UNIVERSIDAD TÉCNICA DE MACHALA
FACULTAD DE INGENIERÍA CIVIL

MAESTRÍA EN SOFTWARE

TEMA

DESARROLLO DE SOFTWARE DE PREDICCIÓN DE CARGA DE ENERGÍA BASADO EN MACHINE LEARNING

AUTORA
PAQUITA ALEJANDRA CUADROS GARCÍA

PROPUESTA METODOLÓGICA Y TECNOLÓGICA AVANZADA

TUTOR
ING. JOHNNY PAUL NOVILLO VICUÑA

COTUTOR
ING. DIXYS LEONARDO HERNANDEZ ROJAS

MACHALA
2023

PENSAMIENTO

“No es inteligencia artificial lo que me preocupa, es la estupidez humana”.

- Neil Jacobstein

DEDICATORIA

A mi madre quién con su amor, sus oraciones, sus consejos y palabras de aliento me han permitido llegar a cumplir hoy un sueño más, gracias por inculcar en mí el ejemplo de no temer las adversidades porque Dios está conmigo siempre.

A Thais, Ashley, Alejandro y Angelito quienes con su infancia me dan ánimos para mejorar cada día.

AGRADECIMIENTOS

Quiero agradecer a mi tutor de esta tesis Ing. Johnny Novillo, por el respeto a mis sugerencias e ideas y por la dirección para culminar esta tesis.

A mis amigos de verdad Sandra, Cristian, Israel, Fernando, David, Mónica, Wilmer, Alexander, Jessica por brindarme su apoyo moral y humano cuando más lo necesité, por extender su mano cada día, siempre los llevo en mi corazón.

Gracias a mi familia, porque con ellos comparto mis días felices y días grises que guardo en el recuerdo y son mi aliento para continuar.

A todos, muchas gracias

RESPONSABILIDAD DE AUTORÍA

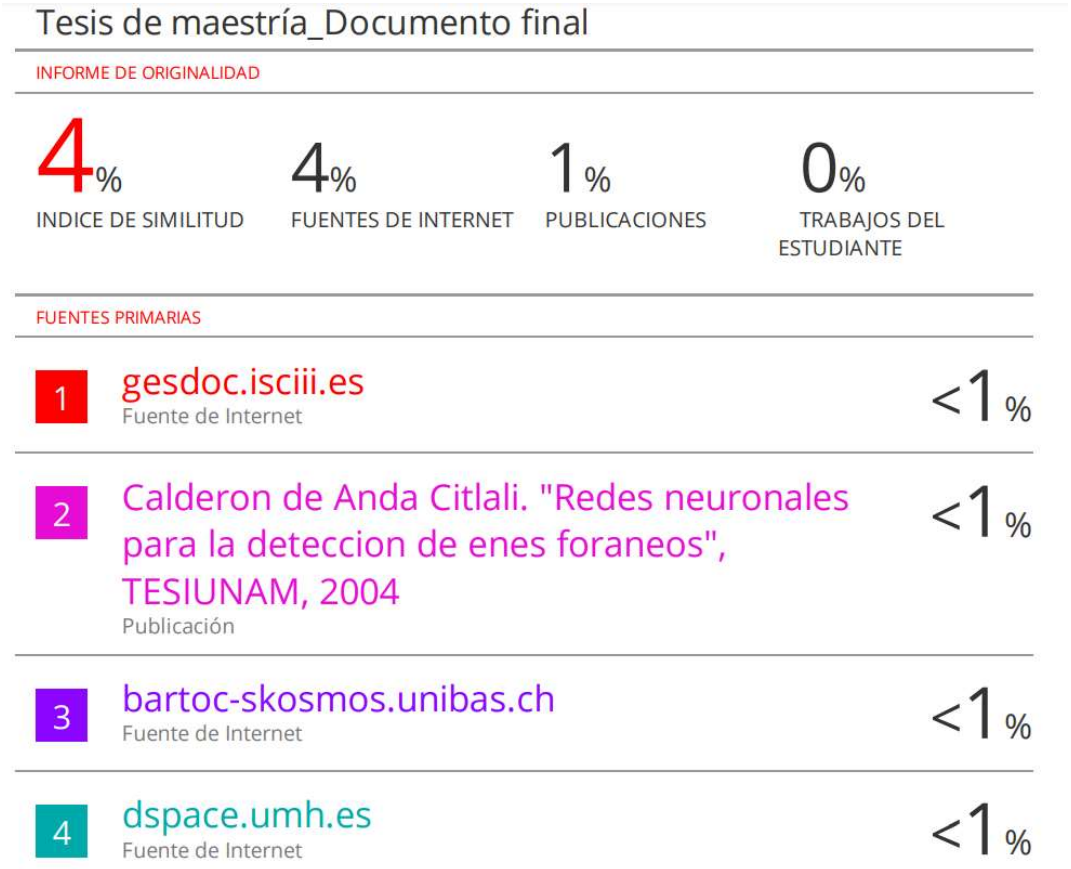
Yo, PAQUITA ALEJANDRA CUADROS GARCÍA con C.I. 0704152065 declaro que el trabajo de "DESARROLLO DE SOFTWARE DE PREDICCIÓN DE CARGA DE ENERGÍA BASADO EN MACHINE LEARNING", en opción al título de Magister en Software, es original y auténtico; cuyo contenido: conceptos, definiciones, datos empíricos, criterios, comentarios y resultados son de mi exclusiva responsabilidad.

