energía no está exenta de esto. El análisis y comprensión de esta variable es importante para el desarrollo del sector energético, especialmente para el ámbito de la generación de energía, donde esta última actividad aún se sigue realizando, en su mayoría, a partir de recursos no renovables (Petróleo: 32.89 %, Carbón: 29.16 %, Gas Natural: 23.40 % son las 3 fuentes de energía mas utilizada en el mundo[2]). Esta característica muestra que es necesario desarrollar modelos que permitan pronosticar la demanda ya que, de esta forma, permitiría administrar de mejor manera la generación y consumo energético.

Al analizar la demanda como serie temporal, es decir, viendo la evolución de esta variable alrededor del tiempo, se pueden encontrar comportamientos interesantes. Si se analizan los valores de demanda energéticas anuales, se aprecia un crecimiento de esta a través de los años, siendo esto la consecuencia del desarrollo de las industrias: aumento de la población, aumento de la tecnología, desarrollo económico, etc (ver figura 1.1). Por otro lado, si se analiza la demanda con valores registrados por hora, entonces se aprecia un aumento de estas durante el día y una reducción durante la noche, donde esto indica una relación estrecha de esta variable con el desarrollo de las actividades laborales y cotidianas. Sin embargo, al analizar la demanda en este último rango de tiempo, en diversas ocasiones se tiene registros que no obedecen este comportamiento, e incluso valores que superan el máximo que se registraron en días anteriores. Como se mencionó anteriormente, la demanda depende en gran medida del desarrollo de las actividades humanas, donde estas están sujetas a una gran cantidad de factores los cuales, en muchos casos, poseen comportamientos estocásticos (fallas en los sistemas eléctricos, eventos de gran convocatoria, evolución económica de la

industria, clima, por dar algunos ejemplos). Esto muestra que la demanda, en una gran cantidad de ocasiones, posee un comportamiento aleatorio, lo que dificulta su análisis al desarrollar modelos de predicción[3].



(a) (b)

Figura 1.1: (**a**) Aumento de la demanda energética y PIB español, siendo este último un valor económico se aprecia como la demanda aumenta al crecer la economía[4]. (**b**) Evolución porcentual de la población, consumo de electricidad y emisiones de CO_2 en las últimas décadas. Al igual que el caso anterior, se aprecia un aumento de la demanda al crecer la población[5]

Debido a estas características y tambien a la importancia de la demanda, se han desarrollado muchos sistemas que predicen esta variable, utilizando técnicas de todo tipo, mediante *machine learning supervisado, deep learning*, sistemas autorregresivos como ARIMA, entre otros. Dentro de todos los modelos que realizan regresiones, el **uso de series difusas** han sido sistemas que han tenido mejores rendimientos en este campo, donde estos modelos, ademas de ser muy variados, permite incorporar de manera sencilla la información de variables relacionadas, ya que dentro de su proceso de entrenamiento, se encuentra el análisis profundo de las relaciones entre las variables procesadas.

Dado lo anterior, el objetivo principal de este trabajo es desarrollar un modelo de predicción de la demanda de la energía eléctrica usando lógica difusa, explorando diferentes configuraciones y parámetros mediante el uso de librerías especializadas en `Python` y comparar el rendimiento de estos modelos difusos con algunos sistemas que ya han sido desarrollados en el estado del arte como **redes neuronales o bosques aleatorios**. Esto con la finalidad de desarrollar un modelo que pueda predecir los siguientes valores de la demanda con una mayor precisión. Para ello, en las siguientes secciones de este trabajo, se presentará lo siguiente:

- Una revisión sobre los conceptos más importantes para la comprensión del trabajo desarrollado en el Marco Teórico.
- Análisis profundo sobre la base de datos, mostrando las peculiaridades de la información disponible, especialmente de la demanda y los datos más relacionados a esta.
- La metodología utilizada para el desarrollo de los modelos destacando el preprocesamiento y el entrenamiento realizado.
- Resultados obtenidos tras la aplicación de 35 modelos con configuraciones diferentes junto con el análisis de los rendimientos obtenidos.

- Conclusiones sobre los resultados obtenidos, más puntos a desarrollar en un trabajo futuro para mejorar los estos valores.

Capítulo 2

Marco Teórico

Para una buena comprensión del trabajo desarrollado, se debe tener en claro los siguientes conceptos:

2.1. Series Temporales.

Es una sucesión de datos ordenados cronológicamente en intervalos de tiempo definidos[6]. Los datos pueden estar espaciados equitativamente (por ejemmplo el registro de la generación solar diaria de una planta fotovoltaica), o distintos (cantidad de terremotos anuales de una zona definida). Este tipo de representación ofrece muchas ventajas, ya que su análisis permite encontrar relaciones subyacentes en los datos (pueden ser de varias series temporales o consigo misma) donde estas pueden ser usadas para extrapolar el comportamiento hacia el pasado, periodos de pérdida de datos y futuro. Poseen 2 características relevantes: la **estacionalidad** que muestra si esta posee algún patrón alrededor del tiempo y la **tendencia** que indica la evolución de ésta variable (si los valores crecen, decrecen o se mantienen en el mismo rango).

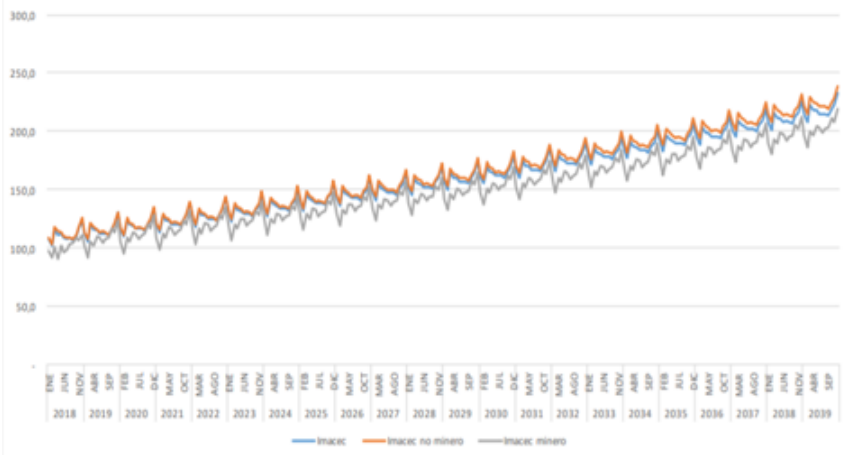


Figura 2.1: Serie temporal del Índice Mensual de Actividad Económica (IMACEC) del sector minero, no minero y en conjunto entre 2019 - 2039[7]

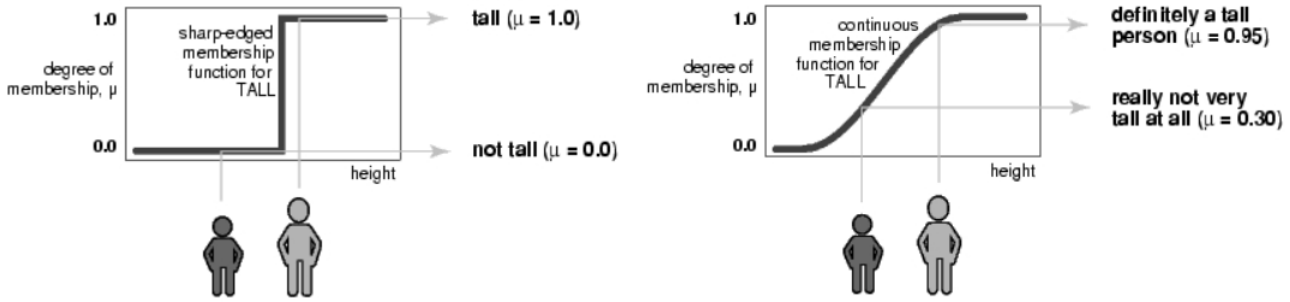
La figura 2.1 muestra la representación temporal de un indicador económico, en donde se puede apreciar una estacionalidad de alrededor de un año y una tendencia que muestra el crecimiento de estos valores.

2.2. Series Temporales difusas.

2.2.1. Lógica Difusa

Este un tipo de lógica paraconsciente (un sistema de lógica que trata de las contradicciones de una forma atenuada), cuya principal diferencia de la lógica clásica es que no toma todas las expresiones como completamente ciertas o falsas.

Esta lógica permite abordar los problemas de decisiones de una forma muy interesante[8], ya que en el mundo real no se puede abstraer todo a un sistema binario. Por ejemplo: considere que se tiene un registro de la altura de las personas en donde esta información es utilizada para saber quien es alto o pequeño de este conjunto de datos (figura 2.2), donde para hacer la diferencia entre estos conjuntos se toma el valor 1.70 mts. Al utilizar la lógica normal se tiene que todos los datos estarán en alguno de estos conjuntos, sin embargo, ¿sería correcto decir que alguien con estatura 1.69 mts es pequeño o 1.71 mts es alto? .



(a) (b)

Figura 2.2: Diferencias entre la asociación de la altura a los conjuntos de alto y bajo mediante lógica clásica (a) y lógica difusa (b), el eje x en ambos ejemplos corresponde a la altura mientras que el eje y corresponde al **grado de pertenencia o membresía**.

El uso de la Lógica Difusa permite establecer un grado de pertenencia a los conjuntos definidos, es decir, las variables pueden pertenecer a más de un conjunto. Cuando estos términos se encuentran en estas representaciones, y, juntando estas características con los valores de pertenencia, se construye lo que se conoce como **variable lingüística**, donde estas en su totalidad mas un margen conforman el **universo de discurso**. El proceso que se encarga de llevar las variables numéricas al dominio difuso (el de las variables lingüísticas) se llama **fuzzyficación**. Entonces al llevar los datos del dominio original (X) al dominio difuso $\tilde{A}$, se tiene que cada término lingüístico corresponde a un **conjunto difuso** $\tilde{a} \in \tilde{A}$, donde cada uno de estos conjuntos esta asociado a una función μ tal que $\mu\tilde{a}(x) : X \longrightarrow [0, 1]$. Es decir, que indican un grado de pertenencia a cada conjunto definido. Para el ejemplo anterior, se pueden definir 2 conjuntos $\tilde{A} = [\tilde{a_1}, \tilde{a_2}] = [pequeño, alto]$, donde definiendo el mismo umbral de 1.70 mts se pueden obtener los siguientes resultados[1] para 1.30, 1.69, 1.90 [mts]:

- $\mu A(1.50) = [0.99, 0.01]$, donde esta muestra pertenece más al conjunto pequeño, que al alto.

[1] Este corresponde a un ejercicio mental para ayudar la comprensión de la lógica difusa.

- $\mu A(1.69) = [0.495, 0.505]$, donde esta muestra tiene una pertenencia mixta a ambos conjuntos con valores similares.
- $\mu A(1.90) = [0.01, 0.99]$, donde esta muestra tiene una pertenencia mayor al conjunto alto.

La Lógica Difusa opera en los llamados conjuntos difusos, donde los valores de están en un rango [0,1] en lugar de los términos binarios. Estos valores son controlados por la **función de membresía** de un conjunto a cada elemento perteneciente al universo de discurso. Entonces, el conjunto difuso A se caracteriza por su función de membresía $\mu_A : X \longrightarrow [0, 1]$ y se define como

$$A = \{(x, \mu_A(x))\} \tag{2.1}$$

Al generar diferentes valores de asociación a través de estas funciones, se tiene que, la finalidad de estas es obtener los mejores valores. Por ende al utilizar un sistema que utilice Lógica Difusa **se espera que este construya la variable lingüística para los términos iniciales.**

2.2.2. Serie Temporal Difusa

Como su nombre lo indica, corresponde a series temporales que utiliza la Lógica Difusa para su procesamiento, convirtiendo cada valor de la serie en elementos pertenecientes a conjuntos difusos. Al tener una serie temporal en el dominio difuso, permite robustecer aún más el análisis sobre la serie, aplicando el análisis de relaciones de conjuntos difusos a los datos originales.

2.2.3. Operaciones difusas[9]

Corresponden a las operaciones relacionadas a la Lógica Difusa, donde los elementos pertenecientes a una serie temporal difusa también pueden aplicarlo.

2.2.3.1. Fuzzyficación.

Sea $(A_i)_{i=1}^N$ una secuencia de conjuntos difusos y $x \in X$ un valor de una serie temporal, entonces la **fuzzyficación** de x se define como.

$$fuzz(x) = (\mu_{A_i}(x))_{i=1}^N \tag{2.2}$$

2.2.3.2. Intersección difusa.

Sean A y B dos conjuntos difusos definidos en X con funciones de membresía μ_A y μ_B, la intersección $A \cap B$ se define por la función de membresía:

$$\mu_{A \cap B} = \min(\mu_A(x), \mu_B(x)) \tag{2.3}$$

2.2.4. Reglas difusas e inferencia

Al estar trabajando con lógica, entonces se tiene que tener el concepto de regla o condición, esta indica que bajo un supuesto (**precedente**) entonces otra información (**consecuente**) puede ser cierta o falsa dependiendo de la regla utilizada. Sin embargo, al estar trabajando con Lógica Difusa entonces la veracidad del consecuente es parcial.

Para aplicar este concepto a la regresión de series temporales se tienen los siguientes modelamientos.

2.2.4.1. Modelamiento Matricial

Desarrollado por Song y Chissom[10], este consiste en el uso de matrices de relaciones difusas que representen la dinámica temporal de las series temporales difusas. En este método, si se tiene una serie difusa F, cada par de secuencia $f(t-1), f(t) \in F$ es agrupado en las *relaciones de Lógica Difusa o FLR* por sus siglas en inglés.

El *FLR* corresponden a reglas difusas que describen patrones temporales $f(t-1) \longrightarrow f(t)$, o $A_i \longrightarrow A_j$ donde el lado de mano izquierda (LHS por sus siglas en inglés) o simplemente el precedente A_i contiene datos históricos fuzzyficados del valor hasta el tiempo $t-1$ y el lado de mano derecha (RHS) de la regla (o consecuente) A_j posee el valor fuzzyficado en el instante t. La expresión $A_i \longrightarrow A_j$ puede ser leída como "Si $f(t-1)$ es A_i, entonces f(t) es A_j". Por ende el dataset difuso F generará $T-1$ FLRs, siendo T el valor temporal asociado al conjunto difuso.

Entonces, para cada FLR, $A_i \longrightarrow A_j$ una matriz $R_t = A_i^T \times A_j = a_{ij}$ serán creadas con dimensiones $k \times k$ donde $a_{ij} = \min\{\mu_{A_i}(t), \mu_{A_j}(t-1)\}$ para $i, j = 1, ..., k$ y $t = 1, ..., T$. Esta matriz contiene las membresías difusas de todas las FLR para todos los conjuntos difusos. La Matriz de Operación $R(t, t-1)$ es calculada como la unión de todas las matrices de relación R_t tal que $R(t, t-1) = \bigcup_{t=1}^{T} R_t$, por lo que esta matriz posee todas las membresías de todas las FLR para todos los conjuntos difusos.

Al usar la inferencia de las Matrices de relaciones difusas se debe encontrar la membresía de la relación $f(t-1) \longrightarrow f(t)$ en $R(t, t-1)$, tal que $f(t) = f(t-1)\ o\ R(t, t-1)$, donde o es el operador difuso relacional máximo - mínimo. La operación $f(t-1)\ o\ R(t, t-1) = \max_j\{\min_i\{\mu_{A_i}(f(t-1), r_{ij})\}\}$ para $i, j = 1, ..., k$ y $r_{ij} \in R(t, t-1)$, genera un vector con las membresías de $f(t)$ para todos los conjuntos difusos A_j.

2.2.4.2. Modelamiento de reglas difusas

Propuesto por Chen en 1996[11], donde este crea los Grupos de Reglas de Lógica Difusa (**FLRG** por sus siglas en inglés), los cuales permiten realizar pronósticos de forma más eficaz evitando complejos procesos matriciales. El FLRG representa el conocimiento base del modelo que es legible para los seres humanos y sencillo de interpretar.

Este grupo difuso tiene la forma $LHS \longrightarrow RHS$, donde todas las FLR con el mismo LHS son agrupados mientras que RHS equivale al conjunto de posible conjuntos difusos que pueden seguir el conjunto de reglas LHS. El patrón $LHS \longrightarrow RHS$ se puede leer como "Si $F(t-1) = LHS$ entonces $\exists A_j\ \forall A_j \in RHS \mid F(t) = A_j$". Un ejemplo del conjunto de reglas se muestra en la siguiente expresión.

$$
\begin{gathered}
A_0 \longrightarrow A_1 \\
A_1 \longrightarrow A_1, A_2 \\
A_2 \longrightarrow A_4 \\
A_3 \longrightarrow A_2, A_3, A_5 \\
A_4 \longrightarrow A_3, A_4 \\
A_5 \longrightarrow A_4
\end{gathered}
$$

La inferencia usada por Chen[11] genera un pronóstico para el siguiente valor futuro $f(t+1)$

dado un retraso $f(t) = A_i$. Una búsqueda es realizada por el modelo de predicción difuso para encontrar el FLRG donde $LHS = A_i$. El RHS del FLRG promedia los posible conjuntos difusos que siguen A_i, donde $f(t) = A_i$, es decir, el pronóstico $f(t+1)$ es el conjunto RHS del FLRG seleccionado.

Los modelos FLRG de Chen permiten una representación compacta y legible por humanos del comportamiento de la serie temporal utilizando reglas difusas, además, tiene una ventaja respecto a los modelos matriciales y es su eficiencia computacional, ya que las operaciones de las matrices de relaciones provocan un aumento en las particiones del Universo de Discurso, lo que genera un aumento en la dimensionalidad que se traduce en un impacto negativo en la potencia de cómputo al realizar predicciones.

2.2.4.3. Extracción de conocimiento

Tanto el modelamiento matricial, como el modelamiento de reglas difusas, corresponden a algoritmos que forman parte del proceso de extracción del conocimiento[12]. El cual es responsable de inducir el conocimiento del modelo mediante la mejora de los patrones de reconocimiento presentes en las series temporales difusas, además, de aprender los patrones temporales presentes en las series temporales.

2.3. Modelos de series temporales difusas.

Son los sistemas que utilizan Lógica Difusa para la predicción de datos en intervalos de tiempo que no tengan registros. Dado que la teoría de la Lógica Difusa es amplia y el análisis de las series temporales también lo es, entonces se tiene también la existencia de una gran cantidad de modelos difusos. Sin embargo, a grandes rasgos, todos comparten una serie de características que permiten agruparlos[13].

2.3.1. Parámetros relevantes

Son las características que se pueden configurar al utilizar los modelos difusos, los más destacados son:

- **Orden (Ω):** este dato indica cuánta información pasada se necesita para obtener el valor siguiente. Cabe destacar que a medida que se tome una mayor cantidad de datos previos, es más probable obtener un modelo con un mejor rendimiento (figura 2.3)

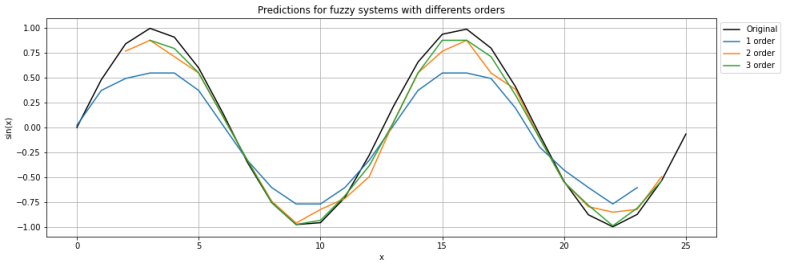


Figura 2.3: Comportamiento de predicciones de una función seno realizados por modelos difusos de diferentes ordenes.

- **Partición ($\prod$):** es la función que controla la forma en la que los datos temporales son llevados a los conjuntos difusos, por ende, se tienen 2 características relevantes en este parámetro: el

tipo de partición y la **cantidad de conjuntos difusos**.

El primer término, indica la forma general que tendrá la partición, algunos de estos se pueden apreciar en las figuras 2.4 y 2.5:

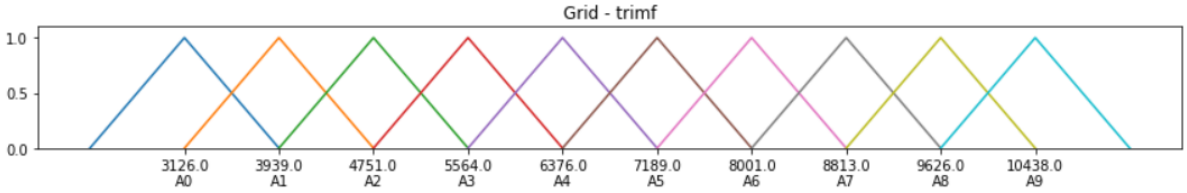


Figura 2.4: Partición Grid aplicado a 10 conjuntos difusos.

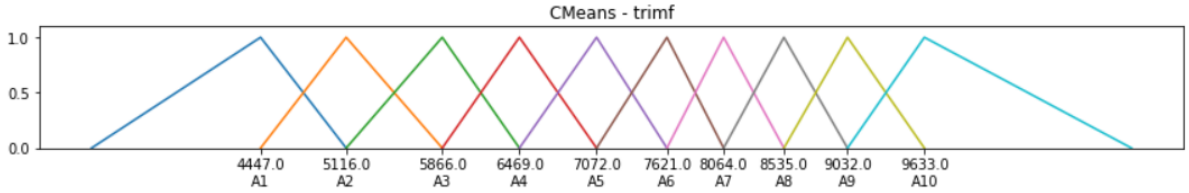


Figura 2.5: Partición tipo CMeans aplicado a 10 conjuntos difusos

Por otro lado, la cantidad de conjuntos difusos, como su nombre lo indica, es el número de estos que estarán presentes en el proceso de fuzzyficación de la información. Cabe destacar la importancia de este parámetro, ya que al tener pocos conjuntos, entonces el sistema no logrará entrenarse de forma adecuada entregando predicciones poco precisas (ver figura 2.6), en cambio, si se tiene una gran cantidad conjuntos definidos, es posible que el sistema aprenda relaciones innecesarias dando como resultado un aumento del ruido y menor eficacia al momento de realizar las predicciones (similar a un sobreajuste de una red neuronal).

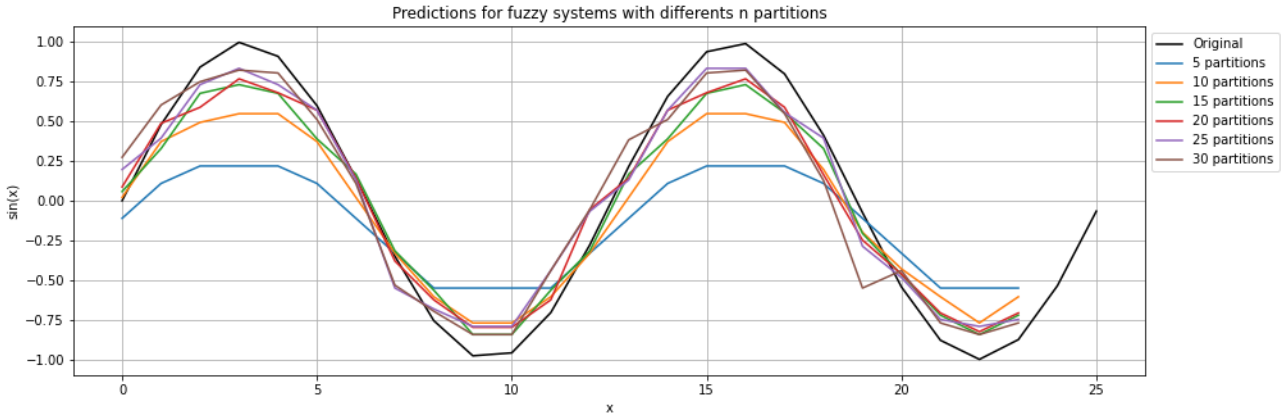


Figura 2.6: Predicciones de un modelo difuso para una función seno con distintas cantidades de particiones

- **Función de membresía (μ):** Este parámetro controla el tipo de mapeo que se realice a la partición, por lo que controla la forma en la cual la data es asociada a conjuntos difusos

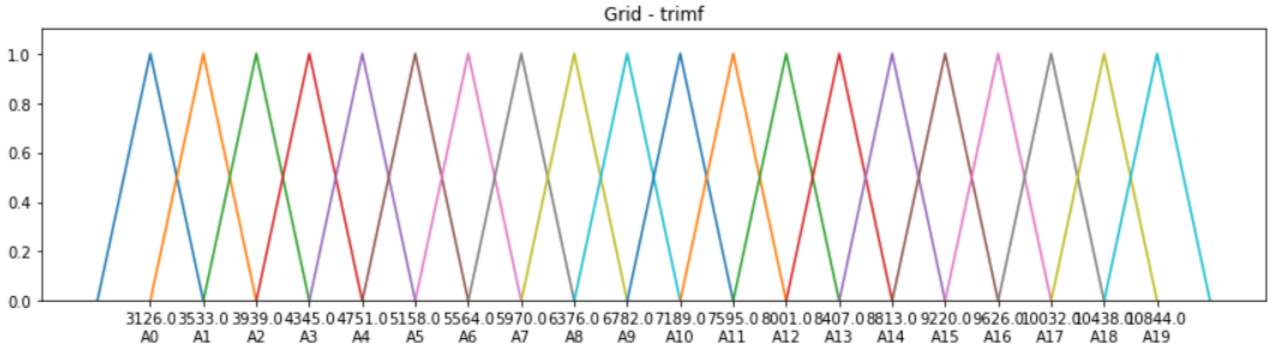


Figura 2.7: Partición Grid con función de membresía triangular aplicado a 20 conjuntos difusos.

