UNIVERSIDAD DE PIURA

**FACULTAD DE INGENIERÍA**

# Predicción de la demanda eléctrica de los edificios de la Facultad de Derecho y Edificio E de la UDEP mediante el uso de redes neuronales LSTM y TCN

Tesis para optar el Título de
Ingeniero Mecánico - Eléctrico

**José Renato Guerrero Meza**
**Bruno Eduardo Renteros Parra**

**Asesor:**
**Mgtr. Ing. José Hugo Fiestas Chevez**

**Piura, setiembre de 2022**

UNIVERSITAS STUDIORUM PIURENSIS

**Dedicatoria**

A Dios, a mis padres, Jorge y Julia, a mis hermanos, Coco y Cinthia. A ellos, a quienes amo y son mi soporte, les dedico todos mis logros.

Bruno Eduardo Renteros Parra

A mis padres Laura y José, y demás familiares por siempre apoyarme y ser mi soporte a pesar de las adversidades. Este y todos mis logros son gracias a ustedes.

José Renato Guerrero Meza

UNIVERSITAS STUDIORUM PIURENSIS

**Agradecimientos**

A Dios y a la Virgen María por ayudarnos a cumplir nuestras metas y permitirnos seguir adelante a pesar de las adversidades

A nuestros amigos y familiares, por darnos soporte y alentarnos en todo momento.

A nuestro asesor, Hugo Fiestas, por atendernos oportunamente y de la mejor manera, y por guiarnos en esa desafiante meta de culminar la tesis.

Los autores

UNIVERSITAS STUDIORUM PIURENSIS

**Resumen**

La inclusión de la inteligencia artificial en la industria trae consigo beneficios como la optimización y reducción de costos, siendo la predicción de series de tiempo una herramienta importante para la identificación de valores anómalos o para la planificación de actividades a ejecutar. El presente trabajo de investigación busca comparar dos modelos de predicción de series temporales y, con el mejor modelo, diseñar una interfaz gráfica para la predicción de la demanda eléctrica de un día entero.

En el primer capítulo se presentan las generalidades, en las que se han incluido los antecedentes, los objetivos generales y los específicos. Ya en el segundo capítulo se desarrolla la parte teórica de esta investigación, brindando una breve introducción sobre la inteligencia artificial y el machine Learning con sus derivados; posteriormente, se realiza una explicación de cada uno de los modelos a utilizar y las partes que los componen.

En el tercer capítulo, se realiza la caracterización de la demanda eléctrica de los edificios seleccionados para el estudio. Este es un capítulo clave, porque gracias a él se tiene un punto de partida para desarrollar el capítulo 4, en el cual se realiza el análisis de la data obtenida basándose en todo aquello que ha sido observado en el capítulo 3.

En el quinto capítulo, se realiza el análisis y discusión de resultados obtenidos. Para el desarrollo de este capítulo se plantearon cuatro casos distintos por cada modelo, se comparó el desempeño entre casos y, posteriormente, se eligió el modelo más preciso para ser implementado en la interfaz gráfica. En este caso, fue el modelo TCN el que tuvo un mejor desempeño, con un MAPE de 8.35%.

En el último capítulo, se explica el desarrollo de una interfaz gráfica que permite al usuario obtener la predicción de la demanda eléctrica de un día entero, en una resolución treintaminutal, a partir de un archivo con datos del día anterior. De esta investigación se evidencia la importancia de la implementación de la inteligencia artificial en la industria eléctrica y sus grandes beneficios.

UNIVERSITAS STUDIORUM PIURENSIS

Tabla de contenido

**Lista de tablas**

UNIVERSITAS STUDIORUM PIURENSIS

**Lista de figuras**

## Introducción

La predicción de la demanda eléctrica es de vital importancia para la planificación de la generación eléctrica y para la toma de decisiones, tanto en la operación de sistemas de potencia como en la negociación de contratos. No solo es muy importante para las empresas del subsector eléctrico, sino que también puede ayudar al cliente de múltiples maneras. Por ejemplo, algunas de las aplicaciones prácticas en las que puede contribuir son: la mejora de la eficiencia energética; la elaboración de estrategias que permitan ahorro en la facturación eléctrica; proporciona una referencia para destacar un consumo de energía anormalmente alto o bajo, para así alertar sobre fallas dentro de la instalación (Soon Chua Chiah et al., 2020); permite a los usuarios que aprovechan energías renovables orientar su producción energética a la demanda, entre otras aplicaciones.

La Universidad de Piura, como usuario libre, puede gozar de los beneficios anteriormente mencionados si pudiera predecir su demanda eléctrica. En este trabajo se proponen modelos capaces de predecir dicha demanda en edificios propios de la universidad. Estos modelos se basan en redes neuronales *Long Short-Term Memory* (LSTM) y *Temporal Convolutional Network* (TCN), mediante el uso de datos históricos climáticos y de consumo eléctrico.

La presente investigación detalla los objetivos generales y específicos del trabajo de tesis, describe el estado de la predicción de la demanda eléctrica en la actualidad y explica los modelos LSTM y TCN. Además, analiza la data obtenida y extrae características para su posterior uso, ejecuta el entrenamiento de los dos modelos para cuatro casos distintos y realiza un análisis de resultados. Finalmente, obtiene una interfaz gráfica para la predicción de la demanda eléctrica de un día, a partir de 48 tomas de un día anterior.

UNIVERSITAS STUDIORUM PIURENSIS

# Capítulo 1

# Generalidades

## 1.1 Antecedentes

A partir del desarrollo de la industria 4.0 y la transformación digital es posible manejar enormes cantidades de datos y extraer información valiosa de ellos. Gracias a esta recolección de datos, se pueden crear sistemas computacionales capaces de realizar tareas que normalmente requerirían de inteligencia humana (Lazzeri, 2020). Dichos sistemas, mediante el uso de técnicas de Machine Learning, son capaces de aprender automáticamente a partir del conjunto de datos, reconociendo los complejos patrones y relaciones que puedan existir entre ellos.

En el subsector eléctrico, con la finalidad de mejorar la eficiencia y la sustentabilidad, se ha empezado a buscar modelos capaces de predecir la demanda eléctrica de una manera confiable y precisa. En la actualidad se está optando por el uso de redes neuronales, las cuales solo necesitan de datos históricos previamente estudiados y analizados. Dichos datos pueden ser de distinta naturaleza, desde datos climáticos hasta datos históricos de consumo eléctrico, es interesante observar que se puede encontrar relaciones entre ellos y, dependiendo de la arquitectura de red neuronal que se emplee, se obtendrá un desempeño característico. A continuación, se mencionarán algunos importantes estudios relacionados:

Electricity Demand Forecasting of a Micro Grid Using ANN (Akarslan & Hocaoglu, 2018) – Este estudio se llevó a cabo en la Universidad Afyon Kocatepe, en Turquía, con el objetivo de predecir la demanda eléctrica en el campus universitario. Desarrollaron una red neuronal sencilla compuesta de tres entradas en la capa inicial, tres capas ocultas y una salida. La estación del año, hora y demanda eléctrica actual fueron las entradas escogidas, teniendo como salida la predicción de la demanda eléctrica en la próxima hora. Como resultado obtuvieron un RMSE (Root mean square error) igual a 64.12 kW, lo que es bueno considerando que las lecturas de demanda eléctrica oscilaban entre 200 – 1200 kW.

Predicting energy consumption in multiple buildings using machine learning for improving energy efficiency and sustainability (Pham et al., 2020) – Este estudio se llevó a

cabo en la Universidad de Danang, en Vietnam, con el objetivo de predecir el consumo energético a corto plazo en una resolución horaria, en múltiples edificios. Se desarrolló un algoritmo *Random Forests* para la predicción, entrenado con datos históricos de cinco edificios diferentes recolectados durante un año. Se evaluó la efectividad del modelo para predecir el consumo energético de 1 hora, 12 horas y 24 horas posteriores, obteniendo mejores resultados en la predicción de las 24 horas posteriores con un MAPE (Mean Absolute Percentage Error) de 21.16%.

Temporal convolutional neural network (TCN) for an effective weather forecasting using time-series data from the local weather station (Hewage et al., 2020) - Esta investigación tiene como objetivo simplificar los complejos sistemas actuales utilizados para pronósticos meteorológicos. Se propone un sistema ligero y novedoso, que consta de una o más estaciones meteorológicas locales y de las más recientes técnicas de Machine Learning para el pronóstico del clima utilizando series de tiempo recolectadas por las estaciones mencionadas. Se exploran las redes TCN y LSTM (Long-short term memory), obteniendo como resultado que el modelo TCN produce un mejor pronóstico comparado con la red LSTM y otros modelos tradicionalmente usados.

Temporal convolutional networks interval prediction model for wind speed forecasting (Gan et al., 2021) – Este estudio se llevó a cabo en la Universidad de Ciencia y Tecnología Huazhong, en China, con el objetivo de predecir la velocidad del viento. Una capa de arquitectura de red convolucional temporal, múltiples capas completamente conectadas que utilizan la función de activación *tanh* y una capa de clasificación sirvieron respectivamente como capas de entrada, ocultas y de salida del modelo de predicción de intervalos basado en redes convolucionales temporales. Se desarrollaron experimentos para comparar la precisión y confiabilidad de la predicción entre el modelo propuesto y otras técnicas convencionales, los resultados mostraron que el modelo propuesto es superior en la probabilidad de cobertura del intervalo de predicción.

Short-Term Residential Load Forecasting based on LSTM Recurrent Neural Network (Kong et al., 2019) – Esta investigación tiene como objetivo la predicción de la demanda eléctrica a corto plazo para clientes eléctricos individuales. Se propone un marco basado en la red neuronal recurrente LSTM, la cual es una de las técnicas más populares de aprendizaje profundo. El marco propuesto se prueba en un conjunto de datos reales de medidores inteligentes residenciales, comparando su rendimiento con otros métodos convencionales como el BPNN (Backpropagation neural network) y KNN (K-nearest neighbor). Como resultado, el enfoque LSTM se mostró superior en la tarea de predicción de la demanda eléctrica a corto plazo para hogares residenciales individuales, obteniendo valores MAPE (Mean Absolute Percentage Error) cercanos al 8%.

Short-Term Load Forecasting based on ResNet and LSTM (Choi et al., 2018) – Este estudio se realizó en la Universidad Sogang, en Corea, con el objetivo de predecir la demanda

eléctrica de clientes industriales. Proponen un marco basado en un modelo ResNet/LSTM combinado, que tiene la ventaja de pronosticar datos de carga que tienen tanto regularidad como inconsistencia. Para demostrar el rendimiento, comparan el modelo propuesto con otros modelos de aprendizaje profundo: perceptrón multicapa (MLP), ResNet, LSTM. Los resultados muestran que el modelo combinado ResNet/LSTM propuesto tiene un 21% de mejora de MAPE en general, comparado con los otros modelos estudiados.

Los estudios mencionados sobre la predicción de series de tiempo a partir de redes neuronales artificiales han obtenido resultados bastante satisfactorios y han permitido demostrar la variedad de modelos que existen, las diferencias entre ellos, y su gran aplicabilidad frente a distintos problemas prácticos, mostrándose como una propuesta relativamente sencilla en comparación a los métodos convencionales utilizados.

## 1.2 Objetivos

### *1.2.1 Objetivos Generales*

Predecir la demanda eléctrica de los edificios de la Facultad de Derecho y Edificio E de la Universidad de Piura, haciendo uso de modelos de redes neuronales LSTM y TCN.

### *1.2.2 Objetivos Específicos*

- Conocer el perfil de consumo de los edificios mencionados para el estudio en la Universidad de Piura.
- Analizar los datos históricos de demanda eléctrica y definir las características que permitirán tener una predicción acertada.
- Entrenar los algoritmos con los datos históricos, modificando los hiperparámetros hasta obtener modelos precisos.
- Analizar los resultados y comparar el desempeño de ambos modelos.
- Elegir el modelo más destacado e implementarlo en una interfaz gráfica amigable con el usuario.

UNIVERSITAS STUDIORUM PIURENSIS

# Capítulo 2

# Marco teórico

## 2.1 Inteligencia artificial

La inteligencia artificial nace en la década de 1950, cuando surge la pregunta por parte de un grupo de pioneros en el campo de la informática, si es que se podía lograr que las computadoras "piensen", pregunta que hoy en día sigue siendo explorada (Ketkar & Moolayil, 2018).

La inteligencia artificial puede ser definida de una manera más concisa como el esfuerzo por automatizar las tareas intelectuales que normalmente realizan los humanos (Ketkar & Moolayil, 2018).

Como tal, la inteligencia artificial es un campo general que abarca tanto el aprendizaje automático y el aprendizaje profundo; asimismo, engloba muchos otros enfoques que no implican el aprendizaje. En los siguientes acápites se desarrollan el Machine Learning, las redes neuronales artificiales y el Deep Learning, con la finalidad de cubrir los conocimientos necesarios para el correcto entendimiento de los modelos a desarrollar en el presente trabajo de investigación.

### *2.1.1 Machine Learning*

Es la ciencia y el arte de programar computadoras para que puedan aprender de los datos (Géron, 2019). Definido por Arthur Samuel (1959), es el campo de estudio que les brinda a las computadoras la habilidad de aprender sin estar programadas explícitamente (Patterson & Gibson, 2017).

El termino *Machine Learning* hace referencia a la detección automática de diferentes y significativos patrones en datos. En las últimas décadas se ha convertido en una herramienta común en cualquier tarea que involucre la extracción de información de grandes conjuntos de datos (Shalev-Shwartz & Ben-David, 2014).

**2.1.1.1 Aprendizaje supervisado.** Este enfoque mantiene cierta relación con el aprendizaje humano bajo la supervisión de un mentor. Entrando en materia, este aprendizaje se da cuando el algoritmo aprende a partir de data que se tiene de ejemplo y de respuestas objetivo-asociadas, que pueden ser valores numéricos o etiquetas de texto, con la finalidad de luego poder predecir nuevos ejemplos (El-amir & Hamdy, n.d.).

El aprendizaje automático supervisado puede ser dividido en dos tipos de problemas:

- Los problemas de clasificación, que usan un algoritmo para asignar con precisión datos de prueba en categorías específicas. En el día a día tiene diferentes tipos de aplicaciones, una de ellas es la identificación de correo "Spam", que consiste en enviar a una bandeja diferente todo aquel correo clasificado con esa etiqueta (Juliana Delua, n.d.).

- Los problemas de regresión son otro tipo que utiliza un algoritmo para entender la relación que existe entre las variables dependiente e independientes. Su aplicación en la vida diaria son las proyecciones que se hacen para predecir valores numéricos basados en diferentes puntos de datos, como las proyecciones de ingresos por ventas para una empresa determinada.(Juliana Delua, n.d.)

**2.1.1.2 Aprendizaje no supervisado.** Utilizan algoritmos con la capacidad de aprender y organizar información sin proporcionar una función de error para evaluar el desempeño. Lo mencionado anteriormente representa una gran ventaja en ciertos casos ya que permite al algoritmo identificar patrones que no han sido considerados anteriormente (Sathya & Abraham, 2013).

Los algoritmos de aprendizaje no supervisado son utilizados en 3 casos principales:

- La agrupación en clústeres, que es una técnica utilizada en la minería de datos, para agrupar datos sin etiquetar en función de sus similitudes o diferencias. Ejemplo de la aplicación de esta técnica es su utilidad para la segmentación del mercado (Juliana Delua, n.d.).

- La asociación, otro tipo de método de aprendizaje no supervisado que utiliza diferentes reglas para determinar relaciones entre variables en un conjunto de datos determinado. Dentro de sus aplicaciones, se tienen los motores de recomendación, los cuales se realizan en base a las recomendaciones de otros clientes que ya han comprado (Juliana Delua, n.d.).

- La reducción de dimensionalidad, técnica de aprendizaje que es utilizada cuando el número de características o dimensiones en un conjunto de datos determinado es demasiado alto. Consiste en la reducción de la cantidad de entradas de datos a un tamaño manejable, preservando siempre la integridad de los datos. Es utilizado mayormente en la etapa de preprocesamiento de datos, para así eliminar el ruido que pueda existir (Juliana Delua, n.d.).

**2.1.1.3 Aprendizaje semi supervisado.** El aprendizaje semi supervisado es un término medio entre supervisado y no supervisado, en el que se usa un conjunto de datos de entrenamiento que tiene un grupo etiquetado y otro sin etiquetar. Esto resulta de gran utilidad cuando es difícil extraer características notables de los datos y cuando se tiene un gran volumen de estos (Juliana Delua, n.d.).

### *2.1.2 Redes neuronales artificiales*

Una red neuronal artificial es un modelo computacional inspirado en la estructura del comportamiento del cerebro (Shalev-Shwartz & Ben-David, 2014). Las redes neuronales artificiales (ANN por sus siglas en inglés) están compuestas de una estructura de capas de nodos, que contienen una capa de entrada, una o más capas ocultas y finalmente una capa de salida. Cuando la salida de cualquier nodo individual tiene un valor por encima del umbral especificado, el nodo se activa y este envía datos a la siguiente capa de la red; de lo contrario, no se realiza la transmisión de datos a la siguiente capa (IBM Cloud Education, 2020).

Las redes neuronales artificiales se basan en conjuntos de datos de entrenamiento para aprender y mejorar su precisión con el tiempo. Cuando estos logran ajustarse con precisión, llegan a ser herramientas poderosas en informática e inteligencia artificial, permitiendo al usuario clasificar y agrupar datos a alta velocidad. Una de las aplicaciones de redes neuronales más conocidas es el algoritmo de búsqueda utilizado por Google (IBM Cloud Education, 2020).

### *2.1.3 Deep Learning*

El aprender es el proceso de establecer una relación entre dos o más variables (W. J. Zhang et al., 2018). El *Deep Learning* no trata solamente de aprender la relación entre dos o más variables, sino también el conocimiento que gobierna la relación, así como el conocimiento que le da sentido a la relación (W. J. Zhang et al., 2018).

Dentro de sus grandes ventajas, frente a otros algoritmos tradicionales de aprendizaje automático, está la extracción automática de características, que es el proceso de determinar las características de un conjunto de datos que pueden usarse como indicadores para etiquetar los datos de manera confiable (Patterson & Gibson, 2017).

**2.1.3.1 Redes neuronales convolucionales.** Las redes neuronales convolucionales, conocidas como CNN, son un tipo específico de redes neuronales que generalmente están compuestas por las siguientes capas:

- Red convolucional, hace uso de filtros que realizan operaciones de convolución mientras que escanea la entrada con respecto a sus dimensiones. Dentro de sus hiperparámetros se incluyen el tamaño del filtro y el desplazamiento. La salida resultante se le denomina mapa de características o mapa de activación (Afshine Amidi, n.d.).

- Agrupación, su principal objetivo es reducir progresivamente el tamaño espacial (ancho y altura) del volumen de entrada. Esto permite reducir la cantidad de parámetros y cálculos en la red; asimismo, ayuda a controlar el sobreajuste (Rosebrock, 2017).
- Capa completamente conectada, las neuronas en estas capas están completamente conectadas a todas las activaciones de la capa anterior. Este tipo de capas siempre se colocan al final de la red (Rosebrock, 2017).

**2.1.3.2 Redes neuronales recurrentes.** Las redes neuronales recurrentes, también conocidas como RNN, son una clase de redes neuronales que permiten que las salidas anteriores se utilicen como entradas mientras mantienen estados ocultos. (Amidi, n.d.)

## 2.2 Arquitecturas usadas en el estudio

### *2.2.1 LSTM*

Las redes neuronales recurrentes LSTM (*Long Short-Term Memory*) han surgido como modelos eficaces y escalables para numerosos problemas de aprendizaje relacionados con datos secuenciales. Los métodos más primitivos que han abordado estos problemas, como la versión simple de RNN que consta de una sola conexión de retroalimentación, han sido adaptados a casos específicos o no han escalado correctamente para largas secuencias temporales. Las redes neuronales LSTM, por otro lado, son tanto generales como efectivas en la captura de dependencias temporales a largo plazo, ya que no sufren de los problemas propios de las redes neuronales recurrentes convencionales en su proceso de optimización de la función de costo (Greff et al., 2015).

Desde su publicación original en 1997 (Hochreiter, 1997), numerosos trabajos teóricos y experimentales sobre la red LSTM han sido publicados, muchos de ellos informan los impresionantes resultados obtenidos a través de una amplia variedad de aplicaciones donde los datos son secuenciales. El modelado del lenguaje, la transcripción de voz a texto y la traducción son algunos ejemplos que demuestran el notable impacto de la red LSTM (Sherstinsky, 2020).

En los últimos años también se ha observado la aplicación de redes LSTM para la predicción de la demanda eléctrica, a gran escala a nivel de países (Dudek et al., 2021) como a pequeña escala a nivel de clientes residenciales (Kong et al., 2019). En este último caso suele ser difícil obtener modelos precisos dado que el consumo eléctrico en un solo hogar tiende a ser volátil; sin embargo, el modelo LSTM se muestra robusto y confiable incluso en este tipo de casos (Wang et al., 2019).

La red neuronal LSTM tiene ventajas y desventajas respecto a un modelo de red neuronal recurrente convencional (Mebsout, 2020), estas se pueden resumir en los siguientes puntos:

- Ventajas
  - ✓ Las redes LSTM han sido especialmente diseñadas para evitar el problema de la dependencia a largo plazo. Recordar información durante largos periodos de tiempo es prácticamente su comportamiento predeterminado, no es algo que les cueste aprender.
  - ✓ Las redes LSTM también tienen una estructura en forma de cadena como las RNN. Sin embargo, en lugar de tener una red neuronal de una sola capa, tiene una de cuatro capas que interactúan entre sí.

- Desventajas
  - ✓ Aumentan la complejidad computacional en comparación con la RNN, ya que introduce más parámetros para el aprendizaje.
  - ✓ La memoria requerida es mayor que en la RNN convencional debido a la presencia de varias celdas de memoria.

**2.2.1.1 Operaciones dentro de una celda LSTM**. Una red LSTM tiene un flujo de control similar al de una red neuronal recurrente, ya que procesa datos que transmiten información a medida que se propagan. Las diferencias son las operaciones dentro de las celdas de LSTM, estas operaciones se utilizan para permitir que la red LSTM mantenga u olvide información (Phi, 2018).

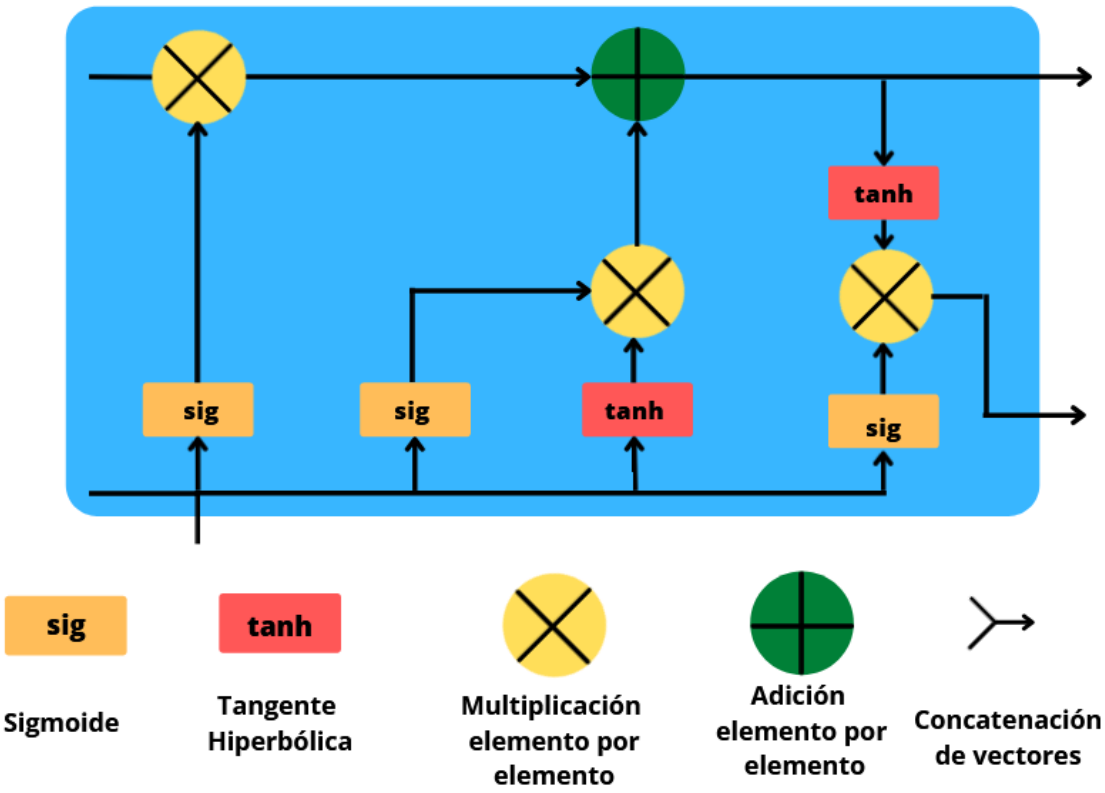


**Figura 1**. Operaciones dentro de una celda LSTM
**Fuente:** Elaboración propia, basado en la página web de Phi, 2018

A continuación, se explicará cada una de estas operaciones:

- Función sigmoide (Mehreen Saeed, 2021)

La función sigmoide es una función matemática cuya gráfica es una curva con forma de "S". Generalmente se denota por $\sigma(x)\ or\ sig(x)$, se define de la siguiente manera:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

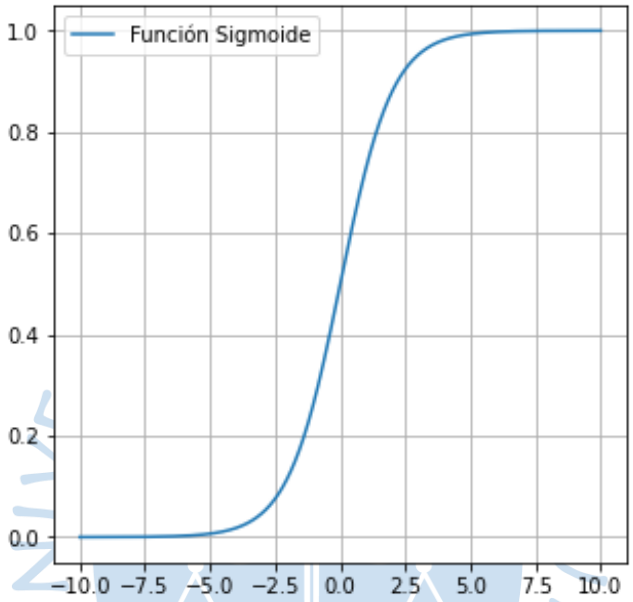


**Figura 2.** Función sigmoide.

Esta función cuenta con las siguientes propiedades:

- ✓ $Dominio < -\infty, +\infty >$
- ✓ $Rango < 0, +1 >$
- ✓ $\sigma(0) = 0.5$
- ✓ Es una función continua
- ✓ La función es creciente en todo momento
- ✓ La función es diferenciable en su dominio

- Función Tangente Hiperbólica (Mathematics, 2011)

La función tangente hiperbólica es una función matemática cuya gráfica es una curva con forma de "S", similar a la función sigmoide. Se denota por $tanh(x)$ y se define como la división entre las funciones seno y coseno hiperbólicos:

$$tanh(x) = \frac{senh(x)}{cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2}$$

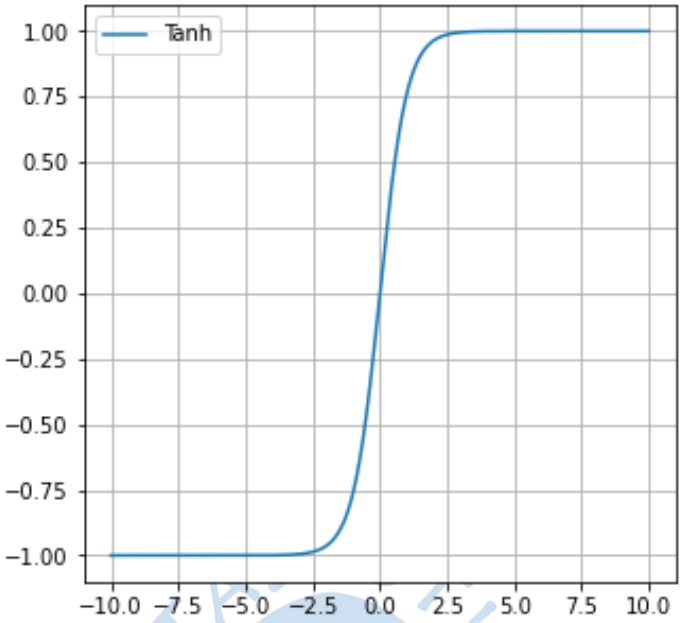


**Figura 3.** Función tangente hiperbólica

Esta función cuenta con las siguientes propiedades:

- ✓ $Dominio < -\infty, +\infty >$
- ✓ $Rango < -1, +1 >$
- ✓ $tanh(0) = 0$
- ✓ Es una función continua
- ✓ La función es creciente en todo momento
- ✓ La función es diferenciable en su dominio

➢ Multiplicación elemento por elemento (RTMath & Deltix Inc, 2020)

Consiste en multiplicar cada elemento de un determinado vector por un escalar o por el elemento correspondiente de otro vector dado.