PAQUITA ALEJANDRA CUADROS GARCÍA

C.I. 0704152065

Machala, 2023/junio/30

REPORTE DE SIMILITUD URKUND



Tesis de maestría_Documento final

INFORME DE ORIGINALIDAD

4%	4%	1%	0%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	gesdoc.isciii.es Fuente de Internet	<1%
2	Calderon de Anda Citlali. "Redes neuronales para la deteccion de enes foraneos", TESIUNAM, 2004 Publicación	<1%
3	bartoc-skosmos.unibas.ch Fuente de Internet	<1%
4	dspace.umh.es Fuente de Internet	<1%

CERTIFICACIÓN DEL TUTOR

Yo, JOHNNY PAUL NOVILLO VICUNA con C.C. 0702947409; tutor del trabajo de "DESARROLLO DE SOFTWARE DE PREDICCIÓN DE CARGA DE ENERGÍA BASADO EN MACHINE LEARNING", en opción al título de Magister en Software, ha sido revisado, enmarcado en los procedimientos científicos, técnicos, metodológicos y administrativos establecidos por el Centro de Posgrado de la Universidad Técnica de Machala (UTMACH), razón por la cual doy fe de los méritos suficientes para que sea presentado a evaluación.

JOHNNY PAUL NOVILLO VICUNA

Firmado digitalmente porJOHNNY PAUL NOVILLO VICUNA
DN: cn=JOHNNY PAUL NOVILLO VICUNA c=EC o=SECURITY DATA S.A. 1 ou=ENTIDAD DE CERTIFICACION DE INFORMACION
Motivo:JohnnyNovillo
Ubicaci- n:
Fecha:2023-07-20 11:13-05:00

JOHNNY PAUL NOVILLO VICUNA

C.C. 0702947409

Machala, 2023/junio/30

CESIÓN DE DERECHOS DE AUTOR

Yo, Paquita Alejandra Cuadros García con C.C.0704152065, autora del trabajo de titulación "DESARROLLO DE SOFTWARE DE PREDICCIÓN DE CARGA DE ENERGÍA BASADO EN MACHINE LEARNING", en opción al título de Magister en Software, declaro bajo juramento que:

- El trabajo aquí descrito es de mi autoría, que no ha sido presentado previamente para ningún grado o calificación profesional. En consecuencia, asumo la responsabilidad frente a cualquier reclamo o demanda por parte de terceros de manera exclusiva.
- Cede a la Universidad Técnica de Machala de forma exclusiva con referencia a la obra en formato digital los derechos de:
    a. Incorporar la mencionada obra en el repositorio institucional para su demostración a nivel mundial, respetando lo establecido por la Licencia *Creative Commons Atribution-NoCommercial* – Compartir Igual 4.0 Internacional (CC BY NCSA 4.0); la Ley de Propiedad Intelectual del Estado Ecuatoriano y el Reglamento Institucional.
    b. Adecuarla a cualquier formato o tecnología de uso en INTERNET, así como correspondiéndome como Autora la responsabilidad de velar por dichas adaptaciones con la finalidad de que no se desnaturalice el contenido o sentido de la misma.

PAQUITA ALEJANDRA CUADROS GARCÍA

C.I. 0704152065

Machala, 2023/junio/30

RESUMEN

La presente investigación se centra en el desarrollo de un sistema de predicción de carga de energía eléctrica para validar el comportamiento energético utilizando redes neuronales convolucionales (CNN), redes neuronales artificiales (ANN) y la memoria de corto y largo plazo (LSTM). Se emplean métricas como el Error Porcentual Absoluto Medio (MAPE) y el coeficiente de determinación (R2) para evaluar la precisión de las predicciones.

La metodología utilizada en la investigación es mixta, combinando enfoques cuantitativos y cualitativos. Se sigue la metodología CRISP-DM, que consta de diferentes etapas como comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue.

El enfoque de investigación es experimental, donde se manipulan y monitorean variables de acuerdo a las métricas establecidas. Además, el estudio se desarrolla en un contexto descriptivo, presentando y analizando las variables relevantes.

El objetivo principal del trabajo es desarrollar un sistema de predicción de carga de energía eléctrica utilizando las redes neuronales CNN, ANN y LSTM, y evaluar su desempeño utilizando las métricas MAPE y R2. El propósito es mejorar la capacidad de predicción de la carga de energía eléctrica y proporcionar resultados confiables y precisos. Se realizan pruebas y experimentos para comparar el rendimiento de los diferentes modelos y técnicas utilizadas. Se busca identificar el modelo más eficiente y efectivo en la predicción de la carga de energía eléctrica.