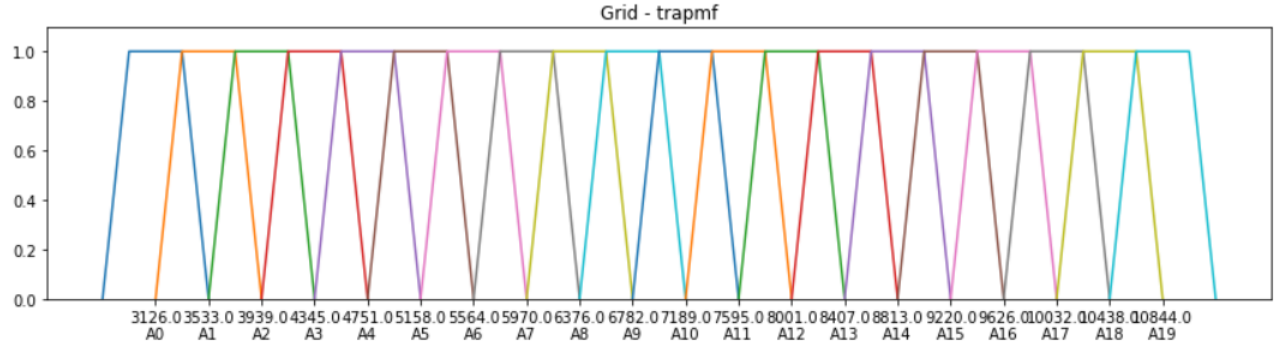


Figura 2.8: Partición Grid con función de membresía trapecio aplicado a 20 conjuntos difusos.

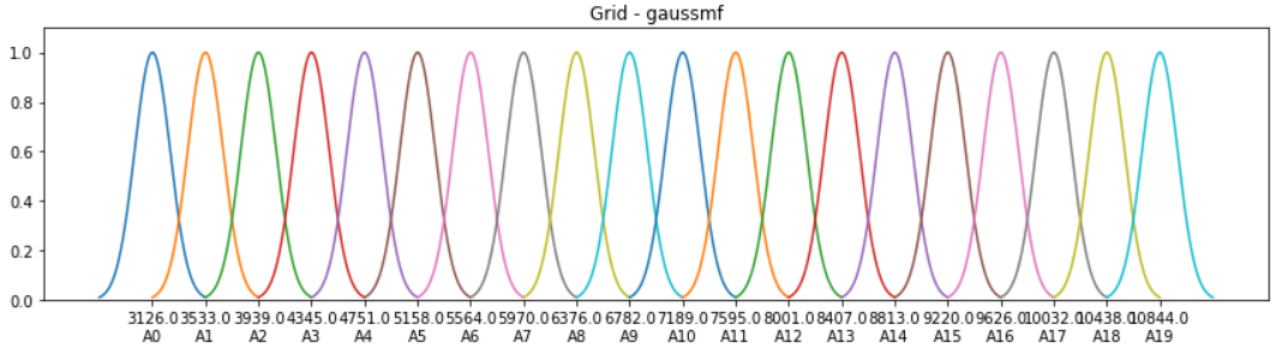


Figura 2.9: Partición Grid con función de membresía gaussiana aplicado a 20 conjuntos difusos.

Si bien se tiene una gran cantidad de particiones inspirados en diversas funciones matemáticas (figuras 2.7, 2.8, 2.9), la más utilizada y con mejores resultados corresponde a la partición Grid con función de membresía triangular (figura 2.7). También, hay que destacar que en diversos modelos se puede configurar el valor mínimo que puede tener esta asociación, es decir, controlar el valor numérico que se tengan en los conjuntos difusos; a este umbral se le conoce como **corte alfa** y se representa como α **cut**.

2.3.2. Modelos monovariables y multivariables.

Como la Lógica Difusa analiza los grados de pertenencia de las variables a conjuntos difusos, entonces la información a procesar puede tener datos no solamente de una variable objetivo, sino también, de datos de la información relacionada a esta variable. Por ende se pueden categorizar estos modelos en monovariables y multivariables, donde los más destacados son los siguientes:

Modelos monovariables

- **HOFTS** (*High Order Fuzzy Time Series*): modelos cuyo principal parámetro para la creación de las reglas es el orden, no realiza ponderaciones de los conjuntos desarrollados, asumiendo que cada uno es igual de importante.
- **WHOFTS** (*Weighted High Order Fuzzy Time Series*): modelo que tras la obtención de las reglas de partición, pondera los conjuntos difusos, por lo que se tendrán valores mas influyentes al momento de realizar las predicciones.
- **PWFTS** (*Probabilistic Weighted Fuzzy Time Series*): modelo que tambien pondera la importancia de cada conjunto difuso, sin embargo, este no solo tiene en consideración las reglas obtenidas, sino también la configuración inicial de la partición y el rendimiento de los modelos sin ponderaciones, por lo que la inclusión de estos pesos se realiza de manera probabilística. Cabe destacar que este modelo funciona mejor al tener una mayor cantidad de datos por lo que un aumento del orden puede ser contraproducente al utilizar este modelo.

Modelos multivariables

- **MVFTS** (*MultiVariable Fuzzy Time Series*): es un sistema que analiza las relaciones entre las variables y no añade importancia a los conjuntos difusos creados, siendo similar al modelo HOFTS pero con la facultad de tener un análisis más robusto al obtener una predicción, debido a la presencia de información relacionada a la variable objetivo.
- ***Weighted MVFTS***: similar al modelo anterior, sin embargo le añade importancia a los conjuntos por lo que tiene un mejor análisis al desarrollar predicciones.
- **FIG-FTS** (*Granular Multivariate Weighted Fuzzy Time Series*): son modelos que tienen un énfasis en la obtención de las relaciones de las variables mediante algoritmos de clusterización, además, incorpora ponderaciones en los conjuntos difusos creados.

Estos modelos poseen métodos de entrenamiento particulares, la teoría de estos sistema es explicada con detalle en el capitulo 5.

2.4. Modelos del estado del arte

Para analizar la eficacia de los modelos difusos implementados en el capitulo anterior, es necesario el desarrollo de modelos de predicción presentes en la literatura. Dado que se esta trabajando con datos de la demanda eléctrica, se tiene una gran variedad de sistemas regresores que ya se han desarrollado este trabajo.

En esta se mostrarán las implementaciones de algunos modelos extraídos del estado del arte, su arquitectura y una explicación de sus hiperparámetros más relevantes.

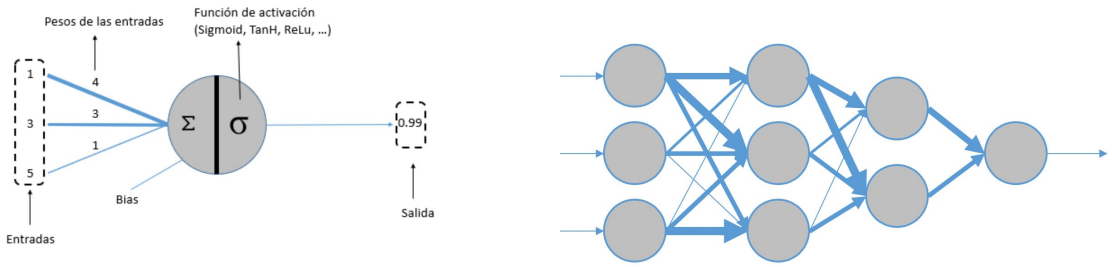
2.4.1. Redes Neuronales Artificiales.

Es el algoritmo de Inteligencia Computacional más popular debido a su enorme aplicación en una gran cantidad de áreas. Está inspirado en las redes neuronales naturales y sus conexiones presentes en el cerebro.

2.4.1.1. Arquitectura.

Corresponden a conjuntos de funciones matemáticas que ponderan los valores de una entrada para ajustarlo a un valor objetivo (salida). cada una de estas funciones se denominan *neuronas* y simplemente realizan una suma ponderada (figura 2.10a), donde el valor de estas sumas se procesa finalmente a una función para obtener su valor final. Esta última función se llama *función de activación* y permite a la neurona convertirse en un sistema no lineal.

Para que estos modelos puedan resolver problemas requiere el uso de varias neuronas, donde estas se agrupan por capas (figura 2.10b). El conjunto de todas estas capas dan lugar a una Red Neuronal Artificial, las cuales pueden ser de **entrada, oculta o salida** que indica si la capa esta al principio, en medio o final de la red, respectivamente.



(a) (b)

Figura 2.10: Diagrama de neurona artificial (a), y agrupación de estas en capas de una pequeña red (b)

2.4.1.2. Entrenamiento

El algoritmo que realiza el proceso de aprendizaje de estos sistemas se llama *backpropagation*[14], el cual consiste en calcular el error obtenido en las ultimas capas y propagarlo hacia atrás, con el cual se realiza un ajuste de los valores de los pesos y bias (conexiones entre las neuronas y los valores independientes de esta), capa por capa hasta llegar al principio de la red. El entrenamiento de estos sistemas es un proceso iterativo, el cual se realiza durante una cantidad de **épocas** definidas, donde en cada una se realiza el cálculo de una función de costos o *loss*, que muestra el error del sistema alrededor del entrenamiento. El objetivo del algoritmo *backpropagation* es hacer disminuir los valores que se obtengan de la función *loss*, es decir, provocar una convergencia hacia el valor más pequeño posible (o en caso de que se tenga un métrica de eficacia del sistema, maximizar el valor de esta alrededor del entrenamiento).

2.4.1.3. Hipeparámetros Relevantes

El proceso de entrenamiento y la forma del modelo, es controlado por los **hiperparámetros de la red**, los cuales configuran la forma de este modelo y también colocan pautas sobre cómo debe ser el entrenamiento que debe tener. Los más destacables son los siguientes:

- Tamaño de las capas ocultas: cantidad de neuronas que utilizan las capas ocultas de la red.
- Función de activación: corresponde al tipo de función no lineal que se aplica al resultado de la suma ponderada (los valores procesado por la capa anterior con los pesos de las conexiones de la capa actual). Se tienen diversas funciones, una de las más usadas y que entrega modelos con buen rendimiento corresponde a la **función RELU**[15] la cual solo procesa valores positivos dejando nulo el resto, es decir:

$$\text{RELU}(x) = \max(0, x) \tag{2.4}$$

  Otra función de activación sumamente usada aunque posee una lenta convergencia es la **función sigmoide**[16], la cual lleva los valores a un rango entre 0 y 1, convirtiendo las cantidades más altas a valores cercanos a 1 y los más bajas a cero, es decir aplica la siguiente ecuación:

$$f(x) = \frac{1}{1 - e^x} \tag{2.5}$$

- Cantidad de épocas: corresponde al número de iteraciones que debe realizar el sistema durante el entrenamiento.
- Tasa de aprendizaje: es un valor utilizado al momento de actualizar el valor que poseen los pesos[17], esta se encuentra incluida en la función que define el tipo de cambio que se le debe aplicar a estas conexiones (optimizador). Su valor es un parámetro muy importante ya que controla la convergencia del sistema. En caso de que sea un valor muy pequeño, el modelo requerirá una mayor cantidad de épocas de entrenamiento lo que se traduce en una ralentización del proceso de aprendizaje. Por otro lado, si se escoge un valor muy alto, es posible que el modelo no pueda obtener los valores apropiados de los pesos, lo que conlleva a un mayor error al momento de calcular el *loss* de la época, por lo cual el modelo no logra aprender correctamente.
- Optimizador: corresponde al método por el cual se ajustan los valores de los pesos a partir del error obtenido. Un optimizador muy utilizado es el sistema **ADAM**, el cual utiliza estrategias de aprendizaje adaptativo para cambiar el valor de los pesos de la red junto con la tasa de aprendizaje, logrando de esta forma acelerar el aprendizaje junto con la convergencia de este proceso[18].

2.4.2. Random Forest.

Un modelo de *machine learning supervisado* muy conocido debido a sus buenos resultados en varia s tareas de regresión y clasificación. Este sistema utiliza los conocidos árboles de decisión, siendo estos modelos que cambian el valor de un dato procesado dependiendo de las configuraciones que posea cada nodo del árbol[19]g, donde realiza *separaciones, hojas o splits* en cada nodo para extraer características e información importante de los datos procesados.

2.4.2.1. Arquitectura

El algoritmo Random Forest utiliza diversos árboles de decisión con diferente configuración cada uno (figura 2.11), donde los datos procesados pueden obtener diversos valores durante el procesamiento de cada árbol. Sin embargo, dependiendo del valor objetivo, se descubre cuál es la

configuración óptima de todos estos árboles para procesar los datos de una forma más correcta, logrando así acercarse a los valores deseados.

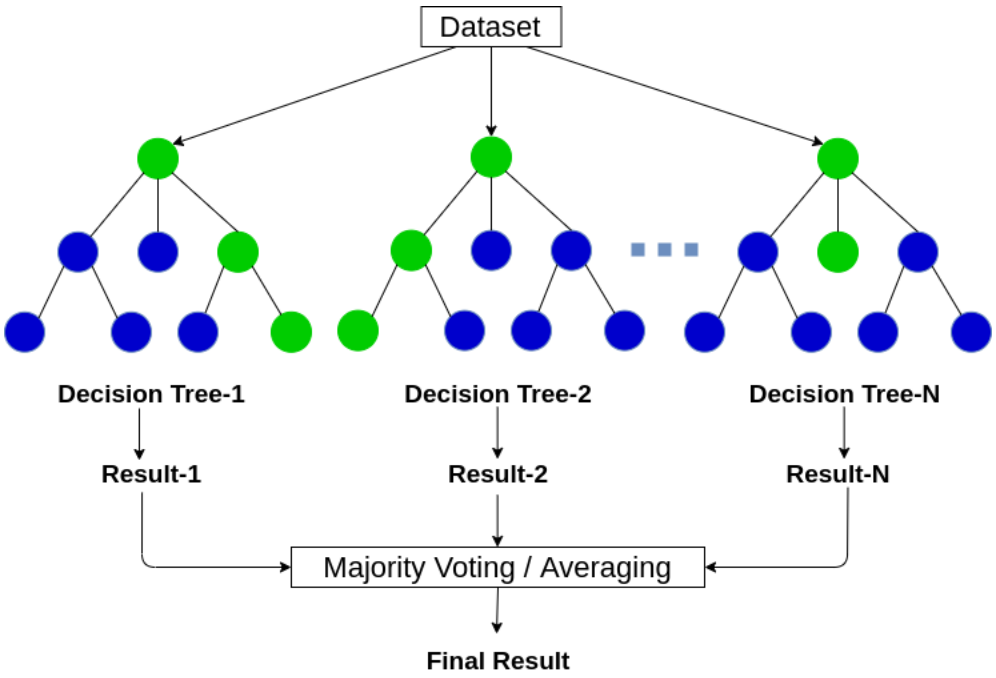


Figura 2.11: Diagrama del proceso de un algoritmo random forest.

2.4.2.2. Entrenamiento[20]

Este algoritmo como se mencionó anteriormente, utiliza una gran cantidad de árboles de decisión, por lo que este sistema sería un modelo de unión o **ensemble** de otros modelos. La fortaleza de estos modelos es que permiten un mejor análisis de las variables si estas son procesados por sistemas que sean lo más diferentes posibles, es decir, donde estos tengan una menor varianza.

Para reducir la relación entre los árboles, el Random Forest utiliza un proceso llamado *bagging* el cual consiste en el uso de un muestreo repetido con reposición, el cual promedia un conjunto de observaciones de entrada para reducir la varianza del modelo *ensemble*. Sin embargo, al aplicar el proceso de *bagging* estos estarán dominados por un mismo predictor, por lo que la generación de los árboles darán modelos muy parecidos, aumentando la varianza en esto. Para abordar este problema el algoritmo Random Forest realiza una selección aleatoria de m predictores antes de realizar la división, logrando de esta forma desarrollar una media de sistema que no tengan un predictor influyente. Este paso permite disminuir la varianza, logrando de esta manera disminuir la correlación entre los árboles generados.

2.4.2.3. Hiperparámetros relevantes

Al igual que las Redes Neuronales, estos sistemas también tiene sus hiperparámetros, que controlan la estructura de estos modelos. Los más destacados son:

- Cantidad de árboles: corresponde al número de árboles de decisión que posee el algoritmo.
- Criterio de separación: es la función que mide la calidad de los *splits* de los árboles. Al utilizar el Random Forest como regresor, entonces este hiperparámetro sirve como la función

de costos del sistema. Estos criterios pueden ser *error cuadrático*, el cual sirve como reducción de varianza como criterio de selección de característica, y el criterio *error absoluto* como error absoluto medio.

- Cantidad de muestras mínimas para una separación: es el número de muestras internas mínimo que son necesarios para realizar una separación de un nodo.
- Profundidad máxima: largo máximo de los árboles. Usualmente, este parámetro se ajusta dependiendo del valor de la cantidad de muestras mínimas para una separación, ya que los nodos se expanden hasta que las hojas obtenidas, contienen menos datos que este valor

2.4.3. Transformer

Estos modelos corresponden a sistemas de *deep learning* cuyo aprendizaje recae en lo que se conoce como *sistemas de atención*. Este modelo propuesto en 2017 por investigadores de Google Brain[21], es la base de diversas tecnologías que han obtenido enorme logros en tareas de machine learning e inteligencia computacional. Dentro de las gran cantidad de problemas que pueden resolverse mediante el uso de estos sistemas, tambien se encuentran las regresiones de series temporales, donde estos ya han sido utilizados en el estado del arte para realizar pronósticos de la demanda energética.

Este modelo surgió como una alternativa para poder solucionar problemas **Seq2Seq o sequence to sequence** que consisten en convertir una secuencia de datos en otra. Inicialmente, los problemas de esta naturaleza fueron abordados mediante el uso de Redes Neuronales Recurrentes[2] (RNN por sus siglas en inglés) ya que estos modelos pueden procesar las salidas de las capas de las redes para tomarlos como entradas en otra parte del sistema (figura 2.12)

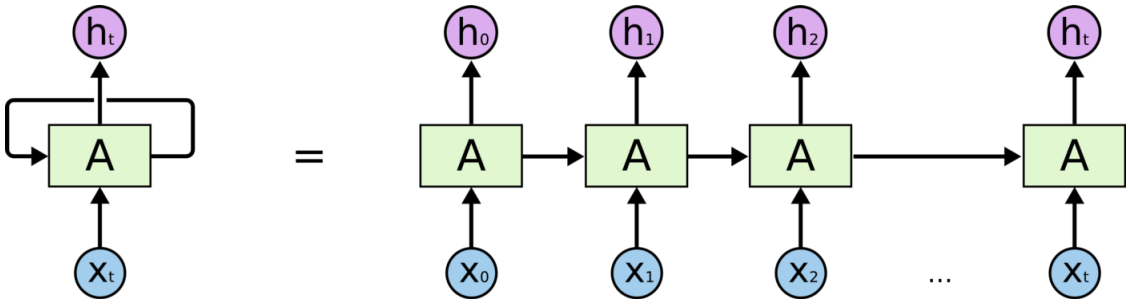


Figura 2.12: Diagrama de procesamiento de una secuencia en una Red Neuronal Recurrente

Este tipo de sistemas permiten obtener una salida final que tenga en consideración la importancia contextual de cada elemento anterior, por lo que es muy útil para trabajos dentro del Procesamiento del Lenguaje Natural[3][22].

2.4.3.1. Arquitectura.

El modelo principal desarrollado en 2017 para poder abordar problemas de traducción se aprecia en la figura 2.13[21].

[2] Redes Neuronales que poseen nodos entre capas no consecutivas.
[3] Campo de las ciencias de la computación y la Inteligencia Artificial que estudia las interacciones entre las computadoras y el lenguaje humano

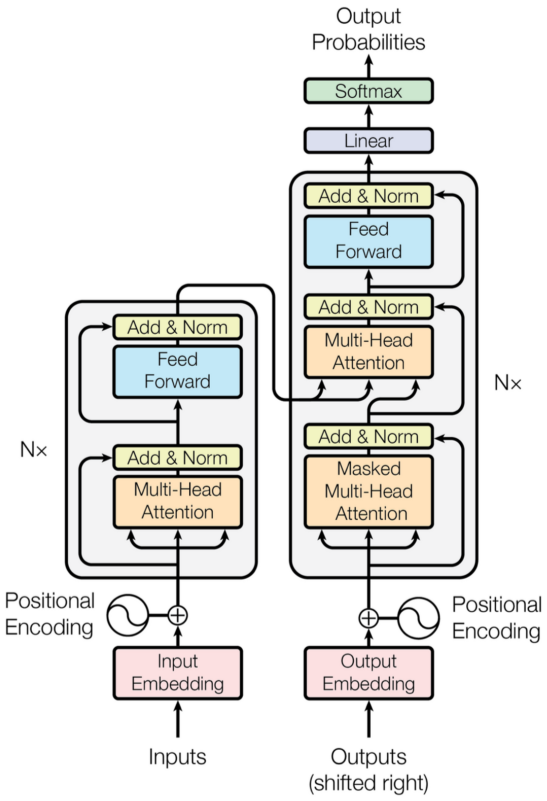


Figure 1: The Transformer - model architecture.

Figura 2.13: Arquitectura de un transformer

Este modelo posee una ventaja en comparación a las RNN al procesar secuencias, ya que el contexto aprendido toma en consideración cada elemento pasado de la secuencia procesada[23], no solo el anterior. La estructura de este modelo se puede sintetizar en 2 módulos principales:

- Un **Encoder** que contiene un módulo de atención junto con una capa *feed forward*[4] (parte izquierda de la figura 2.13).
- Un **Decoder** que posee una estructura similar al Encoder, pero incluye un módulo de atención extra que solo aplica esta a los datos de salida anteriores, además de tener un módulo *Linear* que ajusta las dimensiones de salida junto con una función *softmax*, la cual convierte el vector de salida en uno de probabilidades (parte derecha de la figura 2.13). La entrada a esta parte del modelo corresponde a las salidas deseadas pero desplazadas en una posición a la derecha.

Cabe destacar que este sistema posee varias capas de normalización[24] a la salida de diversos módulos, y, también, procesa las secuencias completas. Al realizar esto, se pierde la noción de la posición de los elementos procesados, por ende, también se incluye un **codificación posicional o positional encoding** a la serie para que esta pueda tener información sobre cual es la posición de cada elemento de la secuencia.

2.4.3.2. Self Attention

El módulo más destacado dentro de este sistema corresponde al sistema de atención o *self attention*. Dado que se desea obtener un sistema que pueda el contexto de la serie procesada, es decir,

[4] Capa de una red neuronal

que sea capaz de de tener **atención** entonces la entrada a este módulo corresponde a 3 vectores, los cuales mapean un *query (Q) ,key (K) y value(V)*. Estos vectores se obtienen a través de redes neuronales, se operan entre ellos para obtener una relación de compatibilidad entre estos.