- ✓ Multiplicación elemento por elemento Vector-Escalar: Multiplica cada elemento del vector por el escalar dado. Si se multiplica un vector **u** por un escalar *a* se tendría:

$$a\boldsymbol{u} = (au_0, au_1, \dots, au_{n-1}) \quad (3)$$

- ✓ Multiplicación elemento por elemento Vector-Vector: Multiplica cada elemento del vector por el elemento correspondiente del otro vector dado. Si se multiplica un vector **u** por otro vector **v** se tendría:

$$\boldsymbol{u} * \boldsymbol{v} = (u_0 * v_0, u_1 * v_1, \dots, u_{n-1} * v_{n-1}) \quad (4)$$

- Adición elemento por elemento (RTMath & Deltix Inc, 2020)

A cada elemento de un determinado vector le suma un escalar o el elemento correspondiente de otro vector dado.

- ✓ Adición elemento por elemento Vector-Escalar: A cada elemento del vector le suma el escalar dado. Si se suman un vector **u** y un escalar *a* se tendría:

$$a + \boldsymbol{u} = (a + u_0, a + u_1, \dots, a + u_{n-1}) \quad (5)$$

- ✓ Adición elemento por elemento Vector-Vector: A cada elemento del vector le suma el elemento correspondiente del otro vector dado. Si se suman un vector **u** y otro vector **v** se tendría:

$$\boldsymbol{u} + \boldsymbol{v} = (u_0 + v_0, u_1 + v_1, \dots, u_{n-1} + v_{n-1}) \quad (6)$$

- Concatenación de vectores (MathWorks, n.d.)

La concatenación de vectores consiste en la unión de dos o más vectores para formar un vector más largo o una matriz. Se formará un vector más largo si se concatenan vectores a lo largo de una misma fila; en cambio, si dichos vectores se concatenan en filas diferentes se formará una matriz.

De la misma manera en que se pueden concatenar vectores, también se puede concatenar matrices. Sin embargo, dichas matrices deben tener tamaños compatibles. Es decir, si se concatenan matrices horizontalmente, deben tener el mismo número de filas; si se concatenan verticalmente, deben tener el mismo número de columnas.

**2.2.1.2 Celda básica LSTM**. La idea central detrás de la arquitectura LSTM es el estado de celda, que puede mantener información a lo largo del tiempo, y múltiples compuertas no lineales, que regulan el flujo de información hacia adentro y hacia afuera de la celda (Greff et al., 2015). Esto forma el módulo o celda básica de la arquitectura LSTM, que toma decisiones acerca de qué, en qué medida, y cuándo almacenar o liberar información. Aprende a través del proceso iterativo de hacer predicciones, retropropagar el error y ajustar los pesos a través del descenso por gradiente (Lazzeri, 2020). La Figura 4 muestra una celda básica LSTM y detalla el flujo de información a través de ella.

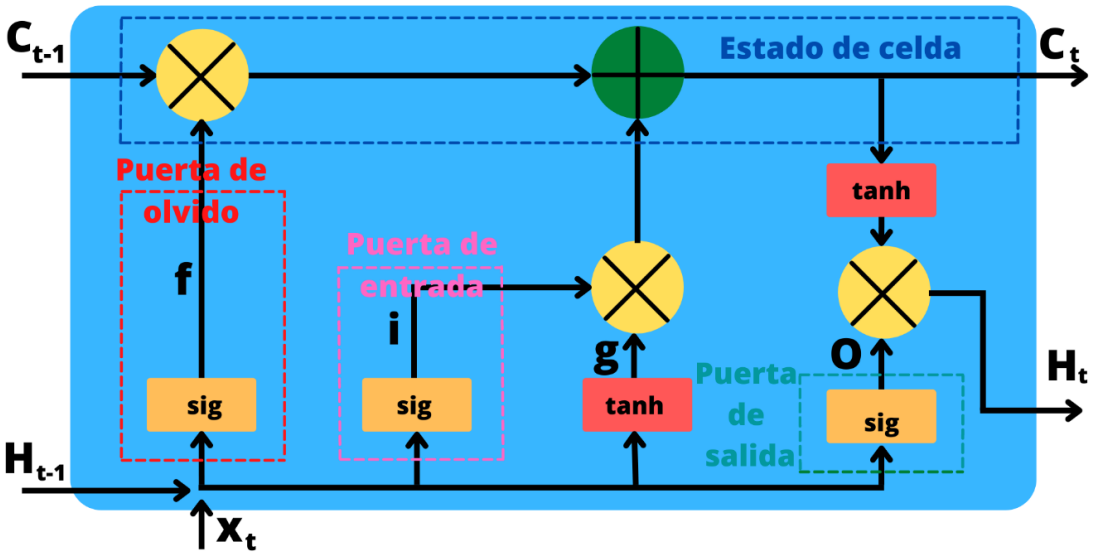


**Figura 4.** Celda básica LSTM
**Fuente:** Elaboración propia, basado en la página web de Phi, 2018

El flujo de información dentro del módulo LSTM, en cada paso de tiempo *t,* se describe mediante las siguientes ecuaciones:

$$
\begin{aligned}
&Puerta\ de\ olvido{:}f{=}\sigma \mathrm{W}f.\mathrm{H}t{-}1,Xt \\
&Puerta\ de\ entrada{:}i{=}\sigma \mathrm{W}i.\mathrm{H}t{-}1,Xt \\
&Nuevo\ estado\ de\ celda{:}g{=}tanh\mathrm{W}g.\mathrm{H}t{-}1,Xt \\
&Puerta\ de\ salida{:}o{=}\sigma \mathrm{W}o.\mathrm{H}t{-}1,Xt \\
&Estado\ de\ celda{:}Ct{=}f{\times}Ct{-}1{+}i{\times}g \\
&Estado\ oculto{:}Ht{=}o{\times}\tanh(Ct)
\end{aligned}
\tag{7}
$$

La primera puerta, la puerta de olvido [ $f$ ], decide qué información se debe desechar del estado de celda [ $C_t$ ]. Esto lo hace gracias a la función sigmoide, que genera valores entre 0 y 1: con el valor de 0 desecha información, mientras que con el valor de 1 la información se mantiene. La función sigmoide en esta compuerta funciona como un filtro.

La segunda puerta, la puerta de entrada [ *i* ], trabaja en conjunto con el nuevo estado de celda [ *g* ] para actualizar el estado de celda [ $C_t$ ]. El nuevo estado de celda genera un vector con valores candidatos que posteriormente serán filtrados por la puerta de entrada, los valores candidatos que sean seleccionados podrán actualizar el estado de celda.

Finalmente, se tiene la puerta de salida [ *o* ], que decide cuál debería ser el próximo estado oculto. Primero, el antiguo estado oculto y la entrada actual ingresan a una función sigmoide. Luego, el estado de celda recién actualizado pasa a través de la función *tanh.* La salida de la función *tanh* se multiplica por la salida de la función sigmoide para decidir qué información debe guardar el estado oculto (R. Zhang et al., 2019).

### *2.2.2 TCN*

La Red Convolucional Temporal (TCN por sus siglas en inglés), hace referencia a la familia de arquitecturas que incorporan capas convolucionales de una sola dimensión. Siendo más específicos, es una arquitectura compuesta por convoluciones causales que se caracterizan por no permitir la filtración de información del futuro al pasado. (Alla & Adari, 2019; Bai et al., 2018)

La TCN tiene ventajas y desventajas respecto a otros modelos de predicción de series temporales, las cuales se pueden resumir en los siguientes puntos:

- Ventajas
  - ✓ Cálculos paralelos: los cálculos matriciales de las redes convolucionales se adaptan bien a la estructura de las GPU, las cuales están configuradas para realizar los cálculos matriciales que forman parte del procesamiento gráfico, es por eso que las TCN pueden entrenarse mucho más rápido en comparación a las RNN.
  - ✓ Flexibilidad: las TCN pueden cambiar la longitud de los datos de entrada, el tamaño del filtro, aumentar los factores de dilatación, apilar más capas, etc., todo esto la convierte en un modelo bastante flexible para varios dominios.
  - ✓ Gradientes consistentes: las TCN están compuestas por capas convolucionales, estas se propagan hacia atrás de forma diferente a las RNN, por lo que los gradientes se guardan. De esta manera se evita el problema existente en las RNN, que es el gradiente explosivo o desvanecido, donde a veces el gradiente calculado es extremadamente pequeño o grande, lo que conlleva a un ajuste de pesos muy extremo o un cambio relativamente inexistente.
  - ✓ Memoria más ligera: las TCN son relativamente más sencillas que otros modelos debido a que están compuestas por varias capas que comparten sus respectivos filtros, haciéndolas mucho más ligeras en cuanto a memoria con respecto a otros modelos, que dependiendo de la longitud de su data de entrada consumirá más o menos memoria.

- Desventajas
  - ✓ Uso de la memoria durante la evaluación: Las TCN necesitan toda la secuencia hasta el punto actual para realizar la evaluación, lo cual lleva a un uso de memoria potencialmente mayor que una RNN.
  - ✓ Aprendizaje por transferencia: debido al problema mencionado en el punto anterior, el modelo podrá presentar ciertos inconvenientes con el rendimiento porque va a depender de cuánta información histórica se tenga para realizar las predicciones.

En los siguientes apartados se expondrá a mayor detalle diferentes características importantes para el entendimiento del comportamiento de la red.

**2.2.2.1 Red Convolucional 1D.** Lo que realiza una convolución en una dimensión es pasar un filtro sobre la entrada unidimensional y multiplicar los valores por los pesos del kernel para crear una salida (Alla & Adari, 2019). A continuación, se procede a ejemplificar el proceso de la convolución en una dimensión:

$$x = [10\ 5\ 15\ 20\ 10\ 20]$$

$$k = [1\ 0.2\ 0.1]$$

Siendo *"x"* el tensor de entrada y *"k"* el kernel con los pesos a aplicar para el proceso de la convolución.

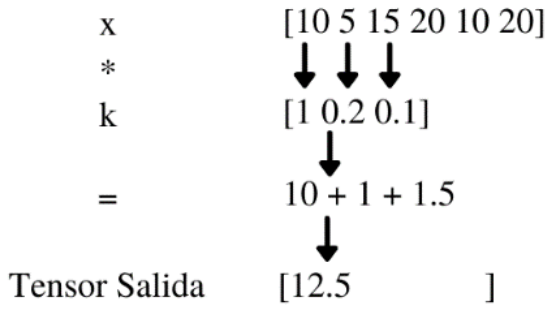


**Figura 5** Cálculo del primer elemento del tensor de salida

En la Figura 5, se observa que, para calcular el primer término del tensor de salida, se realiza el producto escalar con los 3 primeros elementos que coinciden con la longitud del kernel. Posteriormente, se realiza la suma de estos para la obtención del primer término del tensor de salida.

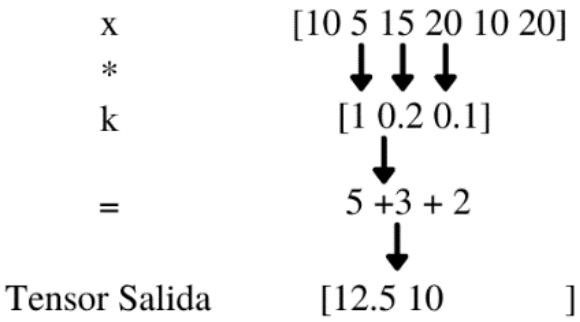


**Figura 6** Cálculo del segundo elemento del tensor de salida.

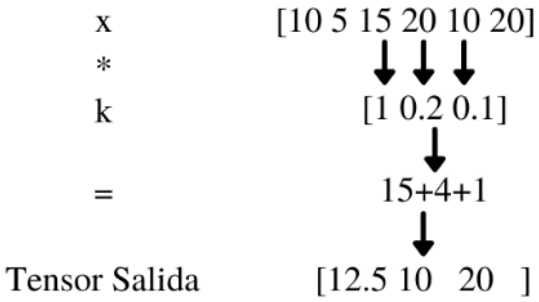


**Figura 7** Cálculo del tercer elemento del tensor de salida.

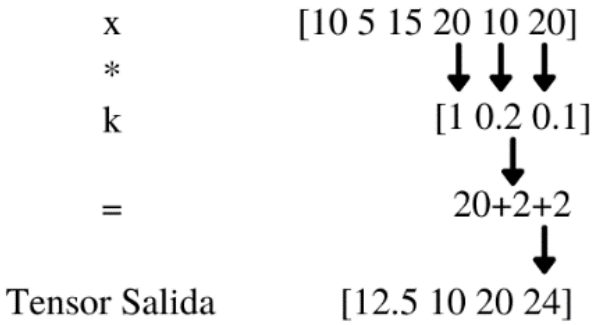


**Figura 8** Cálculo del cuarto elemento del tensor de salida

En las Figuras 6, 7 y 8 se observa cómo el procedimiento inicialmente descrito en la Figura 5 ha sido realizado de la misma manera tomando en cuenta que el kernel se ha ido desplazando en un elemento hacia la derecha para así poder obtener todos los elementos del tensor de salida.

Analizando el tensor de salida, se puede notar que posee un menor número de elementos que el de entrada, es por ello que se recurre al relleno de ceros para así garantizar que la secuencia de salida tenga la misma longitud que la de entrada. En otras palabras, se agregarán entradas adicionales de valor cero al final de cada capa para así garantizar lo anteriormente mencionado y, al ser estas de valor 0, no afectan en el resultado final.

**2.2.2.2 Convolución causal.** Convolución donde el tensor de salida es obtenido desde un instante *"t"* y sus antecedentes, es decir, un elemento del tensor de salida solo va a depender de los elementos que lo anteceden en el tensor de entrada(Liu et al., 2019).

Como se mencionó, el relleno de ceros es aplicado en la parte izquierda del tensor de entrada con la finalidad de garantizar la causalidad de la convolución. De esto se puede deducir que, para cada elemento del tensor de salida, su última dependencia de entrada tendrá el mismo índice que él. En la siguiente figura se ilustrará lo mencionado. Tomando como referencia un tensor de entrada con una longitud de valor 4 y un kernel de longitud 3.

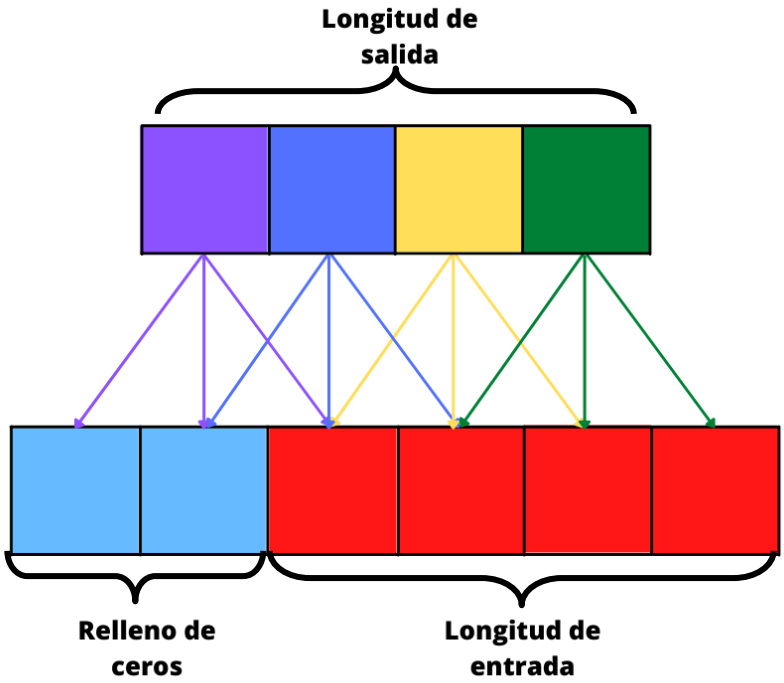


**Figura 9.** Esquematización del relleno de ceros.
**Fuente:** Elaboración propia, basado en la página web de Lässig, 2021

En la Figura 9, se observa que se utilizó un relleno de ceros igual a 2, para así poder obtener un tensor de salida de la misma longitud que el de entrada. Siempre que no exista dilatación, el número de entradas de relleno de ceros, necesarias para lograr mantener la longitud del tensor de salida igual al de entrada, será siempre igual al tamaño del kernel menos uno, como se puede comprobar en la figura antes mencionada.

**2.2.2.3 Dilatación.** Es la distancia entre los elementos de la secuencia de entrada que se utilizarán para el cálculo de la secuencia de salida. Trae como beneficio el aumento exponencial del campo receptivo sin aumentar el costo computacional. (Chaudhary et al., 2021; Liu et al., 2019)

Una cualidad deseada en un modelo de predicción es que el valor de una entrada específica, en la salida, dependa de todas las entradas anteriores presentes, es decir, todas las entradas que tienen un índice menor o igual a sí mismo.

Lo mencionado en el párrafo anterior se llevará a cabo siempre y cuando el campo receptivo, el cual es el conjunto de datos de la secuencia de entrada original que afectan en la salida, posea el tamaño de la longitud de entrada cumpliéndose así la “Cobertura de historial completo” (Bai et al., 2018)

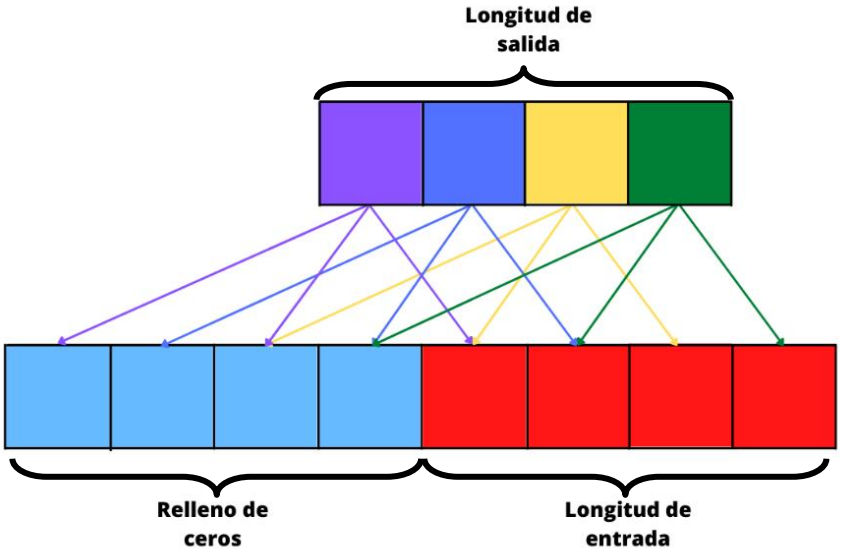


**Figura 10.** Convolución con dilatación de 2.
**Fuente:** Elaboración propia, basado en la página web de Lässig, 2021

Comparando la Figura 10 con la Figura 9 que posee una dilatación igual a 1, esta capa tiene un campo receptivo que se extiende a lo largo de una longitud de valor 5 en lugar de 3.

De manera general, una capa con una dilatación "d" y un kernel de valor "k" poseerá un campo receptivo que se extenderá a lo largo de una longitud de 1+d*(k-1). Partiendo de esto, si "d" es un valor fijo a lo largo de las capas, se requerirá una relación lineal en la longitud del tensor de entrada para lograr así una cobertura del campo receptivo completo.

Por lo mencionado anteriormente, se busca aumentar exponencialmente el valor de "d" a medida que se avanza por las capas. Por ello, se escoge un número entero "b" que será la dilatación base, que va a permitir calcular la dilatación "d" de una capa específica en función del número de capas debajo de ella, siendo "i" el número de capas debajo, obteniendo la siguiente fórmula (Bai et al., 2018):

$$d = b^i \tag{8}$$

**2.2.2.4 Bloques residuales.** Un bloque residual apila dos capas de convoluciones causales dilatadas, y los resultados de la operación se vuelven a agregar a las entradas para así obtener las salidas del bloque (Bai et al., 2018).

Cuando se requiere un campo receptivo muy grande se tienen que usar muchas capas, produciéndose así el problema del desvanecimiento de gradiente, el cual es resuelto por los bloques residuales (Y. Lin et al., 2020; Liu et al., 2019).

En el caso que el número de canales de las entradas y el número de filtros de las segundas capas de convolución causal dilatada difieran, se aplicará una convolución 1D a las entradas antes de agregar las salidas de la operación, para que así coincidan los anchos.

Entonces, el tamaño del campo receptivo “r” en una TCN, con una dilatación base “b” y un kernel de tamaño “k” se podrá calcular de la siguiente manera:

$$r = 1 + \sum_{i=0}^{n-1} 2 \cdot (k-1) \cdot b^i = 1 + 2 \cdot (k-1) \cdot \frac{b^n - 1}{b-1} \quad (9)$$

Lo cual conduce al mínimo número de bloques residuales “n” para lograr una cobertura histórica completa del tensor de entrada:

$$n = \left\lceil \log_b \left( \frac{(l-1) \cdot (b-1)}{(k-1) \cdot 2} + 1 \right) \right\rceil \quad (10)$$

**2.2.2.5 Activación, normalización y regularización.** Es necesario agregar funciones de activación sobre las capas convolucionales para lograr introducir no linealidades, por eso se agregan activaciones ReLU a los bloques residuales después de la capa convolucional. Estas funciones de activación tienen la característica de que devuelven valor cero cuando reciben valores negativos, evitando así que la red sufra del desvanecimiento de gradiente (Li et al., 2020).

$$f(x) = \max(0, x) \quad (11)$$

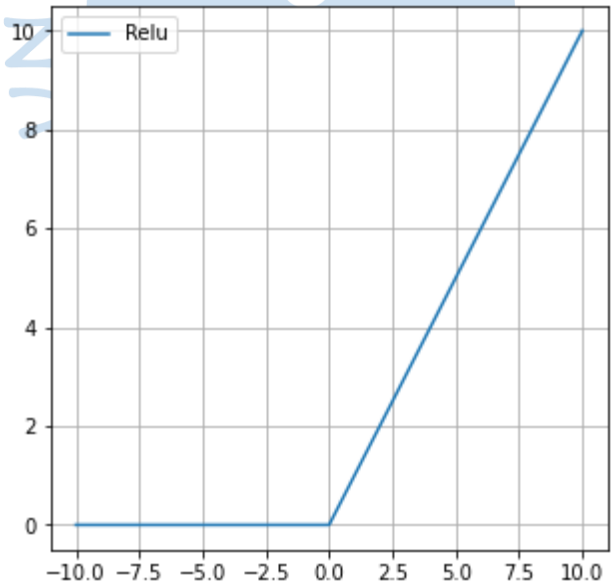


**Figura 11.** Gráfica de función ReLU
**Fuente:** Elaboración propia

Así mismo, para la normalización de la entrada de capas ocultas, que logran contrarrestar el problema del gradiente explosivo entre otras cosas, se aplica una normalización de pesos a cada capa convolucional. Finalmente, para evitar el sobreajuste, se introduce la regularización por abandono o *“dropout”* después de cada capa convolucional en cada bloque residual.

Habiéndose explicado las características principales de una TCN, en la Figura 12 se representa su estructura.

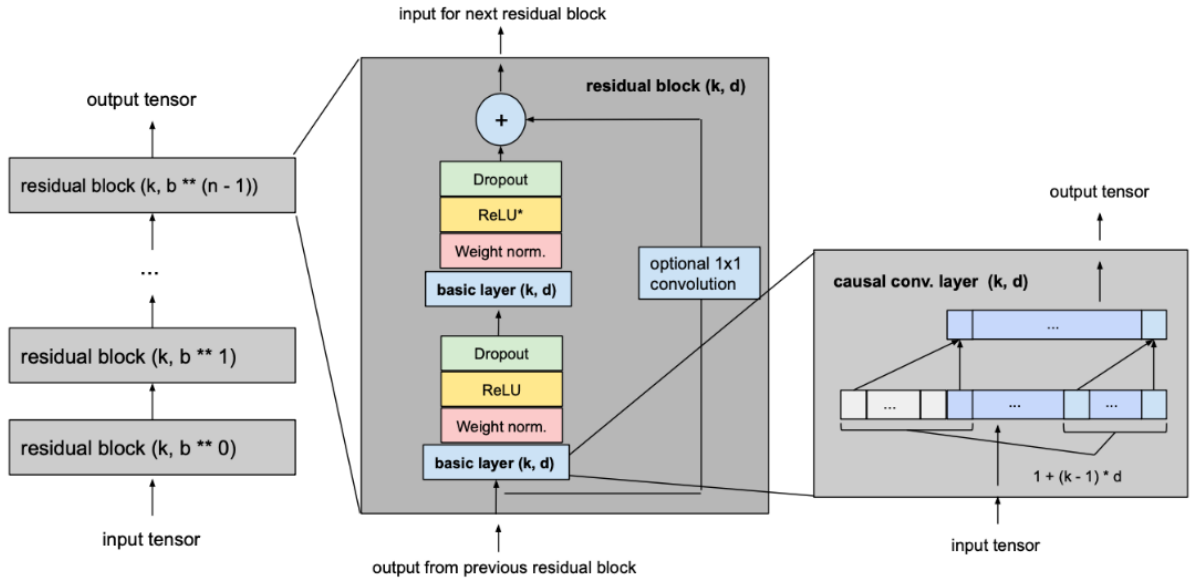


**Figura 12** Representación de una TCN
**Fuente:** (Lässig, 2021)

# Capítulo 3

## Caracterización de la demanda eléctrica de edificios seleccionados

Este capítulo se redactó gracias a las contribuciones de la tesis realizada por Guillermo Perea Vasquez y Julio Omid Vasquez Silva, titulada “Metodología para realizar auditoría de energía eléctrica. Caso aplicativo: Edificio de educación superior”. Debido a la situación de emergencia sanitaria que atraviesa el país, se consideró propicio recoger la información que permita caracterizar la demanda eléctrica del Edificio E y Edificio de derecho, como los equipos eléctricos utilizados, régimen de actividades y datos característicos de los edificios.