En resumen, la tesis se enfoca en el desarrollo de un sistema de predicción de energía eléctrica utilizando redes neuronales CNN, ANN y LSTM. Se utilizan métricas como MAPE y R2, y se sigue la metodología CRISP-DM. La investigación se lleva a cabo mediante un enfoque mixto, con un enfoque experimental y descriptivo. El objetivo es mejorar la precisión y confiabilidad de la predicción de la carga de energía eléctrica, lo que puede tener aplicaciones importantes en la planificación y gestión eficiente de la energía.

PALABRAS CLAVES: aprendizaje automático, redes neuronales, predicción de carga eléctrica, software, phyton, CRISP-DM.

ABSTRACT

This research focuses on the development of an electrical power charge prediction system to validate power behavior using Convolutional Neural Networks (CNN), Artificial Neural Networks (ANN) and Long and Short Term Memory (LSTM). Metrics such as the Mean Absolute Percentage Error (MAPE) and the coefficient of determination (R2) are used to assess the accuracy of the predictions.

The methodology used in the research is mixed, combining quantitative and qualitative approaches. The CRISP-DM methodology is followed, which consists of different stages such as understanding the business, understanding the data, data preparation, modeling, evaluation and use.

The research approach is experimental, where variables are manipulated and monitored according to the established metrics. In addition, the study is carried out in a descriptive context, presenting and analyzing the relevant variables.

The main objective of the work is to develop an electric power charge prediction system using the CNN, ANN and LSTM neural networks, and to evaluate its performance using the MAPE and R2 metrics. The purpose is to improve the predictability of electrical power load and provide reliable and accurate results.

Tests and experiments are carried out to compare the performance of the different models and techniques used. It seeks to identify the most efficient and effective model in the prediction of the electric power load.

In summary, the thesis focuses on the development of an electrical energy prediction system using CNN, ANN and LSTM neural networks. Metrics such as MAPE and R2 are used, and the CRISP-DM methodology is followed. The research is carried out using a mixed approach, with an experimental and descriptive approach. The goal is to improve the accuracy and reliability of electrical power load forecasting, which may have important applications in efficient power planning and management.

KEYWORDS: machine learning, neural network, electric charge prediction, software, phyton, CRISP-DM.

ÍNDICE GENERAL

LISTA DE ILUSTRACIONES Y TABLAS

LISTA DE ABREVIATURAS Y SÍMBOLOS

LSTM: Long Short-Term Memory

ML: Machine Learning

ANN: Artificial Neural Network

CNN: Convolutional Neural Network

CRISP-DM: Cross-Industry Standard Process for Data Mining

MAPE: Mean Absolute Percentage Error

R2: R cuadrado, coeficiente de determinación

IoT: Internet of Things

GUI: Interfaz gráfica de usuario

INTRODUCCIÓN

La energía eléctrica es indispensable para la vida diaria y esto aumenta la emisión de gases invernaderos, que son perjudiciales e incluso amenazan con la futura existencia de la raza humana. El constante incremento en el uso de la energía hace que exista la posibilidad de que los sistemas eléctricos colapsen.

En las últimas dos décadas la demanda de energía eléctrica se incrementó mundialmente entre 50% en el área industrial y residencial, las nuevas tecnologías pretenden poner a su disposición dispositivos y elementos que aporten a la sostenibilidad, permitiendo tener un futuro más eficiente gracias a la reducción del consumo energético [1].

El internet de las cosas (IoT) hoy es aplicado en una infinidad de áreas, que van desde la agricultura, pasando por la salud, la meteorología, el tráfico vehicular, educación, la conservación ambiental, la visión artificial, etc., [2].

La predicción de carga energética es una parte importante del sistema de potencia, tiene como objetivo principal ampliar la eficiencia de sistemas energéticos, tanto para consumidores y generadores de electricidad, permitiendo ahorrar costos y energía. La recopilación de información permite planificar estratégicamente la relación entre producción y consumo. Al contar con datos precisos sobre la oferta y la demanda, es posible establecer programas y estrategias que permitan una relación óptima entre lo que se produce y lo que se consume. De esta manera, se evita el desperdicio y se maximizan los recursos, generando beneficios a los involucrados. Además, la programación estratégica permite asegurar la calidad de los productos y la satisfacción de los clientes, contribuyendo a un desarrollo sostenible y equilibrado [6]. Manteniendo un registro a largo plazo de las previsiones de carga, se podría planificar a futuro los requisitos de los sistemas de energía, se lograría celebrar contratos con empresas energéticas que puedan tener escasez o exceso de energía, hacer intercambios energéticos, etc. Por el contrario, un registro a corto plazo permite asignar suficiente potencia de energía para responder a las expectativas de demanda y mantener una reserva rotatoria [7].

Los datos generados por los sistemas IoT deben ser analizados, para lo cual se aplicarán técnicas de Machine Learning (ML), las cuales han sido ampliamente utilizadas para solucionar problemas prácticos complejos en diversas áreas, y están ganando cada vez más popularidad [8]. Dentro de la inteligencia artificial constantemente surgen nuevas técnicas y algoritmos orientados a predecir con exactitud el consumo de energía, y el rendimiento de los recursos renovables. Existen diversos modelos de predicción elaborados mediante redes neuronales, los mismos que han sido llevados a cabo por universidades de Colombia, Perú y otras de habla española [9].