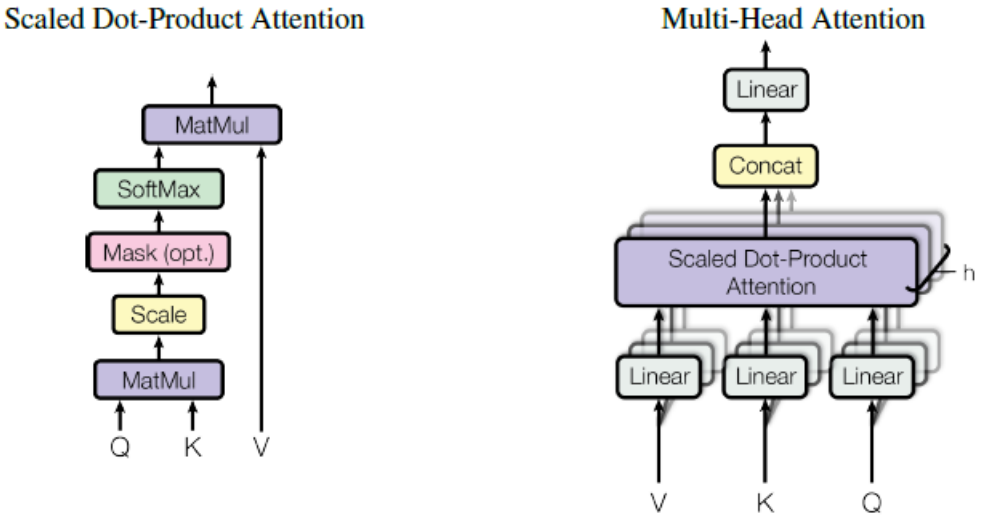


Figura 2.14: Diagramas de operaciones de la Atención escalada por Producto Punto y Atención Multicabezas

Los vectores Q y K tienen dimensión d_k y V tiene V_d. La relación se obtiene mediante el uso de la siguiente expresión:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.6}$$

$$\text{softmáx}(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_K}} \quad j = 1, ..., K \tag{2.7}$$

Donde la ecuación 2.7 representa z es un vector K-dimensional de valores reales, donde al ser procesado por esta función sus valores se convierten en densidades de probabilidades[25], es decir entre 0 y 1.

La aplicación de la función 2.6 entrega un vector en la salida, la cual tiene información sobre la compatibilidad de las relaciones entre el elemento de la secuencia con el resto de términos. Esta función se puede escalar sin problemas para procesar matrices en vez de vectores, las cuales contengan la información *Q, K y V* de todos los términos de las secuencia. Sin embargo, en la práctica, se tiene que en vez de utilizar una sola función de atención a vectores de tamaño d_{model} para Q, Ky V, es mejor utilizar diferentes proyecciones lineales h veces a d_k, d_ky d_v, respectivamente, donde una vez obtenidas estas atenciones proyectadas entonces se concatenan para obtener los valores finales como se muestra en la figura 2.14.

Esta función Atención multicabezas o **multihead attention**, permite al modelo obtener información de representaciones en diferentes subespacios y posiciones. Este proceso queda expresado

de la siguiente manera:

$$\text{Multihead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{2.8}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.9}$$

Donde las proyecciones son matrices de parámetros $W_i^Q \in R^{d_{model} \times d_k}, W_i^K \in R^{d_{model} \times d_k}, W_i^V \in R^{d_{model} \times d_v}$ y $W^0 \in R^{hd_v \times d_{model}}$.

Este modelo atencional es un novedoso sistema de *deep learning*, el cual, gracias a su proceso de aprendizaje contextual a partir de las secuencias permite desarrollar una gran cantidad de trabajos de análisis de datos, entre los cuales se encuentran la regresión de series temporales.

2.5. Métricas de error

Al estar trabajando con regresores, se tienen las siguientes métricas para analizar la efectividad de las predicciones realizadas por los modelos.

2.5.1. MSE

Es la simbología de **Mean Squared Error**[26], corresponde al error cuadrático medio. Dada una serie de valores $[y_1, y_2, ..., y_n]$ y una serie de predicciones $[\hat{y}_1, \hat{y}_2, ..., \hat{y}_n]$, entonces la fórmula para el RMSE es:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.10}$$

Donde este valor indica en términos del orden de los valores de la variable cual fue el error medio, cabe destacar que esta métrica es útil para entrenar modelos inteligentes al tomarla como función de costos, donde un paso importante es normalizar los datos para evitar grandes al valores al estar realizando el proceso de entrenamiento.

2.5.2. MAPE.

Similar al anterior, el **Mean Absolute Percentage Error**[27] mide el error medio porcentual de las predicciones obtenidas. La obtención de esta se realiza a partir de la siguiente relación:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{y_i - \hat{y}_i}{y_i} \tag{2.11}$$

Ésta métrica mide cuál fue el error porcentual analizando la relación entre el error y la data verdadera.

Capítulo 3

Base de datos.

3.1. Dataset.

La información utilizada en este trabajo son datos recopilados por la Red Europea de Gestores de Redes de Transporte de Electricidad (**ENTSO-E** por sus siglas en inglés), un organismo de gestores de electricidad que contiene registros sobre diversas variables relacionadas al mercado eléctrico de varios países europeos. Los datos mostrados corresponden a la información de consumo y generación española durante el 2015 y 2019[5]. El formato del dataset se muestra en la tabla 3.1:

time	Generation biomass [MW]	...	Forecast solar [MW]	...	Price Forecast [Eur/ MWh]	Price actual [Eur/ MWh]	Total load forecast [MW]	Total load actual [MW]
2015/01/01 00:00:00 + 01:00	447	...	17	...	50.1	65.41	26118	25385
2015/01/01 01:00:00 + 01:00	449	...	16	...	48.10	64.92	24934	24382
...	...	...	...	...	...	...	...	...
2018/12/31 23:00:00 + 01:00	290	...	26	...	64.27	69.88	24424	24455

Tabla 3.1: Información resumida disponible en la base de datos utilizada, debido a la gran cantidad de columnas, solo se mostraron estos datos.

En total se tiene un dataset con 29 características y 35064 muestras. Esta información contiene datos de la fecha registrada (en formato yyyy/mm/dd - hh/mm/ss), 21 fuentes de generación de energía (renovables y no renovables, los pronósticos de la generación de energías renovables en MW), el precio de la energía junto con pronósticos de esta (en EUR/MWh) y, finalmente, los datos de interés para este trabajo: **la demanda de la energía eléctrica horaria** junto con valores predichos por modelos del **TSO** español (Transmision System Operator, sociedad mercantil que maneja la operación del sector eléctrico).

[5] Esta base de datos es pública y se puede obtener pulsando aquí

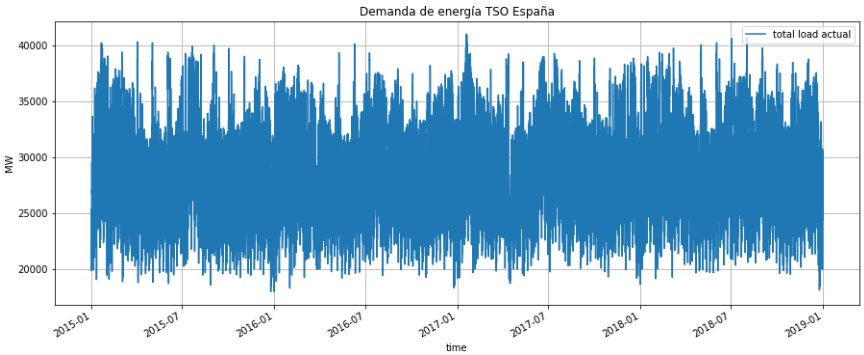
Los datos más utilizados fueron los de la demanda junto con los valores predichos por el TSO. Estos se utilizaron para comparar el rendimiento de los modelos con los sistemas utilizados por este organismo hasta 2019.

3.2. Análisis exploratorio de la demanda disponible.

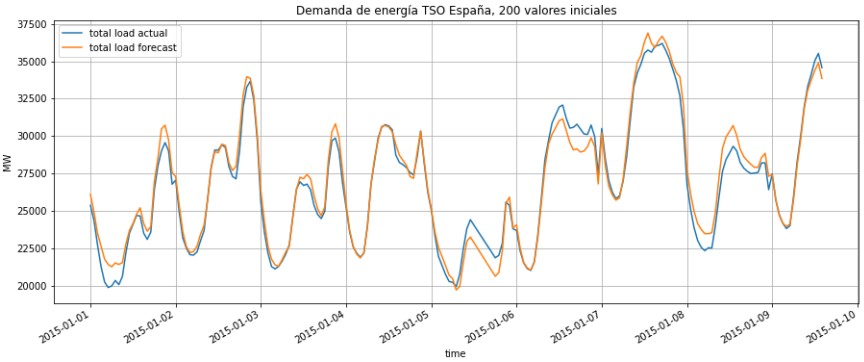
La información de mayor interés es la demanda, por lo que realizando un pequeño análisis se puede destacar lo mostrado en la tabla 3.2:

Fechas registradas	Desde 01/01/2015 - 00:00:00 hasta 31/12/2018 - 23:59:59
Rango de la demanda	Mínimo: 18041 [MW], Máximo: 41015 [MW]
MAPE TSO forecast	1.096 %

Tabla 3.2: Datos de interés de la demanda disponible.



(a) Visualización de la demanda total



(b) Muestra de los primeros 200 valores de la demanda junto con el pronóstico

Figura 3.1: Visualización gráfica de la demanda total y los valores de esta durante los primeros 10 días de 2015.

3.3. Preprocesamiento de datos.

Dada la información disponible en la base de datos (sintetizada en la tabla 3.1), se realizó un preprocesamiento de datos. Todo este proceso fue realizado en `python`, donde utilizando las fun-

ciones de la librería `pandas` (figura 3.2) se pudo obtener una visualización clara del dataset.

Figura 3.2: Logo de la librería `pandas`, la cual corresponde a una biblioteca desarrollada como una extensión de numpy el desarrollo de ciencia de datos.

EL primer filtrado realizado consistió en eliminar las columnas que tuvieron mas de un **90 %** de información nula usando la función definida en el código 3.1, este pequeño algoritmo remueve las características de un `dataframe` cuyo porcentaje de valores nulos sea superior a un límite indicado (`nan_threshold`). Entonces para realizar el proceso mencionado al principio, solo se cargó los datos en la entrada de esta función y se ajustó el valor de `nan_threshold` a 0.9, donde además tambien se removieron las muestras que no tuvieran información en el resto de columnas restantes.

Código 3.1: Función de eliminación de columnas con poca información.

```
import pandas as pd

def nans_remove(df, nan_threshold, reset_index=True):
  df_process = df.copy()
  assert nan_threshold < 1 and nan_threshold > 0 , ('nan threshold must be between 0 and 1')
  cols_names = df.columns
  for name in cols_names:
    len, drop_len = df_process[name].___len___(), df_process[name].dropna().___len___()
    difference_reason = (len - drop_len)/(len)
    if difference_reason > (1-nan_threshold):
      df_process.drop(columns=[name], inplace=True)
    else:
      continue
  df_process.dropna(inplace=True)
  df_process.reset_index(drop=reset_index, inplace=reset_index)
  return df_process
```

Tras la aplicación de esta limpieza de datos, se realizó un preprocesamiento para cada modelo que fuese probado, construyendo diversas series temporales (tabla 3.4) o aplicando particiones a la data.

Si bien se tiene una gran cantidad de información, los datos utilizados fueron la fecha de registro (***time***), la demanda (***total load actual***) y el pronóstico realizado por el TSO español (***total load forecast***), por ende se seleccionaron los valores de estas columnas, formando de esta manera un dataset el cual se puede apreciar en la tabla 3.3.

Los datos de la demanda (`total load actual`) fueron procesados de diferente manera, depen-

time	total load actual	total load forecast
2015/01/01 00:00:00 + 01:00	25385	26118
2015/01/01 01:00:00 + 01:00	24382	24934
...	...	...
2018/12/31 22:00:00 + 01:00	25801	25450
2018/12/31 23:00:00 + 01:00	24455	24424

Tabla 3.3: Demanda registrada por el TSO español, junto con el pronostico de esta y su fecha

diendo de cual fuese el modelo predictivo que se desease evaluar, y los valores del pronóstico solo se utilizaron para comparar la eficacia de los regresores con las predicciones realizadas por este organismo.

Algunos modelos del estado del arte utilizaron *machine learning supervisado* por lo que para estos sistemas se tomaron los valores de la demanda y se construyeron diversas series temporales, como estos modelos necesitan una serie de características y una variable objetivo, entonces las características fueron los valores de la demanda previos a la actual.

time	load-72h	load-71h	...	load-1h	load actual
2015/01/02 00:00:00+01:00	25385	24382	...	26798	27070
2015/01/02 01:00:00+01:00	24382	22734	...	27070	24935
...	...	...	...	...	...
2018/12/31 22:00:00+01:00	29145	26934	...	28071	25801
2018/12/31 23:00:00+01:00	26934	24312	...	25801	24455

Tabla 3.4: Series temporales creadas a partir de los datos de la demanda junto con la fecha de registro.

Las series construidas tuvieron un largo de 73 valores, 72 valores previos de la demanda energética junto con el valor actual, por lo que los valores de la demanda de los primeros 3 días fueron perdidos (no se tuvo información previa). Con esta información se definió lo siguiente:

- Características del dataset: Demanda registrada durante las últimas 72 horas. Se escogió esta cantidad de características debido a que la información de la demanda horaria fue registrada.
- Variable objetivo: Valor actual de la demanda.

3.4. Definición de conjuntos de entrenamiento y prueba.

La distribución de estos conjuntos se aprecia en la tabla 3.5. El conjunto de entrenamiento contuvo el 70 % de los datos dejando lo restante para la evaluación del modelo.

Conjunto	Tamaño	Inicio	Fin
Entrenamiento	24519	2015/01/01 00:00:00	2017/10/19 23:00:00
Prueba	10509	2017/10/20 00:00:00	2018/12/31 23:00:00

Tabla 3.5: Distribuciones de datos para los conjuntos de entrenamiento y prueba

Todos los resultados expuestos en el capitulo 6 se hicieron utilizando solamente los datos del conjunto de prueba .Un punto a considerar es que dependiendo del modelo utilizado, se hicieron preprocesamientos extras o apartes a partir de los datos mostrados en este capitulo, estos serán explicados en la metodología e implementación computacional de cada modelo visto en los siguientes capítulos.

Capítulo 4

Estado del arte

El trabajo mostrado en este informe no corresponde a un estudio original. La predicción de la demanda energética es un estudio realizado desde hace varias décadas, obteniendo mejores resultados y la creación de nuevos modelos a lo largo de los años.

4.1. Investigaciones sobre modelos de predicción de la demanda de energía eléctrica

En este trabajo de memoria se investigó a través de diversos documentos y artículos pero los trabajos más destacados son destacados en esta sección

4.1.1. Pronóstico de demanda y potencia eléctrica para la red Chilectra S.A

- Titulo completo: *Pronóstico de demanda de energía y potencia eléctrica en el largo plazo para la red de CHILECTRA S.A. utilizando técnicas de minería de datos.*
- Fecha: Diciembre de 2015
- Autor: Andrés Felipe Palma Llewellyn.

4.1.1.1. Resumen

Este trabajo consiste en una tesis de magíster, cuya investigación consiste en la predicción de la potencia y demanda eléctrica mediante diversos regresores, cuya finalidad consistió en encontrar un modelo que permita mejorar las estimaciones realizadas por la empresa hasta esa fecha.

Se realizaron diversos modelos de proyección de demandas energía eléctrica. las cuales fueron separadas en 3 tipos de sectores: Residencial, Industrial y Comercial. Con estos modelos se realizaron validaciones estadísticas de los modelos de la empresa y los desarrollados en esta investigación para lograr las comparaciones y analizar los rendimientos y corroborar si los modelos desarrollados fueron superiores a los de la compañia.

La metodología aplicada en este trabajo consiste en la extracción del conocimiento a partir de los datos, lo que se conoce como *KDD* (Knowledge Discovery in Databases). Este consiste en llegar a las conclusiones apropiadas a través de la transformación de datos de bajo nivel en información

que sea más compacta, abstracta o más útil, el cual depende del problema al que se este aplicando. Esta metodología contiene una serie de etapas, donde una de ellas utiliza herramientas analíticas en búsqueda de patrones e información relevante. En este paso se implementan los regresores mencionados los cuales son Redes Neuronales, Random Forest, cuyas estructuras fueron investigadas mediante una búsqueda de grilla (GridSearch) para encontrar la combinación de hiperparámetros que maximizaron el rendimiento de estos modelos. También, se implementaron modelos estadísticos SARIMA y SARIMAX, para comparar el desempeño con los métodos tradicionales de pronóstico. Para este último modelo se utilizó información relacionada al IMACEC el cual corresponde a un índice que mide el desarrollo económico de algún sector industrial.

4.1.1.2. Base de Datos

Al ser un trabajo de investigación realizado en una empresa, los datos utilizados en este proyecto fueron proporcionados por esta, siendo la información relacionada a la demanda y consumo eléctrico de los clientes de la compañia. Estos fueron los datos mensuales históricos de energía y potencia eléctrica máxima mensual en el período correspondiente de Junio de 2001 hasta Agosto de 2014. También se utilizaron datos climatológicos y del IMACEC del mismo período de tiempo para la confección de los modelos multivariados.

4.1.1.3. Resultados

Al realizar diversas pruebas con una gran cantidad de modelos, el sistema más eficaz para realizar las predicciones fueron las Redes Neuronales cuya estructura e hiperparámetros fueron encontradas mediante una búsqueda de grilla. Este obtuvo un MAPE de 2.78 %.

4.1.2. *A Transformer-Based Framework for Time Series Learning*

- Titulo completo: *A Transformer-Based Framework for Multivariate Time Series Representation Learning*.
- Fecha: Diciembre de 2020.
- Autores: George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, Carsten Eickhoff.

4.1.2.1. Resumen

Los desarrolladores de este trabajo proponen un Transformer adaptado para realizar aprendizaje sobre las series temporales aplicando los sistemas de atención de estos. Construyendo de esta forma regresores de series temporales multivariables.

La adaptación del modelo original (figura 4.1) consistió en utilizar solamente una parte de la arquitectura: el *encoder*. Donde previamente al uso de este implementan un módulo, el cual, realiza un preprocesamiento especial a los datos ingresados. Este aplica un *positional encoding* a la serie, una máscara a la entrada la cual se encarga de filtrar las atenciones obtenidas debido al cálculo de valores posteriores con respecto al término que este procesando el *encoder* del transformer (ecuación 2.4 y 2.6), y, al final una capa *linear* al final de este proceso.

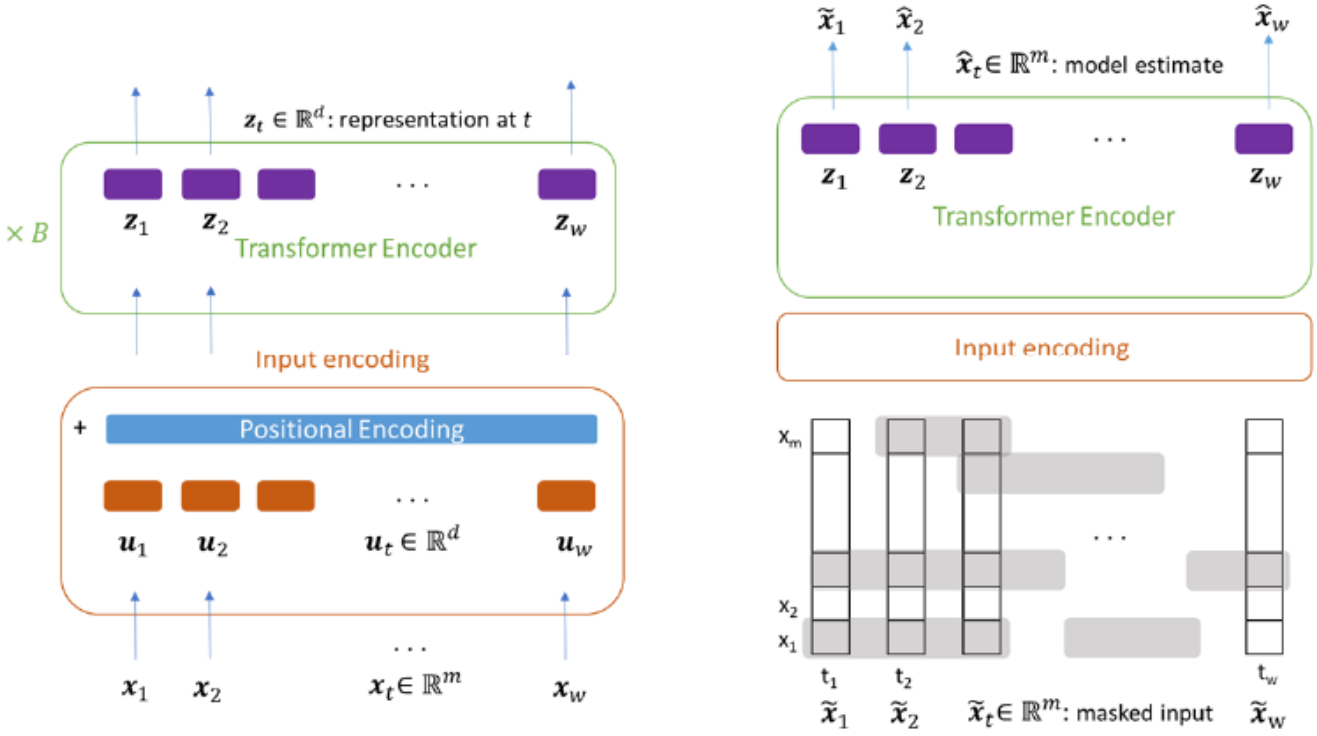


Figura 4.1: Arquitectura general del modelo propuesto, aplicable a todo tipo de tareas (**izquierda**) vs configuración del entrenamiento del modelo donde se aprecia la máscara la cual se aplica para realizar un entrenamiento no supervisado (**derecha**).

Para evaluar el rendimiento de este modelo, se realizan, comparaciones con diversos modelos de machine learning supervisado como redes neuronales o SVR[6] además de mencionar que este trabajo corresponde a una mejora del modelo propuesto por Li y su equipo[28] debido a que en este último solo se abordo el *transformer* para que procesara series temporales univariables.

4.1.2.2. Base de Datos

Las pruebas desarrolladas abordan una gran cantidad de bases de datos, entre las cuales, se destaca el uso dataset *Appliances Energy*[29], el cual, tiene información sobre el consumo energético, temperatura y condiciones de humedad de un edificio de bajo consumo energético en Bélgica[30]. Esta base de datos contiene 19735 muestras y posee una gran cantidad de características, sin embargo, la variable principal corresponde al consumo energético el cual es predicho por el modelo propuesto en esta investigación.

4.1.2.3. Resultados

Para la base de datos descrita se tiene un MAPE de 2.8 % en comparación a los 3.457 % del SVR y 4.227 % de las redes neuronales[31]

4.1.3. *Scalable Models with Fuzzy Time Series*

- Titulo Completo: *Scalable Models for Probabilistic Forecasting with Fuzzy Time Series.*

[6] *Support Vector Regression*: modelo que entrena mediante la creación de diversos hiperplanos.

- Fecha: Noviembre de 2019.
- Autor: Petrônio Cândido de Lima e Silva.

4.1.3.1. Resumen

Esta investigación corresponde a una tésis de doctorado presentado en la universidad Minas Gerais en Belo Horizonte, Brasil, la cual muestra diversos modelos difusos que pueden ser utilizados para predicciones de series temporales. Además presenta la implementación computacional de estos modelos en `Python` cuyos sistemas se encuentran agrupados en una librería particular de este lenguaje.

Si bien en esta investigación se analizan varios modelos de series temporales difusos algunos de estos ya eran existentes, por lo que en estos casos la investigación exploró las fortalezas y debilidades de estos regresores al utilizarlos como modelos predictivos. Sin embargo, también se hicieron presentes nuevos modelos difusos, especialmente aquellos sistemas relacionados a pronósticos probabilísticos, siendo este sistema el más estudiado debido a que este explora las incertidumbres presentes en las bases de datos y desarrolla un nuevo sistema que convierte esta información en reglas difusas. Este modelo se llama **Series Temporales Difusas con Ponderaciones probabilísticas** o **PWFTS** por sus siglas en inglés *(Probabilistic Weighted Fuzzy Time Series)*.

Este último modelo es la mayor innovación realizada en este trabajo. Dentro de esta investigación se mejora la escalabilidad para que pueda resolver problemas que requieran grandes series temporales y, además, se extiende su uso para aplicarlo como modelo multivariable, el cual utiliza diversas series temporales de las que extrae información difusa granulada[7], dándole a este modelo una gran versatilidad y efectividad al aplicarse sobre una gran gama de problemas.