Para la presente investigación se utilizaron datos históricos del suministro eléctrico 16196527, el cual alimenta a los dos edificios mencionados anteriormente. Dicho suministro cuenta con un transformix de medición indirecta para la facturación y con un contrato de usuario libre, por lo que la facturación no se encuentra sujeta a una tarifa convencional.

El motivo por el cual se elige, para este estudio, el suministro eléctrico mencionado es porque la carga que alimenta tiene potencial de crecimiento y podría ser útil predecir con precisión la demanda eléctrica del suministro en el futuro. Estas predicciones pueden ser información muy valiosa en proyectos de mejora de eficiencia energética, en proyectos fotovoltaicos, o en cualquier tipo de proyectos donde se busque ahorro económico a partir de una estrategia de uso eficiente de la energía eléctrica.

A continuación, se realizará una caracterización de la demanda eléctrica de los edificios seleccionados para el estudio.

### 3.1 Equipos eléctricos característicos

En la Tabla 1 se detallan los equipos eléctricos y la carga promedio que representan. Para la identificación de estos se utilizaron los diagramas unifilares de los edificios.

Se identifica que en cada aula existen tres tableros: el tablero general estabilizador, que alimenta los equipos utilizados por los docentes como proyector, Ecran, etc.; el tablero general de alumbrado, que alimenta las luminarias del salón; y el tablero general de emergencia, que alimenta los tomacorrientes y aires acondicionados de cada salón. Adicionalmente, se observa que las cargas suelen ser similares para cada salón.

**Tabla 1**. Equipos eléctricos

| Equipo | Potencia de carga |
|---|---|
| Luminarias led | 25 W |
| Ventiladores | 80 W |
| Aire acondicionado | 1500 W |
| Laptops | 200 W |
| Celulares | 20 W |
| Secadora de manos | 300 W |
| Computadora de escritorio | 500 W |
| Ecran | 200 W |
| Cámaras de seguridad | 500 W |
| Máquinas expendedoras | 1500 W |
| Microondas | 1000 W |
| Proyector | 1000 W |
| Puerta enrollable | 800 W |
| Refrigeradora | 1000 W |

**Fuente:** (Vasquez & Perea, 2020)

## 3.2 Régimen de actividades y horario de funcionamiento

En el trabajo de tesis mencionado (Vasquez & Perea, 2020), realizaron un análisis de la ocupación del Edificio “E” y Edificio de Derecho, a partir de la obtención de datos representativos brindados por el Departamento de Mantenimiento de la universidad.

A partir de estos aportes, se puede conocer un régimen de actividades característico de los edificios, de lunes a sábado, como se muestra a continuación.

**Tabla 2.** Régimen de actividades característico - Lunes

| | Edificio de Derecho | | Edificio "E" | | | |
|---|---|---|---|---|---|---|
| **Horario** | **Número de aulas utilizadas** | **Aforo acumulado (personas)** | **Número de aulas utilizadas** | **Aforo acumulado (personas)** | **Aforo total acumulado** | **Número total de aulas utilizadas** |
| 7:00-7:59 am | 2 | 176 | 16 | 1714 | 1890 | 18 |
| 8:00-8:59 am | 4 | 336 | 16 | 1714 | 2050 | 20 |
| 9:00-9:59 am | 4 | 336 | 15 | 1606 | 1942 | 19 |
| 10:00-10:59 am | 4 | 336 | 15 | 1606 | 1942 | 19 |
| 11:00-11:59 am | 5 | 416 | 16 | 1714 | 2130 | 21 |
| 12:00-12:59 pm | 5 | 416 | 14 | 1516 | 1932 | 19 |
| 1:00-1:59 pm | 0 | 0 | 16 | 1714 | 1714 | 16 |
| 2:00-2:59 pm | 0 | 0 | 16 | 1714 | 1714 | 16 |
| 3:00-3:59 pm | 6 | 486 | 15 | 1633 | 2119 | 21 |
| 4:00-4:59 pm | 8 | 620 | 15 | 1633 | 2253 | 23 |
| 5:00-5:59 pm | 7 | 524 | 16 | 1714 | 2238 | 23 |
| 6:00-6:59 pm | 4 | 336 | 16 | 1714 | 2050 | 20 |
| 7:00-7:59 pm | 4 | 336 | 11 | 1124 | 1460 | 15 |
| 8:00-8:59 pm | 1 | 96 | 11 | 1124 | 1220 | 12 |
| 9:00-9:59 pm | 0 | 0 | 2 | 151 | 151 | 2 |
| 10:00-10:59 pm | 0 | 0 | 2 | 151 | 151 | 2 |
| 11:00-11:59 pm | 0 | 0 | 0 | 0 | 0 | 0 |

**Fuente:** (Vasquez & Perea, 2020)

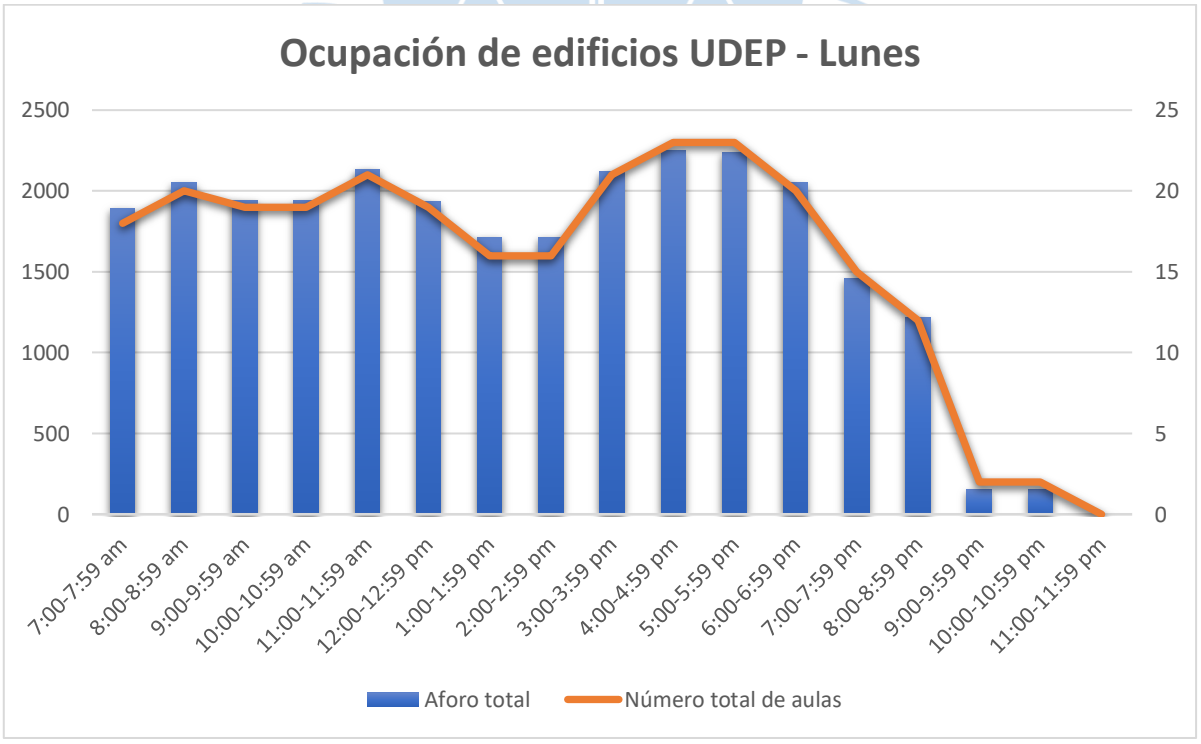


**Figura 13.** Ocupación de edificios Facultad de Derecho y Edificio E – Lunes

**Fuente:** (Vasquez & Perea, 2020)

**Tabla 3.** Régimen de actividades característico - Martes

| | Edificio de Derecho | | Edificio "E" | | | |
|---|---|---|---|---|---|---|
| **Horario** | **Número de aulas utilizadas** | **Aforo acumulado (personas)** | **Número de aulas utilizadas** | **Aforo acumulado (personas)** | **Aforo total acumulado** | **Número total de aulas utilizadas** |
| 7:00-7:59 am | 0 | 0 | 15 | 1606 | 1606 | 15 |
| 8:00-8:59 am | 1 | 96 | 15 | 1606 | 1702 | 16 |
| 9:00-9:59 am | 7 | 556 | 16 | 1714 | 2270 | 23 |
| 10:00-10:59 am | 8 | 620 | 16 | 1714 | 2334 | 24 |
| 11:00-11:59 am | 6 | 470 | 15 | 1614 | 2084 | 21 |
| 12:00-12:59 pm | 5 | 416 | 15 | 1614 | 2030 | 20 |
| 1:00-1:59 pm | 0 | 0 | 3 | 494 | 494 | 3 |
| 2:00-2:59 pm | 0 | 0 | 3 | 494 | 494 | 3 |
| 3:00-3:59 pm | 1 | 80 | 15 | 1515 | 1595 | 16 |
| 4:00-4:59 pm | 2 | 160 | 15 | 1515 | 1675 | 17 |
| 5:00-5:59 pm | 8 | 620 | 16 | 1614 | 2234 | 24 |
| 6:00-6:59 pm | 7 | 566 | 16 | 1614 | 2180 | 23 |
| 7:00-7:59 pm | 2 | 160 | 12 | 1244 | 1404 | 14 |
| 8:00-8:59 pm | 2 | 160 | 12 | 1244 | 1404 | 14 |
| 9:00-9:59 pm | 0 | 0 | 3 | 237 | 237 | 3 |
| 10:00-10:59 pm | 0 | 0 | 3 | 237 | 237 | 3 |
| 11:00-11:59 pm | 0 | 0 | 0 | 0 | 0 | 0 |

**Fuente:** (Vasquez & Perea, 2020)

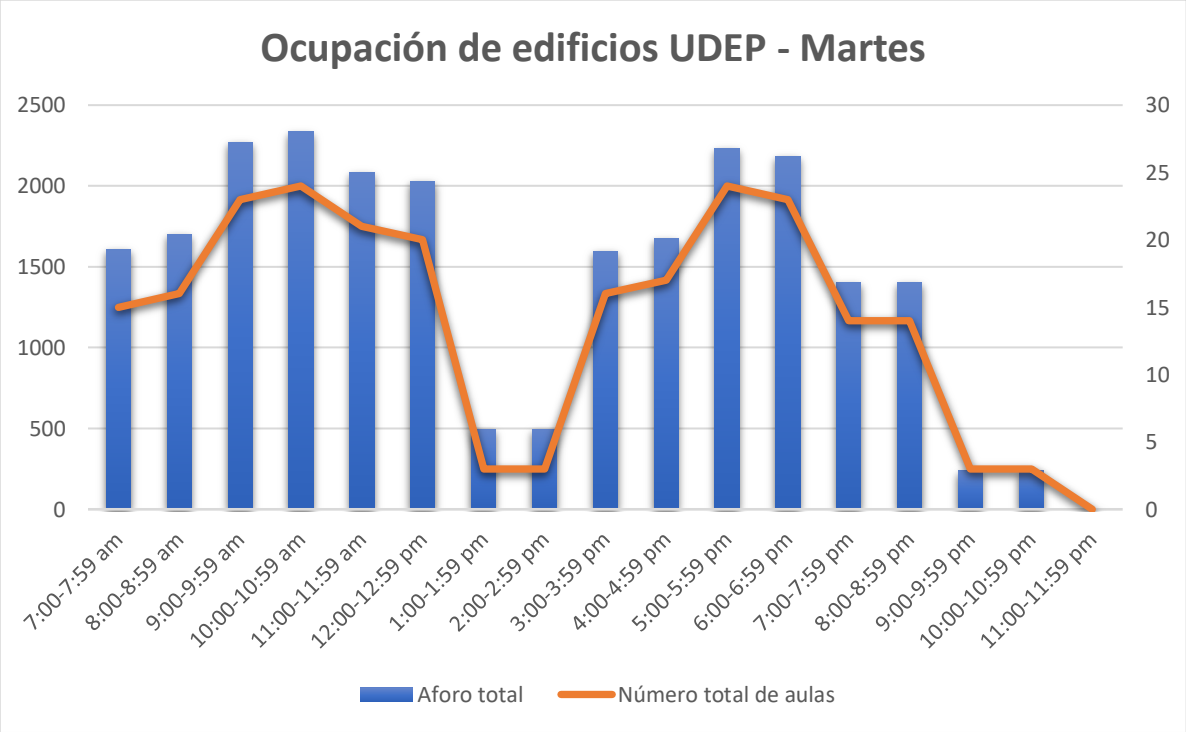


**Figura 14.** Ocupación de edificios Facultad de Derecho y Edificio E – Martes

**Fuente:** (Vasquez & Perea, 2020)

**Tabla 4.** Régimen de actividades característico - Miércoles

| | Edificio de Derecho | | Edificio "E" | | | |
|---|---|---|---|---|---|---|
| **Horario** | **Número de aulas utilizadas** | **Aforo acumulado (personas)** | **Número de aulas utilizadas** | **Aforo acumulado (personas)** | **Aforo total acumulado** | **Número total de aulas utilizadas** |
| 7:00-7:59 am | 1 | 80 | 15 | 1615 | 1695 | 16 |
| 8:00-8:59 am | 2 | 176 | 16 | 1714 | 1890 | 18 |
| 9:00-9:59 am | 6 | 512 | 15 | 1606 | 2118 | 21 |
| 10:00-10:59 am | 6 | 512 | 15 | 1606 | 2118 | 21 |
| 11:00-11:59 am | 4 | 336 | 13 | 1428 | 1764 | 17 |
| 12:00-12:59 pm | 4 | 336 | 13 | 1437 | 1773 | 17 |
| 1:00-1:59 pm | 0 | 0 | 12 | 1130 | 1130 | 12 |
| 2:00-2:59 pm | 0 | 0 | 12 | 1130 | 1130 | 12 |
| 3:00-3:59 pm | 6 | 512 | 14 | 1498 | 2010 | 20 |
| 4:00-4:59 pm | 8 | 620 | 15 | 1606 | 2226 | 23 |
| 5:00-5:59 pm | 6 | 470 | 16 | 1714 | 2184 | 22 |
| 6:00-6:59 pm | 6 | 470 | 14 | 1515 | 1985 | 20 |
| 7:00-7:59 pm | 0 | 0 | 9 | 1049 | 1049 | 9 |
| 8:00-8:59 pm | 0 | 0 | 9 | 1049 | 1049 | 9 |
| 9:00-9:59 pm | 0 | 0 | 5 | 565 | 565 | 5 |
| 10:00-10:59 pm | 0 | 0 | 2 | 159 | 159 | 2 |
| 11:00-11:59 pm | 0 | 0 | 0 | 0 | 0 | 0 |

**Fuente:** (Vasquez & Perea, 2020)

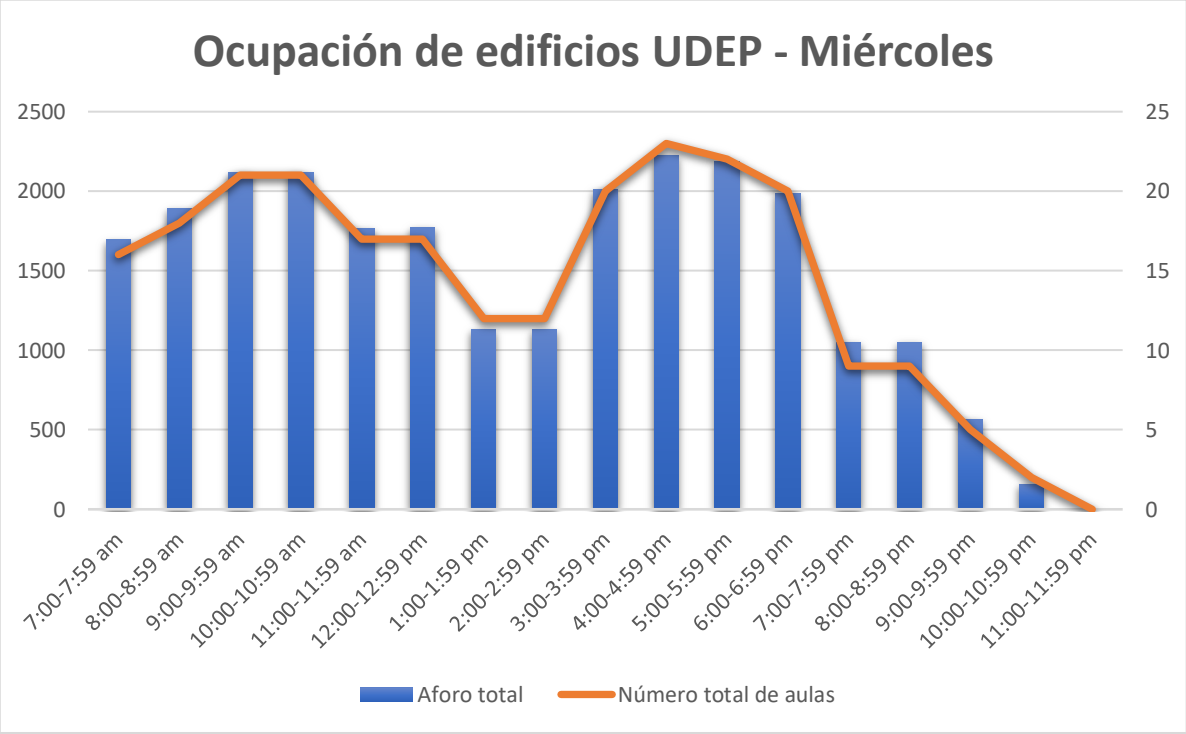


**Figura 15**. Ocupación de edificios Facultad de Derecho y Edificio E – Miércoles

**Fuente:** (Vasquez & Perea, 2020)

**Tabla 5.** Régimen de actividades característico - Jueves

| | Edificio de Derecho | | Edificio "E" | | | |
|---|---|---|---|---|---|---|
| **Horario** | **Número de aulas utilizadas** | **Aforo acumulado (personas)** | **Número de aulas utilizadas** | **Aforo acumulado (personas)** | **Aforo total acumulado** | **Número total de aulas utilizadas** |
| 7:00-7:59 am | 0 | 0 | 11 | 1240 | 1240 | 11 |
| 8:00-8:59 am | 2 | 160 | 16 | 1714 | 1874 | 18 |
| 9:00-9:59 am | 7 | 566 | 16 | 1714 | 2280 | 23 |
| 10:00-10:59 am | 7 | 566 | 16 | 1714 | 2280 | 23 |
| 11:00-11:59 am | 7 | 566 | 16 | 1714 | 2280 | 23 |
| 12:00-12:59 pm | 6 | 512 | 16 | 1714 | 2226 | 22 |
| 1:00-1:59 pm | 0 | 0 | 8 | 961 | 961 | 8 |
| 2:00-2:59 pm | 0 | 0 | 8 | 961 | 961 | 8 |
| 3:00-3:59 pm | 6 | 512 | 13 | 1318 | 1830 | 19 |
| 4:00-4:59 pm | 8 | 620 | 13 | 1318 | 1938 | 21 |
| 5:00-5:59 pm | 4 | 326 | 15 | 1636 | 1962 | 19 |
| 6:00-6:59 pm | 5 | 432 | 12 | 1348 | 1780 | 17 |
| 7:00-7:59 pm | 4 | 352 | 14 | 1521 | 1873 | 18 |
| 8:00-8:59 pm | 0 | 0 | 14 | 1521 | 1521 | 14 |
| 9:00-9:59 pm | 0 | 0 | 2 | 159 | 159 | 2 |
| 10:00-10:59 pm | 0 | 0 | 0 | 0 | 0 | 0 |
| 11:00-11:59 pm | 0 | 0 | 0 | 0 | 0 | 0 |

**Fuente:** (Vasquez & Perea, 2020)

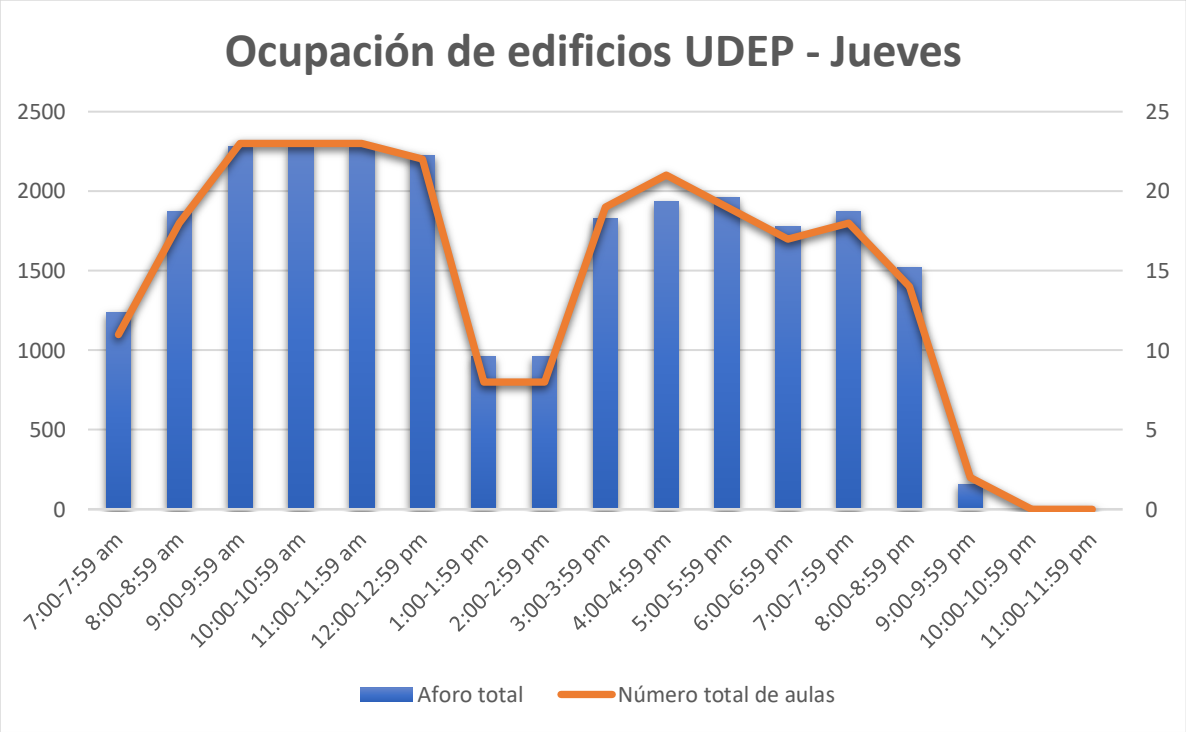


**Figura 16**. Ocupación de edificios Facultad de Derecho y Edificio E – Jueves

**Fuente:** (Vasquez & Perea, 2020)

**Tabla 6.** Régimen de actividades característico - Viernes

| | Edificio de Derecho | | Edificio "E" | | | |
|---|---|---|---|---|---|---|
| **Horario** | **Número de aulas utilizadas** | **Aforo acumulado (personas)** | **Número de aulas utilizadas** | **Aforo acumulado (personas)** | **Aforo total acumulado** | **Número total de aulas utilizadas** |
| 7:00-7:59 am | 1 | 96 | 15 | 1633 | 1729 | 16 |
| 8:00-8:59 am | 1 | 96 | 16 | 1714 | 1810 | 17 |
| 9:00-9:59 am | 5 | 348 | 16 | 1714 | 2062 | 21 |
| 10:00-10:59 am | 8 | 620 | 16 | 1714 | 2334 | 24 |
| 11:00-11:59 am | 7 | 540 | 13 | 1537 | 2077 | 20 |
| 12:00-12:59 pm | 6 | 444 | 12 | 1429 | 1873 | 18 |
| 1:00-1:59 pm | 0 | 0 | 13 | 1426 | 1426 | 13 |
| 2:00-2:59 pm | 0 | 0 | 13 | 1426 | 1426 | 13 |
| 3:00-3:59 pm | 7 | 566 | 12 | 1362 | 1928 | 19 |
| 4:00-4:59 pm | 8 | 620 | 14 | 1521 | 2141 | 22 |
| 5:00-5:59 pm | 6 | 444 | 14 | 1416 | 1860 | 20 |
| 6:00-6:59 pm | 6 | 444 | 14 | 1416 | 1860 | 20 |
| 7:00-7:59 pm | 4 | 268 | 14 | 1422 | 1690 | 18 |
| 8:00-8:59 pm | 3 | 214 | 13 | 1314 | 1528 | 16 |
| 9:00-9:59 pm | 0 | 0 | 5 | 418 | 418 | 5 |
| 10:00-10:59 pm | 0 | 0 | 0 | 0 | 0 | 0 |
| 11:00-11:59 pm | 0 | 0 | 0 | 0 | 0 | 0 |