Dada la existencia de trabajos previos tanto teóricos como prácticos que vinculan las redes neuronales con la predicción del flujo de carga eléctrica, se ha observado que en muchos casos hay una falta de distribución completa, ya que no incluyen una Interfaz Gráfica de Usuario (GUI) [9].

Hoy en día estamos en plena auge del Machine Learning, su implementación dentro del mundo empresarial, permite encontrar estrategias creativas en muchas áreas, incluyendo los sistemas energéticos y eléctricos.

De acuerdo con J. López [10] se utilizó una herramienta para predicciones eólicas en base a datos meteorológicos. De la misma forma López, E., et. al [11] se utilizó en México la herramienta para predecir la demanda de electricidad. El autor García, R. [12] utiliza Machine Learning para efectuar predicciones de mantenimiento, tomando en cuenta elementos estacionales y el incremento de la carga.

Rodríguez, J. et al [13] crean un sistema automatizado que utiliza redes neuronales artificiales para predecir el flujo de carga en subestaciones eléctricas, debido a que existen antecedentes teóricos y prácticos que relacionan estas técnicas con el pronóstico de carga eléctrica. Villarroel González [14] creó un sistema de predicción de la demanda eléctrica a corto plazo utilizando modelos horarios y diarios en Paraguay. Además, Ramírez, M. [15] empleó técnicas de análisis Wavelet y un modelo neuronal auto-regresivo no lineal NAR para la predicción de la demanda de energía eléctrica del Sistema Interconectado Nacional (SIN) de Colombia. Por otro lado, de acuerdo a Castro D., et al [16] utilizaron los modelos Holt-Winters para pronosticar el consumo de energía eléctrica residencial de la Región Cajamarca.

Confirmando que el IoT es una solución viable para el problema de investigación ya que, la recolección de datos resulta muy importante a la hora de mantener un control estricto del comportamiento del consumo de la energía eléctrica [17], y que es necesario desarrollar una interfaz de usuario utilizando los diferentes algoritmos de redes neuronales, que muestren al usuario las predicciones y permitan tomar decisiones que mejoren la gestión energética. Con esta investigación, de tipo experimental se plantea desarrollar un software de predicción de carga energética usando machine learning para precisar el consumo de energía de diferentes elementos, haciendo uso de datos recolectados mediante un medidor inteligente.

Con las problemáticas anteriormente expuestas surgen las siguientes preguntas:

Pregunta General

¿Cómo predecir el comportamiento de la carga de energía eléctrica en un tiempo determinado?

Preguntas especificas

¿Qué técnicas de Machine Learning permiten validar el comportamiento energético más estable de un software de predicción?

¿Qué técnica de Machine Learning me dará un dato más preciso?

¿Cuáles son los parámetros de ejecución utilizados y definidos para implementación del modelo?

¿Cómo se mide la eficiencia de los resultados de predicción con el uso de redes neuronales?

Objetivo General

Desarrollar un software de predicción de carga de energía eléctrica mediante técnicas de machine learning para validar el comportamiento energético.

Objetivos específicos:

Investigar las técnicas de Machine Learning empleadas en los modelos de predicción de carga de energía eléctrica utilizando la revisión sistemática de la literatura.

Diseño y desarrollo del software de predicción de carga de energía para estimar la demanda de energía eléctrica en un intervalo de tiempo determinado mediante técnicas de Machine Learning.

Realizar las pruebas del software mediante la presentación de resultados obtenidos en la predicción con los cambios de parámetros y tipos de predictores utilizados.

Se plantea además la siguiente teoría a validar, el desarrollo de un software aplicando técnicas de Machine Learning optimiza la predicción de carga eléctrica.

Hipótesis

Se plantea la siguiente suposición a validar, si se desarrolla un software aplicando machine learning se optimiza la predicción de carga eléctrica a través de la aplicación de métricas del Error porcentual absoluto medio (MAPE) y R al cuadrado (R2).

Por otro lado, en el estudio se empleó una metodología mixta que combina la recopilación de datos numéricos y cálculos algorítmicos. Asimismo, se utilizó un diseño de investigación experimental en el proyecto, donde se monitorearon y manipularon las variables de acuerdo con las métricas establecidas. En términos de tipología, se trata de un estudio descriptivo, donde se presentan las siguientes variables:

Variable independiente: Datos Históricos de mediciones
Variable dependiente: Carga de energía eléctrica

La investigación se desarrolla en un total de 4 capítulos. El primer capítulo establece el marco teórico sobre el Machine Learning, la metodología de desarrollo y campo de acción. El capítulo dos establece los métodos y materiales utilizados en el trabajo de investigación. En el tercer capítulo se muestran los resultados obtenidos durante la ejecución del proyecto. Por otro lado, el cuarto capítulo presenta la discusión de los resultados obtenidos

CAPÍTULO 1. MARCO TEÓRICO

1.1 Antecedentes Históricos de la Investigación.

Las decisiones que toman los seres humanos regularmente se basan en la experiencia adquirida, al haber cursado por una experiencia pasada se puede asumir qué consecuencias tendrá en un futuro. En la predicción de carga energética, al conocer su comportamiento se puede establecer la conducta que tendrá posteriormente, esto permitirá mejorar la forma en que gestionamos la energía.