4.1.3.2. Base de Datos

Debido al estudio general de modelos difusos presentes en esta investigación, se utilizaron diversos datasets para los modelos de una variables y multivariables

Modelos monovariables

- Se usó el Índice bursátil ponderado por capitalización de la bolsa de valores de Taiwán o TAIEX por sus siglas en inglés[8]. Este dataset es una serie temporal económica bien conocida usada habitualmente en la literatura de series temporales difusas; contiene información de la valorización de la bolsa de Taiwán media diaria desde 1995 hasta 2014.
- Se hace uso de información del NASDAQ, que corresponden a los índices de la bolsa de valores automatizada y electrónica más grande de Estados Unidos. Este índice económico tambien ha sido usado en estudios de series difusas y tiene datos históricos muestreados entre 2000 y 2016.
- El índice Standard & Poor's 500 también fue usado en esta tesis doctoral. Este es uno de los índices más importantes de Estados Unidos y es el que se considera el índice más representativo de la situación real del mercado[32]. Se encuentra basado en capitalización bursátil del mercado de 500 grandes empresas que cotizan en las bolsas más grandes de dicho país. Este dataset contiene los índices diarios medios por día laboral desde 1950 a 2017 con 16000 registros.

[7] La computación granular es un método que combina información precisa con detalles más generales, se centra en incorporar incertidumbres y probabilidades

[8] The Taiwan Stock Exchange Capitalization Weighted Stock Index

Modelos multivariables

- El proyecto SONDA (Sistema de Organização Nacional de Dados Ambientais) es un proyecto gubernamental brasileño cuya agrupación de datos analiza diferentes características ambientales (radiación solar, velocidad del viento, precipitaciones, etc). Las variables analizadas para este trabajo son: la radiación solar horizontal global y la velocidad del viento sobre 10 metros, ambas registradas por la estación de telemetría de Brasilia entre 2012 y 2015 por minuto, por lo que se tuvieron aproximadamente 2 millones de valores para analizar.
- Otro dataset interesante utilizado en esta sección de esta tesis doctoral, fue el dataset de consumo energético y potencia de Malasia, el cual tiene datos de la ciudad de Johor en Malasia, registrados entre 2009 y 2010 con 17519 muestras.

4.2. Adaptación de modelos del estado del arte

Las investigaciones presentadas en la sección 3.1.1. y 3.1.2. presentan el uso de modelos de *machine learning* supervisado, destacándose la adaptación del transformer como regresor de series temporales en este segundo trabajo. Dado esto, se decidió replicar estos modelos con sus características y metodologías para utilizarlos con los datos mostrados en la tabla 3.3.

4.2.1. Modelos de *machine learning* supervisado

Prácticamente todas las investigaciones sobre predicciones de algún fenómeno realiza comparaciones con modelos de machine learning supervisado. Las Redes Neuronales y los Random Forest solo tienen adaptaciones en los valores de sus hiperparámetros. Con esto conservan la forma de sus arquitecturas originales y su método de entrenamiento.

4.2.2. Transformers Adaptados

Este modelo esta inspirado en los *transformers* descritos en la sección 2.4.3. Corresponde a uno de los sistemas más avanzados que se tienen en la actualidad. Sin embargo, el sistema implementado no es precisamente el descrito en la figura 2.13, debido a que este se utilizó para resolver problemas de traducción e interpretación de idiomas, sino al descrito en la sección 3.1.2. de este trabajo, siendo esta adaptación la mejor arquitectura encontrada para aplicar los modelos de atención a los trabajos de predicción de series temporales.

4.2.2.1. Arquitectura

La arquitectura del modelo utilizado se puede apreciar en la figura 4.2

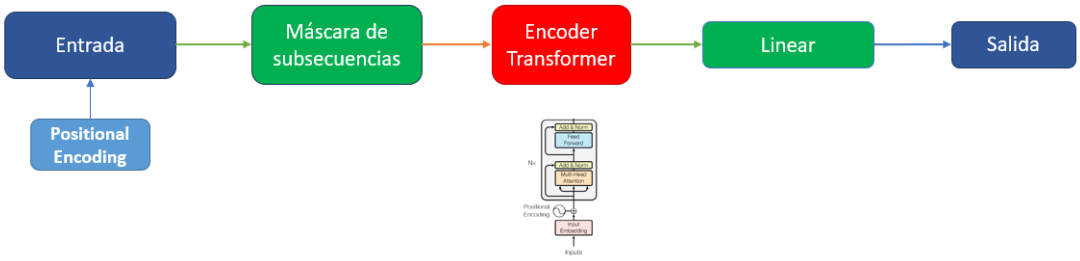
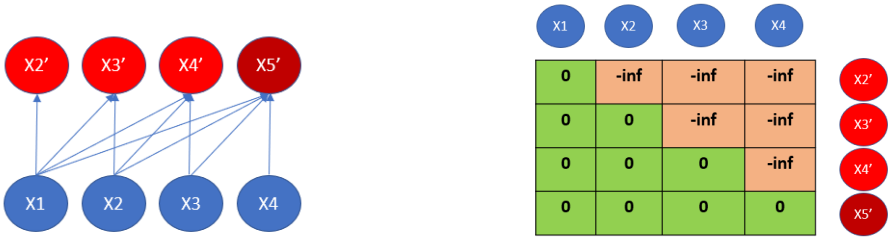


Figura 4.2: Diagrama de módulos del transformer adaptado para realizar regresiones de series temporales

La cantidad de módulos aplicados a este modelo es inferior al sistema original, pero aún conserva una parte de la estructura principal (Encoder Transformer). Las funcionalidades de cada uno de estos módulos son:

- **Entrada:** este representa la serie a procesar. Sin embargo, como el modelo utiliza los sistemas de atención del transformer, entonces este sistema realizará el proceso de la secuencia sin tener noción de la posición de los elementos procesados. Por ende, a los datos se les inserta información extra (es decir, incorpora una codificación posicional) para que al ser procesados, el modelo tenga noción sobre la posición de los elementos de las secuencias.

- **Máscara de subsecuencias:** este módulo realiza un filtrado de las secuencias, las ecuaciones 2.8 y 2.9 descritos en la sección 2.4.3.2, tienen la peculiaridad de afectar a todos los datos procesados, ya sean pasados o futuros. Como se desea realizar regresiones no es conveniente que el sistema procese los datos de salida que afecten a las regresiones ya realizadas, por lo que este módulo se encarga de filtrar este cálculo mediante la implementación de una máscara matricial que anula la importancia al realizar el cálculo de las atenciones de las secuencias.



(a) (b)

Figura 4.3: Objetivo de desarrollo de predicciones (**a**) y matriz triangular que representa la máscara para el filtro de cálculo de atenciones (**b**).

Como muestra la figura 4.3, para desarrollar el filtro de atenciones, es necesario utilizar una matriz con valores infinitos en la parte triangular superior[33]. Esto es debido a la aparición de las exponenciales (ecuación 2.7) presentes en la función *softmax*, al realizar el cálculo de atenciones (ecuación 2.6).

- **Transformer Encoder:** este módulo es el corazón de este sistema. Como su nombre lo indica, posee la arquitectura completa de un Encoder de Transformer por lo que el cálculo de los vectores de atención se realizan en este módulo.

- **Linear:** este módulo corresponde simplemente a una capa de neuronas. Es implementada para dimensionar la salida del módulo anterior y se obtengan los valores de la regresión.

La salida de este sistema corresponde a una secuencia con las predicciones tras los análisis de los t valores anteriores, es decir, el último valor corresponde a la predicción contando todos los valores reales de la secuencia, por lo que, el vector obtenido tendrá las mismas dimensiones que la entrada, pero las posiciones son desplazados en uno (figura 4.4.

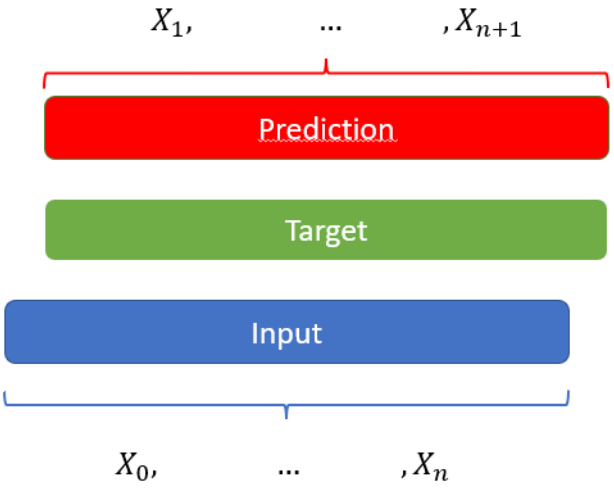


Figura 4.4: Representación del desplazamiento de las series procesadas y obtenidas en este sistema

Analizando la arquitectura de este regresor, se aprecia que tiene una gran similaridad al decoder del transformer original (parte derecha de la figura 2.13). Sin embargo, la capa softmáx es removida. Cabe destacar que la salida durante el entrenamiento puede ser tomada como una nueva entrada para añadir versatilidad al proceso de cálculo de atención.

4.2.2.2. Entrenamiento

El proceso para entrenar este sistema fue el siguiente:

1. De los datos totales disponibles se tomó una secuencia de largo n. Esta secuencia es enviada al módulo que contiene la máscara de subsecuencias, es decir, a la entrada del modelo (figura 4.5. Cabe destacar que, el inicio de esta secuencia es aleatorio, es decir, si i corresponde a la posición inicial de la secuencia seleccionada, entonces $i \in [1, N-n] : i \in \mathbb{N}$ indica que la secuencia puede tomar desde los primeros hasta los últimos n valores de un total de N datos disponibles.

2. La secuencia es procesada por la máscara de subsecuencias para anular el aporte de las atenciones obtenidas por parte de términos futuros.

3. El modelo realiza los cálculos de los vectores de atención siguiendo la ecuaciones 2.8 y 2.9 de la sección 2.4.3.2.

4. Los aportes obtenidos se procesan por una capa linear. Con esto se obtiene un conjunto de datos que corresponden a predicciones one step obtenidas por el análisis atencional de todos los elementos previos a cada valor de la secuencia obtenida.

5. A este vector de predicciones se le aplica la función de costos (loss) para actualizar los parámetros de todo el modelo.

6. Se repite el proceso según la cantidad de veces que el usuario lo haya configurado (cantidad de épocas).

Un punto que debe quedar claro es que cada valor del vector de predicciones obtenido tras el proceso de este modelo corresponden a predicciones one step realizadas con el análisis de todos

los valores anteriores a la posición del valor que se este observando, no solamente el último valor. Por ende, se puede procesar todos los valores de prueba de una sola vez si el usuario solo quiere obtener predicciones one step[9].

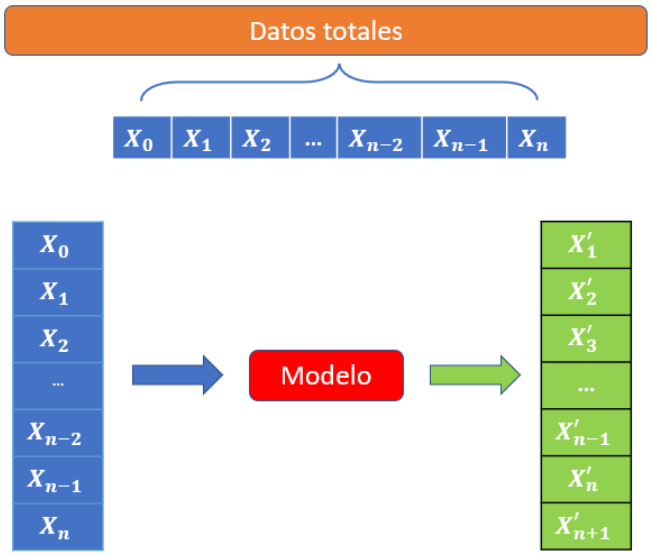


Figura 4.5: Diagrama de entrenamiento para la realización de predicciones one step, donde se selecciona una serie de n elementos de los datos totales y el modelo entrega las predicciones en el instante posterior

Por otro lado si se desea obtener predicciones sobre ventanas más extensas, entonces, el sistema realiza el entrenamiento siguiendo los siguientes pasos.

- Tras la obtención de la secuencia, esta separa sus términos y son añadidos uno a uno al modelo (figura 4.6).
- Dependiendo de la extensión solicitada por el usuario, entonces, el sistema no siempre incorporará datos reales para realizar el cálculo atencional, por lo que, para diversificar el entrenamiento este también toma como entrada las predicciones generadas. La selección del valor previo para la obtención del siguiente esta sujeta a las siguientes condiciones:

$$x_{t-1} = \begin{cases} y(t) & \text{si} \quad v < \dfrac{k}{k + T \cdot \exp\{T/k\}} \\ \hat{y}(t) & \text{caso contrario[} \end{cases} \tag{4.1}$$

  Donde v corresponde a un valor aleatorio $v \in [0, 1]$, k es un valor de frecuencia configurado por el usuario y, T es la época en la que se encuentra el sistema durante el entrenamiento. Claramente, estos dos parámetros poseen una gran relevancia, por lo que es más probable que el sistema tome valores pasados como las predicciones anteriores al estar avanzando en el entrenamiento. Hay que destacar que esta regla se puede configurar para que solo sea aplicable cuando el usuario desee realizar pronósticos sobre ventanas muy extensas (de tamaño j), y,

[9] Esto solo es viable si el usuario posee gran cantidad de memoria disponible, debido a que, mientras mayores sean las dimensiones de las series, mayor tamaño tendrán los tensores generados en el proceso

este solo se realice cuando el sistema ya ha procesado solo con datos reales, los j primeros elementos de su secuencia.

- Tras la obtención de la serie se repiten los pasos 4, 5, 6 del entrenamiento anterior

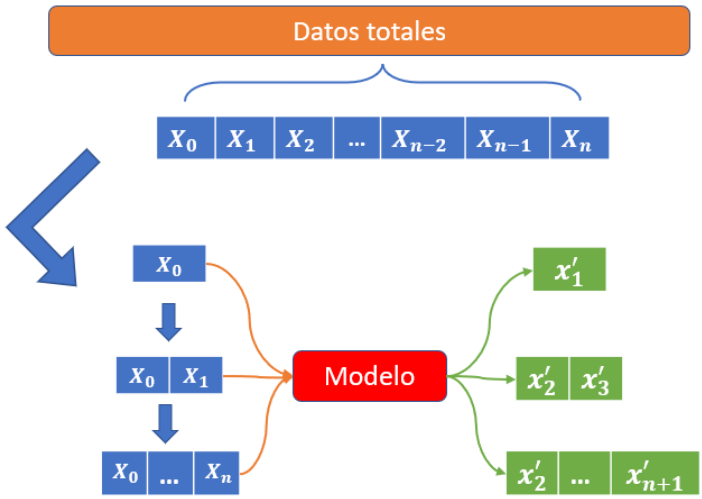


Figura 4.6: Diagrama de representación del entrenamiento con valores predichos del modelo. El vector resultante corresponde a $[X_1, ...X_{n+1}]$ siendo este último valor el de interés para las regresiones con ventanas extendidas ya que es provisto por un entrenamiento con tanto por valores previos reales como predichos.

4.2.2.3. Hiperparámetros relevantes.

Debido a que este sistema utiliza modelos de cálculo de atención, y, realiza el entrenamiento siguiendo la generación de series consecuentes aleatorias dentro de los datos totales. Entonces los hiperparámetros mas importantes en este sistema son los siguientes:

- Dimensión de proyecciones de las capas de atención (d_{model}): este controla el tamaño de las matrices Q, K y V al ser procesadas por los módulos de atención.
- Cantidad de cabezas: Número de cabezas utilizadas para realizar el cálculo de las atenciones paralelas descritas en la ecuaciones 2.8 y 2.9, cabe destacar que debido al cálculo matricial mostrado en estas expresiones el valor de este debe ser capaz de dividir a la dimensión de proyecciones.
- Cantidad de capas de *Transformer Encoder*: valor que representa cuantas veces se desea realizar el proceso de cálculo de atención más la normalización, junto con el proceso de la capa *feed-forward*.
- Tamaño de la capa feed-forward: cantidad de neuronas de las capas de atención.
- Largo de series de entrenamiento: tamaño de las series temporales extraídas del conjunto total para la realización del entrenamiento.
- Largo de ventana: cantidad de predicciones futuras que se desea realizar utilizando valores de predicción previos.

- Cantidad de épocas: este indica cuantas series aleatorias de datos consecutivos se desean generar durante el entrenamiento.
- Método de entrenamiento: Enfoque del entrenamiento, puede ser para *one step* o *sampling* para predicciones extensas.

4.3. Metodología e Implementación Computacional

Debido a la gran cantidad de de modelos de predicción existentes en la actualidad, el desarrollo de redes neuronales, random forest entre otros modelos se encuentran ya implementados computacionalmente en diversas librerías de lenguajes de programación, aunque algunos sistemas no estan del todo desarrollados, por ende en esta sección se hablará sobre la forma en la cual se abordó la programación de estos sistemas y cuales fueron las librerías utilizadas para el desarrollo de estos modelos.

4.3.1. Scikit-Learn y Pytorch

Scikit Learn (figura 4.7a) es un proyecto que es impulsado debido a la colaboración de diversos científicos de datos y desarrolladores alrededor del mundo. Sin embargo, también reciben subvenciones institucionales y privadas para asegurar su sostenibilidad[34].

La librería desarrollada por este proyecto recibe el nombre de `sklearn` y es una de las librerías de machine learning más populares que existen. Desde regresores lineales a complejos modelos ensamblados pueden ser utilizados para realizar tareas de clasificación, regresión y agrupación en python, por lo que, esta librería fue utilizada para la implementación de todos los modelos de machine learning supervisado.

Por otro lado, debido a que los transformers utilizan módulos y funciones propias de aprendizaje profundo o *deep learning*, se trabajó con otra librería que permitiese utilizar estos modelos. Debido a esto, la implementación del transformer adaptado se realizó mediante la librería `Pytorch`[35].

Esta librería esta diseñada para programar modelos de aprendizaje automático, principalmente, aquellos que utilizan *deep learning* en su arquitectura. Para ello utiliza **tensores**, los cuales corresponden a arreglos multidimensionales de valores que pueden ser vectores, matrices o conjuntos en una mayor cantidad de dimensiones. Cabe destacar que esta última característica (uso de arreglos multidimensionales), también, puede desarrollarse con otras librerías, pero la principal ventaja de utilizar `Pytorch` es que esta permite el uso de *Data Paralellism*[36]. Este consiste en realizar paralelismo de datos mediante la distribución del trabajo en diferentes núcleos de CPU o GPU[10].

[10] Unidad de Procesamiento Gráfico, este hardware permite desarrollar los cálculos de forma paralela acelerando los procesos de los modelos.

(a) (b)

Figura 4.7: Logos de librerías

Para utilizar la aceleración mediante la unidad de procesamiento gráfico, es necesario contar con una. Tras instalar correctamente `Pytorch` se debe instalar los drivers necesarios para que esta pueda ser controlada por el IDE donde se desarrolle la programación. Otra alternativa es utilizar entornos virtuales o una nube, donde la plataforma que de el servicio ofrezca una GPU, evitando la programación local y, desarrollando las tareas con el acelerador estrictamente en la nube. El desarrollo del entrenamiento del *transformer adaptado* se realizó utilizando una GPU, lo cual fue posible debido a que *Google Colaboratory* ofrece una al utilizar las librerías de *deep learning*.

4.3.2. Datos

El entrenamiento tomó el 70 % de los datos total disponibles dejando el 30 % restante para las pruebas. La información utilizada es la mostrada en las tablas 3.3 y 3.4, siendo esta última la que se utilizó para el entrenamiento de la Red Neuronal y el Random Forest.

Por otro lado, para el transformer adaptado solo se utilizaron los datos de la demanda y fecha (tabla 4.1), eliminando los valores de las predicciones del TSO español. Sin embargo, se le añadió un preprocesamiento extra.

4.3.2.1. Codificación posicional

Los módulos de atención de los transformers no tienen nociones sobre las posiciones de los elementos procesados, por lo que, se debe incorporar esta información a las series procesadas. Como se trabajaron con valores de la demanda energética registradas por hora, entonces, se usaron los valores de hora, mes y año presentes en los datos de las fechas.

time	total load actual
2015/01/01 00:00:00 + 01:00	25385
2015/01/01 01:00:00 + 01:00	24382
...	...
2018/12/31 22:00:00 + 01:00	25801
2018/12/31 23:00:00 + 01:00	24455

Tabla 4.1: Demanda obtenida por el TSO español junto con su fecha de registro.

Como cada dato difiere en al menos un valor temporal, se extrajeron la hora, día y mes las cuales se dividieron por 24, 30 y 12 respectivamente[11]. Cada uno de estos valores fueron llevados a 2 funciones trigonométricas: **seno y coseno**, por lo que, cada muestra de demanda fue extendida

[11] Esto se hizo considerando horas en un día, días en un mes y meses en un año

con 6 valores posicionales (tabla 4.2). Matemáticamente se realizó lo siguiente:

$$\sin_h(x_i) = \sin\left(2\pi \cdot \frac{t_h}{24}\right) \tag{4.2}$$

$$\cos_h(x_i) = \cos\left(2\pi \cdot \frac{t_h}{24}\right) \tag{4.3}$$

$$\sin_d(x_i) = \sin\left(2\pi \cdot \frac{t_d}{30}\right) \tag{4.4}$$

$$\cos_d(x_i) = \cos\left(2\pi \cdot \frac{t_d}{30}\right) \tag{4.5}$$

$$\sin_m(x_i) = \sin\left(2\pi \cdot \frac{t_m}{12}\right) \tag{4.6}$$

$$\cos_m(x_i) = \cos\left(2\pi \cdot \frac{t_m}{12}\right) \tag{4.7}$$

Donde t_h, t_d, t_m corresponden al valor numérico de la hora, día y mes registrado. Con esto los datos tuvieron referencia posicional para procesarse mediante un módulo de atención.

time	total load actual	$\sin_h$	$\cos_h$	$\sin_d$	$\cos_d$	$\sin_m$	$\cos_m$
2015/01/01 00:00:00	25385	0	1	0.207	0.978	0.5	0.866
2015/01/01 01:00:00	24382	0.258	0.965	0.207	0.978	0.5	0.866
...	...	...	...	...	...	...	...
2018/12/31 22:00:00	25801	-0.5	0.866	0.2079	0.978	0	1
2018/12/31 23:00:00	24455	-0.25	0.965	0.207	0.978	0	1

Tabla 4.2: Demanda del consumo energético español junto con sus posiciones codificadas.

Los datos que se llevaron al transformer adaptado fueron las muestras de la tabla 4.2 sin contar los registros.

4.3.3. Configuración de modelos de machine learning supervisado.

Cada modelo puede tomar una gran cantidad de valores para algunos hiperparámetros, lo que da una enorme cantidad de modelos posibles para un tipo de regresor. Para encontrar la configuración óptima en los modelos de esta subsección, se aplicó una *búsqueda de grilla o* ***grid search***[37], este proceso consiste en ejecutar el entrenamiento del mismo regresor pero con diferentes valores en sus hiperparámetros. Esta característica es uno de los tantos objetos disponibles en la librería `sklearn`.

Como se mencionó anteriormente, se utilizaron 2 modelos: Una Red Neuronal Artificial y un Random Forest. Se buscó la mejor configuración de hiperparámetros para el tamaño de la capa oculta (entre 8, 50 y 100 neuronas) y el tipo de función de activación (sigmoide o RELU). La mejor configuración obtenida se aprecia en la tabla 4.3.

Estos valores muestran la construcción de una red sencilla. Por otro lado los hiperparámetros del sistema Random Forest se encuentran en la tabla 4.4.

Esta configuración se obtuvo al probar 50, 80 y 100 árboles, y variando el criterio de separación

Hiperparámetro	Valor
Cantidad de capas ocultas	1
Tamaño de la capa oculta	200 neuronas
Función de activación de la capa oculta	RELU
Cantidad de épocas	200
Tasas de aprendizaje	0.001
Optimizador	ADAM

Tabla 4.3: Hiperparámetros de la Red Neuronal utilizada.