**Fuente:** (Vasquez & Perea, 2020)

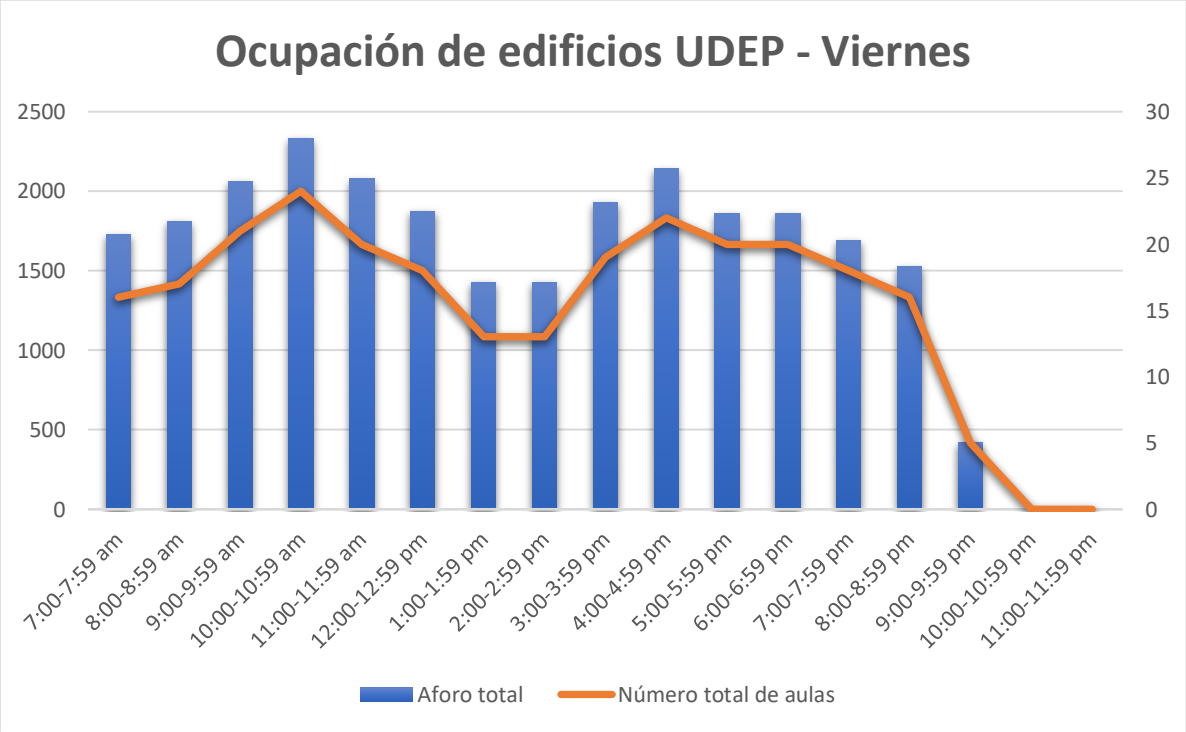


**Figura 17.** Ocupación de edificios Facultad de Derecho y Edificio E – Viernes

**Fuente:** (Vasquez & Perea, 2020)

**Tabla 7.** Régimen de actividades característico - Sábado

| | Edificio de Derecho | | Edificio "E" | | | |
|---|---|---|---|---|---|---|
| **Horario** | **Número de aulas utilizadas** | **Aforo acumulado (personas)** | **Número de aulas utilizadas** | **Aforo acumulado (personas)** | **Aforo total acumulado** | **Número total de aulas utilizadas** |
| 7:00-7:59 am | 1 | 80 | 12 | 1336 | 1416 | 13 |
| 8:00-8:59 am | 5 | 364 | 14 | 1516 | 1880 | 19 |
| 9:00-9:59 am | 6 | 444 | 13 | 1511 | 1955 | 19 |
| 10:00-10:59 am | 6 | 444 | 13 | 1323 | 1767 | 19 |
| 11:00-11:59 am | 6 | 444 | 12 | 1223 | 1667 | 18 |
| 12:00-12:59 pm | 5 | 364 | 13 | 1124 | 1488 | 18 |
| 1:00-1:59 pm | 5 | 364 | 7 | 766 | 1130 | 12 |
| 2:00-2:59 pm | 2 | 108 | 4 | 382 | 490 | 6 |
| 3:00-3:59 pm | 3 | 188 | 0 | 0 | 188 | 3 |
| 4:00-4:59 pm | 3 | 188 | 1 | 81 | 269 | 4 |
| 5:00-5:59 pm | 2 | 108 | 1 | 81 | 189 | 3 |
| 6:00-6:59 pm | 2 | 108 | 0 | 0 | 108 | 2 |
| 7:00-7:59 pm | 0 | 0 | 0 | 0 | 0 | 0 |
| 8:00-8:59 pm | 0 | 0 | 0 | 0 | 0 | 0 |
| 9:00-9:59 pm | 0 | 0 | 0 | 0 | 0 | 0 |
| 10:00-10:59 pm | 0 | 0 | 0 | 0 | 0 | 0 |
| 11:00-11:59 pm | 0 | 0 | 0 | 0 | 0 | 0 |

**Fuente:** (Vasquez & Perea, 2020)

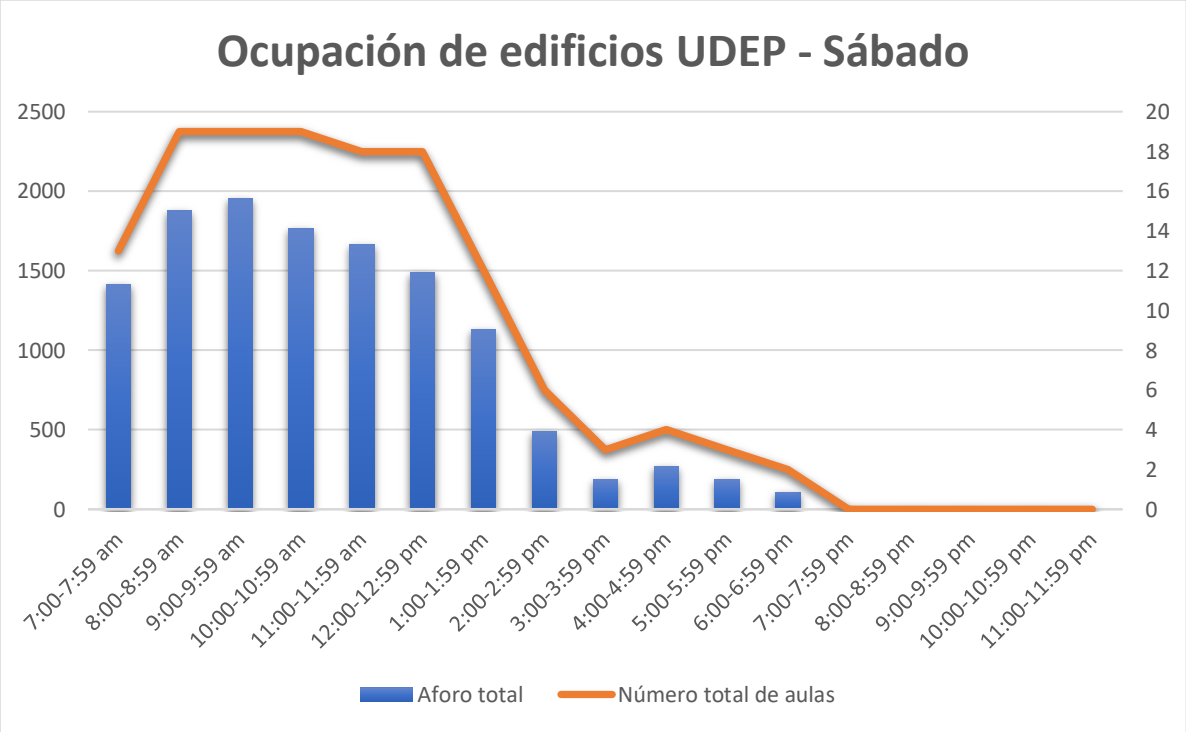


**Figura 18.** Ocupación de edificios Facultad de Derecho y Edificio E – Sábado

**Fuente:** (Vasquez & Perea, 2020)

## 3.3 Datos característicos de edificios y de subestación de distribución

### *3.3.1 Edificios Facultad de Derecho y Edificio E*

- Rubro de negocio: educación universitaria.
- Horario de trabajo: 7:00 a.m. – 9:00 p.m.
- Ubicación geográfica: Piura, departamento de Piura.
- Tiempo de operación del sistema eléctrico: todo el día
- Demanda eléctrica promedio: 154 kW, 230-400 V.

### *3.3.2 Subestación de distribución*

- Tipo: Caseta a subnivel
- Potencia de transformación: 1000 kVA.
- Tensión primaria trifásica: 22.9 kV
- Tensión secundaria trifásica: 0.4-0.23 kV.
- Esquema de conexión a tierra NT.
- Alimentación subterránea
- Disponibilidad del neutro: se presenta disponible y aterrado
- Cargas que alimenta: Edificio “E” y Edificio de Derecho
- Grupo electrógeno: 350 kW (stand by), 3 fases con neutro, 400 – 230 V
- Contrato de usuario libre

UNIVERSITAS STUDIORUM PIURENSIS

# Capítulo 4

# Preparación del set de datos

## 4.1 Datos de entrada y salida deseada

Observar, graficar y analizar el comportamiento de la data son los procedimientos más importantes para el análisis de series de tiempo. Todo esto permite poder entender la complejidad que se va a modelar (Sanjinés Tudela, 2011). Por esta razón se realizará una pequeña introducción a las series temporales, donde aparte de explicarse de manera breve en qué consisten, se mostrarán las gráficas de los datos que forman parte de este trabajo de tesis.

Por otro lado, se desarrollará el manejo de la data, donde se explicará cómo se preparó la data antes de ser analizada a profundidad. Después, se hablará sobre el análisis de la data y sobre cómo este facilita clasificarla y extraer ciertas características de gran importancia que servirán para obtener el set de datos final con el que se entrenará la red.

## 4.2 Series temporales

Una serie temporal puede ser descrita como el resultado de observar valores de una variable a lo largo del tiempo en intervalos regulares (cada media hora, cada hora, cada día, cada año, etc.). Asimismo, se puede considerar una serie de temporal como la observación de la realización de un proceso entre los instantes 1 y X: $A_1, A_2, A_3, \ A_X$. (Orellana Romero, 2012).

Existen muchos programas y lenguajes de programación que permiten graficar y analizar series temporales. Manteniendo la metodología a emplear, se graficará la data haciendo uso del lenguaje de programación Python en la versión 3.7; asimismo, se hará uso de la librería Plotly[1] y Pandas[2].

Es importante mencionar que el set de datos a utilizar, correspondiente a la demanda eléctrica, ha sido suministrado por el área de Mantenimiento de la Universidad de Piura.

---

[1] Librería idónea para graficar series temporales que se caracteriza por ser de código abierto y generar graficas interactivas de diferente tipo .(*Plotly Open Source Graphing Libraries | | Plotly*, n.d.)

[2] Librería de código abierto que permite el manejo de datos de forma fácil y flexible, logrando así ser una herramienta para el análisis y manipulación de data.(*Pandas - Python Data Analysis Library*, n.d.)

```
import pandas as pd
import plotly.express as px
df = pd.read_csv('/content/drive/InitialDataSetFeaturesNoOHE.csv')
fig = px.line(df, 'date', 'PotAct', labels={"PotAct": "Potencia Activa (kW)", "date": "Tiempo"})
fig.update_layout(title_text='Curva de consumo eléctrico Universidad de Piura: Edificio E y Derecho -- Marzo 2019-Febrero 2020', title_x=0.5)
fig.show()
```

En las dos primeras líneas del código mostrado se realizó la importación de las librerías a utilizar; en las siguientes líneas, lo que se hizo fue crear una variable que contenga el set de datos a graficar en formato ".csv". Posteriormente, se utilizó la librería Plotly para poder obtener la gráfica de la potencia activa desde marzo del 2019 a febrero del 2020, la cual corresponde al periodo de tiempo en el que está comprendido el set de datos, del cual se hablará a profundidad en otros acápites.

En la Figura 19, se pueden distinguir ciertos valores atípicos, que pueden pertenecer a apagones o momentos en los que falló la comunicación y no se logró realizar la medición. En los próximos acápites se hablará y analizará a profundidad el set de datos en su totalidad para poder lograr un correcto modelado de este.

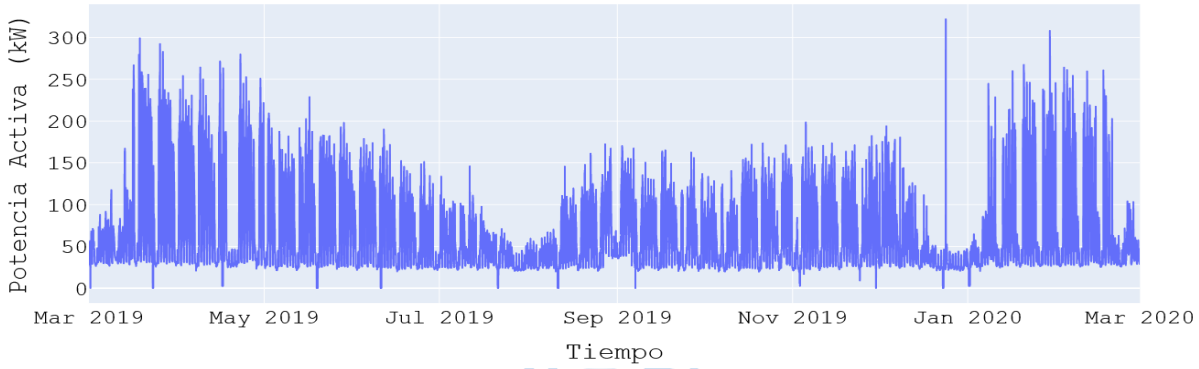


Figura 19 Gráfica del set de datos en su totalidad

## 4.3 Manejo del set de datos

El set de datos se encontraba en un inicio compuesto de dos partes. Una parte era un conjunto de archivos pertenecientes a las lecturas obtenidas por la entidad encargada del suministro eléctrico. La segunda parte, estaba compuesta por las diferentes lecturas de variables extraídas de la estación de la Universidad de Piura, ya que se consideró que podría existir una relación entre algunas variables del clima y la demanda eléctrica. Además, en distinta bibliografía se hacía uso de estas variables para lograr una mejor precisión en la predicción. En la Figura 20 se muestra el flujo de trabajo que se siguió para el manejo del set de datos y la redacción del presente capítulo.

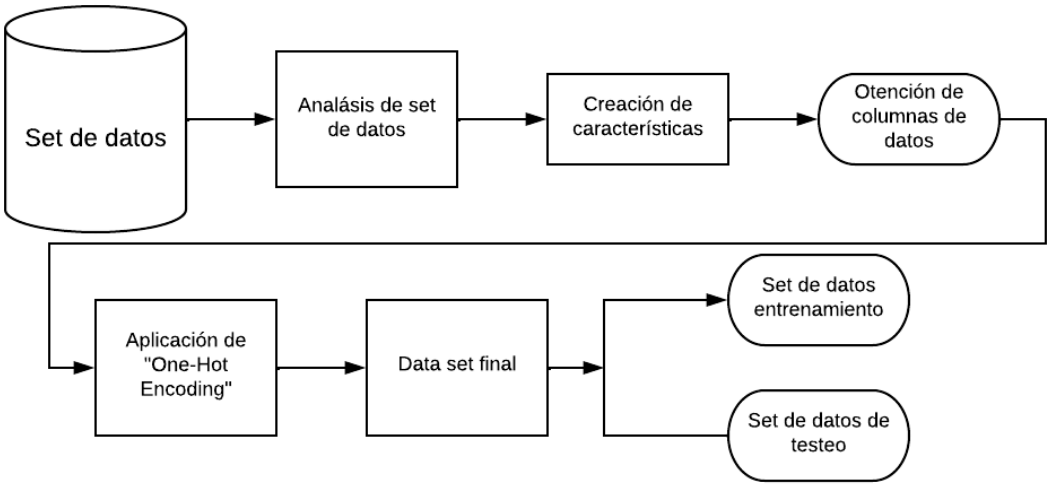


**Figura 20**. Diagrama de flujo para el manejo del set de datos

En la Figura 21 se muestra un extracto del set de datos, donde se tiene una columna asignada para el día y otra columna para la hora en que es tomada la lectura. Por ello, haciendo uso de las herramientas que brinda Excel, primero se procedió a cambiar el formato de la columna de hora ya que se requiere que se muestre en el formato "HH:MM: SS". Luego, se unieron las columnas "Día" y "hora" para lograr el formato "AAAA:MM:DD HH:MM: SS", ya que con este formato se trabaja en el código escrito en lenguaje Python.

| | | Potencia |
|---|---|---|
| **POTENCIA** | | Activa |
| | | Entregado |
| Día | hora | kW |
| 01/03/2019 | 0:15 | 0.317 |
| 01/03/2019 | 0:30 | 0.3108 |
| 01/03/2019 | 0:45 | 0.3151 |
| 01/03/2019 | 1:00 | 0.315 |
| 01/03/2019 | 1:15 | 0.3159 |

**Figura 21** Captura del set de datos Correspondiente a las lecturas del suministro

Con lo anteriormente mencionado, se obtuvo como producto final un set datos que se ve como en la Figura 22:

| date | Pot Act (kW) |
|---|---|
| 2019-03-01 00:00:00 | 29.31199744 |
| 2019-03-01 00:30:00 | 28.22945208 |
| 2019-03-01 01:00:00 | 28.89563384 |
| 2019-03-01 01:30:00 | 28.47927024 |
| 2019-03-01 02:00:00 | 28.64581568 |
| 2019-03-01 02:30:00 | 28.3127248 |

**Figura 22.** Set de datos con formato de fecha requerido

Teniendo el set de datos correspondiente a las lecturas del suministro eléctrico, se procede a revisar el estado del set de datos que corresponde a las diferentes variables que son tomadas por la estación meteorológica.

El set original de datos meteorológicos cuenta con 34 parámetros climáticos, de los cuales solo se consideraron tres: temperatura ambiente, humedad relativa y radiación solar. El set de datos ya se encuentra en el formato de fecha requerido, es por ello que solo se procedió a unirse con el data set correspondiente a las lecturas del suministro eléctrico. Cabe mencionar que ambos sets de datos tienen una diferencia importante, que es la frecuencia con la que han sido tomados los datos, siendo cada 15 minutos para el suministro y cada media hora para la estación meteorológica.

En el data set de las lecturas del suministro eléctrico, se procedió a realizar promedios entre grupos de datos consecutivos, tomados de dos en dos. De esta manera, se pudo cambiar la frecuencia de datos de quince minutos a media hora, pudiendo así realizar la unión de ambos sets de datos.

| date | PotAct | TempOut | OutHum | SolarRad |
|---|---|---|---|---|
| 2019-03-01 00:00:00 | 29.311997 | 26.133333 | 72.333333 | 0.0 |
| 2019-03-01 00:30:00 | 28.229452 | 25.833333 | 74.333333 | 0.0 |
| 2019-03-01 01:00:00 | 28.895634 | 25.500000 | 76.666667 | 0.0 |
| 2019-03-01 01:30:00 | 28.479270 | 25.300000 | 78.666667 | 0.0 |
| 2019-03-01 02:00:00 | 28.645816 | 25.200000 | 80.000000 | 0.0 |

**Figura 23.** Set de datos a utilizar

La Figura 23, corresponde al set de datos a utilizar ya convertido a formato ".csv", que es con el que se trabajará para el análisis, clasificación, posteriormente el modelado y obtención de resultados.

### 4.4 Análisis del set de datos

El análisis del set de datos permite obtener una descripción de los elementos que lo constituyen. En otras palabras, consiste en la verificación de cuáles son los componentes presentes en la serie o cuál es el comportamiento que rige los datos observados.

Como primer paso, se procedió a la eliminación de los atípicos que se pudieron observar en la Figura 19. Estos fueron reemplazados utilizando las observaciones que se presentaron a esa misma hora en una semana anterior o posterior a la fecha del atípico, obteniendo como resultado la siguiente gráfica:

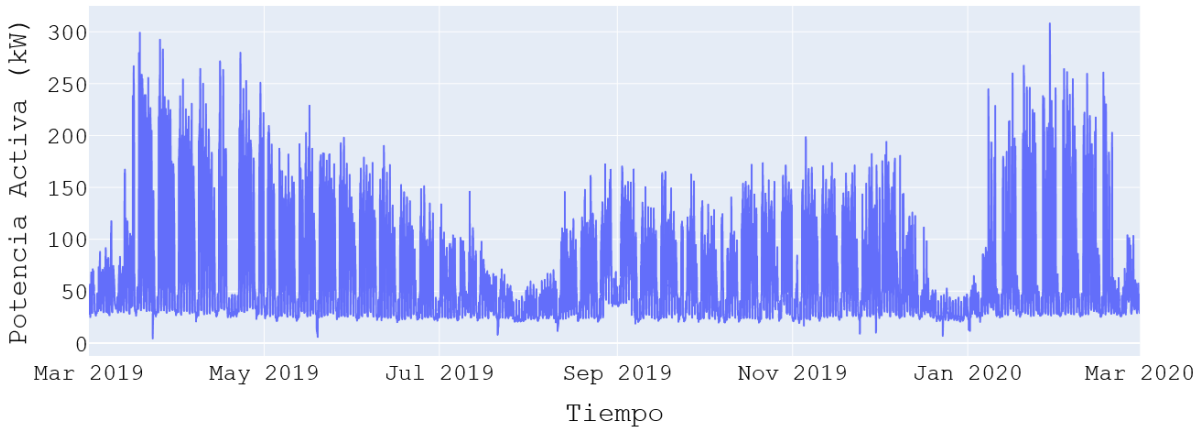


**Figura 24**. Gráfica del set de datos sin atípicos

Realizando un análisis rápido, se puede observar cierto patrón a lo largo de la gráfica, y es que cada cierto tiempo hay un pico y luego un descenso. Asimismo, se observa que hay épocas del año en la que el consumo disminuye notablemente.

Lo mencionado anteriormente se podrá visualizar en las siguientes cuatro figuras donde se graficará un día aleatorio del set de datos, una semana del set de datos, un mes del set de datos y seis meses del set de datos.

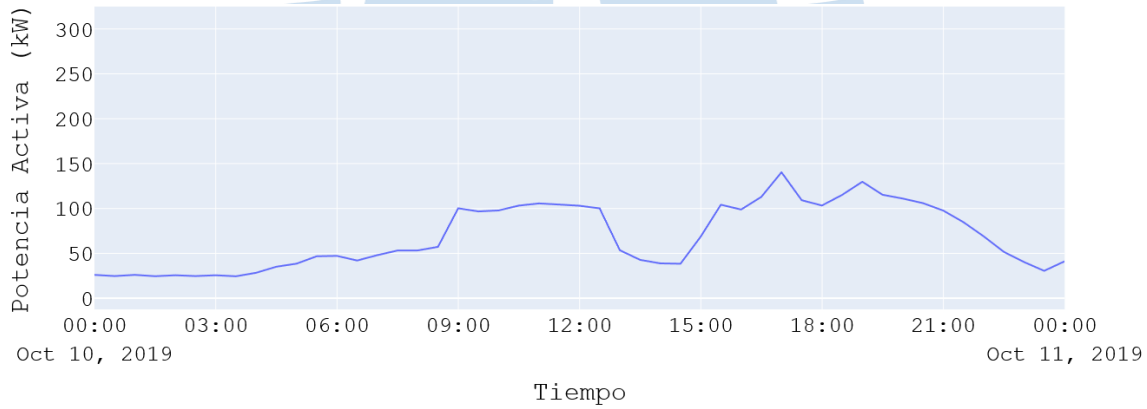


**Figura 25.** Gráfica de 1 día perteneciente al jueves 10 de octubre del 2019.

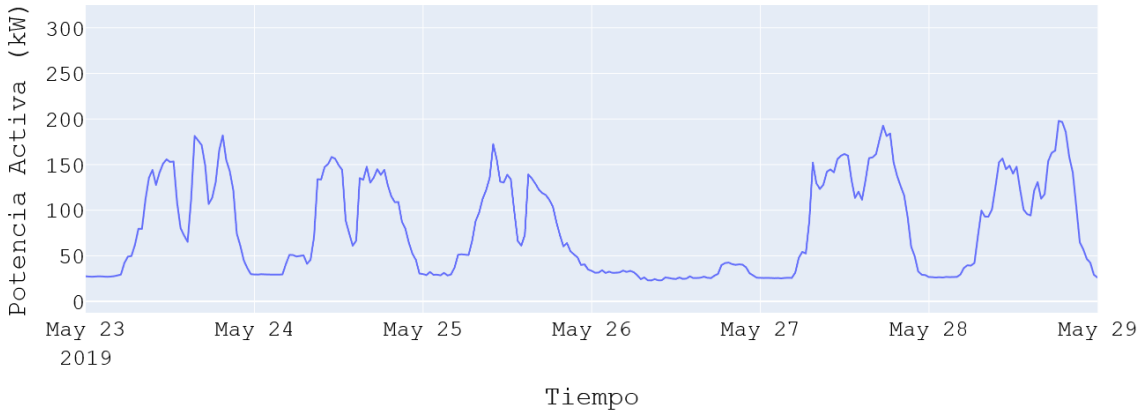


**Figura 26**. Gráfica de 1 semana del jueves 23 de mayo al miércoles 29 de mayo.

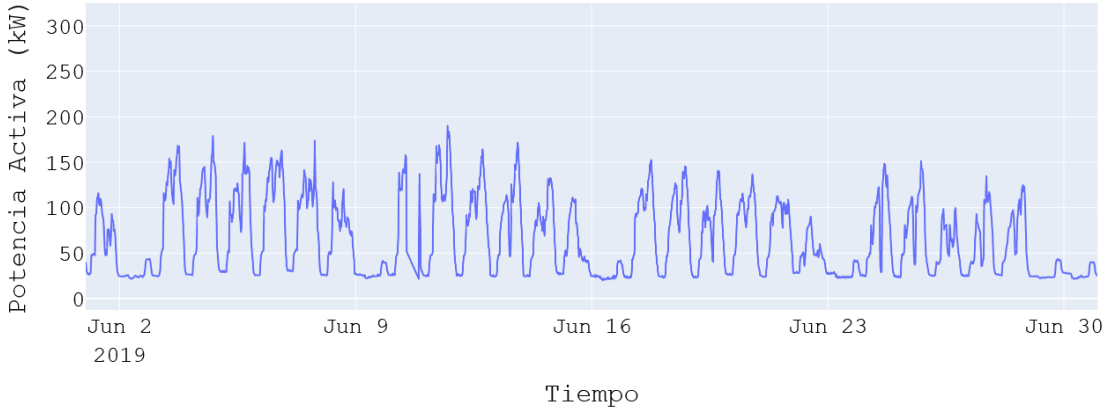


**Figura 27**. Gráfica de 1 mes, perteneciente al mes de junio del 2019.
**Fuente:** Elaboración propia

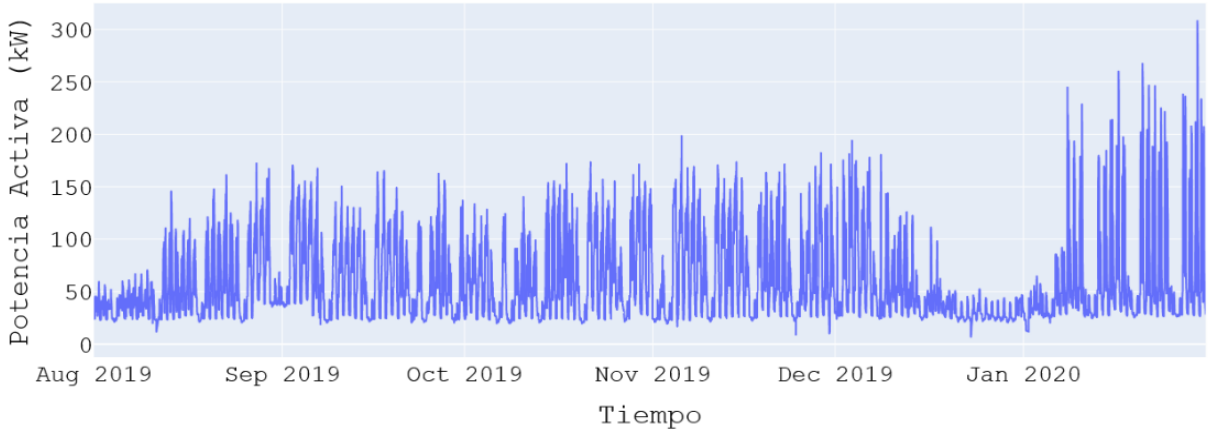


**Figura 28**. Gráfica de 6 meses, desde agosto del 2019 a enero del 2020.
**Fuente:** Elaboración propia

De las gráficas, se puede observar que existen ciertos patrones importantes a identificar. Empezando por la Figura 25, que corresponde a la gráfica de 1 día, se puede observar que el consumo tiene dos picos en horas que coinciden con los horarios de clases; asimismo, en la Figura 26, correspondiente a la gráfica de la semana, se visualiza que el consumo varía de acuerdo con el día de la semana, llegando a tener un comportamiento característico. En la Figura 27, correspondiente a la gráfica del mes, se observa cómo las semanas tienen comportamientos diferentes.

De lo mencionado anteriormente, se puede constatar que el identificar los diferentes horarios en los que se dan las clases y el día de la semana al que pertenecen será de suma importancia para la modelación y obtención de predicciones precisas. Asimismo, se puede observar que en la Figura 28, correspondiente a la gráfica de los 6 meses, hay un periodo en el que el consumo disminuye notablemente, el cual corresponde a la época de vacaciones, que es otra característica importante por identificar en el set de datos.