La Tabla 1 nos detalla una mejor apreciación de los aportes científicos que se han realizado con redes neuronales en el campo del consumo energético.

Tabla 1. *Cronología del uso de redes neuronales en la predicción de la energía eléctrica*

Año	Desarrollo
1990	Uso temprano de redes neuronales en la predicción de la demanda de energía eléctrica. Principalmente redes neuronales de retro propagación y recurrentes. [18]
2000	Continúa la investigación en el uso de redes neuronales para la predicción de la energía eléctrica. Se emplean redes neuronales de múltiples capas y se incorporan técnicas de optimización y selección de características. [19]
2010	Aumento en la popularidad del uso de redes neuronales en la predicción de la energía eléctrica. Se emplean arquitecturas más complejas como redes neuronales convolucionales (CNN) y redes neuronales recurrentes de memoria a largo plazo (LSTM). [20]

2015	Se exploran las redes neuronales generativas adversariales (GAN) para la predicción de la energía eléctrica. Estas redes permiten generar datos sintéticos realistas y simular diferentes escenarios. [21]
2018	Se emplean redes neuronales con estructuras de atención en la predicción de la energía eléctrica. Estas redes asignan pesos a diferentes partes de la serie temporal, lo que les permite enfocarse en las características más relevantes. [22]
Actualidad	Continúa la investigación y desarrollo del uso de redes neuronales en la predicción de la energía eléctrica. Se incorporan técnicas de aprendizaje automático avanzadas, como el aprendizaje por refuerzo, para mejorar la precisión de las predicciones y optimizar la gestión de la energía. [23]

Fuente: Elaboración Propia

1.2 Revisión de la literatura: Establecer las técnicas de Machine Learning

Para el proceso de búsqueda se establece como guía el modelo propuesto por Barbara Kitchenham [59]. Este proceso se basará en la búsqueda de fuentes primarias y secundarias de artículos científicos y conferencias.

1.2.1 Preguntas de revisión sistemática

P1. ¿Qué técnicas de Machine Learning permiten validar el comportamiento energético más estable de un software de predicción?

P2. ¿Qué técnica de Machine Learning me dará un dato más preciso?

P3. ¿Cuáles son los parámetros de ejecución utilizados y definidos para implementación del modelo?

1.2.2 Criterios de inclusión y exclusión

Tabla 2. *Criterios de inclusión y exclusión*

Criterios de Inclusión	Criterios de Exclusión
Publicaciones dentro de los últimos 5 años	Estudios no incluidos en las bases de datos seleccionadas
Estudios en el campo de la inteligencia artificial	Estudios duplicados
Estudios que abarquen información de predicciones de carga con machine learning	Fuentes secundarias o terciarias
Fuentes primarias (Artículos científicos y conferencias)	Estudios duplicados

Fuente: Elaboración Propia

1.2.3 Grupo de control

El grupo de control, permite establecer la línea base para el desarrollo del proyecto presentado; todos los artículos analizan cuestiones importantes que aportarán a la construcción de la solución reflejada en esta investigación.

1.2.4 Cadena de búsqueda

Para la cadena de búsqueda se usaron términos claves, se buscaron palabras relacionadas con predicción de carga, Machine Learning, redes neuronales, dentro de los motores de búsqueda definidos en el instructivo para el proceso de titulación de posgrado de la Universidad Técnica de Machala. Para una búsqueda más eficiente se agregaron los operadores lógicos AND y OR para agregar atributos y sinónimos respectivamente, quedando definida la siguiente cadena de búsqueda.

(software and (machine learning OR neural networks) AND energy load prediction)

1.2.5 Selección de búsqueda

En esta sección, fueron seleccionados los estudios de fuentes primarias que cumplen con el criterio de inclusión, tomando en cuenta artículos de revistas desde el año 2019, que abordan el tema de investigación. En la Tabla 7, se describe los artículos encontrados.

Tabla 3. *Artículos que cumplen con los criterios de inclusión y exclusión*

Base de Datos	Resultados
IEEE Explorer	13
Dialnet	1
Scielo	5
DOAJ	3
Taylor & Francis	3

Fuente: Elaboración Propia

1.2.6 Resultados de la revisión

Para seleccionar los estudios más cercanos al objeto de investigación se realizó un filtro secundario mediante la lectura completa de los artículos, dando como resultado un listado de ocho estudios primarios.