Hiperparámetro	Valor
Cantidad de árboles	150
Criterio de separación	Error cuadrático
Cantidad de muestras mínimas para una separación	2
Profundidad máxima	Ajustable

Tabla 4.4: Hiperparámetros del sistema Random Forest utilizado.

entre *error cuadrático y absoluto*. El resto fueron los valores definidos por la librería, donde las profundidades máximas se extendieron extienden hasta ser menor que las muestras mínimas de separación.

4.3.4. Configuración del transformer adaptado.

Los datos mostrados en la tabla 4.2 fueron procesados por un *transformer* adaptado cuyo sistema tuvo la configuración mostrada en la tabla 4.5.

Hiperparámetro	Valor
d_{model}	7
Nº cabezas	7
Nº capas *trans. encoder*	3
Tamaño ff atencional	2048
Largo series entrenamiento	2000
Largo ventana de predicción	48
Épocas	6000
Método de entrenamiento	One-Step

Tabla 4.5: Configuración del transformer adaptado

Los resultados de todos estos modelos se muestran en el capítulo 6 de este informe.

Capítulo 5

Modelos difusos

Los sistemas principales de la investigación desarrollada en esta memoria corresponden a modelos de series temporales difusas implementadas computacionalmente, las cuales se aplicaron a los datos de la demanda energética explicados en el capitulo anterior. En esta sección se explica la implementación computacional de estos modelos junto con una explicación teórica detallada del proceso de entrenamiento

Los modelos descritos a continuación fueron extraidos de la tesis doctoral **Scalable Models for Probabilistic Forecasting with Fuzzy Time Series** de Petronio Candido e Silva. En este trabajo se encuentran una mayor cantidad de modelos difusos que los mostrados en este capítulo, pero algunos de los más importantes son descritos a continuación.

5.1. Modelos difusos con una variable

Para estos modelos solo se utilizaron los valores de la demanda (tabla 3.3, columna `total load actual`) para su construcción, entrenamiento y prueba.

5.1.1. Método de Series Temporales difusas de alto Orden (HOFTS)

El modelo presentado aquí corresponde a un desarrollo realizado por Petronio e Silva en 2019[38]. El método *High Order Fuzzy Time Series* o **HOFTS** define sus reglas mediante la forma $LHS \longrightarrow RHS$, es decir como antecedente y consecuente tomando el orden en consideración para el proceso de predicción. Es decir, sigue la siguiente expresión.

$$f(t - L(\Omega - 1)), ..., F(t - L(0)) \longrightarrow f(t + 1) \tag{5.1}$$

Al implementar este sistema, se tienen valores definidos para su uso el orden Ω y la cantidad de particiones k deben ser definidas por el usuario, es decir:

Este modelo no añade importancia a las reglas y conjuntos difusos generados, por lo que este sistema se puede extender para aplicar esta característica.

5.1.1.1. Proceso de entrenamiento

Este modelo toma como entrada una muestra de entrenamiento Y, el número de particiones k, el número de retrasos Ω y los otros valores definidos en la tabla 5.1 y procede de la siguiente manera:

- *Paso 1: Partición*

Parámetro	Valor
Ω	Definido por el usuario
k	Definido por el usuario
$\prod$	Grid
μ	triangular
α cut	0
L	$\{1, ..., \Omega\}$

Tabla 5.1: Hiperparámetros definidos para los modelos HOFTS y WHOFTS

1. **Definición del Universo de Discurso:** el UoD define el espacio muestral, es decir, los límites conocidos de la serie temporal Y tal que $U = [\min - D_1, \max(Y) + D_2]$, donde $D_1 = \min(Y) \times 0.2$ y $D_2 = \max(Y) \times 0.2$ son usados para extrapolar los límites conocidos a un margen seguro.
2. **Partición del Universo de Discurso:** se realiza una separación a U en k intervalos U_i con puntos medios c_i y se aplica la partición definida en $\prod$ para definir los largos de todos los intervalos.
3. **Definición de la variable lingüística $\tilde{A}$:** para cada intervalo U_i se crea un conjunto difusos A_i superpuesto con la función de membresía μ_A. EL punto medio del conjunto difuso A_i se convierte en c_i, el límite inferior $l_i = c_{i-1}$ y el límite superior $u_i = c_{i+1}$ $\forall i > 0$ y $i < k$, y $l_0 = \min U, l_k = \max U$. Cada conjunto difuso $A_i \in \tilde{A}$ es un término lingüístico de la variable lingüística $\tilde{A}$.

- *Paso 2: Fuzzyficación*

1. Se transforma la series temporales original Y en series temporales difusas F, donde cada dato $f(t) \in F$ corresponde a un arreglo de dimensiones $1 \times k$, el cual contiene los valores fuzzyficados de $y(t) \in Y$ con respecto a los términos lingüísticos $A_i \in \tilde{A}$, donde la membresía definida es mayor que el valor α cut, es decir $f(t) = \{A_i \mid \mu_{A_i}(y(t)) \geq \alpha_i \in \tilde{A}\}$

- *Paso 3: Inducción de reglas*

1. **Generación de patrones con alto orden** (Función 5.1) : los patrones temporales difusos tienen formato $A_{i0}, ..., A_{i\Omega} \longrightarrow A_j$, donde el precedente (o LHS) es $f(t - L(\Omega)) = A_{i0}, f(t - L(\Omega - 1)) = A_{i1}, ..., f(t - L(0)) = A_{i\Omega}$ y el consecuente (o RHS) es $f(t+1) = A_j$
2. **Generación de las reglas base:** se seleccionan todos los patrones temporales con el mismo precedente y se agrupan con sus conjuntos de consecuentes creando una regla con el formato $A_{i0}, ..., A_{i\Omega} \longrightarrow A_k, A_j, ...$, donde LHS es $f(t+1) \in \{A_k, A_j, ...\}$. Cada regla puede entenderse como un conjuntos ponderado de probabilidades las cuales pueden suceder en tiempo $t + 1$ (consecuente) donde un cierto precedente $A_{i0}, ..., A_{i\Omega}$ es identificado en los retrasos en L.

5.1.1.2. Proceso de pronóstico

Para esta parte, el modelo toma como entrada una muestra de prueba Y y la realiza las predicciones en un horizonte H (el valor predeterminado es de 1) cuya salida corresponde al arreglo de predicciones $\hat{y}_H$

1. **Fuzzyficación:** se calcula el grado de membresía μ_{ti} para cada $y(t) \in Y$ donde $t \in L$ y cada conjunto difuso $A_i \in \tilde{A}$ tal que $\mu_{ti} = \mu_{A_i}(y(t))$.

2. **Matching de Reglas:** se seleccionan las K reglas donde todos los conjuntos difusos A_i en LHS tienen $\mu_{ti} = \mu_{A_i}(y(t))$. Se muestra el grado de membresía usando la función como norma T[12].

$$\mu_j = \min_{t \in L, i \in \tilde{A}} \mu_{ti} \tag{5.2}$$

3. **Deffuzyficación:**

   a) **Puntos medio de reglas:** para cada regla j, se calcula el punto medio mp_j siguiendo la siguiente expresión, donde c_i es el parámetro c de la función μ del conjunto difuso A_i:

$$mp_j = |RHS|^{-1} \sum_{i \in RHS} c_i \tag{5.3}$$

   b) **Defuzzyficación:** se calcula la predicción como una suma ponderada de los puntos medios de las reglas mp_j con los valores de membresía μ_j para cada regla j seleccionada:

$$\hat{y}(t+1) = \frac{\sum_{j \in K} \mu_j \cdot mp_j}{\sum_{j \in K} \mu_j} \tag{5.4}$$

4. **Pronóstico para ventanas extensas:** si el horizonte $H > 1$, se define $\hat{y}_H = \{\hat{y}(t+1)\}$ como el conjunto de predicciones y se repite los siguientes pasos para cada $h = 2, ..., H$, en caso contrario, se retorna $\hat{y}(t+1)$.

   a) Se llama recursivamente el método de pronóstico usando $\hat{y}(t+h-1)$ como entrada para generar $\hat{y}(t+h)$.

   b) Se agrega $\hat{y}(t+h)$ a $\hat{y}_H$ y cuando $h = H$ se retorna $\hat{y}_H$

5.1.2. Extensión con Ponderaciones (WHOFTS)

Este método se llama **Método de Series Temporales difusas de alto Orden Ponderadas**(*Weighted Fuzzy Time Series*) y, como mencionó previamente, este modelo es una extensión del sistema difuso anterior, donde las reglas poseen importancias distintas, es decir, ponderaciones las cuales se añaden en la fase de defuzzyficación. Para realizar este proceso se debe realizar un cambio en el paso 3.2 del proceso de entrenamiento mostrado anteriormente.

- Paso 3.2 **Generación de reglas base:** se seleccionan todo los patrones temporales que tengan el mismo precedente y se agrupan sus conjuntos de consecuentes para crear una regla con el formato $A_{i0}, ..., A_{i\Omega} \longrightarrow w_k \cdot A_k, w_j \cdot A_j, ...$, donde LHS es $f(t - L(\Omega)) = A_{i0}, f(t - L(\Omega - 1)) = A_{i1}, ..., f(t - L(0)) = A_{i\Omega}$ y RHS es $f(t+1) \in \{A_k, A_j, ...\}$ y las ponderaciones $w_k, w_j, ...$ son normalizadas por cada patrón temporal tal que:

$$w_i = \frac{\#A_i}{\#RHS} \; \forall A_i \in RHS \tag{5.5}$$

[12] Norma triangular, es la generalización del operador de intersecciones

donde $\#A_i$ es el número de ocurrencias de A_i en los patrones temporales con el mismo precedente LHS y $\#RHS$ es el total de patrones temporales con el mismo precedente LHS. Cada regla puede ser entendida como el conjunto ponderado de probabilidades en el cual puede suceder en el tiempo $t+1$ (consecuente) cuando un cierto precedente $A_{i0}, ..., A_{i\Omega}$ es identificado con el los L retrasos previos (precedente).

Las ponderaciones w_i se ajustan a la condición $\sum_{i=1}^{k} = 1$. Además, dado que es un proceso importante durante el entrenamiento del modelo, estos también afectan el proceso de pronóstico presentado en la sección 5.2.1.2, por ende, este proceso también tiene un cambio, este en el punto 3.a:

- Paso 3.a **Puntos medios de reglas**: para cada regla j seleccionada, se calcula el punto medio mp_j de la siguiente forma:

$$mp_j = \sum_{i \in RHS} w_i \cdot c_i \tag{5.6}$$

  Donde c_i es el parámetro c de la función μ del conjunto difuso A_i.

5.1.3. Modelo de ponderaciones probabilísticas (PWFTS)

El método de **Series Difusas Ponderadas Probabilística o PWFTS[13]**, corresponde a un modelo desarrollado por Petronio e Silva en 2019[39]. Este sistema también permite desarrollar predicciones en el siguiente instante o en ventanas extensas. Este método genera predicciones que manejan 2 tipos de incertidumbre: medidas difusas y comportamiento estocástico. Estas características son inducidas en el proceso de entrenamiento y pronóstico.

Este modelo posee los mismos parámetros mostrados en la tabla 5.1 y el modelo de reglas que sigue se denomina **FLRG ponderados probabilísticamente**. Este describe el comportamiento futuro más probable (RHS) dado un comportamiento pasado (LHS). Dada una muestra, existen muchas reglas aplicables con diferentes activaciones (las membresías ponderadas) y todas estas son tomadas en cuenta.

Dado un FLRG de la forma $LHS \longrightarrow RHS$ donde $F(t-1) = LHS$ y $F(t) \in RHS$, el mejor regresor puede ser descrito como $\mathbb{E}[F(t+1) \mid F(t), ...]$ para $\mathbb{E}[RHS \mid LHS]$. Las ponderaciones asignadas a estas reglas son densidad de probabilidad de los conjuntos difusos medidos durante el entrenamiento.

Dado que este modelo extrae su conocimiento a través de reglas ponderadas, este ayuda en diversas tareas de extracción de conocimiento, ademas, sirve para poder analizar los procesos de distribución del método en la extensión multivariable.

5.1.3.1. Probabilidades empíricas difusas

Este concepto es el núcleo del modelo PWFTS. Esto es utilizado para calcular las ponderaciones del modelo. La noción inicial de probabilidad difusa fue propuesta por Zadeh en 1968[40], $P(A) = E[\mu_A]$. Este indica que se necesita conocimiento previo de la densidad de probabilidad

[13] Probabilistic Weighted Fuzzy Time Series

sobre el universo de discurso. Como las distribuciones para Y son desconocidas, una distribución empírica debe tomar lugar. La definición más simple para esta es la frecuencia relativa sobre un valor discreto en un rango continuo de valores. La teoría difusa entrega diferentes visiones en la Teoría de Probabilidad tradicional debido a que afecta a la forma en la que los eventos son contados.

Sin embargo, en los conjuntos difusos la noción de evento es más complejo debido a que un mismo valor puede estar en diferentes conjuntos con distinto valor de membresía. Dado esto, en lugar de contabilizar las ocurrencias del evento, las ocurrencias parciales son contadas usando el valor de membresía. Este método se conoce como *frecuencias difusas* el cual fue desarrollado por Luo y Bridges[41].

Dado un espacio muestral U y los conjuntos difusos $\tilde{A}$ sobre U , la función de partición $Z_{A_j}, \forall A_j \in \tilde{A}$, es la integral de la función de membresía μ_{A_j} sobre el espacio muestral U, tal que $Z_{A_j} = \int_U \mu_{A_j}(y)dy$ o la aproximación discreta $Z_{A_j} = \sum_{y \in U} \mu_{A_j}(y)$. Con Z_{A_j} es posible encontrar una aproximación a la probabilidad empírica de un conjunto difuso $A_j \in \tilde{A}$ como la suma de las membresías $\mu_{A_j}(y), \forall y \in U$ dividido por la suma de las funciones de partición Z_{A_j} de todos los conjuntos difusos $A_j \in \tilde{A}$. Es decir, realizando la siguiente expresión:

$$P(A_j) = \frac{\sum_{y \in U} \mu_{A_j}(y)}{\sum_{A_j \in \tilde{A}} Z_{A_j}} \tag{5.7}$$

Esta ecuación representa la probabilidad empírica $P(A_j)$ y es distribuida uniformemente sobre la forma de la función de membresía μ_{A_j}, y el punto y es una parte de esta forma cuya área equivale al valor $\mu_{A_j}(y)$, por lo que el área de μ_{A_j} es Z_{A_j}. $P(A_i)$ es medido desde una muestra de Y y su valor empírico es una aproximación de la verdadera probabilidad. Esta aproximación es utilizada en la siguiente expresión.

$$P(y|A_j) = P(A_j) \cdot \frac{\mu_{A_j}(y)}{Z_{A_j}} \tag{5.8}$$

Esta expresión muestra la aproximación de la probabilidad condicional de un valor $y \in U$ dado un conjunto difuso $A_j \in \tilde{A}$. Utilizando las ecuaciones 5.7 y 5.8 junto con la ley de la probabilidad total, la probabilidad empiríca $P(y)$ se aproxima mediante la siguiente expresión:

$$P(y|\tilde{A}) = \sum_{A_j \in \tilde{A}} P(y|A_j) \cdot P(A_j) \tag{5.9}$$

La ventaja de esta aproximación es que se puede obtener $P(A_j)$ de una muestra del dataset de series temporales Y. La precisión de esta probabilidad es determinada, principalmente, por el número de particiones k y el universo de discurso U.

5.1.3.2. Proceso de entrenamiento

Este modelo posee 7 pasos para aprender la dinámica temporal de las series temporales a través de los datos de entrenamiento Y y representarlo en un modelo difuso probabilístico, a saber: el Grupo de Patrones Temporales Difusos Ponderados Probabilísticamente o PWFTPG[14]

1. **Definición del Universo de Discurso:** se define U como el espacio muestral de las muestras

[14] Siglas en inglés de Probabilistic Weighted Fuzzy Temporal Pattern Group.

de entrenamiento Y tal que $U = \min(Y) - D_1, \max(Y) + D_2$, donde $[\min(Y), \max(Y)]$ es el rango de las muestras de los datos de entrada y D_1, D_2 son valores utilizados para extrapolar el rango teniendo valores $D_1 = 0.2 \cdot \min(Y), D_2 = 0.2 \cdot \max Y$

2. **Partición:** se separa U en k intervalos uniformes u_i, para $i = 1, ..., k$ con puntos medios mp_i.

3. **Definición de la variable lingüística $\tilde{A}$:** se crean k conjuntos difusos superpuestos con funciones de membresía μ_{A_j}, relacionados en un intervalo u_j y puntos medios mp_j. Cada conjunto difuso $A_j \in \tilde{A}$ es un término lingüístico de la variable lingüística $\tilde{A}$.

4. **Fuzzyficación:** se transforman los valores númericos de las series temporales Y a Series Temporales Difusas F (FTS), donde cada punto medio $f(t) \in F$ es una k-tupla con el valor de membresía de $y(t)$ con respecto al término lingüístico $A_j \in \tilde{A}$ tal que:

$$f(t) = [\mu_{A_1}(y(t)), \mu_{A_2}(y(t)), ..., \mu_{A_k}(y(t))] \tag{5.10}$$

5. **Generación del conjunto FPT:** el Patrón Temporal Difuso o FTP[15] es una regla difusa con formato $A_i \longrightarrow A_k$ que indica la sucesión temporal donde el precedente (LHS) es $A_i \in f(t)$, y el consecuente (RHS) es $A_k \in f(t+1)$, para cada posible par de $A_i \times A_k$ de valores de membresía mayores que 0, i.e $\{A_i \longrightarrow A_k\} \forall A_i \in f(t) | \mu_{A_i}(y(t)) > 0$ y $\forall A_k \in f(t+1) | \mu_{A_k}(y(t+1)) > 0$. Por ende, $A_i \longrightarrow A_k$ puede ser leído como "Si $y(t)$ es A_i entonces $y(t+1)$ es A_k". Como cada $f(t) \in F$ es un vector disperso de valores de membresía, entonces, existen muchas combinaciones posibles de conjuntos difusos de 2 vectores secuenciales $f(t)$ y $f(t+1)$, por lo que, para cada par secuencial en F posiblemente más de una FTP es generada.

6. **Generación del conjunto FTPG:** un grupo de patrones temporales difusos o FTPG[16] representa el conjunto de todos los FTP con el mismo LHS y la unión de sus RHS, con el formato $A_i \longrightarrow A_k, A_j, ...$, donde el LHS es $f(t) = A_i$ y el RHS es $f(t+1) \in \{A_k, A_j\}$. Cada FTPG puede ser entendido como un conjunto de posibilidades las cuales pueden suceder en el tiempo $t+1$ (consecuente) cuando un cierto conjunto A_i es identificado en el tiempo t (precedente).

7. **Cálculo de probabilidades empíricas:** las ponderaciones probabilísticas FTPG-PWFTPG añade ponderaciones en LHS y RHS que miden su probabilidad empírica difusa. Cada PWFTPG tiene el formato $\pi_j \cdot A_j \longrightarrow w_{j0} \cdot A_0, ..., w_{jk} \cdot A_K$ para $j = 1, ..., k$. El conjunto de todos los PWFTPG forma el modelo, y su tamaño depende del número de particiones k y puede ser representado de forma matricial, pero las ponderaciones w_{ij} forman una matriz muy dispersa, lo que justifica el uso de estructura de datos optimizadas para su representación. Esta se puede apreciar en la siguiente expresión.

$$\begin{aligned} \pi_1 \cdot A_1 &\longrightarrow w_{11} \cdot A_1, ..., w_{1k} \cdot A_k \\ ... \ ... \ ... \ ... \ ... \ ... \ ... \ ... \ ... \ ... \ ... \ ... \ ... &\\ \pi_1 \cdot A_k &\longrightarrow w_{k1} \cdot A_1, ..., w_{kk} \cdot A_k \end{aligned} \tag{5.11}$$

Cada ponderación w_j es asociada con un conjunto difuso A_i en RHS del FTP cuyo LHS es A_j y la suma normalizada de todas los valores de membresía de todos los FTP donde $LHS = A_j$ y

[15] Siglas en inglés de Fuzzy Temporal Pattern
[16] Siglas en inglés de Fuzzy Temporal Pattern Group

$RHS = A_i$, entonces la ponderación w_{ji} puede ser entendida como la probabilidad condicional empírica del conjunto difuso A_i en el tiempo $t+1$ cuando el conjunto difuso A_j es identificado en el tiempo t o en $P(A_i^{t+1}|A_j^t)$ tal que la condición de la ecuación 5.12 deba ser cumplida para cada $A_j \in LHS$.

$$\sum_{i \in \tilde{A}} w_{ji} = 1 \quad \forall A_j \in \tilde{A} \tag{5.12}$$

5.1.3.3. Proceso de pronóstico

Este procedimiento posee 4 pasos, donde este toma como entrada el tipo de predicciones que se desea hacer a partir de una muestra $y(t) \in U$ y usa el modelo PWFTPG, resultado del procedimiento anterior.

1. **Fuzzyficación:** dada una entrada $y(t) \in Y$, se encuentran los valores fuzzyficados $f(t) = \{A_j | \mu_{A_j}(y(t)) > \alpha\}$.
2. **Matching de Patrones:** se localizan todos PWFTPGs cuyo LHS sea $f(t)$.
3. **Predicción:** la densidad de $f(t+1)$ es dada por los conjuntos RHS por cada PWFTPG enlazado (match).
4. **Defuzzyficación:** en este punto se pueden desarrollar 3 tipos de predicciones: probabilística, intervalos o puntual. Dada la naturaleza de esta memoria se explicará solo el proceso para predicciones puntuales.

   Para generar la predicción $\hat{y}(t+1)$ desde la distribución obtenida $P(\cdot|y(t))$ solo se debe aplicar la esperanza, i.e $\hat{y}(t+1) = \mathbb{E}[P(y(t+1)|y(t))]$. Esto es costoso computacionalmente debido al cálculo de $P(y(t+1)|y(t))$. Para generar las predicciones puntuales se calcula la esperanza $\mathbb{E}[A_j]$ de cada PWFTPG (ecuación 5.13), donde mp_i es el punto medio de cada conjunto difuso $A_i \in RHS$. La esperanza $\mathbb{E}[A_j]$ para cada PWFTPG A_j es constante y puede ser precalculada. La predicción final $\hat{y}(t+1)$ entonces es la suma de todas estas esperanzas ponderadas por $P(y(t)|A_j)$ (ecuación 5.14)

$$\mathbb{E}[A_j] = \sum_{i \in A_j^{RHS}} w_{ji} \cdot mp_i \tag{5.13}$$

$$\hat{y}(t+1) = \mathbb{E}[\tilde{A}|y(t)] = \sum_{A_j \in \tilde{A}} \frac{P(y(t)|A_j) \cdot \mathbb{E}[A_j]}{\sum_{A_j \in \tilde{A}} P(y(t)|A_j)} \tag{5.14}$$

Pronósticos para ventanas extensas

Las predicciones mostradas en la sección anterior es un método *one step ahead*[17]. Para poder extender el pronóstico a ventanas más extensas se debe aplicar un método iterativo, el cual en el paso (t+1) sea calculado con los métodos presentes y la salida sea retroalimentada como entrada para los H pasos necesarios. Desde el paso $t+2$ en adelante, $h \in [t+2, t+H]$ es el nuevo indice temporal. El enfoque más sencillo es realizar la predicción de $y(h+1)$ con la entrada $y(h)$.

[17] Métodos para predecir solo el valor siguiente.

5.2. Modelos difusos multivariables

A pesar de los diversos enfoques que se encuentran en la literatura, el tratar con problemas multivariables y series espacio-temporales representa un desafío para los métodos de series temporales difusas, debido a que estos aumentan la complejidad de las reglas generadas y aumentan la dimensión de estas.

Las series temporales multivariables son conjuntos de secuencias de vectores de la forma $Y \in \mathbb{R}^n$ donde $n = |V|$, siendo V la cantidad de características de Y. Cada vector $y(t) \in Y$ contiene todas las características $V_i \in V$ y esta una dependencia temporal entre los datos de interés con las características, por ende, el índice temporal $t \in T$ debe ser respetado.

En la literatura de Series Temporales Difusas es común encontrar métodos de *clusterización* para abordar el problema de dimensionalidad, convirtiendo los datos multivariables en solamente de una variable. Un ejemplo de estos fue propuesto por Chen y Chang en 2010[42] y Sun en 2015[43] Medias Difusas-C (Fuzzy C-Means) junto con algoritmos de clusterización para crear los FLRG multivariables.