Por todo lo mencionado y observado anteriormente, se complementará con una serie de análisis para poder determinar todas aquellas características que permitirán obtener una predicción bastante acertada.

En la Figura 29, se puede observar que los lunes, martes, miércoles, jueves y viernes poseen un gráfico acumulado muy parecido, pero aun así estos varían en cuanto a sus potencias acumuladas. Observando los días sábado y domingo, se ve que la concentración de datos se centra en potencias muy pequeñas, lo cual nos brinda una característica importante frente al patrón de consumo en estos días.

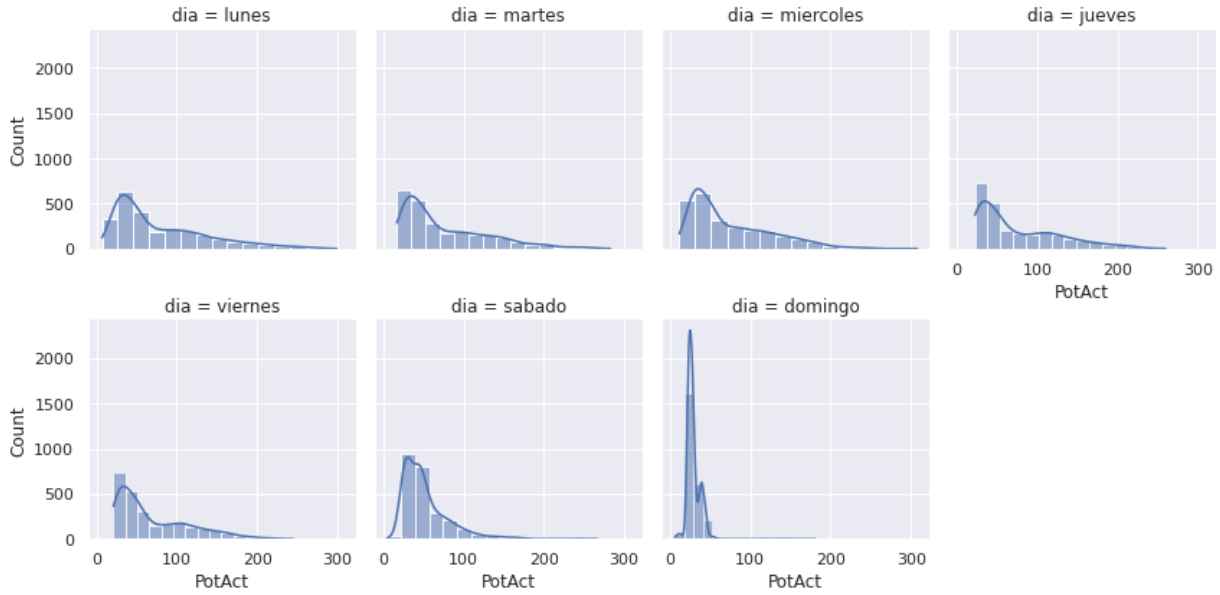


**Figura 29** Gráficos de barras acumulados según día de la semana

En la Figura 30, en el turno “Libre” hay una gran concentración de datos en la parte izquierda del gráfico que corresponde a potencias menores a 100 kW. Esto se debe a que las aulas no están siendo utilizadas por los alumnos y, por ende, los artefactos que consumen una considerable cantidad de energía no están en uso.

Adicionalmente, se observa una concentración de datos bastante pareja en los turnos “mañana” y “tarde”, esto se debe a que las aulas están en uso y, por ende, el consumo es mayor. También se puede observar que en el turno tarde hay mayor concentración de datos entre 100 kW a 200 kW, lo cual se debe al uso de luminarias, aire acondicionado, etc. Por último, en el turno prácticas hay una gran concentración de datos de potencias menores a 100 kW, ya que en este turno solo se utilizan aulas específicas y no todas las disponibles.

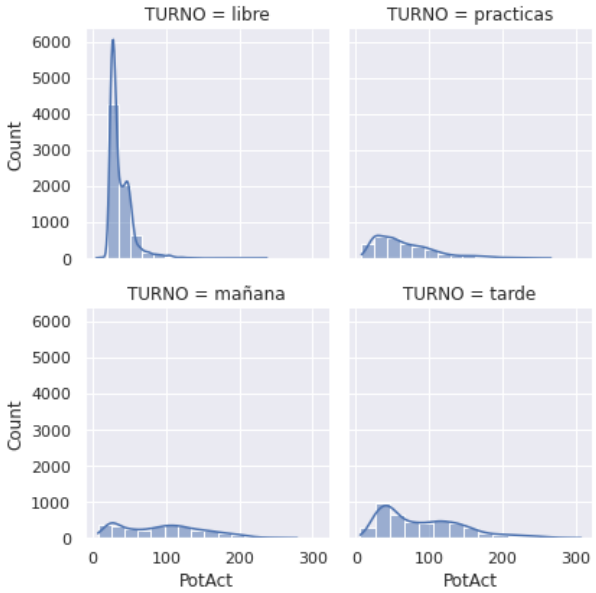


**Figura 30** Gráficos de barras acumulados según turno

En la Figura 31, es notorio cómo el consumo varía según la etapa en la que se encuentre la universidad. De esta manera, la etapa de “clases” es la que posee una mayor densidad de datos a diferencia de las otras etapas, las cuales a pesar de ser etapas en las que el alumnado disminuye notablemente, poseen densidades distintas en lo que respecta a la concentración de datos.

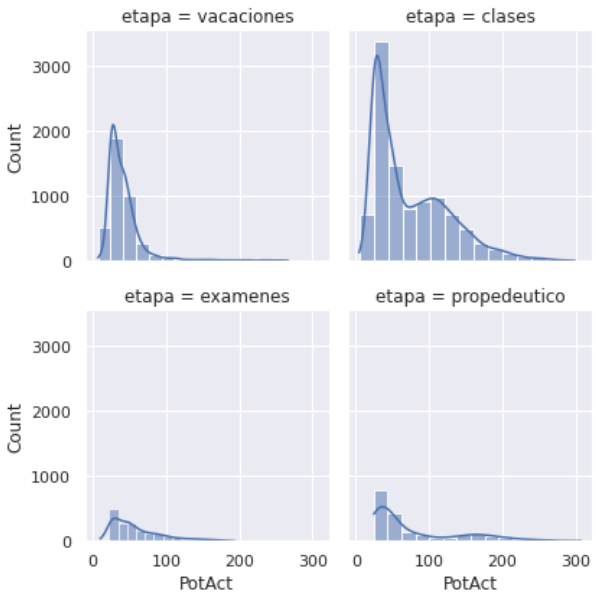


**Figura 31** Gráficos de barras acumulados según etapa

En la Figura 32, se observa que existe una alta densidad de datos en lo que respecta a los días que no son feriados, siendo los días feriados una fracción bastante pequeña del total de datos. Sin embargo, es de gran importancia tener identificados los feriados, ya que estos

representan cambios bruscos en la demanda, que, si no son identificados, pueden generar ruido en las predicciones realizadas y distorsionar los valores obtenidos.

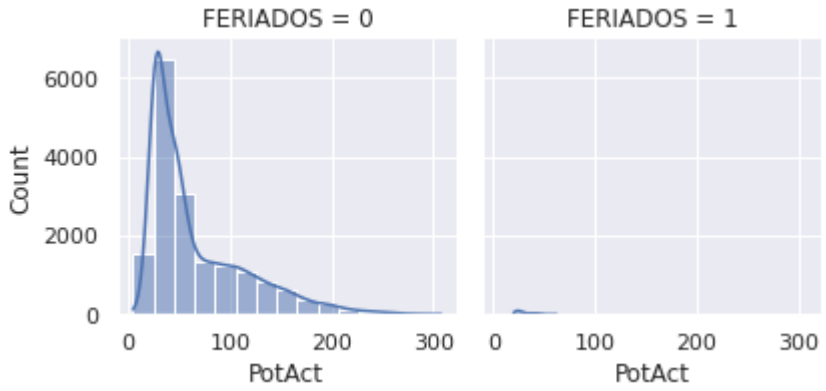


**Figura 32** Gráficos de barras acumulados según día feriado

Por último, se procede a analizar la data del clima para observar si es que hay indicios de patrones en la temperatura y humedad relativa a lo largo del año en las diferentes horas del día.

Se realizó el análisis de la desviación estándar en la temperatura y la humedad relativa para cada una de las horas del día a lo largo del año, se obtuvieron valores bastante reducidos como se puede observar en las figuras 33 y 34. De lo anteriormente mencionado, se puede suponer que estas dos características presentan un comportamiento bastante similar a lo largo del año y que probablemente no tengan una gran influencia en el proceso de predicción del modelo.

```
Desviación Estandar Temperatura a las 0am     3.010676
Desviación Estandar Temperatura a la 1am      2.959819
Desviación Estandar Temperatura a las 2am     2.908820
Desviación Estandar Temperatura a las 3am     2.851639
Desviación Estandar Temperatura a las 4am     2.797870
Desviación Estandar Temperatura a las 5am     2.753390
Desviación Estandar Temperatura a las 6am     2.699301
Desviación Estandar Temperatura a las 7am     2.708015
Desviación Estandar Temperatura a las 8am     2.859172
Desviación Estandar Temperatura a las 9am     3.107457
Desviación Estandar Temperatura a las 10am    3.316464
Desviación Estandar Temperatura a las 11am    3.350757
Desviación Estandar Temperatura a las 12m     3.199902
Desviación Estandar Temperatura a la 1pm      3.019082
Desviación Estandar Temperatura a las 2pm     2.946424
Desviación Estandar Temperatura a las 3pm     2.853737
Desviación Estandar Temperatura a las 4pm     2.799969
Desviación Estandar Temperatura a las 5pm     2.941805
Desviación Estandar Temperatura a las 6pm     3.072167
Desviación Estandar Temperatura a las 7pm     3.100605
Desviación Estandar Temperatura a las 8pm     3.117409
Desviación Estandar Temperatura a las 9pm     3.124878
Desviación Estandar Temperatura a las 10pm    3.101342
Desviación Estandar Temperatura a las 11pm    3.070934
```

**Figura 33** Desviación estándar de la temperatura para cada hora del día [°C]

```
Desviación Estandar Humedad Relativa a las 0am     5.441173
Desviación Estandar Humedad Relativa a la 1am      5.283420
Desviación Estandar Humedad Relativa a las 2am     5.089210
Desviación Estandar Humedad Relativa a las 3am     4.783201
Desviación Estandar Humedad Relativa a las 4am     4.532158
Desviación Estandar Humedad Relativa a las 5am     4.333102
Desviación Estandar Humedad Relativa a las 6am     4.226425
Desviación Estandar Humedad Relativa a las 7am     4.472255
Desviación Estandar Humedad Relativa a las 8am     5.381560
Desviación Estandar Humedad Relativa a las 9am     6.339427
Desviación Estandar Humedad Relativa a las 10am    6.993062
Desviación Estandar Humedad Relativa a las 11am    7.026389
Desviación Estandar Humedad Relativa a las 12m     6.410902
Desviación Estandar Humedad Relativa a la 1pm      5.781346
Desviación Estandar Humedad Relativa a las 2pm     5.455660
Desviación Estandar Humedad Relativa a las 3pm     4.996734
Desviación Estandar Humedad Relativa a las 4pm     4.810590
Desviación Estandar Humedad Relativa a las 5pm     5.316496
Desviación Estandar Humedad Relativa a las 6pm     5.724974
Desviación Estandar Humedad Relativa a las 7pm     5.765119
Desviación Estandar Humedad Relativa a las 8pm     5.680562
Desviación Estandar Humedad Relativa a las 9pm     5.681646
Desviación Estandar Humedad Relativa a las 10pm    5.662847
Desviación Estandar Humedad Relativa a las 11pm    5.623932
```

**Figura 34** Desviación estándar de la humedad relativa para cada hora del día [%]

Todas las características seleccionadas y analizadas en las imágenes anteriores permiten ver que, desde diferentes criterios de análisis, se puede extraer información valiosa para construir un set de datos robusto que permita obtener predicciones precisas.

## 4.5 Set de datos final

El set de datos final está compuesto por todas aquellas características identificadas en el acápite anterior. Para agilizar el cálculo durante el entrenamiento del modelo, se procedió a aplicar "*One Hot Encoding*" a todas las nuevas características añadidas según el análisis realizado anteriormente por los autores.

El "*One Hot Encoding*" es una estrategia que implementa una columna por cada valor distinto que exista en la característica que se está codificando. Por ejemplo, para la columna de día se creó una columna para cada día de la semana y se marcó con un "1" cada vez que el valor cumpla con la característica y "0" cuando no.(One Hot Encoding | Interactive Chaos, n.d.)

A partir del "*One Hot Encoding*" se obtuvo como resultado un total de 16 columnas, entre las cuales están las variables correspondientes al día de la semana, etapa, feriado y turno durante el día.

Con el set de datos listo para entrenar y posteriormente validar, se procede a establecer el horizonte de predicción. Se escogió 1 día, que corresponde a 48 datos, debido a la cantidad de datos que se tiene, que corresponde a 1 año de consumo que va de marzo del 2019 a febrero del 2020.

Con el set de datos obtenido, se procede a realizar su estandarización, para así lograr que el modelo procese la información de manera mucho más rápida. Para la estandarización se utilizó la librería *Sckit-Learn*, la cual es una librería que tiene diversas funciones que son claves en el Machine Learning, así como en la estadística. De *Scikit-Learn*, se hizo uso de la función "*StandardScaler*", la cual consiste en eliminar la media y escalar a la varianza de la unidad (*Sklearn.Preprocessing.StandardScaler — Scikit-Learn 1.0.2 Documentation*, n.d.), obteniendo así un set de datos más prolijo y fácil de procesar.

Un dato importante es que para las 4 primeras columnas se aplicó un *StandardScaler* diferente, como se puede apreciar en el siguiente código. La razón por la cual se aplicó esto fue debido a que cada columna poseía diferentes valores máximos, y en algunos casos los rangos de valores eran muy desiguales entre columnas. Adicionalmente, se aplicó la estandarización a las 4 primeras columnas porque en las demás los valores eran "0" o "1", debido al *"One Hot Encoding"*.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
training_set_scaled = sc.fit_transform(training_set_A[:,0:1])
training_set[:,0:1]=training_set_scaled
sc1 = StandardScaler()
training_set_scaled1 = sc1.fit_transform(training_set_A[:,1:2])
training_set[:,1:2]=training_set_scaled1
sc2 = StandardScaler()
training_set_scaled2 = sc2.fit_transform(training_set_A[:,2:3])
training_set[:,2:3]=training_set_scaled2
sc3 = StandardScaler()
training_set_scaled3 = sc3.fit_transform(training_set_A[:,3:4])
training_set[:,3:4]=training_set_scaled3
training_set_scaled=training_set
```

Con los datos ya procesados y separados en entradas y salidas deseadas, se procedió a realizar una división del set de datos en 2 partes: una para entrenamiento y validación, y otra para poder testear en la interfaz gráfica desarrollada.

Ambos modelos por entrenar estuvieron estructurados para que puedan recibir múltiples variables de entrada y tengan como salida una única variable que en este caso fue la potencia activa. Las variables de entrada y salida fueron obtenidas con el código presentado a continuación:

```
X_train = []
y_train = []
```

```
n_future = 24*2
n_past = 24*2
mov_window= 24*2
for i in range(n_past, len(training_set_scaled) - n_future +1,mov_win
dow):
    X_train.append(training_set_scaled[i - n_past:i, 0:dataset_train.
shape[1]])
for i in range(n_past, len(training_set_scaled) - n_future +1,mov_win
dow):
    y_train.append(training_set_scaled[i:i + n_future, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
y_train=y_train.reshape(y_train.shape[0], n_future,1)
print('X_train shape == {}.'.format(X_train.shape))
print('y_train shape == {}.'.format(y_train.shape))
```

En el código, se pueden observar tres variables importantes que indican lo siguiente:

- *n_future*: es el número de datos a predecir. Para la aplicación actual se escogieron 48 datos debido a que las lecturas de la demanda del suministro eléctrico fueron registradas cada media hora. Por este motivo, para lograr el horizonte de predicción de un día se necesitó de 48 datos.
- *n_past*: es la cantidad de datos del pasado que se tomarán para predecir un día, en este caso se escogió ingresar un día para poder predecir el día siguiente.
- *mov_window*: este es la cantidad de datos que irá saltando para ir armando los grupos de entrenamiento; es decir, se irán formando grupos de 48 datos consecutivos, que corresponden a un día entero de tomas.

Es importante mencionar que el último día en el set de datos no será visto por el modelo, pues se usará para evaluar su desempeño. Se obtiene un criterio para la obtención de dos sets de datos, uno que permitirá entrenar el modelo y otro que permitirá ponerlo a prueba con datos que no ha visto. El set de datos para el entrenamiento está comprendido desde el 1 de marzo del 2019 hasta el 28 de febrero del 2020, mientras que el set de datos de testeo comprende los días 28 y 29 de febrero del 2020.

## Capítulo 5

## Entrenamiento de modelos y análisis de resultados obtenidos

Para el diseño de la arquitectura de los modelos se tuvo en consideración la teoría anteriormente explicada en el Capítulo 2; sin embargo, es importante mencionar que la experimentación cumple un rol importante en este diseño. Es por ello que, para llegar a obtener los valores óptimos que aseguran predicciones precisas, es necesario realizar múltiples experimentos que permitan ir consolidando los hiperparámetros.

Una vez definida la arquitectura de los modelos, se realizaron entrenamientos para cuatro casos distintos, los cuales difieren según el número de características consideradas:

- Caso 1: la variable de entrada tiene como única característica la potencia activa.
- Caso 2: la variable de entrada tiene como características a la potencia activa y tres variables climáticas, las cuales son temperatura ambiente, humedad relativa y radiación solar.
- Caso 3: la variable de entrada tiene como características a la potencia activa y variables extraídas por los autores según patrones identificados, que han sido mencionados en el capítulo 4.
- Caso 4: la variable de entrada contiene todas las características mencionadas en los casos 1, 2 y 3.

Como parte de la comparativa a realizar para los casos mencionados, se tomaron en cuenta las 3 métricas obtenidas durante la fase de entrenamiento y validación, la cuales serán explicadas a continuación:

- RMSE: Es la raíz cuadrada de la media de las diferencias entre los valores de las predicciones y valores observados (Berzal, 2018).
- MAE: Es el valor absoluto de la media de las diferencias entre las predicciones y valores observados (Berzal, 2018).
- MAPE: Es el error medio obtenido en la estimación con respecto al valor observado, expresado de manera porcentual (Berzal, 2018).

Para el entrenamiento de los modelos LSTM y TCN, se hará uso de la nube que en este caso será la de Google Colaboratory, producto de Google Research. Tiene como principal característica permitir al usuario escribir y ejecutar códigos en el lenguaje de Python en Jupyter notebooks haciendo uso del navegador. Es una herramienta de acceso libre y así mismo brinda acceso gratuito a GPUs, las cuales serán utilizadas para el entrenamiento de los dos modelos. Cabe resaltar que no se puede saber con exactitud qué tipos de GPUs han sido utilizados, ya que, para que este siga siendo un recurso gratuito, varían a lo largo del tiempo (Google, n.d.).

## 5.1 Modelo LSTM

### *5.1.1 Arquitectura de modelo*

```
# LSTM

model = Sequential()

model.add(LSTM(units=48, return_sequences=True, activation='relu',inp
ut_shape=(48, dataset_train.shape[1])))

model.add(LSTM(units=64))

model.add(Dropout(0.05))