Tabla 4. Artículos que cumplen con los criterios de inclusión y exclusión

Autores	Año	Título	Revista	Número	Páginas
Luis Fernando Rueda-Vásquez Jaime Guillermo Barrero-Pérez Cesar Duarte	2017	El conmutador inteligente de potencia y la sub-medición por circuito como herramientas para la gestión energética residencial	UIS Ingenierías	Vol. 16, Núm. 1	35-46
Eduar Jamis Mejía Vásquez, Salome Gonzales Chávez	2019	Predicción del consumo de energía eléctrica residencial de la Región Cajamarca mediante modelos Holt - Winters	Ingeniería Energética	Vol. 40, Núm. 3	181-191
Arlys Michel Lastre Aleaga, Erik Fernando Méndez Garcés, Alexis Cordovés García		Sistema automatizado para la predicción de flujo de carga en subestaciones eléctricas mediante redes neuronales artificiales	Enfoque UTE	Vol.6 Núm.3	
Isa S. Qamber	2022	Predicción del consumo de energía mediante la técnica de desarrollo Petri Nets-ANFIS	Arab Journal of Basic and Applied Sciences	Vol. 29, Núm. 1	193-207

Darío Javier Benavides, Páúl Arévalo Cordero, Luis G. González, Luis Hernández-Callejo, Francisco Jurado, José A. Aguado	2022	Method of monitoring and detection of failures in PV system based on machine learning	Revista Facultad de Ingeniería, Universidad de Antioquia	No.102	26-43
Joseline Sánchez Solís Marvin Coto Jiménez	2022	Estado del Arte de la Predicción de Variables en Sistemas de Ingeniería Eléctrica Basada en Inteligencia Artificial	e-Ciencias de la Información	Vol. 12, Num. 1	59-78
Mohamed Y. AL-Hamad e Isa S. Qamber	2019	Modelado de pronóstico de carga máxima eléctrica a largo plazo de GCC utilizando métodos ANFIS y MLR	Arab Journal of Basic and Applied Sciences	Vol. 26, núm. 1	269-282
Isa S. Qamber	2022	Predicción del consumo de energía mediante la técnica de desarrollo Petri Nets-ANFIS	Arab Journal of Basic and Applied Sciences	Vol. 29, Núm. 1	193-207

Fuente: Elaboración propia

1.3 Antecedentes Conceptuales y referenciales

1.3.1 Machine Learning.

Es una rama de la inteligencia artificial que experimenta con algoritmos de computación basado en patrones. Tom Mitchell [24], en su libro Machine Learning indica lo siguiente, "un programa de computadora aprende de la experiencia E con respecto a alguna clase de tareas T y medida de desempeño P, si su desempeño en tareas en T, medido por P, mejora con la experiencia E" [24]. Es decir, significa que un programa de computadora se ha diseñado para aprender a realizar una tarea específica a través de la experiencia.

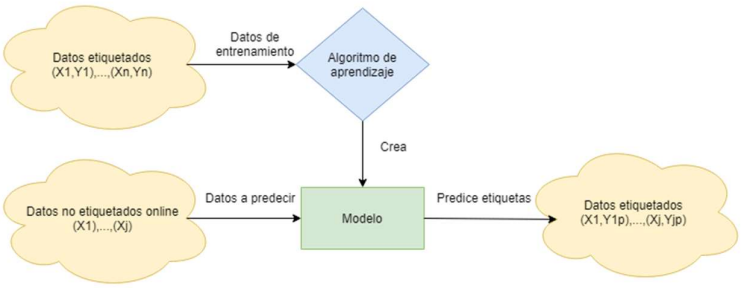
El Machine Learning permite que las computadoras adquieran conocimiento a partir de los datos, en vez de tener que programar soluciones específicas para cada problema de forma manual, como se hace en la programación convencional. En este enfoque, se desarrollan algoritmos genéricos que pueden analizar diferentes tipos de datos para extraer patrones y conocimientos relevantes [25]. La auténtica capacidad de aprendizaje de una máquina reside en un algoritmo que analiza los datos y es capaz de predecir comportamientos futuros. Este algoritmo utiliza la información recopilada para identificar patrones y tendencias, y en base a ellos genera modelos predictivos que permiten anticipar posibles resultados [26].

1.3.2 Técnicas de aprendizaje del Machine Learning.

Las técnicas de aprendizaje del Machine Learning se clasifican en:

Aprendizaje supervisado. Se proporciona un grupo de ejemplos de entrenamiento con las respuestas correctas (objetivos) y, el algoritmo utiliza este conjunto para generalizar y responder correctamente a cualquier entrada posible. Este proceso también se conoce como aprendizaje a partir de ejemplos [27]. Hasta la precisión alcanza el nivel deseado se prosigue con el entrenamiento [28]. Las redes bayesianas, las redes neuronales, y los árboles de decisión son algoritmos que emplean este proceso de aprendizaje. [29].

Ilustración 1. *Esquema general de un modelo de aprendizaje supervisado*

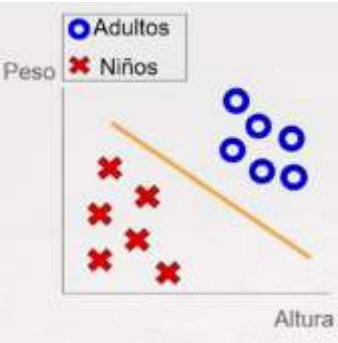


Fuente: Gonzalo A. (2019)

Dentro del aprendizaje supervisado, encontramos modelos de clasificación y modelos de regresión.