5.2.1. Método Multivariable de Series Temporales Difusas (MVFTS)

Este sistema extiende los modelos difusos simples de una sola variable. Este modelo fue diseñado para permitir que varios modelos se entrenen individualmente con subconjuntos de un conjunto de datos mayor y, luego, se fusionen en un solo modelo, característica que mejora el rendimiento de la creación de modelos al permitir su distribución.

Alias	Parámetro	Dominio	Descripción
k_i	Número de particiones	$\mathbb{N}^+$	Cantidad de conjuntos difusos que se crean en la variable lingüística $\tilde{V}_i$
μ	Función de membresía	$\mu : U \longrightarrow [0, 1]$	Función que mide el valor de membresía $y \in U$ a un conjunto difuso
α	α cut	[0,1]	El valor mínimo de membresía para que pueda ser parte de la fuzzyficación

Tabla 5.2: Hiperparámetros del modelo difuso multivariable.

Para cada variable seleccionada $V_i \in V$ en Y, MVFTS incorpora varias características presentes en al literatura, representadas en la tabla 5.2, dando versatilidad y flexibilidad al modelo. Este método también es compuesto por un **proceso de entrenamiento** y **proceso de pronóstico**, teniendo, también una extensión para que pueda añadir importancia a las reglas generadas.

5.2.1.1. Proceso de entrenamiento.

Este tiene 3 escenarios responsables de crear un método multivariable de series temporales difusas. El modelo final consiste en un conjunto de de variables V, una variable lingüística $\tilde{V}_i$ para cada $V_i \in V$ y un conjunto de reglas difusas ponderadas sobre las variables lingüísticas $\tilde{V}_i$. El proceso de entrenamiento es detallado a continuación.

1. **Partición:**

a) **Definición de U_V:** el Universo de Discurso U_{V_i} define el espacio muestral, i.e los límites conocidos de de la variable V_i , tal que $U_{V_i} = [\min(Y^{V_i}) - D_1, \max(Y^{V_i}) + D_2]$, donde $D_1 = \min(Y^{V_i}) \times 0.2$ son usados para extrapolar los límites conocidos como un margen de seguridad $_i \in V$

b) **Partición de U_{V_i}:** se separa U_{V_i} en k_i intervalos U_j con puntos medios c_j para $j = 0, ..., k_i$, donde se encuentran todos los intervalos con el mismo largo.

c) **Definición de la variable lingüística $\tilde{V_i}$:** para cada intervalo $U_j \in U_{V_i}$ se crea un conjuntos difuso superpuesto $A_j^{V_i}$ con función de membresía $\mu_{A_j^{V_i}}$. El punto medio del conjunto difuso $A_j^{V_i}$ sera c_j, el límite inferior $l_j = c_{j-1}$ y el límite superior $u_j = c_{j+1} \quad \forall j > 0$ y $j < k_i$, y $l_0 = \min(U_V), l_k = \max(U_{V_i})$. Cada conjunto difuso $A_j^{V_i}$ es un término lingüístico de la variable lingüística $\tilde{V_i}$

2. **Fuzzyficación:** se transforman los valores originales numéricos de las series temporales Y en Series Temporales Difusas F, donde cada muestra $f(t) \in F$ es un arreglo con dimensiones $n \times k$ que contiene los valores fuzzyficados de $y(t) \in Y$ con respecto a los términos lingüísticos $A_j^{V_i} \in V_i$, donde los valores de membresía deben ser mayores que α cut, es decir

$$f(t) = \{A_j^{V_i} | \mu_{A_j}^{V_i}(y(t)^{V_i}) \geq \alpha \quad \forall A_j^{V_i} \in \tilde{V_i} \tag{5.15}$$

3. **Inducción de reglas:**

a) **Generación de patrones temporales:** los patrones temporales difusos asocian los valores fuzzyficados V a un conjunto de posibles valores de la variable objetivo $*V$, tal que $V \longrightarrow *V$, con el formato $A_j^{V_0}, ..., A_j^{V_n} \longrightarrow A_j^{*V}$ donde el LHS es $f(t-1) = A_j^{V_i}, \forall V_i \in V$ y el RHS es $f(t+1) = A_j^{*V}, A_j^{*V}, A_j^{*V} \in *\tilde{V}$.

b) **Generación de las reglas base:** se seleccionan todos los patrones temporales que tengan el mismo precedente y se agrupan con sus consecuentes creando una regla con el formato $V \longrightarrow w_k \cdot A_k^{*V}, w_j \cdot A_j^{*V}, ...,$ donde LHS es $f(t-1) = A_j^{V_i}, \forall V_i \in V$ y RHS es $f(t+1) \in \{A_k^{*V}, A_j^{*V}, ...\}$. Cada regla puede ser entendida como un conjunto ponderado de probabilidades que pueden suceder en el tiempo $t+1$ dado un cierto precedente $A_{i0}, ..., A_{i\Omega}$ es identificado en un previo retraso (precedente).

5.2.1.2. Proceso de pronóstico.

Este proceso tiene como finalidad obtener una estimación $\hat{y}(t+1)$ para la variable objetivo $*V$ dada una entrada Y, usando las variables lingüísticas $\tilde{V_i}$ y las reglas difusas inducidas obtenidas en el entrenamiento del modelo.

1. **Fuzzyficación:** se calcula el valor de membresía μ_{ji} para $y(t-1) \in Y$ tal que $\mu_{ji} = \mu_{A_j^{V_i}}(y(t-1))$ para cada $A_j \in \tilde{V_i}$ de cada $V_i \in V$.

2. **Matching de reglas:** se seleccionan las K reglas donde todos los conjuntos difusos $A_j^{V_i}$ en LHS, para cada $V_i \in V$ tenga $\mu_{ji} > \alpha_i$ El valor de la membresía se obtiene utilizando la función mínima como norma T mediante la siguiente expresión:

$$\mu_q = \min_{j \in \tilde{V_i}; i \in V} \mu_{ji} \tag{5.16}$$

3. **Puntos medios de regla:** para cada regla q seleccionada, se obtiene el punto medio mp_q de la variable objetivo $*V$ siguiendo la siguiente expresión.

$$mp_q = \sum_{j \in \tilde{V}_i} c_j \tag{5.17}$$

Donde c_j es el parámetro c de la función μ del conjunto difuso A_j^{*V}.

4. **Defuzzyficación:** se calcula la predicción como una suma ponderada de los puntos medios de regla mp_q con los valores de membresía μ_q para cada regla j.

$$\hat{y}(t+1) = \frac{\sum_{q \in K} \mu_q \cdot mp_q}{\sum_{q \in K} \mu_q} \tag{5.18}$$

5.2.2. Extensión Ponderada de Método Multivariable

De igual manera que en los modelos de una variable, al modelo tambien se le puede aplicar una extensión para que incorpore la importancia de los conjuntos y reglas difusas generadas. Para ello, se realizan los siguientes cambios en los pasos mencionados.

5.2.2.1. Proceso de entrenamiento

- **Paso 3.b: Generación de reglas base:** se seleccionan todos los patrones temporales que tengan el mismo precedente y se agrupan con sus conjuntos de consecuentes para crear la regla con el formato $V \longrightarrow w_k \cdot A_k^{*V}, w_j \cdot A_j^{*V}, ...$, donde LHS es $f(t-1) = A_j^{V_i}, \forall V_i \in V$ y RHS es $f(t+1) \in \{A_k^{*V}, A_j^{*V}, ...\}$ y las ponderaciones $w_k, w_j, ...$ con frecuencias normalizadas de cada patrón temporal tal que:

$$w_i = \frac{\#A_j^{*V}}{\#RHS} \quad \forall A_j^{*V} \in RHS \tag{5.19}$$

Donde $\#A_i$ es el número de ocurrencias de A_i en patrones temporales con el mismo precedente LHS y $\#RHS$ es la cantidad total de patrones temporales con el mismo precedente LHS.

5.2.2.2. Proceso de pronóstico

Debido al cambio realizado, entonces el método de predicción tambien debe ser modificado.

- **Paso 3: Puntos medios de reglas:** para cada regla q seleccionada, se calculan el punto medio mp_q de la variable objetivo $*V$ siguiendo la siguiente expresión:

$$mp_q = \sum_{j \in \tilde{V}_i} w_j \cdot c_j \tag{5.20}$$

Donde c_j es el parámetro c de la función μ del conjunto difuso A_j^{*V}.

Cabe destacar que los métodos MVFTS y WMVFTS toman **particiones separadas** para cada variable procesada y sus reglas contienen referencias de las diferentes variables.

5.2.3. Series Temporales Difusas con Información Difusa Granular (FIG-FTS)

FIG-FTS[18] es un modelo envolvente que permite que un modelo monovariable (PWFTS) aborde series temporales multivariables. Además de ampliar las funciones de PWFTS para datos multivariables, FIG-FTS también agrega la capacidad de pronóstico multivariado, actuando con métodos de multiples entradas y salidas (MIMO), donde todas las variables son objetivos y explicatorias.

Alias	Parámetro	Dominio	Descripción
k_i	Número de particiones	$\mathbb{N}^+$	Cantidad de conjuntos difusos que se crean en la variable lingüística $\tilde{V}_i$
μ	Función de membresía	$\mu : U \longrightarrow [0,1]$	Función que mide el valor de membresía $y \in U$ a un conjunto difuso
α	α cut	[0,1]	El valor mínimo de membresía para que pueda ser parte de la fuzzyficación
Ω	Orden	$\mathbb{N}^+$	Número de retrasos pasado en el precedente para cada regla difusa
L	Retrasos		Vector con índices de retrasos, de largo Ω y $1 \leq L[i] < L[i+1]$ para $t = 0, ..., \Omega$
κ	k- vecinos cercanos (knn)	$\mathbb{N}^+$	Número de vecinos cercanos que el índice espacial busca en FIG durante el proceso de fuzzyficación.

Tabla 5.3: Hiperparámetros del modelo FIG-FTS

La finalidad de este modelo es reemplazar las etapas de partición y fuzzyficación del proceso de entrenamiento del modelo PWFTS, el paso de fuzzyficación en el proceso de pronóstico y añadir las predicciones multivariables. Dada una serie temporal con n variables $Y = (y_1(t), ..., y_n(t)), t = 0, ..., T$ corresponden a las variables V_i definidas para cada $y_i(t)$. La serie temporal difusa F resultante esta compuesta por datos $f(t) \in F$ que representa una secuencia de información difusa granulada G_i. Cada gránulo contiene un conjunto de variables lingüísticas $\tilde{V}_i$ relacionadas a cada variable V_i.

La variable lingüística FIG corresponde a la unión de toda la información granulada G_i, que, a su vez, es la combinación de un conjunto difuso con cada variable, tal que $G_i = \{A_j^{V_i}\}, \forall V_i \in V$ y la función de membresía esta dada por $\mu_{G_i} = \bigcap \mu_{A_j^{V_i}}$ donde $\bigcap$ es la norma-T mínima. El conjunto FIG esta indexado por los puntos medios de sus conjuntos difusos internos permitiendo la búsqueda espacial optimizada usando árboles KD[19] . Con la variable lingüística FIG el proceso de fuzzyficación convierte cada datos multivariado $y(t) \in Y$ en información granular $G_i \in FIG$ tal que $f(t) = G_i$. Los hiperparámetros de este modelo se aprecian en la tabla 5.3

[18] Siglas provenientes del inglés Fuzzy Information Granule Fuzzy Time Series
[19] Estructura de datos de espacio particionado que organiza los puntos en un espacio euclídeo de k dimensiones.

5.2.3.1. Proceso de entrenamiento.

En este método cada variable es independiente de las otras. Cada una de estas tiene su propio término lingüístico $\tilde{V}_i$. Para esto es necesario informar, para cada variable seleccionada los hiperparámetros k_i, μ y α. El orden de este modelo es controlado por Ω y los índices de retrasos son controlados por L.

1. **Partición:**

   a) **Definición de U_{V_i}:** el Universo de Discurso U_{V_i} define el espacio muestral, i.e los límites conocidos de de la variable V_i , tal que $U_{V_i} = [\min(Y^{V_i}) - D_1, \max(Y^{V_i}) + D_2]$, donde $D_1 = \min(Y^{V_i}) \times 0.2$ son usados para extrapolar los límites conocidos como un margen de seguridad $_i \in V$.

   b) **Partición de U_{V_i}:** se separa U_{V_i} enk_i intervalos U_j con puntos medios c_j para$j = 0, ..., k_i$, donde se encuentran todos los intervalos con el mismo largo.

   c) **Definición de la variable lingüística $\tilde{V}_i$:** para cada intervalo $U_j \in U_{V_i}$ se crea un conjunto difuso superpuesto $A_j^{V_i}$ con función de membresía $\mu_{A_j^{V_i}}(y_{V_i}(t))$, donde $y_{V_i}(t)$es el valor de la variable V_i en la instancia $y \in Y$. El punto medio del conjunto difuso $A_j^{V_i}$ sera c_j, el límite inferior $l_j = c_{j-1}$ y el límite superior $u_j = c_{j+1} \quad \forall j > 0$ y $j < k_i$, y $l_0 = \min(U_V), l_k = \max(U_{V_i})$. Cada conjunto difuso $A_j^{V_i}$ es un término lingüístico de la variable lingüística $\tilde{V}_i$.

2. **Fuzzyficación:** se convierten los valores numéricos de las series temporales Y en series temporales difusas F, donde cada muestra $f(t) \in F$ es una $G_i \in FIG$. Para cada $y(t) \in Y$ se ejecutan los siguientes pasos.

   a) **Fuzzyficación individual de la variable:** para cada variable $V_i \in V$, se encuentran los términos lingüísticos $A_j^{V_i} \in \tilde{V}_i$ donde cada valor de membresía debe ser mayor a $\alpha-$cut , es decir:

   $$f_{V_i}(t) = \{A_j^{V_i} | \mu_{A_j^{V_i}}(y_{V_i}(t)) \geq \alpha_i \quad \forall A_j^{V_i} \in \tilde{V}_i\} \tag{5.21}$$

   b) **Búsqueda en FIG:** por cada combinación de conjuntos difusos $A_j^{V_i}$ en $f_{V_i}(t)$ se verifica si hay un $G_i \in G$ donde $G_i \supset \{A_j^{V_i}, \forall A_j^{V_i} \in f_{V_i}(t)\}$. Si existe entonces el valor valor fuzzyficado de $y(t)$ es G_i esta búsqueda se realiza con árboles KD comparando los puntos medios de los datos fuzzyficados con los puntos medios de los conjuntos difusos encontrados en FIG.

   c) **Creación de un nuevo G_i en FIG:** si no se encuentra un G_i en el paso anterior, entonces se añaden a FIG. Para esto cada combinación de conjuntos difusos $A_j^{V_i} \in f_{V}i(t)$ crea una información difusa granulada G_i tal que $G_i = \{A_j^{V_i}, \forall A_j^{V_i} \in f_{V_i}(t)\}$ y $\mu_{G_i} = \mu_{A_j^{V_i}}$, siendo este G_i el valor fuzzyficado de $y(t)$.

3. **Inducción de reglas:** la data fuzzyficada F donde $f(t) = [(G_0, \mu_{G_0}, ..., G_i, \mu_{G_i})]$ es pasado por la **Inducción de reglas del método PWFTS**, lo cual genera un modelo PWFTPG. Cada regla de este último modelo tiene el formato $\pi_j G_{i0}, ..., G_{i\Omega} \longrightarrow w_{j0} \cdot G_{i0}, ..., w_{ji} \cdot G_{ji}$, donde LHS es $f(t - L(\Omega)) = G_{i0}, f(t - L(\Omega - 1)) = G_{i1}, ..., f(t - L(0)) = G_{i\Omega}$ y RHS es $f(t + 1) \in \{G_k, G_j, ...\}$ donde las ponderaciones π_j, w_{jk} son las probabilidades empíricas difusas.

5.2.3.2. Proceso de pronóstico

Este tiene como finalidad obtener una estimación $\hat{y}(t+1)$ para cada variable V dado una muestra Y como entrada usando las variables lingüísticas FIG y las reglas inducidas encontradas por el modelo.

1. **Fuzzyficación:** se encuentran los κ conjuntos $G_{i\Omega}, ..., G_{i0} \in FIG$ más cercanos a la muestra de entrada $y(t-\Omega), ..., y(t)$.

2. **Matching de reglas:** la muestra de entrada fuzzyficada es trasladada a un modelo PWFTS que busca las reglas aplicables. Para cada regla PWFTPG j encontrada su valor de membresía es dado por:

$$\mu_j = \min_{t \in L, i \in FIG} \mu_{G_{it}} \tag{5.22}$$

3. **Defuzzyficación:**

   a) **Selección de variable objetivo:** para predicciones no multivariables, una variable objetivo $*V$ debe ser seleccionada entre las variables V.

   b) **Adaptación PWFTPG:** tras la selección de la variable objetivo, el RHS de todas las reglas PWFTPG son modificadas, reemplazándolos por los conjuntos difusos $A_j^{*V} \in *\tilde{V}$, pero manteniendo las ponderaciones.

   c) **Defuzzyficación:** este equivale a la defuzzyficación del modelo PWFTS (sección 5.1.3.3, paso 4), la cual puede ser aplicada sin cambios.

4. **Pronósticos Multivariables:**

   a) Si la variable objetivo no es especificada, se calcula una predicción $\hat{y}^{V_i}(t+1)$ utilizando el método de pronóstico PWFTS tomando cada variable $V_i \in V$ como una variable objetivo.

   b) Se unen las predicciones de las variables individuales para crear la predicción multivariada:

$$\hat{y}(t+1) = \bigcup_{V_i \in V} \hat{y}^{V_i}(t+1) \tag{5.23}$$

5. **Pronósticos para ventanas extensas:** dada una ventana de pronósticos de tamaño H (horizonte de predicciones), se repiten los pasos 1 al 4 H veces, añadiendo la salida $\hat{y}(t+1)$ del paso 4 al final de la entrada $y(t)$ para el siguiente paso iterativo.

5.3. Metodología e Implementación Computacional.

5.3.1. pyFTS

Esta librería está especializada en procesar series temporales difusas (*Fuzzy Time Series*). Esta fue desarrollada en 2018 por en el laboratorio de Ciencia de Datos e Inteligencia Computacional de la Universidad Minas Gerais en Belo Horizonte.

Dentro de esta librería se encuentran implementados todos los modelos propuestos en la tesis doctoral **Scalable Models for Probabilistic Forecasting with Fuzzy Time Series** (Sección

2.3 de este informe), por lo que ofrece una gran versatilidad al utilizar series temporales difusas de varios tipos por lo que todas las implementaciones de modelos difusos se hicieron mediante esta librería.

5.3.2. Proceso computacional de un modelo difuso.

El procedimiento de uso de los sistemas difusos para realizar predicciones de series temporales sigue los siguientes puntos[44]:

1. **Preprocesamiento de la data:** es un paso genérico al utilizar modelos de predicción. Aquí se realiza toda la limpieza y filtro de la información para llevarla a los modelos.

2. **Configuración de la partición:** este es el **paso más importante** al utilizar los modelos difusos. Se debe realizar un análisis preciso sobre la variable objetivo para saber cual sería el tipo de partición y cantidad de conjuntos difusos más apropiados. También, se configura el tipo de modelo a utilizar y el orden de este.

3. **Fuzzyficación de la data:** en este paso, los datos son traslados al dominio difuso, por lo que en este punto se generan las series temporales difusas.

4. **Generación de las reglas difusas:** en este punto se obtienen reglas de transición temporal, donde estas dependen de los parámetros con las que fue configurada la partición. Dependiendo del modelo, se obtiene la importancia de los conjuntos generados y las relaciones entre las variables procesadas

5. **Forecast:** se realizan las predicciones tomando una serie de valores cuyo largo debe ser de al menos el orden configurado en la partición, para poder realizar la predicción para los siguientes periodos.

6. **Defuzzyficación de la data:** proceso contrario a la fuzzyficación; en este punto los datos vuelven a su dominio original.

Cabe destacar que, dependiendo de los modelos utilizados para la generación de reglas difusas, el punto 4 puede tener más de un proceso (especialmente si se utiliza el sistema de generación de reglas de forma probabilística **PWFTS**)

5.3.3. Partición.

Se utilizó una partición general: **Grid Partition**, la cual distribuye equitativamente el grado de pertenencia según la cantidad de conjuntos difusos creados (figura 2.7). Para definir la cantidad de conjuntos difusos se utilizó lo siguiente:

- Al estar trabajando con valores de demanda de energía, entonces se crearon **5 niveles:** Muy Bajo, Bajo, Medio, Alto y Muy Alto, o por sus siglas en inglés **VL, L, M, H, VH**

- Cada nivel se dividió a su vez en 7 subniveles, por ende, los conjuntos difusos generados tuvieron nombres [VL0, VL1, ... VH5, VH6], lo que dió un total de 35 conjuntos.

Para las funciones de membresía, se utilizaron 3 tipos: **triangular, trapecio y gaussiana**, siendo la función triangular la más utilizada.

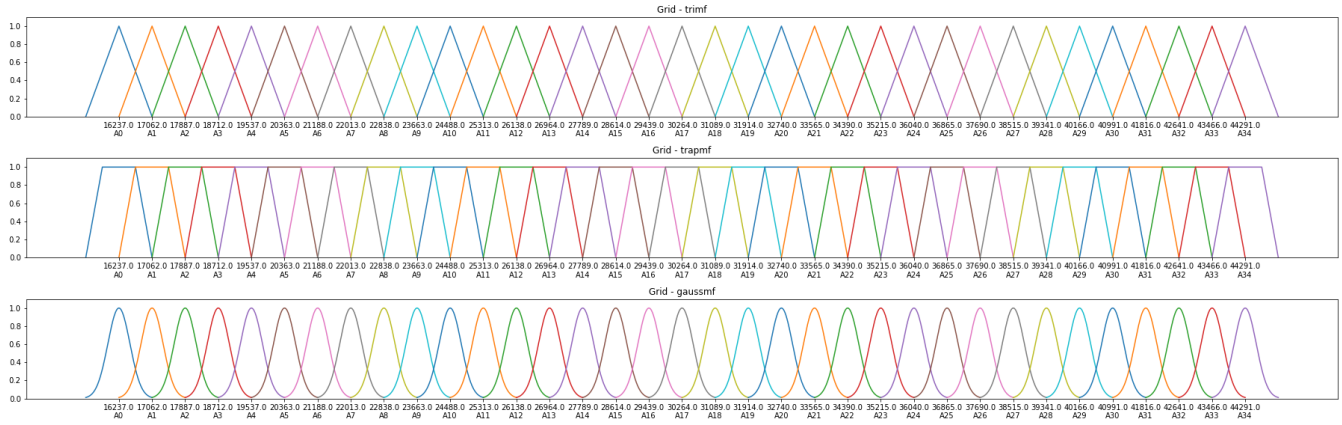


Figura 5.1: Visualización de la partición Grid configurada con 35 niveles y 3 tipos de funciones de membresía. El eje y representa el grado de pertenencia (entre cero y uno) de algún valor de la demanda (eje x). Cada uno de los valores del eje x corresponde al valor medio de los conjuntos difusos asociados cuyo nombre se encuentra escrito debajo.

Partición temporal

Hay que destacar que las particiones mostradas en la figura 5.1 se aplicaron a los datos de la demanda en los modelos monovariables. Al utilizar los modelos multivariables se tuvo que aplicar una partición diferente. Los datos relacionados fueron las fechas, por lo que se aplicó una partición temporal a estos datos cuyas características se pueden apreciar en la figura 5.2.

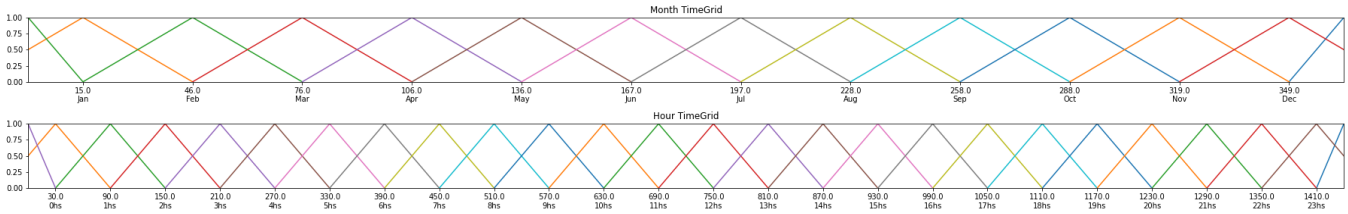


Figura 5.2: Partición temporal aplicado a las fechas registradas. Arriba se aprecia la partición mensual mientras que la segunda es la partición diaria.