model.add(Dense(units=48))
```

El extracto de código mostrado permite definir la arquitectura de una red LSTM. A continuación, se explicarán los hiperparámetros utilizados:

- *units,* es la dimensión de los vectores del estado de celda y del estado oculto, también coincide con el número de neuronas presentes en las compuertas de olvido, entrada y salida. En esta ocasión, se optó por colocar un número de unidades igual al horizonte de predicción en las capas de entrada y salida; en las capas intermedias se optó por tener 64 unidades, según la ecuación 12.

$$N_h = \frac{2}{3} * (N_i + N_o) \tag{12}$$

  $N_i$ : es el número de unidades de entrada, siendo su valor 48.
  $N_o$ : es el número de unidades de salida, también de valor 48.
  $N_h$ : es 64 a partir de la ecuación 12, siendo este el número de unidades de las capas ocultas.
- *return_sequences=True,* al colocar este hiperparámetro como verdadero se obtiene la secuencia completa de estados ocultos de todos los pasos de tiempo. Dicha secuencia será la salida de una capa y la entrada de la capa posterior.
- *activation='relu'.* Si bien la función de activación predeterminada en LSTM es la tangente hiperbólica (*tanh*), experimentalmente se obtuvieron mejores resultados colocando la función ReLU (*Rectified Linear Unit*). Es importante mencionar que al modificar la función de activación se sacrificó velocidad de entrenamiento por precisión.

- *Dropout(0.05),* es una técnica en la que se ignoran neuronas aleatoriamente seleccionadas durante el entrenamiento. Estas neuronas excluidas no contribuyen en la activación de neuronas posteriores y en la actualización de pesos. El valor probabilístico se suele escoger experimentalmente, en este caso el valor de 0.05 mostró buenos resultados.

### *5.1.2 Entrenamiento y resultados*

En la Tabla 8 se muestran los resultados obtenidos para los 4 casos. Se procederá a realizar un análisis tomando como puntos de comparación el tiempo de ejecución y tres tipos de cálculo del error: RMSE, MAE y MAPE.

**Tabla 8.** Métricas obtenidas para los casos considerados - LSTM

| Casos | Fase | Tiempo de ejecución | RMSE [kW] | MAE [kW] | MAPE |
|---|---|---|---|---|---|
| **Caso 1** | **Train** | 2 min 3 s | 33.725 | 19.466 | 0.322 |
| | **Test** | 354 ms | 9.516 | 6.996 | 0.149 |
| **Caso 2** | **Train** | 2 min 7 s | 36.254 | 20.277 | 0.324 |
| | **Test** | 270 ms | 6.589 | 4.776 | 0.122 |
| **Caso 3** | **Train** | 6 min 19 s | 20.977 | 10.71 | 0.152 |
| | **Test** | 1.32 s | 5.669 | 4.061 | 0.104 |
| **Caso 4** | **Train** | 4 min 56 s | 19.385 | 9.982 | 0.145 |
| | **Test** | 1.27 s | 6.188 | 5.034 | 0.119 |

**Fuente:** Elaboración propia

En cuanto al tiempo de ejecución, se observa una tendencia a ser más veloz a medida que se entrena con menos características. El caso 1 confirma lo mencionado, siendo el entrenamiento más veloz. Por otro lado, los entrenamientos 3 y 4, que consideraban el mayor número de características, resultaron ser los entrenamientos más lentos.

En cuanto al error en la predicción, se mantienen bastante diferenciados los valores obtenidos durante el entrenamiento respecto a los obtenidos durante el testeo. Esto se explica porque el conjunto de datos utilizado para la validación durante el entrenamiento corresponde al mes de febrero, que se caracteriza por tener un comportamiento bastante diferenciado comparado al resto del año. Por este motivo, el error en la validación durante el entrenamiento es considerablemente mayor, porque es difícil predecir datos de comportamiento atípico, más aún si no se ha entrenado con una buena cantidad de datos similares.

En los casos 3 y 4 se han obtenido los modelos más precisos, tanto en el entrenamiento como en el testeo. El caso 3 se muestra superior en la fase de testeo, mientras que el caso 4 es superior en la fase de entrenamiento. La precisión de los modelos en ambos casos es bastante similar; sin embargo, se escogerá el caso 3 como el más destacado entre los cuatro casos observados, ya que considera solamente la potencia activa y variables creadas por el usuario, manteniendo una precisión tan buena como la del caso 4. Es decir, el modelo del caso

3 resulta más práctico y, por lo tanto, puede ser empleado de una manera más eficiente en la interfaz gráfica.

La precisión del modelo es cercana a la obtenida en investigaciones recientes en predicción de series de tiempo. Por ejemplo, en el artículo *"Wind Power Forecasting with Deep Learning Networks: Time-Series Forecasting"* (W. H. Lin et al., 2021), se obtuvo un MAPE de 6.46% para el modelo LSTM, que no difiere considerablemente del MAPE de 10.4% obtenido en el caso 3.

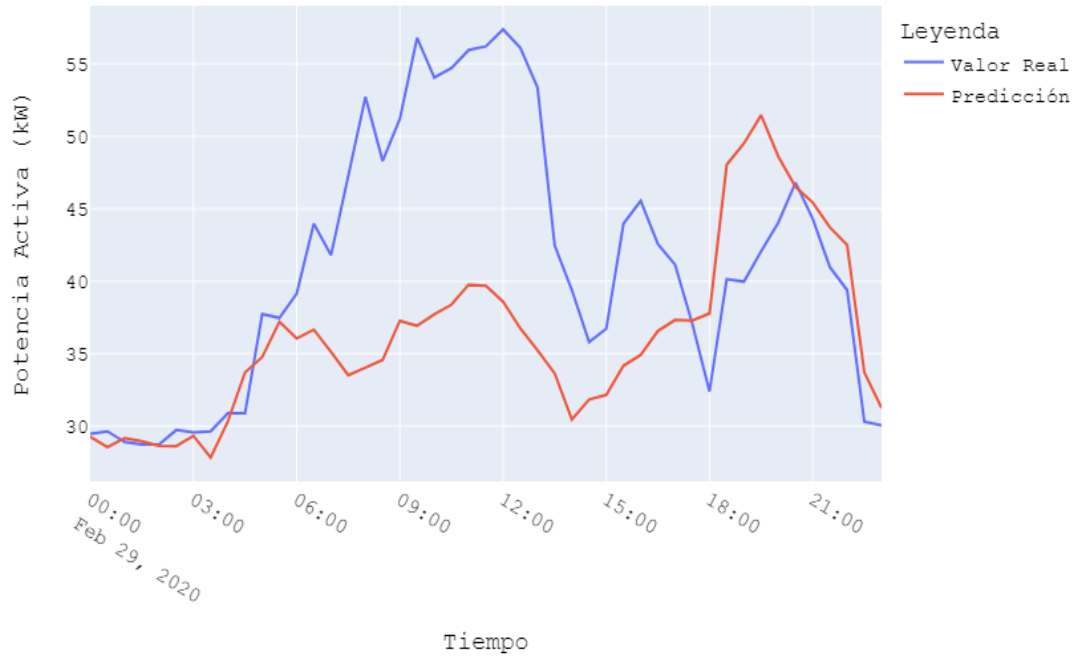


**Figura 35.** Curva de demanda eléctrica: Edificio E y Derecho - 29 de febrero – Caso 1

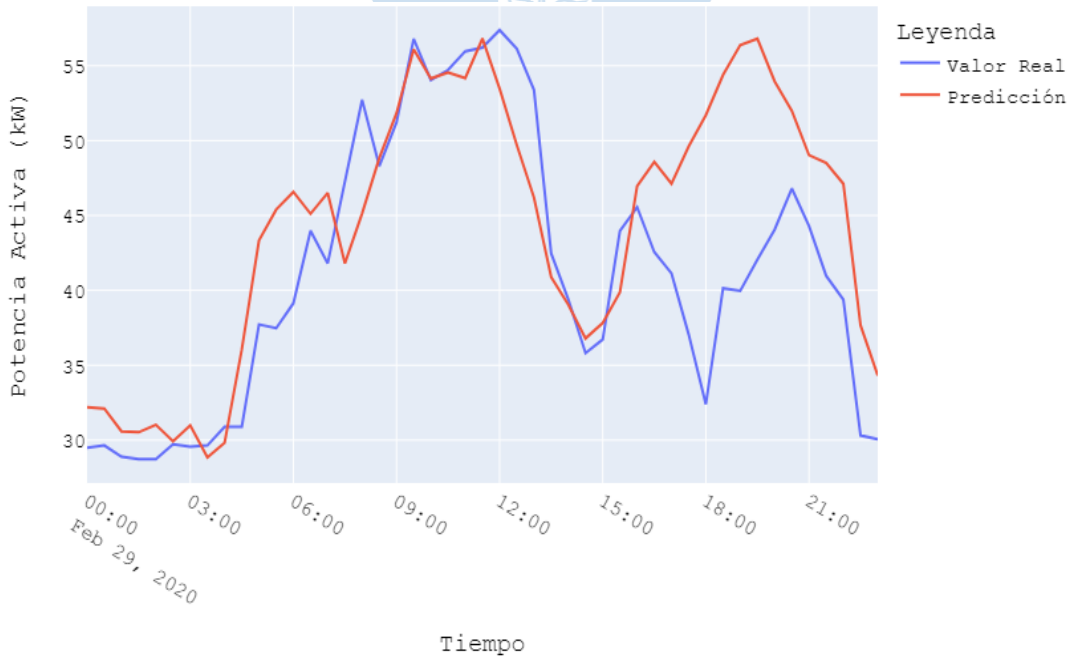


**Figura 36**. Curva de demanda eléctrica: Edificio E y Derecho - 29 de febrero – Caso 2


**Figura 37.** Curva de demanda eléctrica: Edificio E y Derecho - 29 de febrero – Caso 3

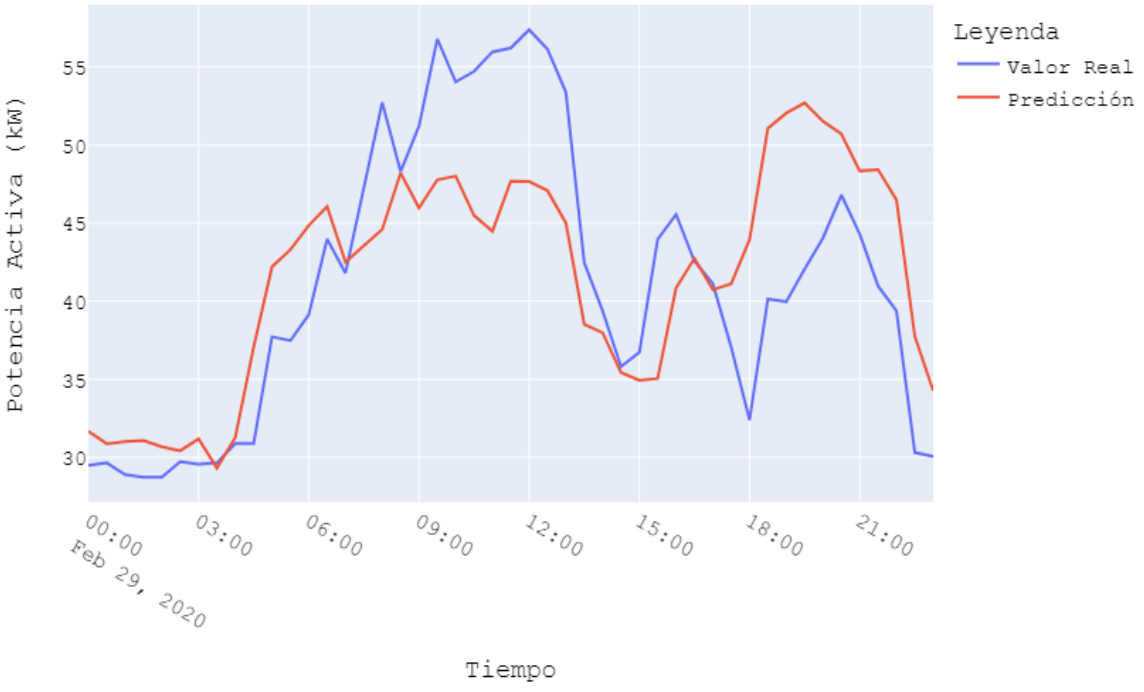


**Figura 38.** Curva de demanda eléctrica: Edificio E y Derecho - 29 de febrero – Caso 4

Las Figuras 35, 36, 37 y 38 corroboran los resultados mostrados en la Tabla 8. Se observa que la Figura 37, que representa el caso 3, muestra la predicción que más se ajusta al valor real, con un MAPE de 10.4%. El resultado menos satisfactorio se obtuvo en el caso 1, con un MAPE de 14.9%.

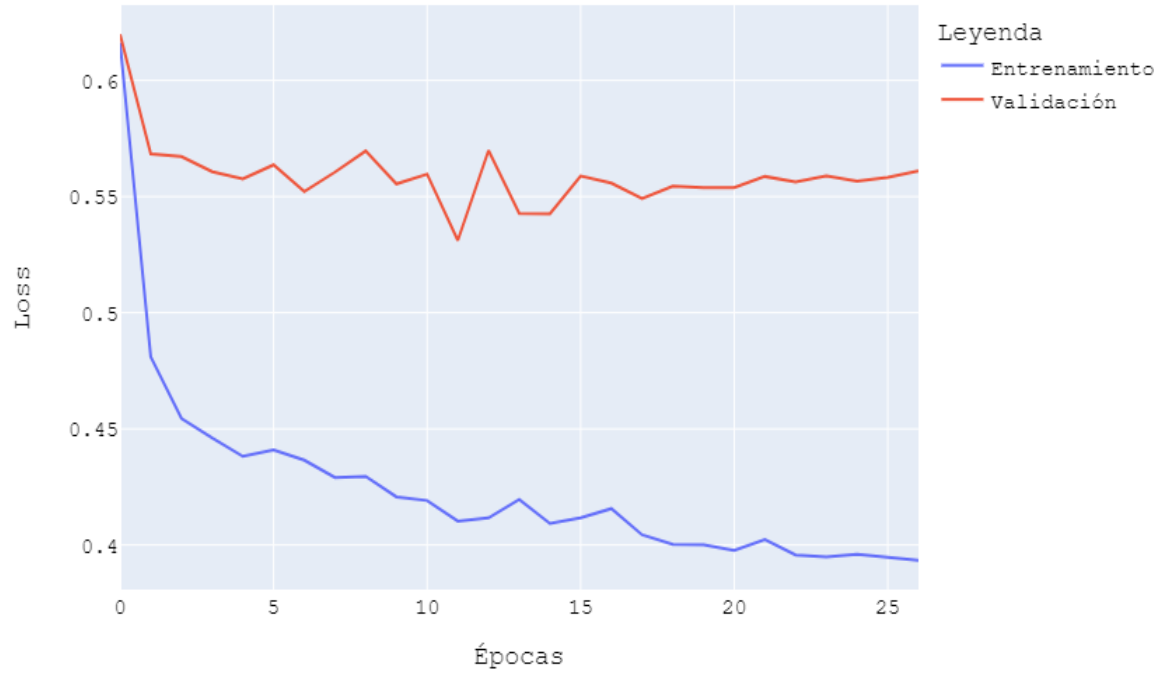


**Figura 39.** Evolución del error durante el entrenamiento - Caso 1

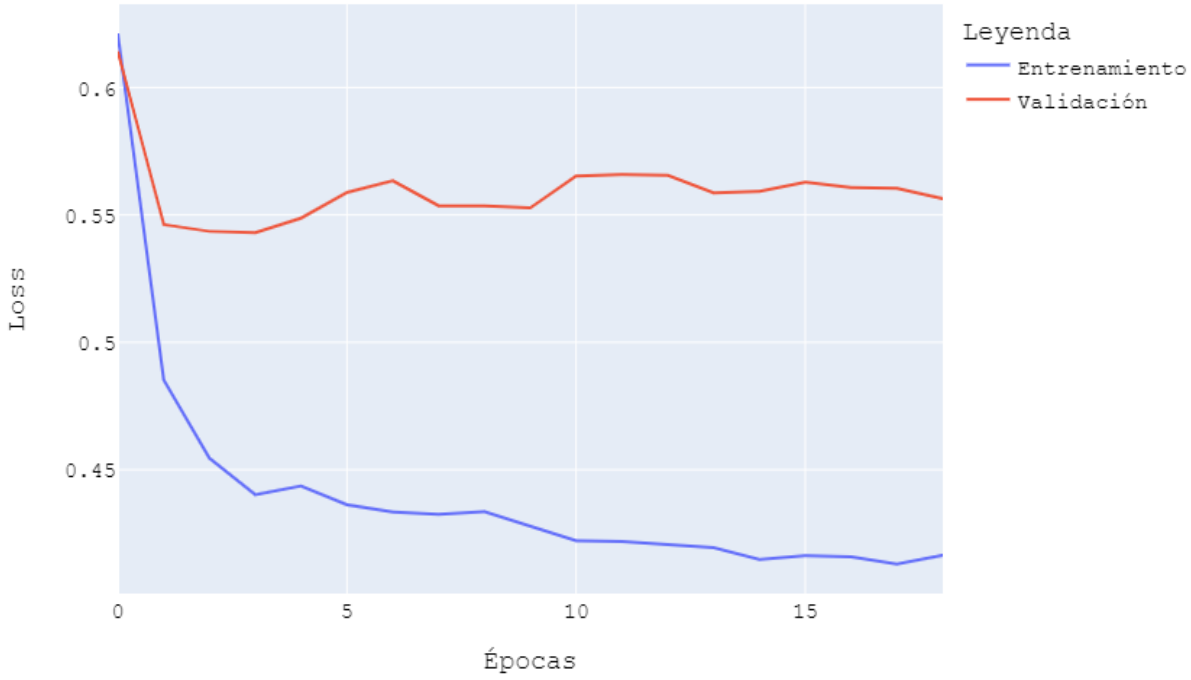


**Figura 40.** Evolución del error durante el entrenamiento - Caso 2.

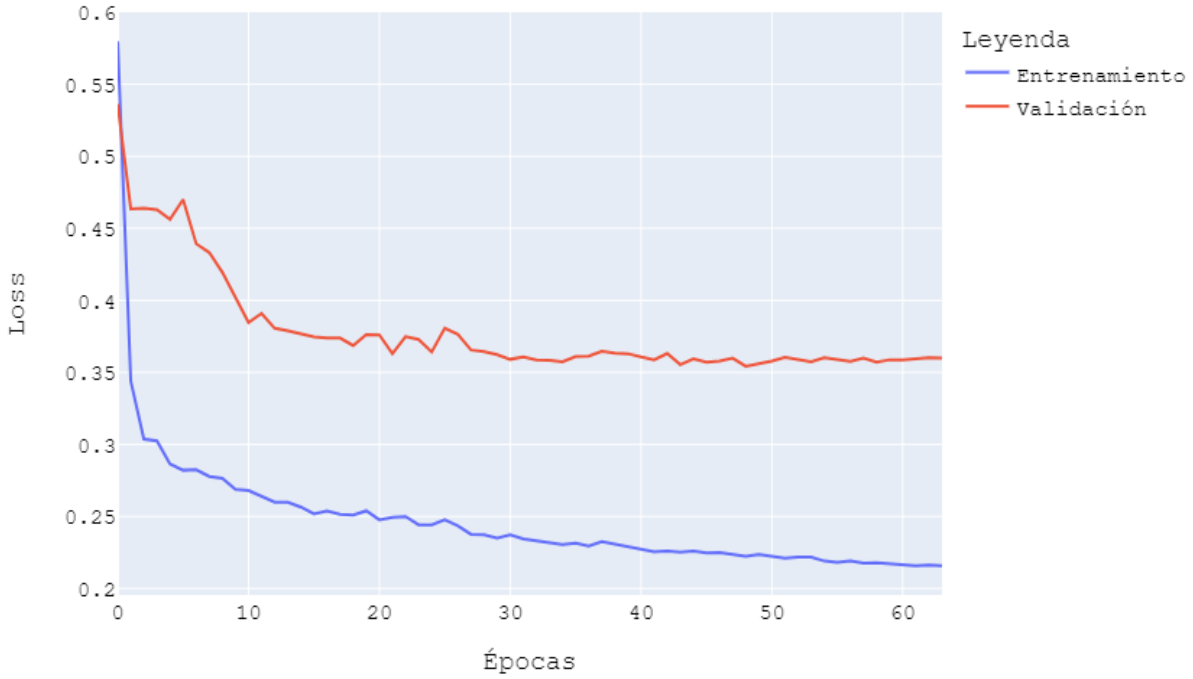


**Figura 41.** Evolución del error durante el entrenamiento - Caso 3.

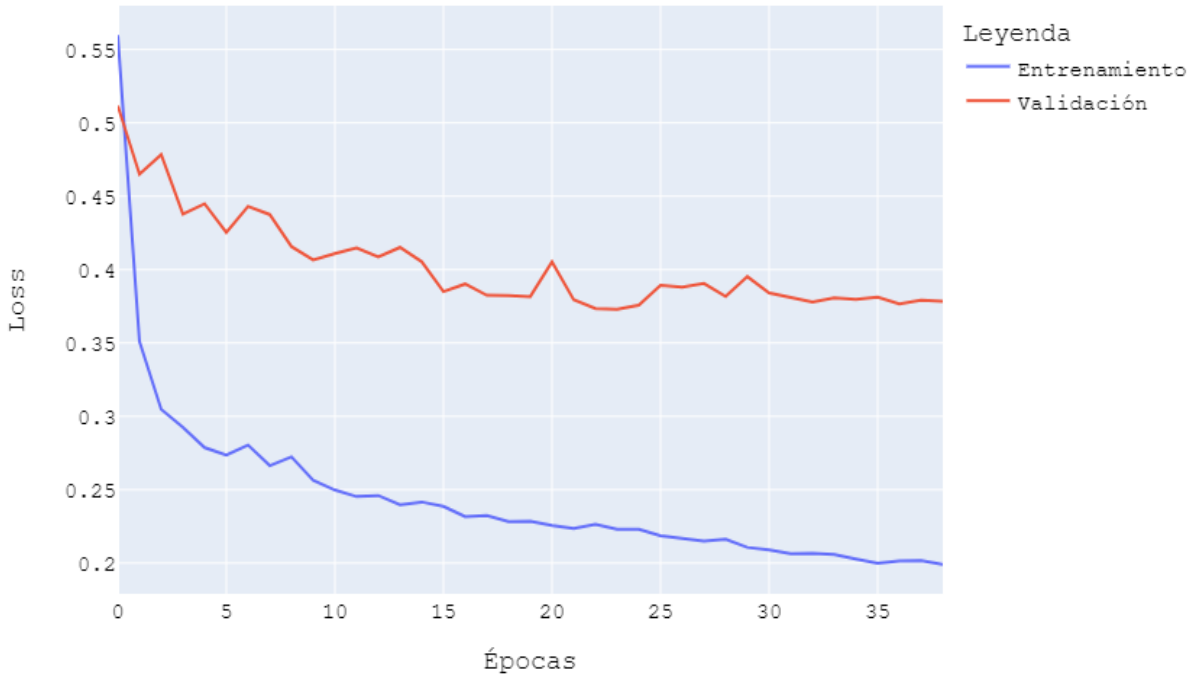


**Figura 42.** Evolución del error durante el entrenamiento - Caso 4

Las Figuras 39, 40, 41 y 42 muestran la evolución del error durante el entrenamiento en los cuatro casos planteados. Se observa una tendencia decreciente en las gráficas del error tanto en entrenamiento como en validación, lo cual demuestra que no hay presencia de "*overfitting*" o sobreajuste, ya que no existen picos pronunciados que indiquen que el modelo tiene dificultad para aprender. Como estrategia principal para evitar el sobreajuste se utilizó la técnica de "*EarlyStopping"*, de la API de Python Keras[3], que permite terminar con el entrenamiento cuando se detecta que el error en validación no ha mejorado en las últimas épocas.

Se observa que en el caso 3 el error en validación llegó a su valor más bajo, comparado con los otros casos planteados. Esta es otra razón para elegir el modelo obtenido en el caso 3 como el más destacado en LSTM, que posteriormente será comparado con el modelo TCN más destacado.

## 5.2 Modelo TCN

### *5.2.1 Arquitectura de modelo*