Modelo de clasificación. Los modelos de clasificación utilizan datos etiquetados y crean modelos de clasificación de nuevos datos en las etiquetas aprendidas, es decir predicen una categoría. Los algoritmos más usados para resolver estos modelos son modelos ocultos de Markov, las máquinas de vectores de soporte (SVM), los bosques aleatorios, el modelo naïve bayes, los modelos gráficos probabilísticos, los de regresión logística y las redes neuronales [29].

Ilustración 2. *Modelo de clasificación*

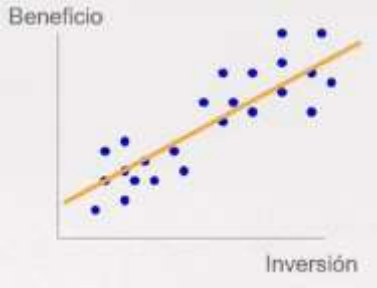


Fuente: Zambrano R. (2022)

Modelo de regresión. La regresión intenta crear un modelo predictivo que optimiza el error en los datos de entrenamiento aprendidos para que el modelo pueda hacer predicciones precisas en nuevos datos [29]. Se centran en modelar las relaciones entre las variables, a fin de crear sistemas que permitan pronosticar la conducta de ciertos fenómenos. Además, se han adaptado para resolver problemas de regresión, lo que permite crear predictores para ciertos eventos [30]. Entre los algoritmos más utilizados

para resolver los modelos de regresión se encuentran el lineal, logístico, el de mínimos cuadrados ordinarios, así como el de splines de regresión adaptativa multivariante [29].

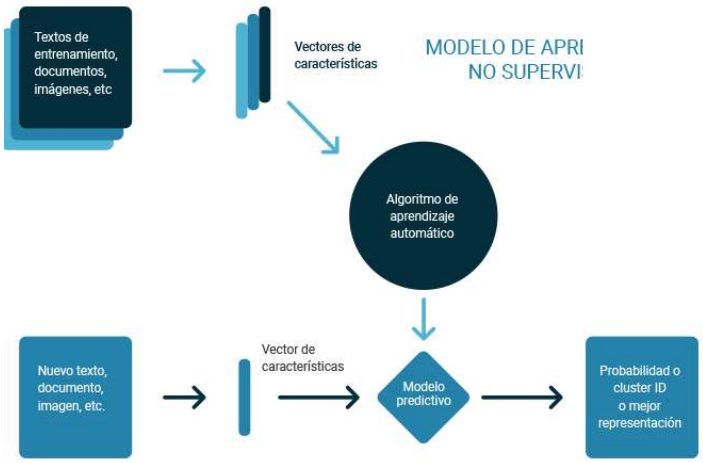
Ilustración 3. *Modelo de regresión*



Fuente: Zambrano R. (2022)

Aprendizaje no supervisado. Durante el proceso de aprendizaje del algoritmo, no se suministran respuestas correctas, sino que se busca que el algoritmo identifique similitudes entre las entradas para clasificar juntas a aquellas que tengan elementos en común [27]. Este objetivo puede lograrse mediante un proceso matemático que reduzca sistemáticamente la redundancia de los datos, o bien, organizando los datos en función de su similitud [31]. La técnica estadística utilizada en el aprendizaje no supervisado es denominada "estimación de densidad" [27]. El aprendizaje no supervisado es utilizado por algoritmos como K-means clustering, clustering jerárquico y análisis de componentes principales [29].

Ilustración 4. *Esquema de un modelo de no supervisión*

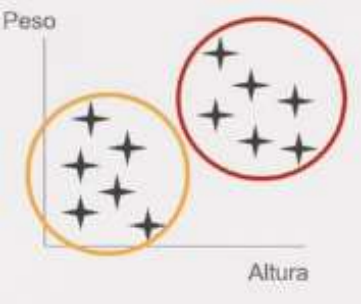


Fuente: Luna J. G. (2022)

Dentro del aprendizaje no supervisado, encontramos modelos de agrupación y modelos de detección de anomalías.

Modelo de agrupación (clustering). El análisis de conglomerados intenta tomar un conjunto de datos y definir conglomerados de elementos similares [29]. Los algoritmos de agrupamiento son similares a los de clasificación, aquí se presenta una circunstancia en la que se desconoce la información previa sobre las características de los datos. En su lugar, estos se agrupan según una medida de igualdad. Estos nuevos casos se pueden asignar a uno de los conjuntos revelados utilizando las mismas medidas de semejanza. [32]. Los algoritmos que emplean esta técnica de aprendizaje incluyen K-medias, jerárquicas y basadas en la densidad (como DBSCAN) [29].