Se aplicaron 2 tipos de particiones: una mensual con 12 conjuntos difusos y una horaria con 24 conjuntos generados. Ambas particiones utilizaron una función de membresía triangular.

5.3.4. Orden.

Se probaron los modelos hasta el 3er orden, donde la extracción de las variables previas se realizó de manera automática debido a los objetos configurados en la librería `pyFTS`.

5.3.5. Tipos de modelos.

Como se ha mencionado, se programaron modelos monovariables y multivariables, donde cada uno de estos tuvo una partición con una función de membresía específica y un orden con presencia de métodos de ponderación de los conjuntos para algunos de estos modelos. Las configuraciones usadas para los modelos difusos monovariables se aprecia en la tabla 5.4.

Modelo	Partición	Función de membresía	Orden
HOFTS	Grid Partition, sin ponderación	triangular, trapecio, gaussiana	1, 2 y 3
WHOFTS	Grid Partition, con ponderación	triangular, trapecio, gaussiana	1, 2 y 3
PWFTS	Grid Partition, con ponderación probabilístca	triangular, trapecio, gaussiana	1, 2 y 3

Tabla 5.4: Características generales de los modelos monovariables usados, las ponderaciones mencionadas hacen referencia al cálculo de importancia de los conjuntos difusos generados.

Los modelos multivariables siguen una estructura similar a los modelos monovariables, sin embargo, estos incorporan cálculos de las relaciones entre las variables a procesar, donde se debe destacar la variable objetivo en donde, para todos los modelos, fue la demanda. Las características principales se aprecian en la tabla 5.5.

Modelo	Características generales
MVFTS	Grid Partition triangular de orden 3 sin ponderación de conjuntos difusos, con particiones mensuales y horarias a las fechas registradas.
Weighted MVFTS	Grid Partition triangular de orden 3 con ponderación de conjuntos difusos, con particiones mensuales y horarias a las fechas registradas.
FIG-FTS	Grid Partition triangular de orden 2, con knn como algoritmo de clusterización para búsqueda de las relaciones, con k = [1,2,3]

Tabla 5.5: Configuraciones generales de los modelos multivariables, la partición inicial fue la aplicada a la variable objetivo.

Capítulo 6

Resultados.

La principal métrica utilizada para evaluar el rendimiento de los modelos, fue el **MAPE**, en donde se aplicó tras obtener las predicciones de cada modelo entrenado con la información de los conjuntos de prueba.

6.1. Modelos de series difusas.

Los modelos difusos definidos en el capitulo anterior fueron aplicados al conjunto de prueba, donde las visualizaciones de la demanda fue entre el 22 y 23 de Diciembre del 2019. Cabe destacar que este fueron los resultados de la aplicación del Grid Partition triangular (figura 5.1).

6.1.1. Modelos monovariables.

6.1.1.1. HOFTS

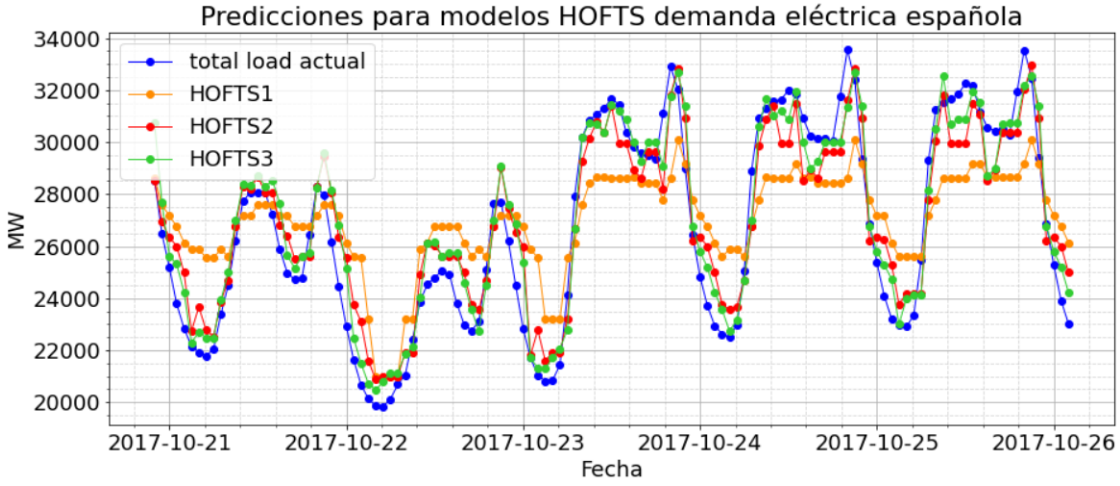


Figura 6.1: Predicciones de la demanda realizado por el modelo monovariable HOFTS para 3 ordenes

Este modelo obtuvo un MAPE igual a **6.90 %, 3.61 % y 2.94 %** para los ordenes 1, 2 y 3, respectivamente. Una parte de sus predicciones se puede apreciar en la figura 6.1

6.1.1.2. WHOFTS

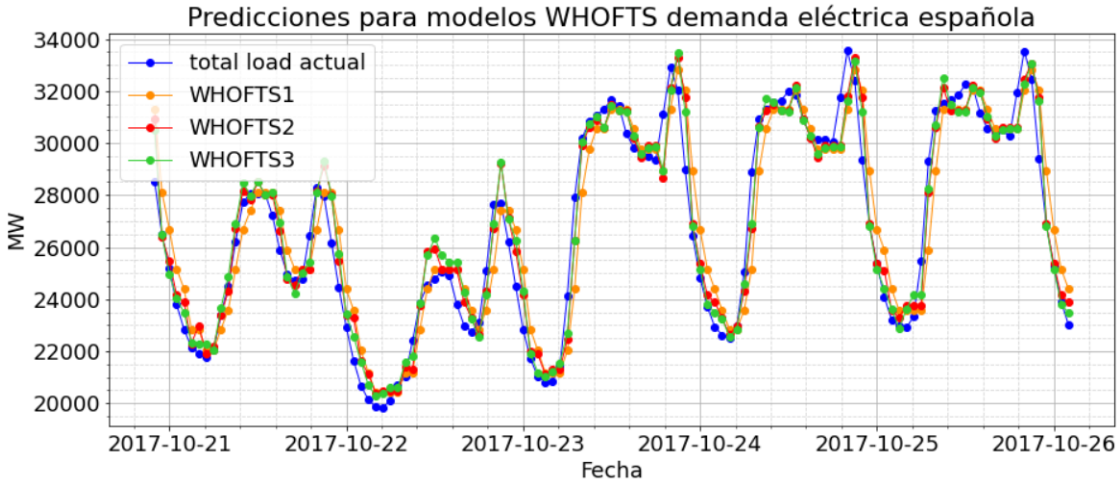


Figura 6.2: Predicciones de la demanda realizado por el modelo monovariable WHOFTS para 3 ordenes

Este modelo tuvo más aciertos que el HOFTS, los MAPE obtenidos fueron de **3.89 %, 2.54 % y 2.33 %** para los ordenes 1, 2 y 3, respectivamente. En la figura 6.2 se aprecia una parte del comportamiento de sus predicciones.

6.1.1.3. PWFTS

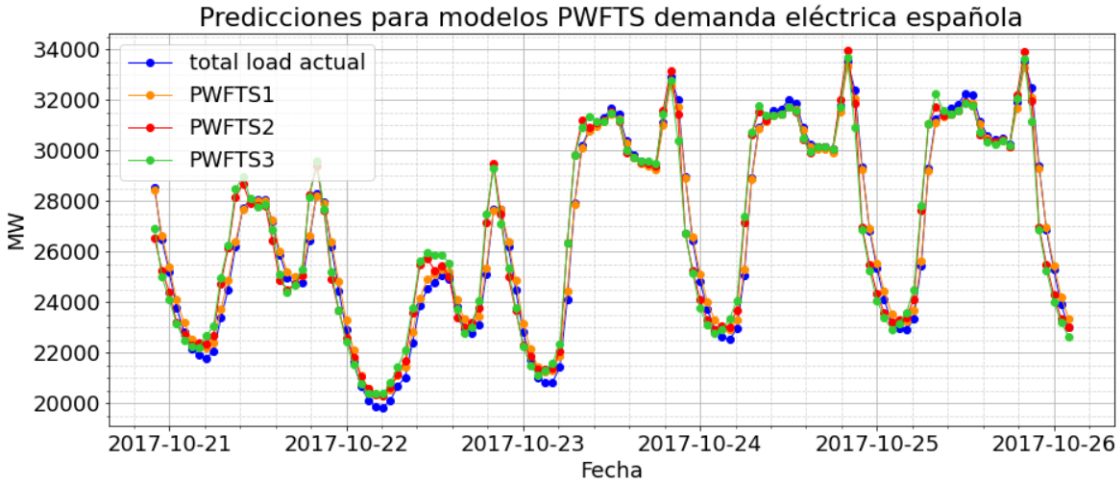


Figura 6.3: Predicciones de la demanda realizado por el modelo monovariable PWFTS para 3 ordenes

Los aciertos aumentaron notoriamente con el uso de este sistema, a tal punto, que el modelo de orden 1 tiene una gran similitud con la demanda real. Los MAPE obtenidos fueron de **0.87 %, 2.46 % y 2.83 %** para los órdenes 1, 2 y 3, respectivamente. Una parte de sus predicciones se aprecian en la figura 6.3.

6.1.1.4. Rendimiento de modelos de una variable.

Ordenando los rendimientos de los modelos según el MAPE obtenido se obtiene la tabla 6.1.

Modelo	Orden	MAPE (%)
PWFTS	1	0.8783
TSO forecast	No aplica	0.9590
WHOFTS	3	2.3359
PWFTS	2	2.4677
WHOFTS	2	2.5408
PWFTS	3	2.8389
HOFTS	3	2.9445
HOFTS	2	3.6125
WHOFTS	1	3.8930
HOFTS	1	6.9030

Tabla 6.1: Error Porcentual Medio Absoluto obtenido por los modelos difusos monovariables.

El modelo PWFTS superó a las predicciones realizadas por el TSO, además se obtuvo un mejor rendimiento general para los modelos difusos de mayor orden (exceptuando el PWFTS).

6.1.2. Modelos multivariables.

Incorporando la partición temporal a las fechas e incluyéndolas en el proceso de predicción se obtuvieron rendimientos diferentes. Las predicciones mostradas se realizaron entre el 21 y 26 de Octubre de 2017.

6.1.2.1. MVFTS

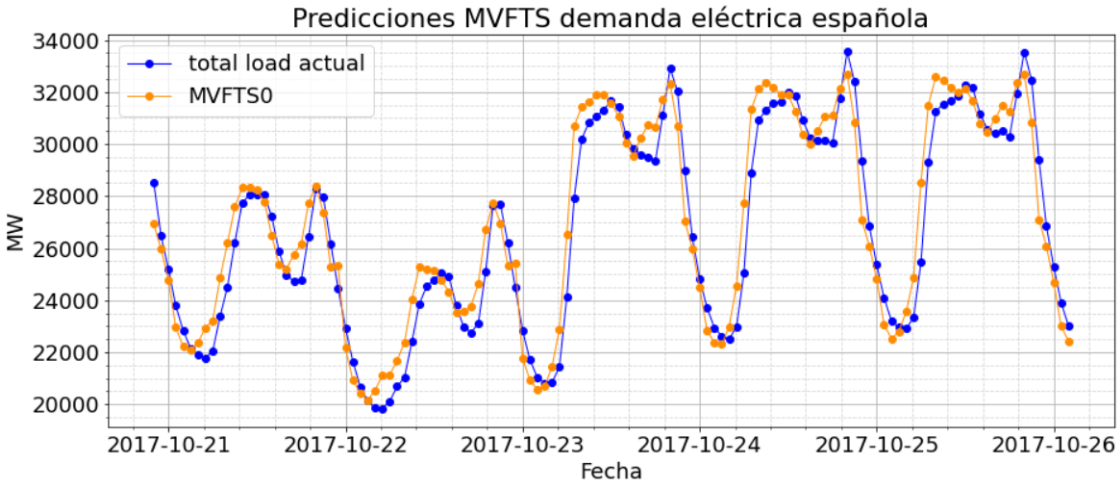


Figura 6.4: Predicciones de la demanda realizado por el modelo multivariable MVFTS

Este sistema tuvo un MAPE de **3.60 %**. Una parte de las predicciones de este modelo se aprecia en la figura 6.4.

6.1.2.2. Weighted MVFTS

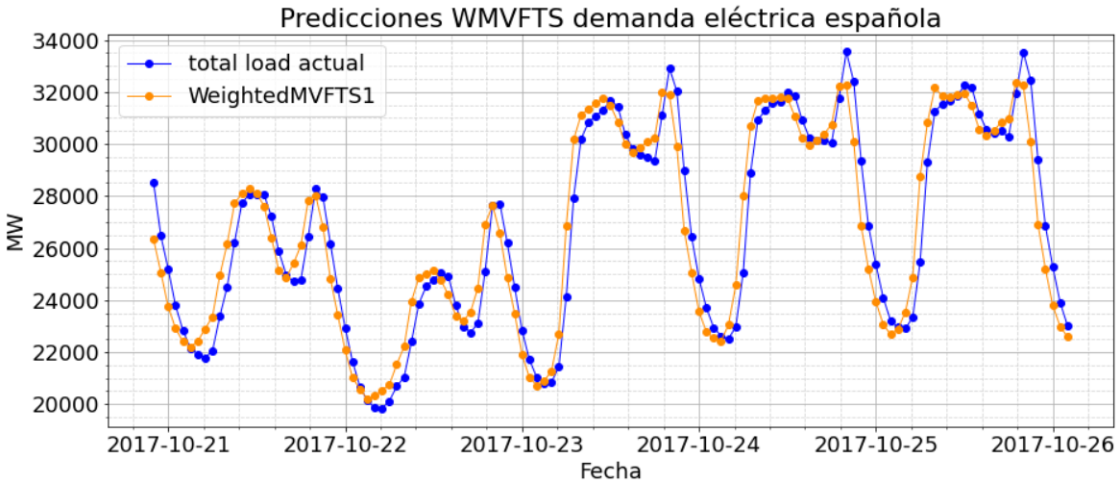


Figura 6.5: Predicciones de la demanda realizado por el modelo multivariable WeightedMVFTS

Incorporando las ponderaciones en los conjuntos difusos, se mejora parcialmente el rendimiento. Este sistema obtuvo un MAPE de **3.48 %**, donde parte de sus predicciones se aprecia en la figura 6.5.

6.1.2.3. FIG - FTS

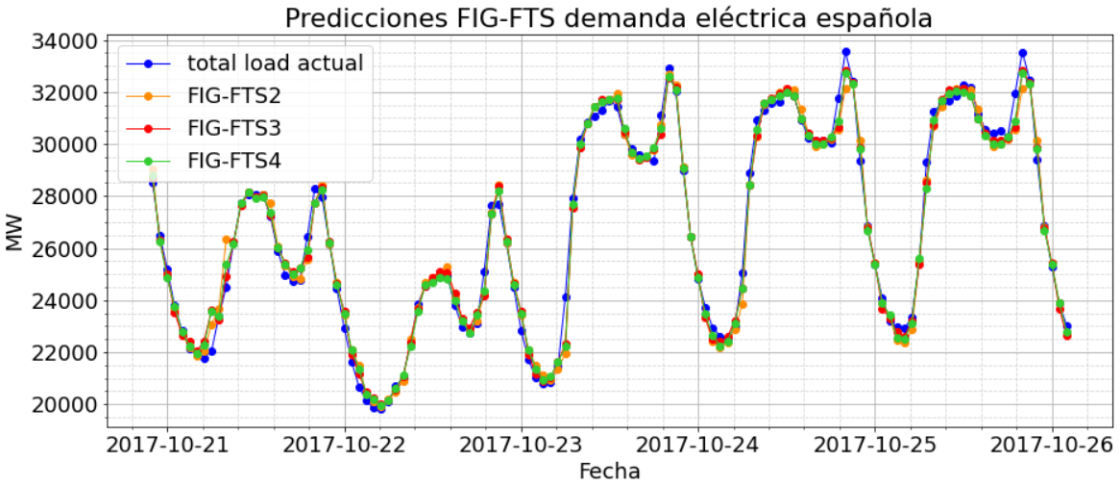


Figura 6.6: Predicciones de la demanda realizado por el modelo multivariable FIG-FTS, hasta 3 vecinos en el uso de KNN.

La búsqueda de agrupaciones mejora el rendimiento. Se obtuvo un MAPE de 1.70 %, 1.54 % y 1.58 % para 1, 2 y 3 vecinos, respectivamente en el uso de knn para el cálculo de relaciones entre las variables. Parte de sus predicciones se aprecia en la figura 6.6

6.1.2.4. Rendimiento de modelos multivariables.

Ordenando los modelos según el MAPE obtenido como en la sección anterior se obtiene la tabla 6.2.

Modelo	MAPE (%)
TSO forecast	0.9590
FIG - FTS (k=2)	1.5421
FIG - FTS (k=3)	1.5866
FIG - FTS (k=1)	1.7079
WeightedMVFTS	3.4820
MVFTS	3.6086

Tabla 6.2: Error Porcentual Medio Absoluto obtenido por los modelos difusos multivariables.

En este caso, no se obtuvo un modelo que tuvo un rendimiento superior al TSO, sin embargo, las predicciones de estos sistemas, en general, tuvieron un error inferior al de los modelos monovariables.

6.1.3. Rendimiento General.

Los resultados mostrados anteriormente, sólo usaron la función de membresía triangular. Se realizaron pruebas también con otro tipo de función de membresía, ordenando el rendimiento obtenido por esos modelos junto con los resultados de las secciones anteriores, se obtiene el ranking de modelos mostrados en la tabla 6.3.

En la gran mayoría de estos sistemas, cuando sólo se cambió la función de membresía el rendimiento no tuvo grandes diferencias, por ende, se espera que el comportamiento gráfico sea similar al ya mostrado para cada modelo.

Modelo	Tipo	Orden	Partición	MAPE (%)
PWFTS	single	1	Grid Partition tri	0.8738
PWFTS	single	1	Grid Partition gauss	0.8779
PWFTS	single	1	Grid Partition trap	0.8851
TSO forecast	No aplica	No aplica	No aplica	0.9590
FIG-FTS (k=2)	multi	2	Grid Partition tri	1.5421
FIG-FTS (k=3)	multi	2	Grid Partition tri	1.5866
FIG-FTS (k=1)	multi	2	Grid Partition tri	1.7079
WHOFTS	single	3	Grid Partition tri	2.3359
WHOFTS	single	3	Grid Partition trap	2.3359
PWFTS	single	2	Grid Partition trap	2.3936
PWFTS	single	2	Grid Partition tri	2.4677
WHOFTS	single	2	Grid Partition tri	2.5408
WHOFTS	single	2	Grid Partition trap	2.5408
WHOFTS	single	3	Grid Partition gauss	2.6005
PWFTS	single	2	Grid Partition gauss	2.6868
PWFTS	single	3	Grid Partition trap	2.7710
PWFTS	single	3	Grid Partition tri	2.8389
WHOFTS	single	2	Grid Partition gauss	2.9332
HOFTS	single	3	Grid Partition trap	2.9445
HOFTS	single	3	Grid Partition tri	2.9445
PWFTS	single	3	Grid Partition gauss	3.0015
WeightedMVFTS	multi	2	Grid Partition tri	3.4820
HOFTS	single	3	Grid Partition gauss	3.6079
MVFTS	multi	1	Grid Partition tri	3.6086
HOFTS	single	2	Grid Partition trap	3.6125
HOFTS	single	2	Grid Partition tri	3.6125
WHOFTS	single	1	Grid Partition tri	3.8930
WHOFTS	single	1	Grid Partition trap	3.8930
WHOFTS	single	1	Grid Partition gauss	3.9295
HOFTS	single	2	Grid Partition gauss	4.7830
HOFTS	single	1	Grid Partition tri	6.9030
HOFTS	single	1	Grid Partition trap	6.9030
HOFTS	single	1	Grid Partition gauss	8.0617

Tabla 6.3: Modelos difusos probados junto con su rendimiento

6.2. Modelos del Estado del Arte.

Aplicando estos modelos para predecir el comportamiento de la demanda entre el 21 y 26 de Octubre del 2017, se obtuvieron los comportamientos mostrados en esta sección.

6.2.1. Red Neuronal

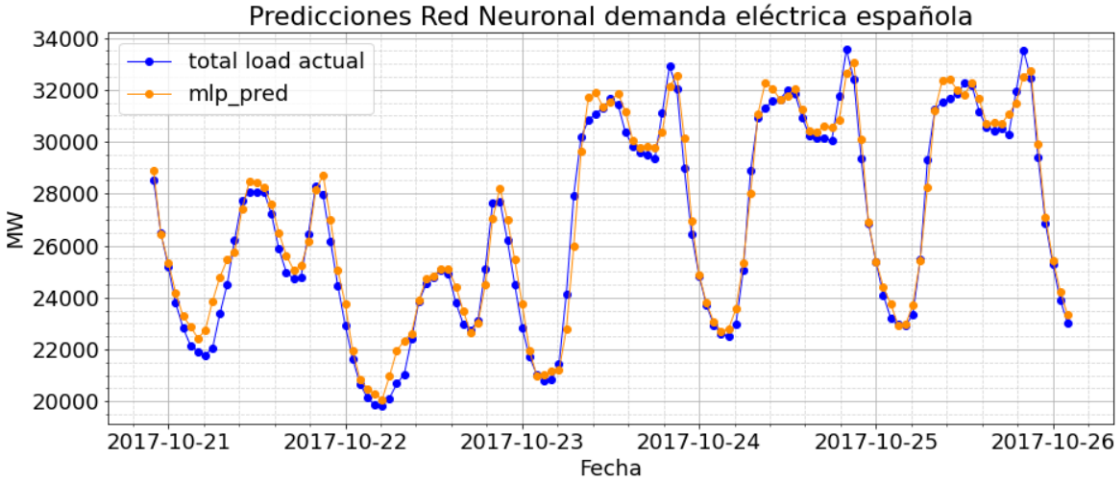


Figura 6.7: Predicciones de la demanda eléctrica de la red neuronal descrita en la tabla 4.3

La figura 6.7 muestra un buen rendimiento de las predicciones. Este comportamiento se mantuvo similar hasta el final de las fechas registradas. El MAPE para este modelo fue de 1.5371 %

6.2.2. Random Forest

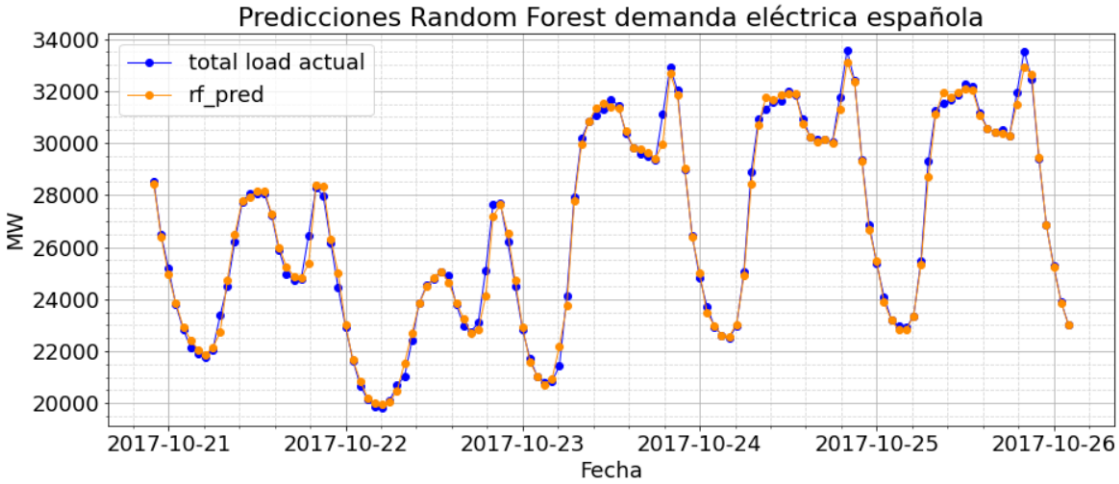


Figura 6.8: Predicciones de la demanda eléctrica del Random Forest descrito en la tabla 4.4

Visualizando las predicciones de este modelo en la figura 6.8, se aprecia que tuvo más precisión que el modelo anterior. Su comportamiento fue parecido hasta el final de la serie. El MAPE

obtenido con este modelo fue de 1.07 %.