```
# TCN
inp = Input(shape=X_train.shape[-2:])
x = TCN(nb_filters = 36, kernel_size = 3, nb_stacks = 4, dilations = [1,2,4,8,16], dropout_rate = 0.04, activation = 'relu', padding = 'causal', use_batch_norm = False, use_layer_norm = False, use_weight_norm = True)(inp)
x = Dense(forecast_horizon)(x)
model = keras.Model(inputs=inp, outputs=x)
```

El extracto de código mostrado permite definir la arquitectura de una red TCN. A continuación, se explicarán los hiperparámetros utilizados:

- *Nb_filters=36*, este valor corresponde al número de filtros a usar en las capas convolucionales, las cuales son similares a las unidades que tiene la red LSTM. El valor escogido fue en base a la cantidad de filtros necesarios para poder cubrir toda la data de entrada y lograr realizar la operación de la convolución en toda la data de entrada.
- *Kernel_size=3*, es el tamaño del kernel a utilizar en cada convolución es decir irá haciendo la convolución con datos tomados de 3 en tres. Cabe destacar que este valor fue con el que mejores resultados se obtuvo seleccionándose, así como el óptimo.
- *Nb_Stacks=4*, esta variable corresponde a la cantidad de bloques residuales a utilizar, la cual fue sacada a partir de la ecuación 10, en la cual toma como datos de entrada el tamaño del campo receptivo que es 48, la dilatación base y el

[3] Keras es una API de aprendizaje profundo escrita en Python, que se ejecuta sobra la plataforma de aprendizaje automático TensorFlow. (*About Keras*, n.d.)

tamaño del kernel, obteniendo así el mínimo número de bloques residuales en este caso se fue probando desde el valor mínimo y con este se obtuvieron mejores resultados a comparación de los valores mayores.

- Dilations= [1,2,4,8,16], son las dilataciones a utilizar a medida que avanza por las capas, así mismo representan la profundidad de la red, en este caso al tener 5 dilataciones tendrá 5 capas la red TCN. Adicionalmente se probaron con mayor y menor cantidad de dilataciones sin tener éxito alguno y en algunos casos llegando a elevar el costo computacional por la gran cantidad de capas presentes.
- Padding= 'causal', se utilizará este tipo de *padding*, ya que esta arquitectura como se mencionó en el capítulo 2 se caracterizaba por tener convoluciones en 1 dimensión y estas eran causales.
- Activation= 'relu', para esta aplicación se utilizará en todas las capas porque se quieren valores ente 0 y 1, a excepción de la capa de salida ya que en la salida no existirá problema alguno si es que existe algún valor negativo, cabe resaltar que en el caso que exista un valor negativo durante nuestro entrenamiento esto nos podrá permitir distinguir de si el entrenamiento está correctamente realizado en el caso de no tener data negativa. Otra razón también es que, muchas veces cuando se normaliza la data, dentro del proceso de normalización se usan datos que oscilan entre los números negativos y positivos es por ello que en la capa de salida no se incluirá la función de activación ReLU.
- Dropout=0.04, este valor corresponde a la fracción de datos que se abandona de manera aleatoria durante el entrenamiento, este valor siempre tiene que estar comprendido entre 0 y 1, es por ello que se realizaron diferentes pruebas con diferentes valores hasta llegar a la conclusión que al aumentarlo el modelo tendía a aumentar el error obteniendo un *dropout* de 0.04 como valor óptimo.
- Use_weight_norm=True, con este parámetro utilizado se logra la normalización de pesos en las capas residuales para así disminuir el problema del gradiente explosivo como se mencionó en el capítulo donde se explica la red TCN.
- Por último, las variables *use_batch_norm* y *use_layer_norm* en estado "False", se debe a que no pueden estar en estado "True" si ya hay otra activada, es por ello que se les asigna "False".

### *5.2.2 Entrenamiento y resultados*

En la Tabla 9, se muestran los valores obtenidos para los cuatro casos, se procederá a realizar un análisis de los resultados, tomando como puntos de comparación el Tiempo de ejecución, RMSE, MAE, MAPE y las diferentes situaciones planteadas.

**Tabla 9.** Métricas obtenidas para los casos considerados - TCN

| Casos | Fase | Tiempo de ejecución | RMSE [kW] | MAE [kW] | MAPE |
|---|---|---|---|---|---|
| **Caso 1** | **Train** | 3min 19s | 29.161 | 16.202 | 0.271 |
| | **Test** | 214 ms | 9.342 | 6.657 | 0.1653 |
| **Caso 2** | **Train** | 4min 19s | 25.599 | 13.619 | 0.21 |
| | **Test** | 333 ms | 15.289 | 12.204 | 0.2966 |
| **Caso 3** | **Train** | 7min 20s | 19.515 | 9.959 | 0.142 |
| | **Test** | 1.65 s | 4.646 | 3.568 | 0.0835 |
| **Caso 4** | **Train** | 4min 17s | 18.501 | 9.582 | 0.143 |
| | **Test** | 1.16 ms | 8.439 | 6.678 | 0.1511 |

**Fuente:** Elaboración propia

Empezando por el tiempo de ejecución, se observa que el modelo del caso 3 logra un tiempo de entrenamiento mayor al de los otros casos planteados, lo cual se debe al número de épocas realizadas durante el entrenamiento. Es importante recordar que el entrenamiento no se verá interrumpido por la técnica *"EarlyStopping"* mientras el error en validación siga mejorando; en el caso 3, el entrenamiento demoró más que en los otros casos porque se observó un progreso constante en cuanto a la minimización del error en validación.

Analizando el error en la predicción, se presenta un acontecimiento que se asemeja a lo explicado en el apartado 5.1.2., respecto a que el conjunto de datos que se está utilizando para validar y testear posee un comportamiento distinto al resto del año. Por ello, en la validación del entrenamiento se ha obtenido errores altos, por la dificultad para predecir con exactitud datos de comportamiento atípico.

En los casos 3 y 4 se han obtenido los modelos con menor error, tanto en la fase de entrenamiento como la de testeo. El caso 4 se muestra superior en la fase de entrenamiento, mientras que el caso 3 se muestra superior en la fase de testeo. Los resultados durante la fase de entrenamiento se caracterizan por tener una precisión bastante similar en ambos casos; sin embargo, en la fase del testeo se puede observar que los valores obtenidos en el caso 3 son casi la mitad de los obtenidos en el caso 4.

Resumiendo, el caso 3 resulta ser el más preciso entre los cuatro casos analizados. Además, también es el caso más práctico porque solo necesita de datos de potencia activa y variables creadas por el usuario a partir de un código, llegando así a ser más eficiente para su empleo en la interfaz gráfica.

La precisión del modelo es cercana a la obtenida en investigaciones recientes en predicción de series de tiempo. Por ejemplo, en el artículo *"Wind Power Forecasting with Deep Learning Networks: Time-Series Forecasting"* (W. H. Lin et al., 2021), se obtuvo un MAPE de 8.03% para el modelo TCN, que no difiere considerablemente del MAPE de 8.35% obtenido en el caso 3.

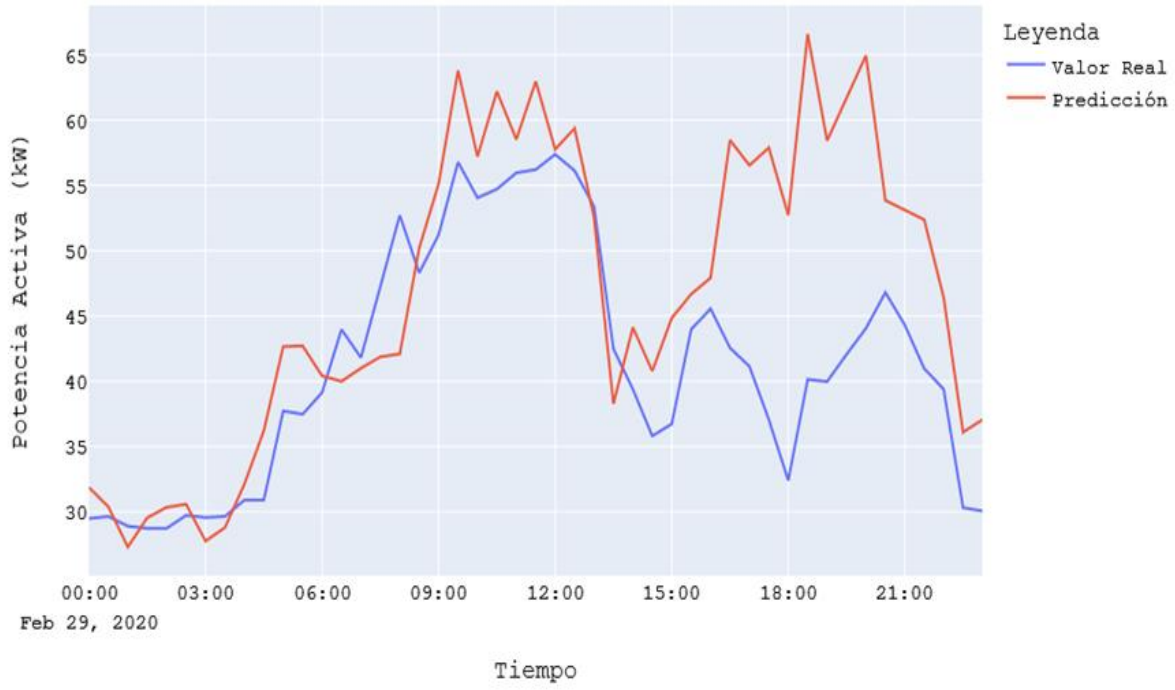


**Figura 43**. Curva de demanda eléctrica: Edificio E y Derecho - 29 de febrero – Caso 1

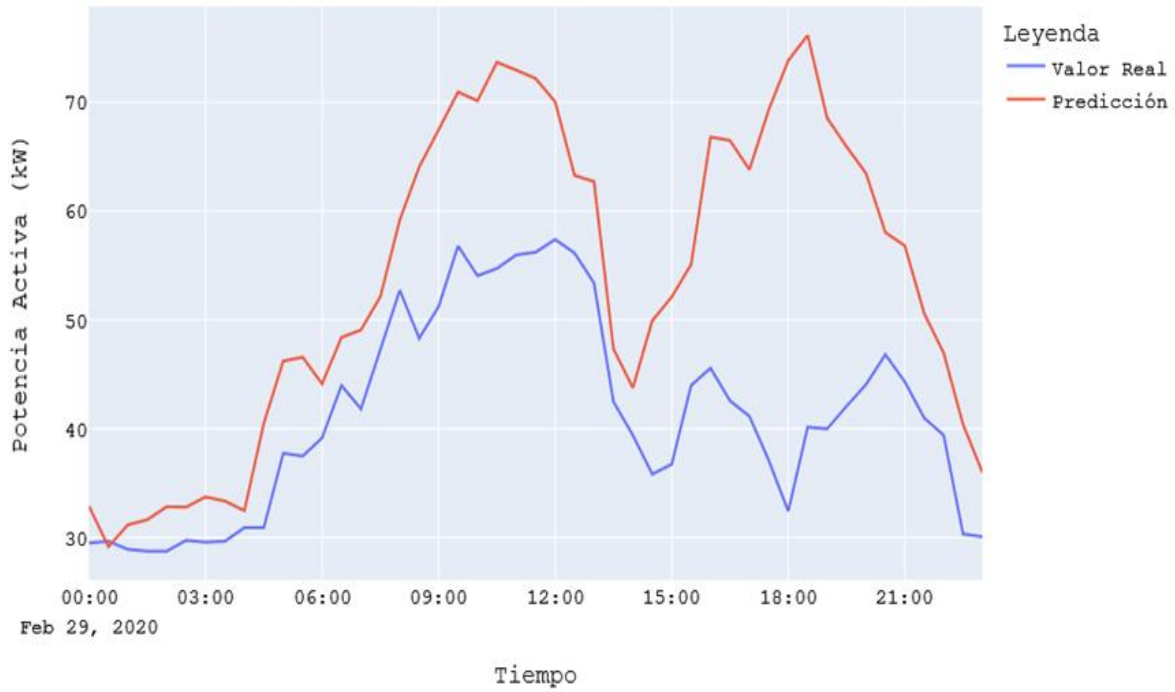


**Figura 44.** Curva de demanda eléctrica: Edificio E y Derecho - 29 de febrero – Caso 2

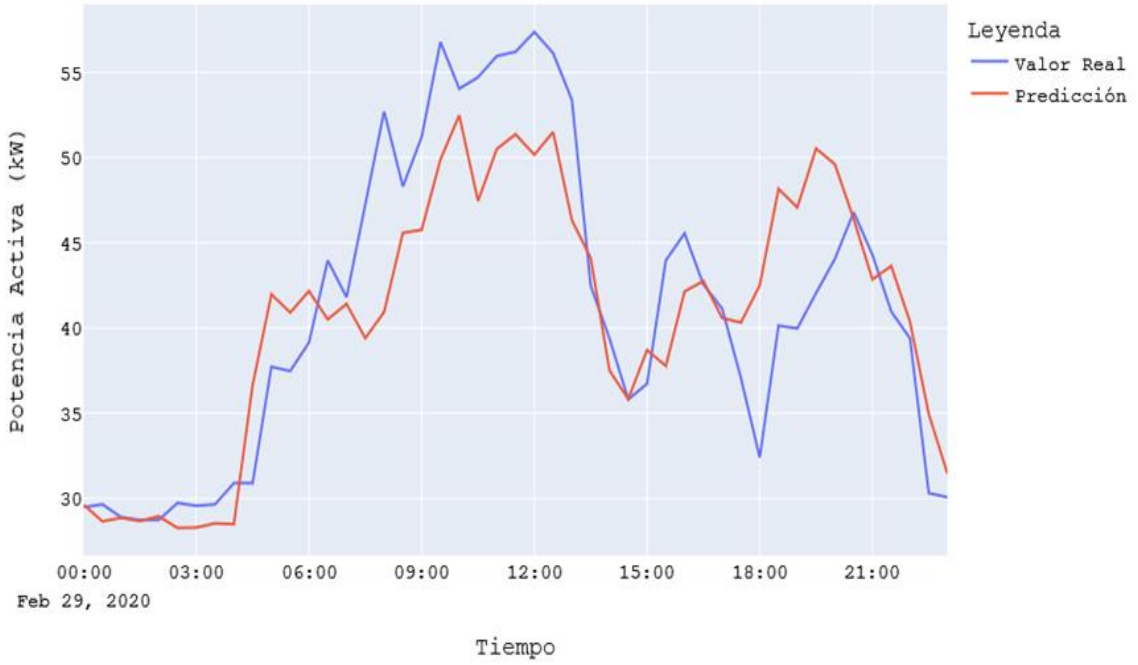


**Figura 45**. Curva de demanda eléctrica: Edificio E y Derecho - 29 de febrero – Caso 3

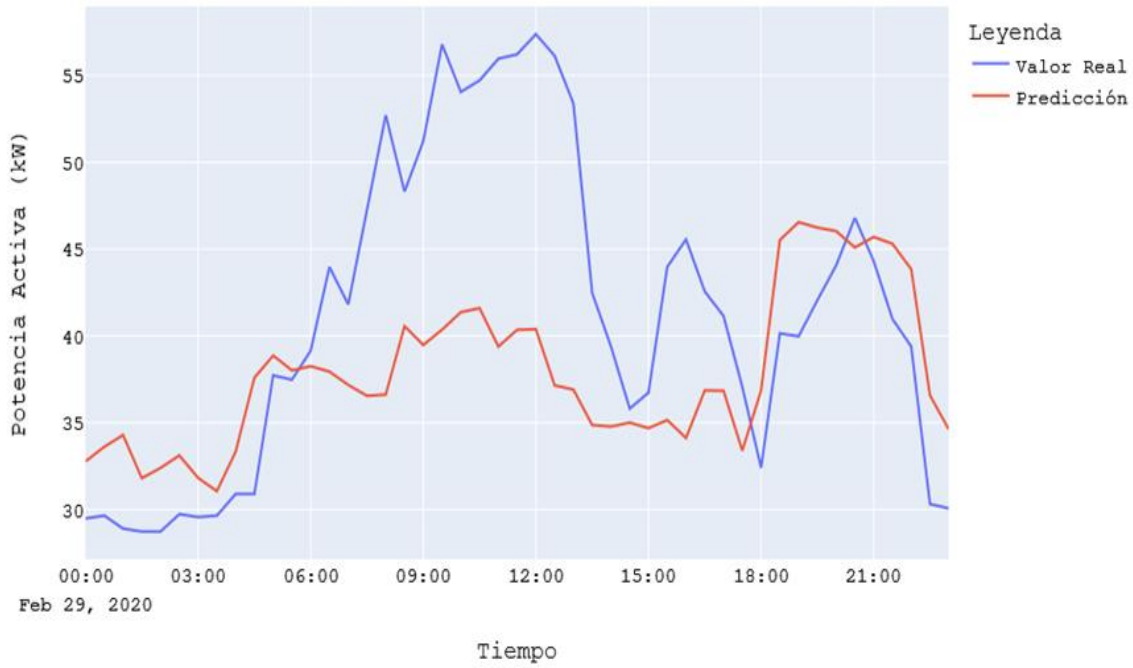


**Figura 46**. Curva de demanda eléctrica: Edificio E y Derecho - 29 de febrero – Caso 4

Las Figuras 43, 44, 45 y 46 corroboran los resultados mostrados en la Tabla 9. Se observa que la Figura 45, que representa el caso 3, muestra la predicción que más se ajusta al valor real, con un MAPE de 8.35%. El resultado menos satisfactorio se obtuvo en el caso 2, con un MAPE de 29.66%.

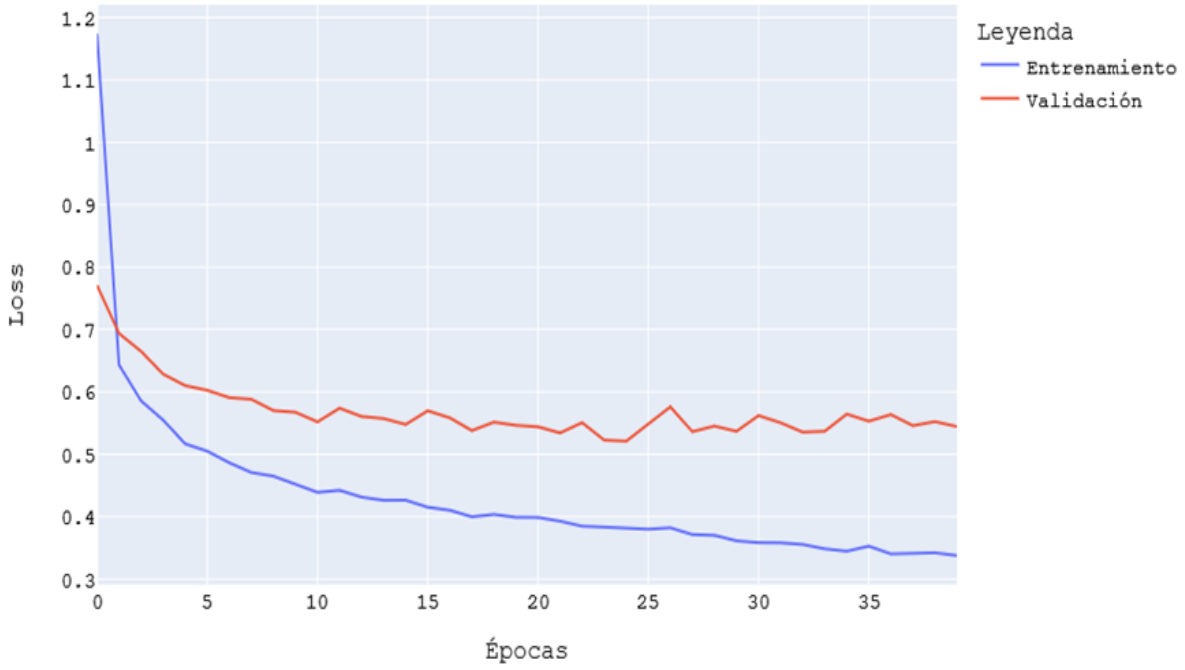


**Figura 47.** Evolución del error durante el entrenamiento - Caso 1

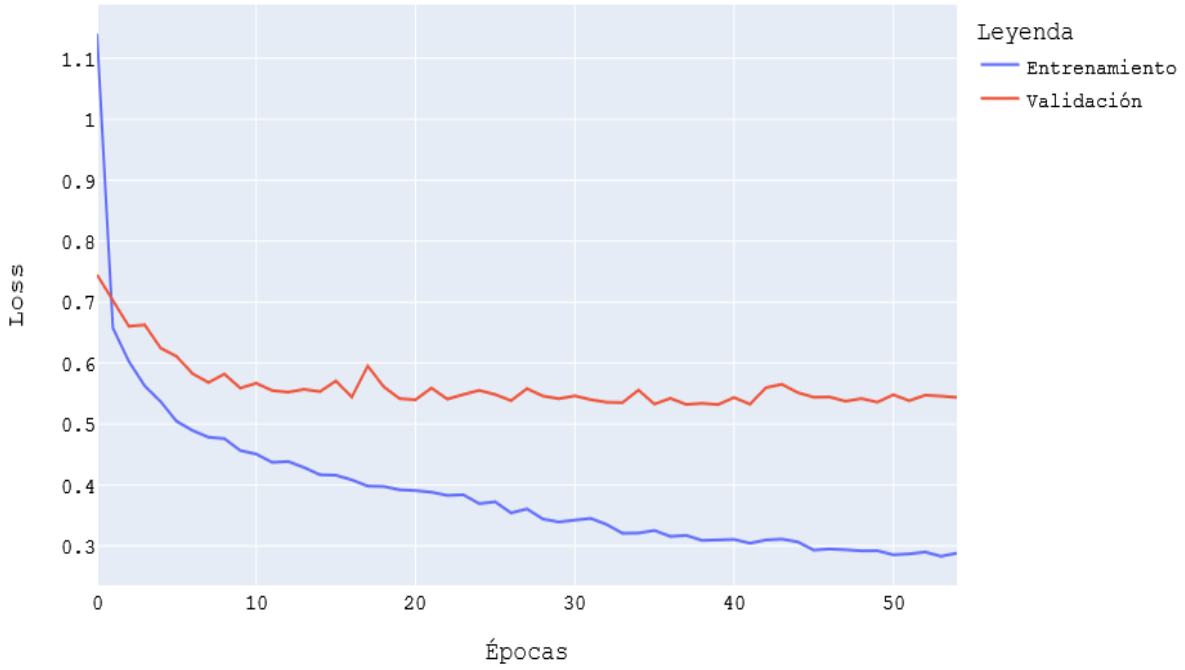


**Figura 48.** Evolución del error durante el entrenamiento - Caso 2

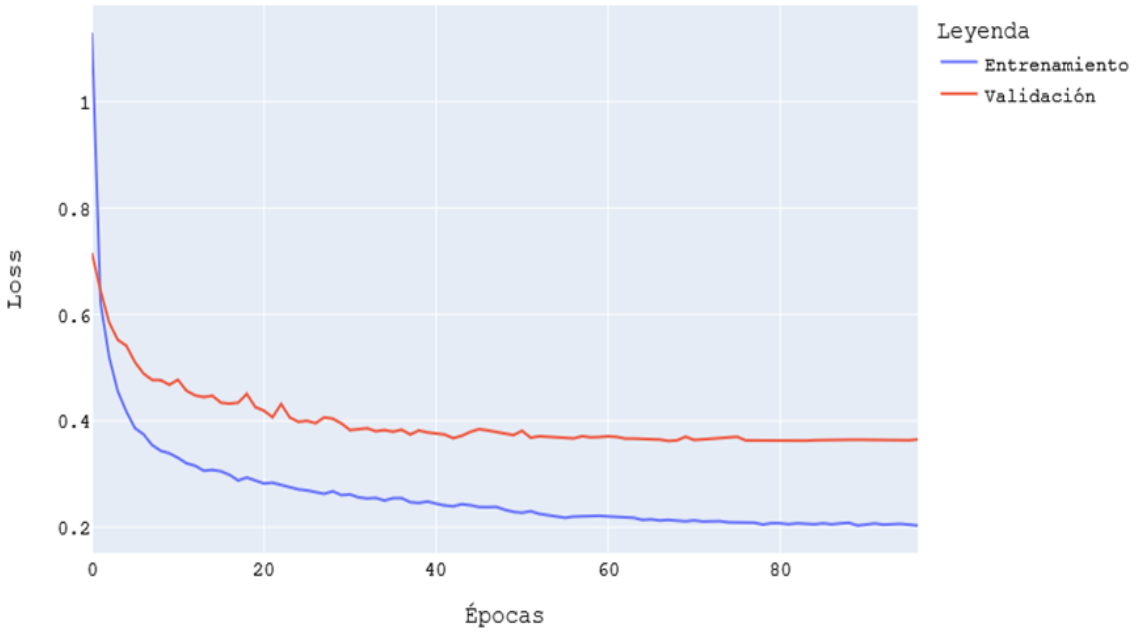


**Figura 49**. Evolución del error durante el entrenamiento - Caso 3

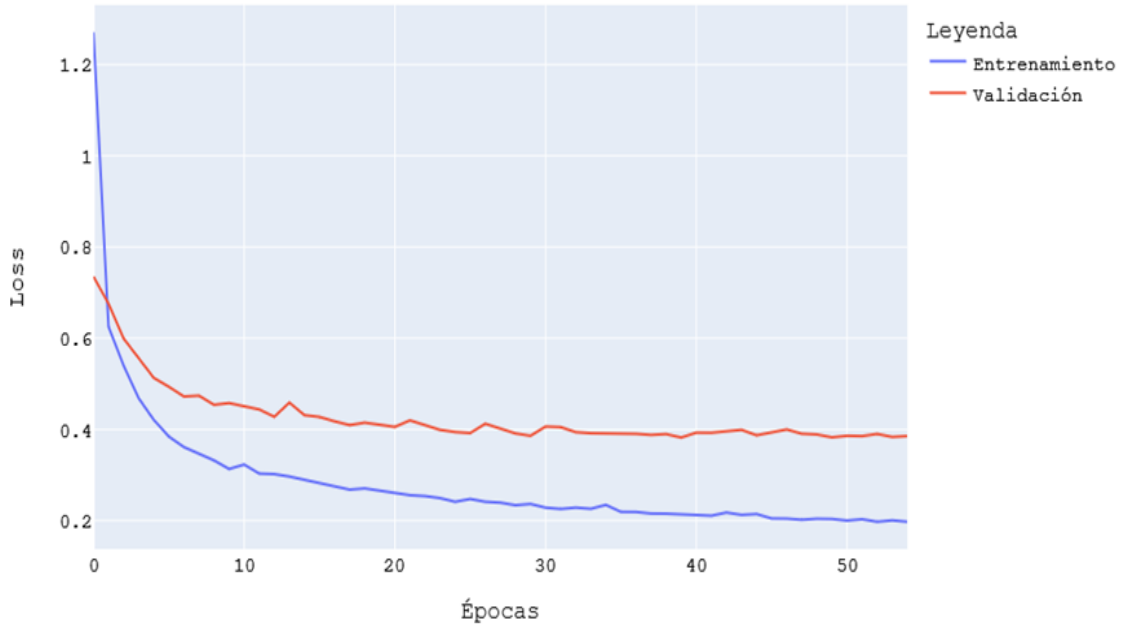


**Figura 50**. Evolución del error durante el entrenamiento - Caso 4

Las Figuras 47, 48, 49 y 50 muestran cómo el error ha ido progresando durante el entrenamiento en cada uno de los cuatro casos planteados. En todas las gráficas se observa una tendencia decreciente tanto en la fase de entrenamiento como en la de validación, lo cual demuestra que no hay presencia de un sobreajuste y que tienden a converger. Para lograr evitar el sobreajuste se utilizó la técnica de “EarlyStopping”, mencionada en el apartado 5.1.2.

En la Figura 49, correspondiente al caso 3, se observa que el error en la validación llegó a su valor más bajo, comparado con los otros casos. Aquí encontramos otra razón para escoger al caso 3 como el más destacado y preciso de la arquitectura TCN.

## 5.3 Comparación entre modelos

**Tabla 10.** Comparación entre las mejores métricas de los modelos LSTM y TCN

| Modelo | Caso | Fase | Tiempo de ejecución | RMSE [kW] | MAE [kW] | MAPE |
|---|---|---|---|---|---|---|
| **LSTM** | **Caso 3** | **Train** | 6 min 19 s | 20.977 | 10.71 | 0.152 |
| | | **Test** | 1.32 s | 5.669 | 4.061 | 0.104 |
| **TCN** | **Caso 3** | **Train** | 7min 20s | 19.515 | 9.959 | 0.142 |
| | | **Test** | 1.65 s | 4.646 | 3.568 | 0.0835 |

**Fuente:** Elaboración propia

Según la Tabla 10, el modelo TCN logró un tiempo de entrenamiento total de 7 minutos con 20 segundos para un total de 97 épocas, mientras que el modelo LSTM logró un tiempo total de 6 minutos con 19 segundos para un total de 64 épocas. Sin embargo, el modelo TCN resultó ser el más veloz si se considera el tiempo medio de entrenamiento por época, ya que obtuvo un tiempo medio de 4.54 segundos por época mientras que el LSTM tuvo un tiempo medio de 5.92 segundos por época.

En términos del MAPE, el modelo LSTM logró obtener durante la fase de testeo un MAPE de 10.4%, mientras que el modelo TCN logró un MAPE de 8.35%, situándolo como el más preciso.

Observando el RMSE en ambos modelos, se puede observar que tienen valores bastante bajos, lo cual nos demuestra que ambos modelos han podido aprender ciertos comportamientos atípicos como lo son los días feriados o los periodos de vacaciones, ya que el RMSE se caracteriza por ser una métrica bastante sensible ante errores grandes, permitiendo lograr modelos bastante precisos.

Por otra parte, la métrica MAE, que es el promedio de los errores absolutos, obtuvo resultados bastante buenos para ambos casos, y esto se debe a que esta métrica tiende a ignorar a todos aquellos valores atípicos presentes en las series de tiempo, obteniendo valores de 3.568 kW para TCN y 4.061 kW para LSTM.

Concluyendo, tanto el MAE como el RMSE han sido métricas claves para evaluar el aprendizaje del modelo. Mientras la métrica MAE es de interpretación sencilla e ignora los atípicos, la métrica RMSE, por otro lado, permitió castigar a aquellos errores grandes; a medida que se iba entrenando, estos errores se han ido reduciendo, logrando así valores bastante satisfactorios. Finalizando, según lo analizado y observado, el modelo más idóneo para implementarlo en la interfaz gráfica sería el modelo TCN del caso 3.

UNIVERSITAS STUDIORUM PIURENSIS

# Capítulo 6

# Interfaz gráfica para la predicción de la demanda eléctrica

## 6.1 Requisitos para el correcto uso de la interfaz gráfica

Para simplificar el uso de la interfaz gráfica, el código fue convertido a un ejecutable, que fue generado a partir de una librería de Python llamada *PyInstaller*. Esta librería permite que todo archivo con extensión ".py" pueda volverse un archivo ejecutable y así no dependa de un intérprete del lenguaje Python instalado en el sistema y de todas sus dependencias para poder ejecutarlo.

Para la correcta ejecución de la interfaz gráfica, se tomaron en cuenta ciertos requisitos que debe cumplir el usuario, se detallarán a continuación:

- Archivo "modelofinal.h5", el cual es el modelo producto del entrenamiento, que tiene toda la información necesaria para poder realizar las predicciones respectivas. Cabe resaltar que este modelo empleado fue el más destacado de los experimentos realizados en el capítulo 5, corresponde al modelo TCN del caso 3.

- Data de entrada, este documento debe ser un archivo con extensión ".csv", el cual debe estar separado por comas y debe estar compuesto por una sola hoja y dos columnas: fecha y potencia activa.

- En el archivo, la columna correspondiente a fecha deberá tener como primera celda el nombre "date" y la columna de potencia activa deberá llevar como primera celda el nombre "PotAct", ya que bajo estos nombres es que el código irá leyendo y realizando las operaciones necesarias.

- Para la correcta ejecución de las operaciones, la fecha se debe encontrar en el siguiente formato "YYYY/MM/DD HH:MM:SS", y la potencia activa deberá encontrarse en formato de número.

- El archivo a utilizar deberá estar compuesto por 48 datos, ya que el modelo ha sido entrenado bajo esos parámetros. La gráfica para la obtención de resultados

toma como referencia que los datos han sido tomados cada media hora, logrando así representar la demanda eléctrica de un día entero con 48 datos.

## 6.2 Desarrollo de interfaz gráfica

El lenguaje de programación utilizado para el desarrollo del trabajo de investigación, Python, tiene dentro de sus componentes paquetes que permiten desarrollar interfaces gráficas con diferentes aplicaciones.

En este caso, la interfaz gráfica a desarrollar será para la implementación del modelo de predicción, y así lograr obtener un ejecutable que permita a cualquier persona poder hacer uso de este.

El paquete utilizado para el desarrollo de la interfaz gráfica es *"Tkinter"*, el cual destaca porque es rápido y generalmente viene incluido con el lenguaje de programación de Python (*Events and Binds in Tkinter | Tkinter | Python-Course.Eu*, n.d.).

A continuación, lo mencionado anteriormente se podrá entender de mejor manera con el código desarrollado para la obtención de la interfaz gráfica.

```
from tkinter import *
from tkinter import filedialog
from PIL import ImageTk,Image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime as dt
from datetime import datetime
```

El código de arriba muestra las principales librerías a importar para el desarrollo de la interfaz gráfica

```
root=Tk()
root.title('Universidad de Piura')
root.geometry("500x280")
global model
global dataset_test
global dataset_test2
global HP
global FP
modelo = r'.\modelofinal.h5'
model=load_model(modelo,custom_objects={'Functional':tf.keras.models.
Model,'TCN': TCN},compile=False)
```

El fragmento del código empieza con la línea *root=Tk()*, la cual genera una instancia de clase Tk que crea el intérprete asociado a Tcl. Asimismo, creará una ventana de nivel superior conocida como la ventana principal o ventana raíz de la interfaz gráfica.(*Interfaces Gráficas de Usuario Con Tk — Documentación de Python - 3.10.2*, n.d.).

Con la ventana principal alojada en una variable que sería *"root",* se procede con la configuración de esta, dándole un ancho de 500 caracteres por 280 caracteres de alto con el comando *"geometry".*

Con la configuración inicial de la ventana principal o raíz, se inicia el comando global que permite hacer globales todas aquellas variables que van a ser utilizadas en diferentes funciones o eventos, sin necesidad de tener que acceder a la función donde fue creada.

Así mismo, se le asigna a la variable "modelo" una dirección en la que buscará el archivo "modelofinal.h5". La razón por la cual va un punto al inicio de la dirección, entre las comillas, es para que cualquier persona que tenga este archivo con extensión .h5 pueda hacer uso de la interfaz con solo tener el archivo ubicado en la misma carpeta donde se encuentra el código.

```
def input():
        filename = filedialog.askopenfilename(initialdir = "/",
                                              title = "Select a File",
                                              filetypes = (("Csv files",
                                                            "*.csv*"),
                                                           ("all files",
                                                            "*.*")))
    dataset_test = pd.read_csv(filename)
```

En el código mostrado arriba se define la función *"input"*, la cual empieza con la variable *"filename"* que crea un diálogo permitiéndole al usuario poder subir un archivo en formato ".csv" y luego la variable *"dataset_test"* se encargará de leer el archivo.

Dentro de la función *"input"* también se encontrarán todas aquellas operaciones que son necesarias para la obtención de las 4 características explicadas en el capítulo 4, que, posterior a la aplicación del "One Hot Encoding", resultan en 16 columnas con valores de 0 o 1 según corresponda.

```
        my_button2=Button(root, text="Predice y grafica", command=graph)
        my_button2.pack()
        my_button3=Button(root, text="Mostrar parámetros importantes",
command=parametros)
        my_button3.pack()
```

Dentro de la función "*input"*, son creadas las variables "*my_button2"* y "*my_button3"*, las cuales generarán dos botones que se explicarán a continuación: la variable "my_button2", tiene asociada la función *"graph",* encargada de graficar la predicción generada por el modelo cargado; la variable "my_button3", tiene asociada la función *"parametros"*, encargada de mostrar en una ventana los cálculos de potencia y energía hallados, que servirán al usuario para su conocimiento.

Es importante resaltar que todos los botones entrarán en ejecución frente a un determinado evento, que en este caso será realizar un clic sobre cada uno de ellos.

```
def parametros():
    global HP
    global FP
    global my_button3
    global a
    global b
    global w
    global msg
    global msg2
    import numpy as np
    from scipy.integrate import simps
    from numpy import trapz
    hp=HP['PotAct'].to_numpy()
    fp=FP['PotAct'].to_numpy()
    total=PREDICTIONS_FUTURE['PotAct'].to_numpy()
    maxvalue_HP = np.max(hp)
    maxvalue_FP = np.max(fp)
    maxvalue=np.array([maxvalue_HP, maxvalue_FP])
    maxvalue=np.max(maxvalue)
    areaHP = trapz(hp, dx=0.5)
    areaTOTAL= trapz(total, dx=0.5)
    areaFP = areaTOTAL - areaHP
     w = Label(root, text ='Parámetros importantes', font = "Verdana
18",fg="Navyblue")
    w.pack()
      a = Label(root, text ='Energia activa', font="Verdana 12
underline",pady=0.3)
    a.pack()
```

```
        var = StringVar()
        msg = Message(root, textvariable = var,justify=LEFT)
        var.set(f"  * HP = {areaHP:.2f} kWh\n  * FP ={areaFP:.2f} kWh\n  *
Total ={areaTOTAL:.2f} kWh")
        msg.pack(expand=True, fill='x')
        msg.bind("<Configure>", lambda e: msg.configure(width=e.width-10))
       b = Label(root, text ='Valor pico de potencia activa',  font="Verdana
12 underline",pady=0.3)
        b.pack()
        var2 = StringVar()
        msg2 = Message(root, textvariable = var2,justify=LEFT)
        var2.set(f"  * HP = {maxvalue_HP:.2f} kW\n  * FP ={maxvalue_FP:.2f}
kW\n  * Del día ={maxvalue:.2f} kW")
        msg2.pack(expand=True, fill='x')
         msg2.bind("<Configure>", lambda e: msg2.configure(width=e.width-
10))
        my_button3.destroy()
```

La función *"parametros"* primero calcula la energía activa consumida, en hora punta (HP) y fuera de punta (FP). Para la obtención de esta variable se aplicó una función propia de la librería de *NumPy* que hace uso del método de los trapecios para el cálculo del área bajo la curva. En cuanto a la potencia activa, solo se mostraron los valores máximos tanto en HP como en FP.

Con los valores obtenidos, se procedió a hacer uso de ciertas funciones del paquete Tkinter para poder mostrar cada uno de ellos en la interfaz desarrollada. Por este motivo, se observa en el código que a diferentes variables se les asigna texto y valores a imprimir en la pantalla una vez ejecutada la función.