6.2.3. Transformer Adaptado

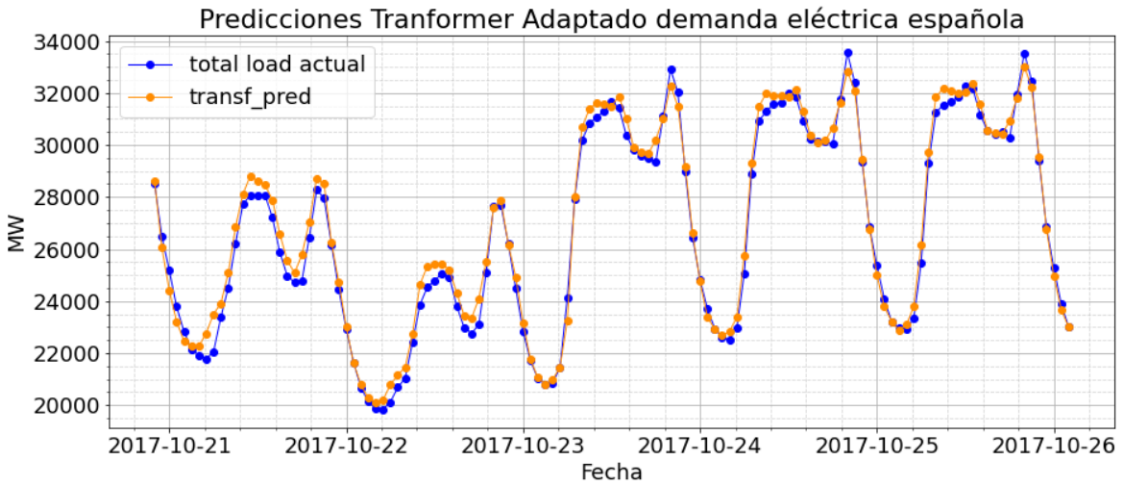


Figura 6.9: Predicciones de la demanda eléctrica mediante el uso del transformer adaptado descrito en la tabla 4.5

Este modelo de atención obtuvo un **1.2478 % MAPE**, lo que lo hace superior a las redes neuronales y a una gran parte de los modelos difusos probados. Parte de sus predicciones se aprecia en la figura 6.9.

6.2.4. Rendimiento de modelos de Estado del Arte

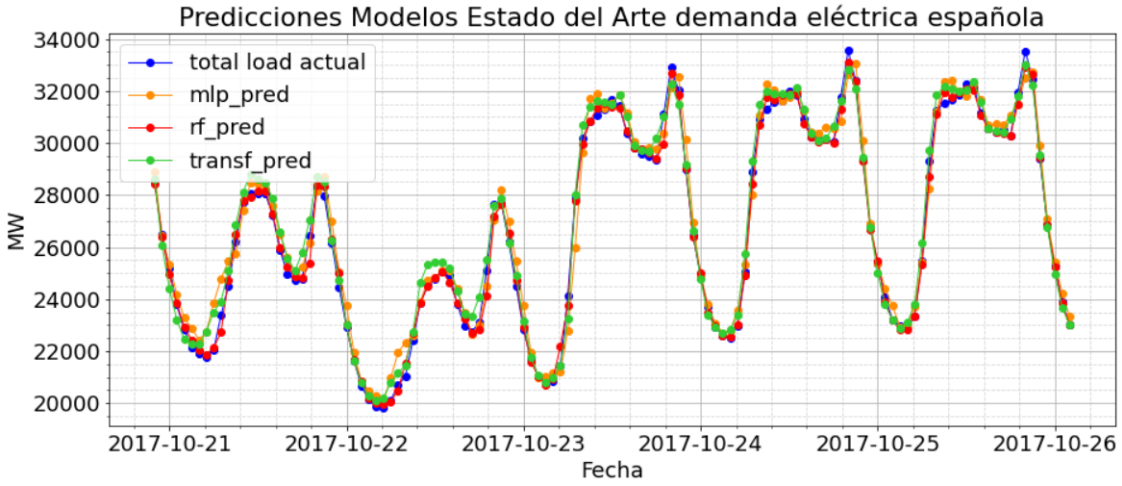


Figura 6.10: Comparación de las predicciones de los modelos junto con la predicción realizada por el TSO español

Sintetizando los rendimientos obtenidos para el mismo rango de fechas, se obtiene, el comportamiento mostrado en la figura 6.10. Juntando los MAPE de cada modelo, se obtienen, los valores mostrados en la tabla 6.4.

Modelo	MAPE
Random Forest	1.066
Transformer Adaptado	1.2478
Red Neuronal	1.9256

Tabla 6.4: MAPE de los modelos de estado del arte para predicciones one step

6.3. Predicciones sobre ventanas extensas

Algunos modelos tambien fueron probados para realizar regresiones sobre grandes pérdidas de datos. Generando una pérdida de datos entren el 23 y 25 de Octubre de 2017 se obtuvo el comportamiento mostrado en la figura 6.11.

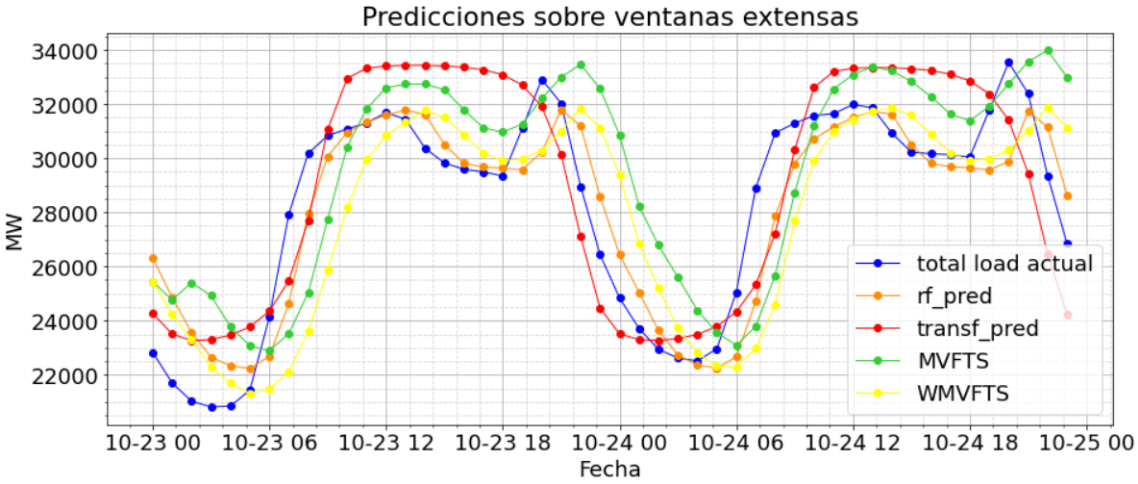


Figura 6.11: Predicciones realizadas durante 2 dias de pérdidas de demanda energética, entre el 23 y 25 de Octubre 2017

Tomando los valores reales de la demanda en este intervalo de tiempo se calcularon los errores de predicción en este intervalo en específico, con esto se obtuvo la tabla 6.5.

Modelo	MAPE
Random Forest	4.86
Transformer Adaptado	7.15
Weighted MVFTS	7.73
MVFTS	9.396

Tabla 6.5: Rendimientos de regresores difusos y del estado del arte sobre un intervalo extenso de pérdida de datos

EL orden de los rendimientos obedece a los rendimientos obtenidos en las predicciones one step, siendo el modelo Random Forest el que obtuvo mejor rendimiento.

6.4. Análisis

De forma general, los modelos pudieron realizar bien las predicciones one step, donde los sistemas difusos fueron los que hicieron el trabajo de mejor y peor manera. Esto es en parte a que si bien todos estos poseen dentro de su implementación computacional, la teoría de la lógica difusa, hay diferencias notorias en la complejidad de la extracción de reglas y el conocimiento. Por lo que los valores finales que posea la variable lingüística pueden diferir bastante, lo que se traduce en rendimientos diferentes.

Modelo	Tipo	Orden	MAPE
PWFTS	Difuso monovariable	1	0.8738
TSO forecast	No aplica	No aplica	0.9590
Random Forest	ML supervisado	No aplica	1.066
Transformer Adaptado	Atención	No aplica	1.2478
FIG-FTS (k=2)	Difuso multivariable	2	1.5421
FIG-FTS (k=3)	Difuso multivariable	2	1.5866
FIG-FTS (k=1)	Difuso Multivariable	2	1.7079
Red Neuronal	ML Supervisado	No aplica	1.9256
WHOFTS	Difuso monovariable	3	2.3359
PWFTS	Difuso monovariable	2	2.4677

Tabla 6.6: Modelos con mejor rendimiento para predicciones one step

Los modelos de machine learning supervisados tuvieron rendimientos elevados al realizar las predicciones, esto fue posible debido a la construcción de las series temporales de la demanda tomando un periodo de 72 muestras, los modelos difusos en general no tuvieron buenos rendimientos al compararlos con los modelos del estado del arte. Sin embargo el modelo que realizó mejor la regresión de las series temporales realizando predicciones one step fue el PWFTS, debido a la forma de extraer las reglas ponderando probabilisticamente los conjuntos difusos.

Analizando los rendimientos obtenidos en la tabla 6.3, se aprecia que las funciones de membresía no afectan mucho al rendimiento de los modelos difusos, en parte, debido a que el poder de predicción se basa en la generación de las reglas de inferencia, por lo que el mapeo de la partición configurada no es un parámetro muy relevante; aunque en algunos casos si se tuvieron cambios al utilizar diferentes funciones. Sin embargo, este impacto pierde fuerza al analizar los modelos con diferente orden, dando a entender el peso de este parámetro al momento de realizar los entrenamientos (todos los modelos difusos al configurarse con mayor orden tuvieron un mejor rendimiento, exceptuando el PWFTS).

Los modelos difusos que no ponderaron los conjuntos fueron los que tuvieron un rendimiento menor. Esto muestra la gran importancia que tiene esta característica al momento de configurar los modelos difusos en general, ya que se tendrán conjuntos difusos más relevantes al momento de generar las reglas de inferencia y, posteriormente realizar las predicciones de la demanda.

Capítulo 7

Conclusiones y Trabajo Futuro

Se logró encontrar un modelo que pudo superar el rendimiento predictivo del TSO, además de que este sistema pudo lograr esto con un menor orden. Si bien este corresponde a un resultado alentador, el sistema PWFTS aún se encuentra en desarrollo y revisión. Una de sus carencias la falta de una implementación computacional más precisa al desarrollar pronósticos extendidos, siendo este el motivo por el cual no se mostraron las predicciones extensas de este modelo en la figura 6.11, por lo que, si bien obtuvo el mejor rendimiento al desarrollar predicciones *one step*, este resultado debe ser analizado minuciosamente si se desea aplicar este modelo a sistemas de predicción de demanda energética similares.

La teoría indicaba la poca relevancia que tenía la función de membresía, y la importancia del orden al momento de configurar un modelo de serie temporal difusa; por lo que los resultados obtenidos apoyan esto. Sin embargo, se obtuvieron casos en donde sí hubo diferencia notable entre mismos modelos, pero con distinta función de membresía, pero este comportamiento al no ser mayoritario, se tiene que esta característica es producto de las peculiaridades del entrenamiento de los modelos y los datos utilizados.

Una de las mayores dificultades fue el poder utilizar modelos que posean un orden elevado. Los resultados logrados indican un mejor rendimiento al subir el orden, sin embargo al aumentar el valor de esta característica tambien se hace mas costoso computacionalmente. Esta misma consecuencia sucedió al incorporar variables relacionadas a la demanda en los modelos multivariables, en donde, al realizar algunos experimentos se apreció un aumento notable del tiempo de ejecución del entrenamiento, además de una reducción del rendimiento obtenido debido al aumento de la complejidad al proceso de la generación de las reglas difusas.

Si bien el uso de esta técnica no es muy conocida, el trabajo desarrollado mostró que sigue siendo útil para desarrollar modelos predictivos, en donde, el modelo PWFTS fue el más destacado. Este modelo fue el sistema más complejo del lado teórico, implementado en este trabajo, a pesar de la simpleza computacional con la que se aplicó a los datos de demanda. Este modelo obtuvo mejores resultados, por ende, este sistema corresponde al modelo difuso más eficiente para realizar predicciones de la demanda energética. Además que sus características permiten que esté presente dentro de otros sistemas más complejos como FIG-FTS. Debido a esto, esta memoria sirvió para validar su eficacia, aunque no es conveniente decir que es uno de los mejores modelos que existen para desarrollar modelos de predicción debido a que, en este campo, se encuentran una gran cantidad de sistemas que realizan este trabajo, en donde, estos se actualizan cada día con las mejoras

en diferentes campos como *deep learning*. Dado esto, como trabajo futuro en este ámbito se espera poder implementar una mayor cantidad de regresores que utilicen este tipo de aprendizaje, aparte del transformer adaptado desarrollado en esta memoria, ya que estos modelos se encuentran entre los mejores que existen en la industria.

Otro punto muy interesante a desarrollar, corresponde a la adaptación de este sistema para el proceso de datos de la demanda local. Una de las dificultades fue conseguir datos para el desarrollo de este modelo, debido a esto, se realizó este trabajo con un enfoque más experimental que práctico. Por lo que un punto a desarrollar, consiste en la adaptación de este sistema para el uso local y práctico, donde el preprocesamiento de la información tenga el papel más relevante en el trabajo para utilizarlo en los modelos presentados en este informe.

Bibliografía

[1] Twenergy, "La demanda eléctrica," Agosto 2019 https://twenergy.com/eficiencia-energetica/como-ahorrar-energia-casa/la-demanda-electrica-953/, https://twenergy.com/eficiencia-energetica/como-ahorrar-energia-casa/la-demanda-electrica-953/.

[2] Arriols, E., "Cuales son las fuentes de energía más utilizadas en el mundo," 2020 https://www.ecologiaverde.com/cuales-son-las-fuentes-de-energia-mas-utilizadas-en-el-mundo-1426.html, https://www.ecologiaverde.com/cuales-son-las-fuentes-de-energia-mas-utilizadas-en-el-mundo-1426.html.

[3] Narvik, "Estudio análisis y determinación de horario de punta en el sistema eléctrico nacional," Diciembre 2018.

[4] Forecasting, A. E., "La eficiencia energética en españa o cómo ha cambiado el uso de la energía desde el récord de demanda de 2007," Enero 2021 https://bit.ly/3qNMkKO, https://bit.ly/3qNMkKO.

[5] integral, P., "El dato del día: evolución del consumo de energía en los últimos cincuenta años," Septiembre 2020 https://bit.ly/3Hyiq45, https://bit.ly/3Hyiq45.

[6] Mauricio, J. A., Introducción al Análisis de Series Temporales. Marzo 2007.

[7] Administración, G. P., "Proyección de demanda eléctrica 2019-2039," Coordinador Eléctrico Nacional, p. 15, Enero 2020.

[8] Caparrini, F. S., Introducción a la lógica difusa. Universidad de Sevilla: Departamento de Ciencias de la Computación, Octubre 2019.

[9] Flores, J. y Lara, C., "Aprendizaje de modelos difusos para predicción de series de tiempo," Octubre 2015.

[10] Song, Q. y Chissom, B., "Fuzzy time series and its models," Fuzzy sets and systems, vol. 54, no. 3, pp. 269–277, Marzo 1996 https://doi.org/10.1016/0165-0114(93)90372-O, doi:https://doi.org/10.1016/0165-0114(93)90372-O.

[11] Chen, S., "Forecasting enrollments based on fuzzy time series," Fuzzy sets and systems, vol. 81, no. 3, pp. 311–319, Agosto 1996 https://doi.org/10.1016/0165-0114(95)00220-0, doi:https://doi.org/10.1016/0165-0114(95)00220-0.

[12] Silva, P., "Scalable models for probabilistic forecasting with fuzzy time series," pp. 38–39, Septiembre 2019 https://bit.ly/35RVY8V, doi:https://doi.org/10.1016/0165-0114(93)90372-O.

[13] Silva, P., "A short tutorial on fuzzy time series," Diciembre 2018 https://bit.ly/3BTGlro, https://bit.ly/3BTGlro.

[14] Kostadinov, S., "Understanding backpropagation algorithm," Agosto 2019 https://bit.ly/3wi

VjWI, https://bit.ly/3wiVjWI.

[15] Perez, G., ¿Por qué es ReLU la función de activación más común utilizada en redes neuronales? Quora: Febrero 2022, https://bit.ly/3K4ijPt.

[16] Calvo, D., "Función de activación – redes neuronales," Diciembre 2018 https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/, https://bit.ly/35xm9S4.

[17] Rodriguez, V., "Conceptos básicos sobre redes neuronales," Octubre 2018 https://vincentblog.xyz/posts/conceptos-basicos-sobre-redes-neuronales, https://bit.ly/35xm9S4.

[18] Bock, S. y Weib, M., A Proof of Local Convergence for the Adam Optimizer. Rengesburgh, Alemania: International Joint Conference, Julio 2019.

[19] Bakshi, C., Random Forest Regression. Level Up Coding: Junio 2022, https://bit.ly/3J5LMac.

[20] Rodrigo, J. A., "Random forest con python," Octubre 2020 https://www.cienciadedatos.net/documentos/py08_random_forest_python.html, https://bit.ly/3Hyiq45.

[21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J. Jones, L., Gomez, A., Kaiser, I., y Polosukhin, I., "Attention is all you need," pp. 2–3, 2017 https://arxiv.org/abs/1706.03762, doi:https://arxiv.org/abs/1706.03762.

[22] Wolf, T., Debut, L., Sanh, V., Chaumand, J., Delange, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., Von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, G., Gugger, S., Drame, M., Lhoest, Q., y Rush, A., "Transformers: State-of-the-art natural language processing," Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45, 2020, https://doi.org/10.18653%2Fv1%2F2020.emnlp-demos.6.

[23] Maxime, "What is a transformer," Enero 2019 https://bit.ly/35xm9S4, https://bit.ly/35xm9S4.

[24] Lei Ba, J., Ryan Kiros, J., y Hinton, G., "Layer normalization," 2016 https://arxiv.org/abs/1607.06450, doi:https://arxiv.org/abs/1607.06450.

[25] Bishop, C. M., "Pattern recognition and machine learning," Springer, pp. 115–117, 2006.

[26] Heras, J. M., Error cuadrático medio para regresión. IArtificial: Octubre 2020, https://www.iartificial.net/error-cuadratico-medio-para-regresion/.

[27] Tutoriales, G., Error Porcentual Absoluto Medio (MAPE) en un Pronóstico de Demanda. Gestión de Operaciones, Enero 2015.

[28] Lei, Q., Xiaoyong, J., Xuan, Y., Zhou, X., Chen, W., Wang, Y., y Yan, X., "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," Advances in Neural Information Processing Systems, pp. 5243–5253, 2019.

[29] Zerveas, G., Jayaraman, S., Patel, D., Bhamidypaty, A., y Eickhoff, C., "A transformer-based framework for multivariate time series representation learning," arXiv, pp. 7–8, 2020, https://arxiv.org/abs/2010.02803.

[30] Candanedo, L., "Appliances energy prediction data set," Febrero 2017 https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction, https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction.

[31] Zerveas, G., Jayaraman, S., Patel, D., Bhamidypaty, A., y Eickhoff, C., "A transformer-based framework for multivariate time series representation learning," arXiv, p. 8, 2020, https:

//arxiv.org/abs/2010.02803.

[32] Klenton, W., "The sp 500 index: Standard poor's 500 index," Diciembre 2021 https://www.investopedia.com/terms/s/sp500.asp, https://www.investopedia.com/terms/s/sp500.asp.

[33] Klingenbrum, N., Transformers for Time-series Forecasting. MLearning.ai: Febrero 2021, https://bit.ly/37a2LLo.

[34] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., y Duchesnay, E., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[35] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., y Chintala, S., "Pytorch: An imperative style, high-performance deep learning library," en Advances in Neural Information Processing Systems 32, pp. 8024–8035, Curran Associates, Inc., 2019, http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[36] Itelligent, ¿Conoces PyTorch? La herramienta de open source con la que puedes crear redes neuronales. Big Data e Inteligencia Artificial: Agosto 2020, https://bit.ly/3LAqjYO.

[37] Joseph, R., Grid Search for model tunning. Towards Data Science: Diciembre 2018, https://bit.ly/3uTbF89.

[38] Silva, P., "Scalable models for probabilistic forecasting with fuzzy time series," pp. 54–57, Septiembre 2019 https://bit.ly/35RVY8V, doi:https://doi.org/10.1016/0165-0114(93)90372-O.

[39] Silva, P., "Scalable models for probabilistic forecasting with fuzzy time series," pp. 101–120, Septiembre 2019 https://bit.ly/35RVY8V, doi:https://doi.org/10.1016/0165-0114(93)90372-O.

[40] Zadeh, L., "Probability measures of fuzzy events," Journal of Mathematical Analysis and Applications, vol. 23, no. 2, pp. 421–427, 1968, doi:https://doi.org/10.1016/0165-0114(93)90372-O.

[41] Luo, J. y Bridges, S., "Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection," International Journal of Intelligent Systems, vol. 15, no. 8, pp. 687–703, 2000 https://doi.org/10.1002/1098-111X(200008)15:8<687::AID-INT1>3.0.CO;2-X, doi:https://doi.org/10.1002/1098-111X(200008)15:8<687::AID-INT1>3.0.CO;2-X.

[42] Chen, S. y Chang, Y., "Multi-variable fuzzy forecasting based on fuzzy clustering and fuzzy rule interpolation techniques," Information Sciences, vol. 180, no. 24, pp. 4772–4783, 2010 http://dx.doi.org/10.1016/j.ins.2010.08.026., doi:https://doi.org/10.1002/1098-111X(200008)15:8<687::AID-INT1>3.0.CO;2-X.

[43] Sun, B., Guo, H., Karimi, H., Ge, Y., y Xiong, S., "Prediction of stock index futures prices based on fuzzy sets and multivariate fuzzy time series," Neurocomputing, vol. 3, no. 151, pp. 1528–1536, 2015 https://doi.org/10.1016/j.neucom.2014.09.018., doi:https://doi.org/10.1002/1098-111X(200008)15:8<687::AID-INT1>3.0.CO;2-X.

[44] Intelligence, M. y Laboratory, D. S., pyFTS Quick Start. 2018.

Anexos

Anexo A. Lógica Difusa.

Algunos conceptos básicos mencionados en el marco teórico tienen las siguientes definiciones.

A.1. Grado de pertenencia o membresía

Corresponde a un valor entre 0 y 1, el cual indica la relevancia que posea algún valor a una característica asociada.

A.2. Conjunto Difuso.

Un conjunto se dice que es difuso si los elementos que conforman parte de este no están presentes solamente en él, sino que tienen presencia y relevancia en otro conjunto, donde los elementos de este ultimo conjunto, también pueden tener relevancia en el conjunto inicial. Cada conjunto difuso obedece a una serie de características y tiene una función de membresía asociada (μ), donde los elementos pertenecientes corresponden a los grados de pertenencia, y estos se obtienen a partir de la aplicación de la función de membresía a cada elemento. Esta característica permite tener una gran cantidad de asociaciones a diferentes tipos de conjuntos difusos de algún fenómeno particular.

A.3. Variable lingüística.

En términos prácticos corresponde corresponden a variables cuyos posibles valores pueden ser palabras. Estas son representadas como conjuntos difusos, donde la agrupación de los valores de pertenencia, junto a las características intrínsecas del conjunto conforman la variable lingüística. Esta permite describir el estado de un fenómeno u objeto, donde los valores descritos se les llama valores lingüísticos.

A.4. Universo de Discurso.

En el ámbito de series temporales difusas, el universo de discurso es el rango de valores que puede tomar algún elemento de una serie temporal, que represente algún fenómeno. Por ejemplo, si se tiene una serie temporal cuyos valores pueden ser $Y \in \mathbb{R}$, entonces el universo de discurso U es $U = [\min(Y) - m, \max(Y) + m]$, siendo m un margen que posibilita la pertenencia de nuevos elementos de la serie temporal, cuyo valor este fuera del rango inicial.