```
    def graph():
        from datetime import datetime
        dfa = PREDICTIONS_FUTURE.reset_index()
        dfa=dfa.rename(columns={'index':'date'})
        import plotly.express as px
        fig = px.line(dfa, x="date", y=dfa.columns,
                      title='Predicción potencia activa [kW]',range_y=[0,100])
        fig.show()
        my_button2.destroy()
```

Líneas arriba se visualiza el código utilizado para la obtención de la gráfica que será mostrada en la interfaz. Cabe resaltar que esta gráfica ha sido realizada con la librería *Plotly*,

la cual tiene como producto una gráfica exportada en formato *html*; por este motivo, cuando esta función sea invocada, la gráfica se mostrará en el navegador del ordenador.

Un punto importante por señalar es que la librería *Plotly* permite ver valores puntuales a lo largo de la gráfica, ya que es una gráfica dinámica que reacciona al pasar el cursor por cada uno de los puntos que la conforman.

```
my_button1=Button(root, text="Cargar datos", command=input)
my_button1.pack()
root.mainloop()
```

La variable "*my_button1*" crea el botón principal, el único que se observa en la fase inicial del programa. Este botón permite cargar el archivo con extensión ".csv", al cual se le asigna la función *"input"*. Entonces, cuando se dé clic sobre el botón llamado "Cargar datos", se abrirá un diálogo que permitirá cargar el archivo y posteriormente se irán ejecutando las demás funciones ya definidas y explicadas.

Finalmente, se observa "*root.mainloop()*", que se encarga de mostrar todo en la pantalla y responde a la entrada del usuario hasta que el programa se termine.(*Tkinter — Interface de Python Para Tcl/Tk — Documentación de Python - 3.10.2*, n.d.)

A continuación, se muestra el flujo de trabajo de la interfaz gráfica, en sus tres fases principales: fase inicial de carga de datos, fase de predicción y ploteo de resultados, fase de cálculo de energía activa consumida y picos de potencia activa.

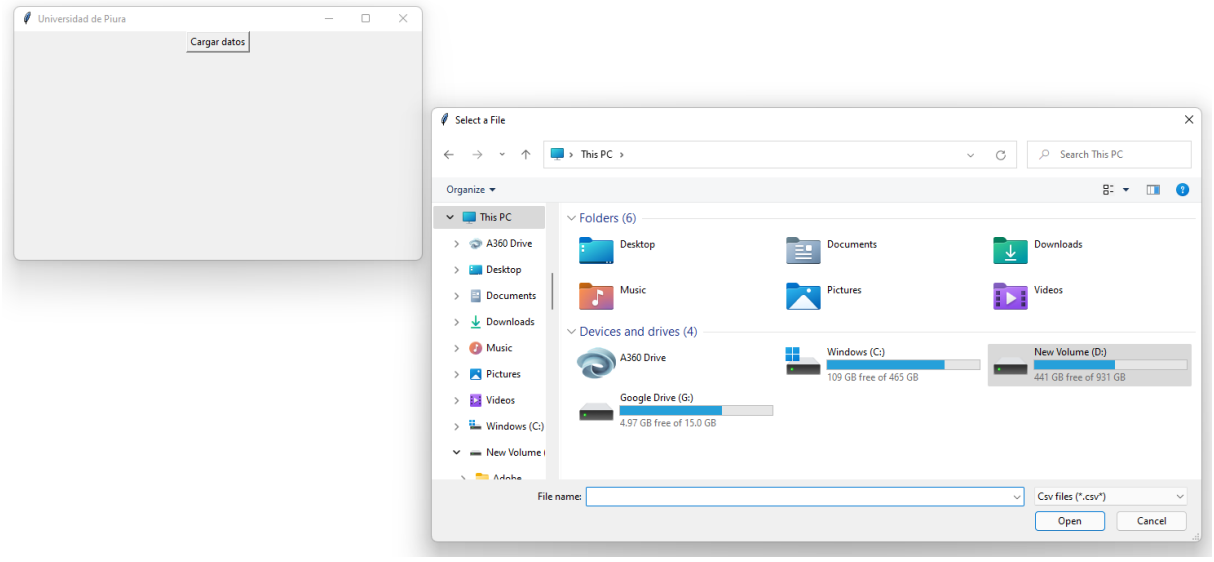


**Figura 51.** Fase inicial del programa donde se solicita cargar datos

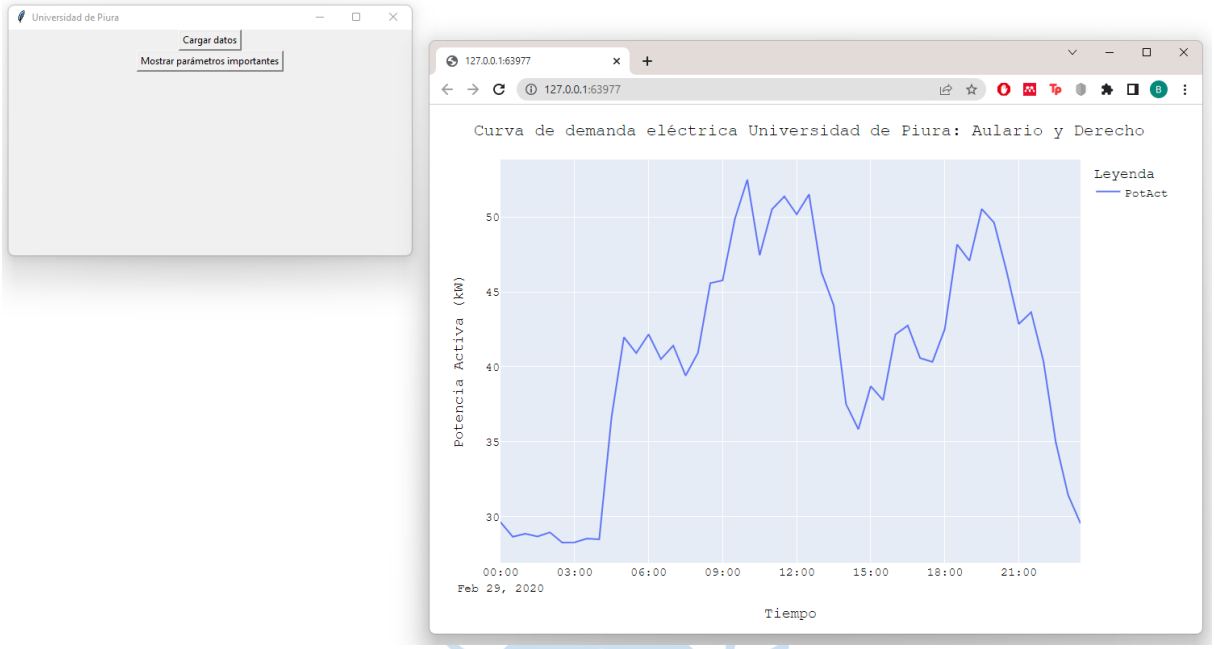


**Figura 52**. Fase de predicción y ploteo de resultados

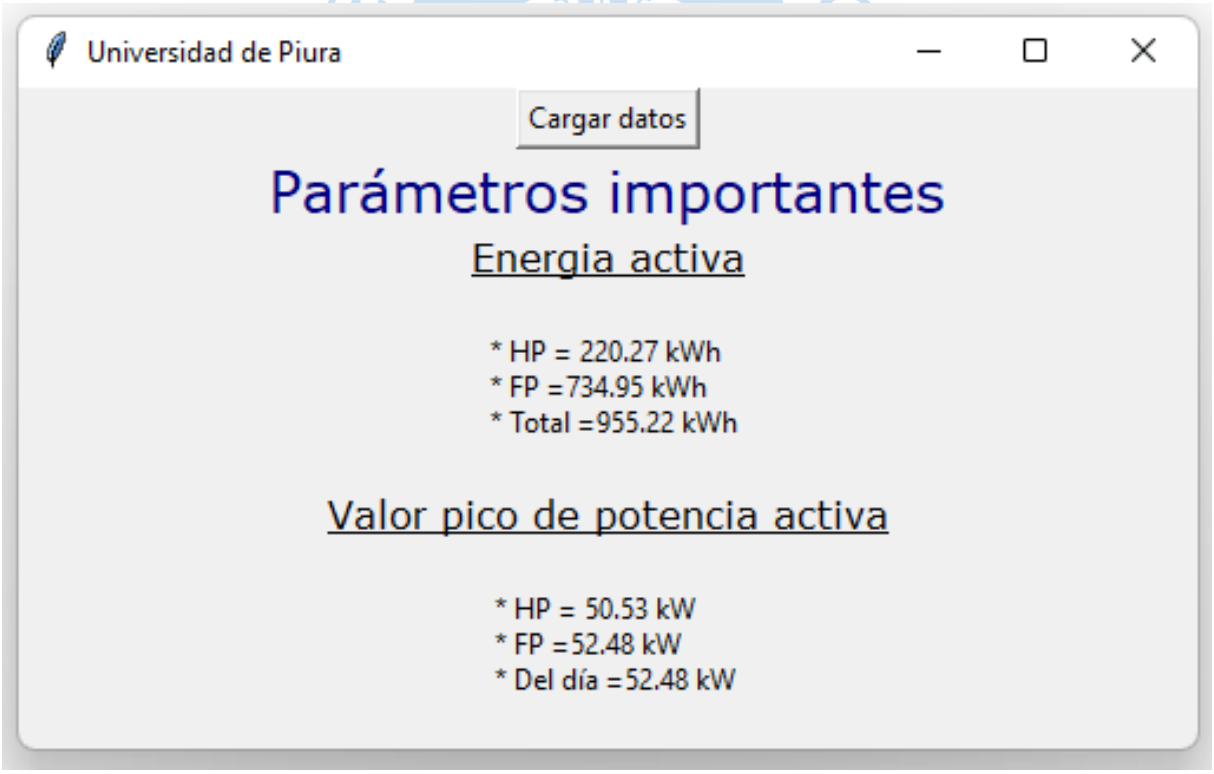


**Figura 53**. Fase de cálculo de energía activa consumida y picos de potencia activa

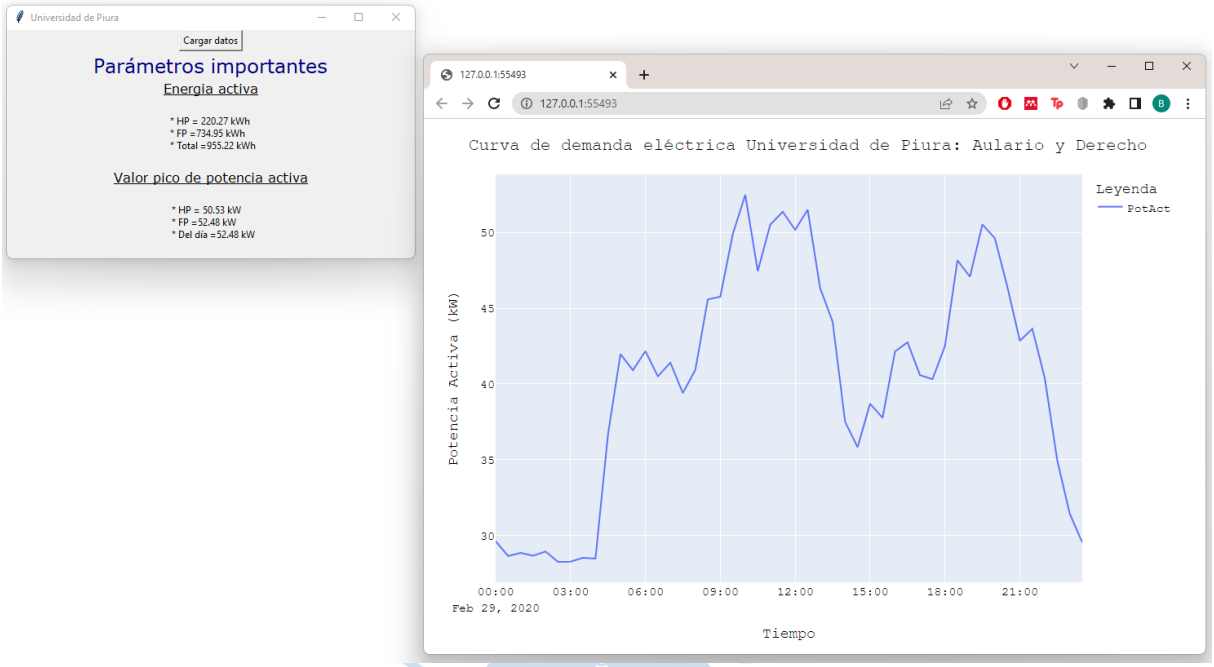


**Figura 54**. Funcionamiento completo de la interfaz gráfica

En la Figura 54 se puede observar el producto final obtenido: una interfaz gráfica que, al recibir un conjunto de datos correspondientes a la demanda eléctrica de un día entero, es capaz de predecir la demanda eléctrica del día siguiente, graficarla e identificar parámetros importantes como energía activa consumida y picos de potencia activa.

## Conclusiones

- La inclusión de Machine Learning en la industria eléctrica trae consigo múltiples beneficios como la reducción de costos, ahorro energético y mejora la experiencia del consumidor. Además, genera nuevas oportunidades en la industria y reduce riesgos porque estos modelos, entrenados solamente con datos históricos, son capaces de predecir e identificar múltiples parámetros importantes para la elaboración de estrategias y toma de decisiones.

- Actualmente, en tiempos de la cuarta revolución industrial, también conocida como Industria 4.0, es posible manejar enormes cantidades de datos y crear sistemas computacionales capaces de realizar tareas que normalmente requerirían de inteligencia humana. Por ejemplo, con la implementación de medidores inteligentes, se puede registrar data histórica de consumo eléctrico y con ello entrenar redes neuronales para la predicción de la demanda eléctrica de viviendas, edificios, fábricas, etc. Por este motivo, es imperativa la educación e investigación de estas nuevas tecnologías, para poder aprovechar la gran cantidad de datos que hoy en día se registran y con ellos crear sistemas inteligentes en el futuro.

- Para la predicción de la demanda eléctrica existen muchos modelos que pueden cumplir la tarea con precisión. En este caso se optó por elegir un modelo tradicionalmente usado y popular en el rubro, el LSTM, y se comparó con un modelo más reciente, el TCN. Los resultados demuestran que ambos modelos son bastante buenos en la predicción de la demanda eléctrica; sin embargo, el modelo TCN se mostró ligeramente superior en cuanto a velocidad de entrenamiento y precisión. Este modelo obtuvo un tiempo medio de 4.54 segundos por época y un MAPE de 8.4%, comparados a los 5.92 segundos por época y 10.4% de MAPE del modelo LSTM.

- El conocimiento del régimen de actividades y horario de funcionamiento del Edificio “E” y el Edificio de Derecho marcó un punto de partida para el análisis y selección de características que, posteriormente, fueron añadidas al conjunto de datos de demanda eléctrica obtenidos desde marzo de 2019 a febrero de 2020. La

adición de estas características tuvo un fuerte impacto en la precisión de los modelos, según los cuatro casos explicados en el capítulo 5. Con el caso 3 se determinó que los patrones identificados por los autores en el capítulo 4 fueron los que tenían mayor relevancia en el aprendizaje del modelo, y que la consideración de parámetros climáticos no contribuyó considerablemente en la disminución del error de predicción, validando así la hipótesis planteada en el capítulo 4, en el cual se analiza la desviación estándar y se concluye que no existe una variación notoria entre las diferentes épocas del año en lo que respecta a los parámetros de temperatura y humedad relativa.

- Con la interfaz gráfica se entrega una herramienta amigable con el usuario, que permite la predicción de la demanda eléctrica del suministro 16196527. Esta interfaz está diseñada para recibir datos de la demanda eléctrica de un día entero, en resolución treintaminutal, para predecir la demanda del día siguiente en la misma resolución. El uso de esta herramienta puede ser beneficioso de múltiples maneras, por ejemplo, hace posible la elaboración de estrategias que permitan ahorro en el pago de tarifas y proporciona una referencia para destacar un consumo de energía anormalmente alto o bajo.

## Recomendaciones

- Continuar con la investigación a profundidad del modelo TCN, ya que es un modelo bastante reciente y con muy buenos resultado, al cual se recomienda implementar técnicas como las de Transfer Learning y Fine Tuning con la finalidad de reutilizar el modelo ya obtenido y patrones ya aprendidos para seguir ampliando y mejorando la precisión del modelo.

- Como futura ampliación de la investigación realizada, se aconseja la implementación de un modelo dinámico el cual al recibir data histórica y realizar predicciones siga aprendiendo nuevos patrones y mejorando así su precisión.

- El presente trabajo realizado sirva como precedente de futuras investigaciones en el campo de las energías limpias, ya que al tener como resultado un modelo con buena precisión en series de tiempo, se pueden desarrollar proyectos para el estudio de la producción de energía solar o eólica y poder evaluar su rentabilidad en base a las predicciones de la producción de esta y poder así optimizar la ubicación de estos.

- Desarrollar algoritmos que permitan identificar todo tipo de patrones dependiendo de la naturaleza de la data, permitiéndole así al usuario poder aprovechar toda la data y lograr predicciones más precisas. Durante el desarrollo del presente trabajo de investigación se demostró la importancia de estos. Así mismo, se puede realizar una evaluación de aquellas características que influyen a grande escala en la predicción de la potencia activa para así solo utilizar aquellos y reducir el costo computacional.

UNIVERSITAS STUDIORUM PIURENSIS

**Referencias bibliográficas**


*About Keras*. (n.d.).

Afshine Amidi, S. A. (n.d.). *CS 230 - Convolutional Neural Networks Cheatsheet*.

Akarslan, E., & Hocaoglu, F. O. (2018). Electricity demand forecasting of a micro grid using ANN. *2018 9th International Renewable Energy Congress, IREC 2018*, *Irec*, 1–5. https://doi.org/10.1109/IREC.2018.8362471

Alla, S., & Adari, S. K. (2019). Beginning Anomaly Detection Using Python-Based Deep Learning. In *Beginning Anomaly Detection Using Python-Based Deep Learning*. https://doi.org/10.1007/978-1-4842-5177-5

Amidi, A. A. and S. (n.d.). *CS 230 - Recurrent Neural Networks Cheatsheet*.

Bai, S., Kolter, J. Z., & Koltun, V. (2018). *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. http://arxiv.org/abs/1803.01271

Berzal, F. (2018). Redes Neuronales & Deep Learning. *Departamento de Ciencias de La Computacion e IA*, 803.

Chaudhary, N., Misra, S., Kalamkar, D., Heinecke, A., Georganas, E., Ziv, B., Adelman, M., & Kaul, B. (2021). Efficient and Generic 1D Dilated Convolution Layer for Deep Learning. *Proceedings Of*, *1*(1). http://arxiv.org/abs/2104.08002

Choi, H., Ryu, S., & Kim, H. (2018). Short-Term Load Forecasting based on ResNet and LSTM. *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids, SmartGridComm 2018*, 1–6. https://doi.org/10.1109/SmartGridComm.2018.8587554

Dudek, G., Pelka, P., & Smyl, S. (2021). A Hybrid Residual Dilated LSTM and Exponential Smoothing Model for Midterm Electric Load Forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 1–13. https://doi.org/10.1109/TNNLS.2020.3046629

El-amir, H., & Hamdy, M. (n.d.). *Deep Learning Pipeline*.

*Events and Binds in Tkinter | Tkinter | python-course.eu*. (n.d.).

Gan, Z., Li, C., Zhou, J., & Tang, G. (2021). Temporal convolutional networks interval prediction model for wind speed forecasting. *Electric Power Systems Research*, *191*(March 2020), 106865. https://doi.org/10.1016/j.epsr.2020.106865

Géron, A. (2019). Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. In

*Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. https://doi.org/10.1201/9780367816377

Google. (n.d.). *Google Colab*. Preguntas Frecuentes. Retrieved July 27, 2022, from https://research.google.com/colaboratory/intl/es/faq.html

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2015). TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS 1 LSTM: A Search Space Odyssey. *ArXiv:1503.04069*, 1–11. https://arxiv.org/pdf/1503.04069.pdf

Hewage, P., Behera, A., Trovati, M., Pereira, E., Ghahremani, M., Palmieri, F., & Liu, Y. (2020). Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*, *24*(21), 16453–16482. https://doi.org/10.1007/s00500-020-04954-0

Hochreiter, S. (1997). Long Short-Term Memory. *Neural Computation*, *1780*, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

IBM Cloud Education. (2020). *What are Neural Networks? | IBM*. https://www.ibm.com/cloud/learn/neural-networks#toc-what-are-n-2oQ5Vepe

*Interfaces gráficas de usuario con Tk — documentación de Python - 3.10.2*. (n.d.).

Juliana Delua. (n.d.). *Supervised vs. Unsupervised Learning: What's the Difference? | IBM*.

Ketkar, N., & Moolayil, J. (2018). Deep Learning with Python. In *Deep Learning with Python*. https://doi.org/10.1007/978-1-4842-5364-9

Kong, W., Dong, Z. Y., Jia, Y., Hill, D. J., Xu, Y., & Zhang, Y. (2019). Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. *IEEE Transactions on Smart Grid*, *10*(1), 841–851. https://doi.org/10.1109/TSG.2017.2753802

Lässig, F. (2021). *Temporal Convolutional Networks and Forecasting*. Unit8. https://unit8.com/resources/temporal-convolutional-networks-and-forecasting/

Lazzeri, F. (2020). Machine Learning for Time Series Forecasting with Python®. In *Machine Learning for Time Series Forecasting with Python®*. https://doi.org/10.1002/9781119682394

Li, X., Hu, Z., & Huang, X. (2020). Combine Relu with Tanh. *Proceedings of 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2020*, *Itnec*, 51–55. https://doi.org/10.1109/ITNEC48623.2020.9084659

Lin, W. H., Wang, P., Chao, K. M., Lin, H. C., Yang, Z. Y., & Lai, Y. H. (2021). Wind power forecasting with deep learning networks: Time-series forecasting†. *Applied Sciences (Switzerland)*, *11*(21). https://doi.org/10.3390/app112110335

Lin, Y., Koprinska, I., & Rana, M. (2020). Temporal Convolutional Neural Networks for Solar Power Forecasting. *Proceedings of the International Joint Conference on Neural Networks*. https://doi.org/10.1109/IJCNN48605.2020.9206991

Liu, Y., Dong, H., Wang, X., & Han, S. (2019). Time series prediction based on temporal convolutional network. *Proceedings - 18th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2019*, 300–305. https://doi.org/10.1109/ICIS46139.2019.8940265

Mathematics, A. (2011, September 19). *Inverse Hyperbolic Tangent -- from Wolfram MathWorld*. https://mathworld.wolfram.com/HyperbolicTangent.html

MathWorks. (n.d.). *Creating, Concatenating, and Expanding Matrices - MATLAB & Simulink*. Retrieved October 6, 2021, from https://es.mathworks.com/help/matlab/math/creating-and-concatenating-matrices.html?lang=en

Mebsout, I. (2020, April 14). *Recurrent Neural Networks | Towards Data Science*. https://towardsdatascience.com/recurrent-neural-networks-b7719b362c65

Mehreen Saeed. (2021, August 25). *A Gentle Introduction To Sigmoid Function*. https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/

*One Hot Encoding | Interactive Chaos*. (n.d.).

Orellana Romero, J. L. (2012). *" Modelación y Pronóstico de la Demanda de Energía Eléctrica de Mediano Plazo de El Salvador ."* 1–190.

*Pandas - Python Data Analysis Library*. (n.d.).

Patterson, J., & Gibson, A. (2017). *Deep Learning: A practitioner's approach*.

Pham, A. D., Ngo, N. T., Ha Truong, T. T., Huynh, N. T., & Truong, N. S. (2020). Predicting energy consumption in multiple buildings using machine learning for improving energy efficiency and sustainability. *Journal of Cleaner Production*, *260*, 121082. https://doi.org/10.1016/j.jclepro.2020.121082

Phi, M. (2018). Illustrated Guide to LSTM ' s and GRU ' s : A step by step explanation The Problem , Short-term Memory. *Medium.Com*, 1–15. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

*Plotly Open Source Graphing Libraries | | Plotly*. (n.d.).

Rosebrock, A. (2017). Deep Learning for Computer Vision with Python - Starter Bundle. *Pyimagesearch*, 330.

RTMath, & Deltix Inc. (2020). *Pointwise Operations*. https://rtmath.net/assets/docs/finmath/html/3545b633-ea89-4fca-a050-df87cc7240d8.htm

Sanjinés Tudela, G. N. (2011). Análisis y pronóstico de la demanda de potencia eléctrica en Bolivia: una aplicación de redes neuronales. *Revista Latinoamericana de Desarrollo Económico*, *15*, 45–78. https://doi.org/10.35319/lajed.201115150

Sathya, R., & Abraham, A. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. *International Journal of Advanced Research in Artificial Intelligence*, *2*(2). https://doi.org/10.14569/ijarai.2013.020206

Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding Machine Learning. In *Understanding Machine Learning*. https://doi.org/10.1017/cbo9781107298019

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, 132306. https://doi.org/10.1016/j.physd.2019.132306

*sklearn.preprocessing.StandardScaler — scikit-learn 1.0.2 documentation*. (n.d.).

Soon Chua Chiah, Zhaochen Li, Jia Yong Quah, & Htoo Lin Min. (2020). Predicting energy demand with neural networks. *Towardsdatascience*, 1–29. https://towardsdatascience.com/forecasting-energy-consumption-using-neural-networks-xgboost-2032b6e6f7e2

*tkinter — Interface de Python para Tcl/Tk — documentación de Python - 3.10.2*. (n.d.).

Vasquez, J., & Perea, G. (2020). *Metodología para realizar auditoría de energía eléctrica . Caso aplicativo : Edificio de educación superior* [Universidad de Piura]. https://hdl.handle.net/11042/4791

Wang, Y., Gan, D., Sun, M., Zhang, N., Lu, Z., & Kang, C. (2019). Probabilistic individual load forecasting using pinball loss guided LSTM. *Applied Energy*, *235*(October 2018), 10–20. https://doi.org/10.1016/j.apenergy.2018.10.078

Zhang, R., Chen, Z., Chen, S., Zheng, J., Büyüköztürk, O., & Sun, H. (2019). Deep long short-term memory networks for nonlinear structural seismic response prediction. *Computers and Structures*, *220*, 55–68. https://doi.org/10.1016/j.compstruc.2019.05.006

Zhang, W. J., Yang, G., Lin, Y., Ji, C., & Gupta, M. M. (2018). On Definition of Deep Learning. *World Automation Congress Proceedings*, *2018-June*, 232–236. https://doi.org/10.23919/WAC.2018.8430387