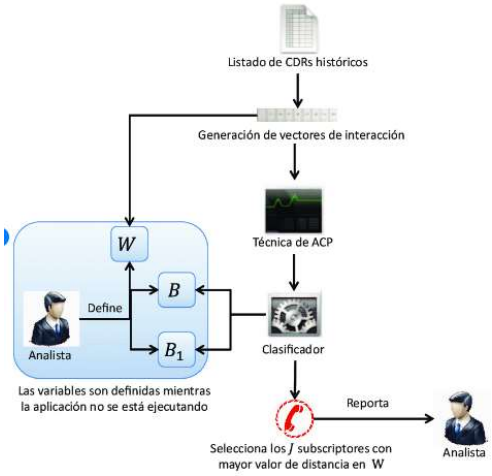
Ilustración 5. *Modelo de agrupación*



Fuente: Zambrano R. (2022)

Modelo de detección de anomalías. Se trata de un proceso de aprendizaje que se utiliza para detectar datos anómalos o inusuales. En este proceso, se utiliza un conjunto de datos "normales", y luego se aplica este modelo para determinar si los nuevos datos son anómalos o tienen una baja probabilidad de ocurrir. Los algoritmos que utilizan este proceso son SVM de una clase, regresión lineal y logística, patrón de crecimiento frecuente (FP-crecimiento) y a priori. [29].

Ilustración 6. *Funcionamiento del modelo de detección de anomalías*



Fuente: V. Herrera Semenets, M. A. Prado Romero y A. Gago Alonso (2014)

Aprendizaje por refuerzo. se ubica en un punto de vista intermedio entre el aprendizaje supervisado y el no supervisado. El algoritmo recibe información cuando su respuesta es incorrecta, pero no se le proporciona información sobre cómo corregir su error. Debe explorar y probar diferentes posibilidades hasta encontrar la forma de obtener la respuesta correcta. A veces se denomina "aprendizaje con un crítico" debido al monitor que califica la respuesta, pero no sugiere mejoras. El aprendizaje por refuerzo utiliza algoritmos como Q-learning, SARSA y DQN [27]. Este tipo de modelos tienen la tarea de aprender tanto la estructura de los datos como de realizar predicciones en base a ella [33]. Algunos algoritmos de aprendizaje automático utilizan el proceso de maximización de expectativa, máquina de vector de apoyo transductivo y los procesos de decisión de Markov para aprender las estructuras de datos y realizar predicciones. [29].

Ilustración 7. *Esquema de un modelo de aprendizaje por refuerzo*

Fuente: Luna J. G. (2022)

1.3.3 Algoritmos de aprendizaje de machine learning.

Regresión lineal: busca establecer la correspondencia entre una o más variables independientes y una variable dependiente, con el propósito de realizar predicciones sobre resultados futuros. En caso de que se tenga una única variable independiente y una variable dependiente, regresión lineal simple es su denominación; por el contrario, si se incrementa la cantidad de variables que son independientes, se denomina regresión lineal múltiple. La meta de la regresión lineal es trazar una línea más precisa que represente la relación entre las variables, la cual se determina a través del método de los mínimos cuadrados. Diferente a otros modelos de regresión, la línea trazada en la regresión lineal siempre es una línea recta cuando se representa en un gráfico [34].

Regresión logística: es adecuada cuando la variable dependiente no es continua, sino que es discreta o categórica, se utiliza un enfoque diferente llamado análisis de clasificación. En este caso, en lugar de predecir un valor numérico, se trata de asignar una categoría a los datos de entrada en función de la relación entre sus variables dependientes e independientes. Esta técnica se utiliza para clasificar los datos en diferentes categorías o clases, como positivo o negativo, sí o no, o diferentes tipos de categorías según el problema específico que se esté abordando, por ejemplo, en la identificación de correos electrónicos de spam [34].

Árboles de decisión: es conocido por su facilidad de interpretación y claridad, lo que lo hace popular debido a su capacidad para presentar los resultados a audiencias no técnicas en la industria. Este algoritmo se representa como un árbol, donde la raíz es el punto de partida y las hojas son el resultado final de las decisiones tomadas en el camino. De esta forma, las reglas de decisión se definen por el camino que sigue el árbol empezando en la raíz y finalizando en las hojas. La visualización y comprensión del proceso de aprendizaje del árbol de decisión resulta sencilla gracias a su similitud con un diagrama de flujo. [34].

Random Forest: se emplea con frecuencia en la clasificación y la regresión debido a su gran flexibilidad, conocido como "Random Forest". El término "Forest" hace referencia a un conjunto de árboles de decisión no correlacionados que se combinan para reducir la varianza y obtener predicciones más precisas de los datos [35].

XGBoost: El modelo se apoya en un árbol de decisión y utiliza un marco de refuerzo de gradiente. Si bien las redes neuronales artificiales suelen ser empleadas para predecir datos no estructurados, como imágenes o texto, los algoritmos basados en árboles de decisión se consideran más apropiados para manejar datos estructurados o tabulares de tamaño moderado. En este sentido, el modelo se enfoca en el análisis de datos tabulares y busca optimizar el rendimiento mediante técnicas de refuerzo de gradiente. De esta forma, se logra una mayor precisión en la predicción de los resultados, lo que resulta especialmente relevante en ámbitos como la ciencia de datos o el aprendizaje automático [36].

Gradient Boosting: Los algoritmos de potenciación del gradiente genera un modelo predictivo que combina varios modelos predictivos débiles, generalmente árboles de decisión, a través de un proceso de ensamblaje que mejora el rendimiento global del modelo [37].

Redes neuronales: son comúnmente utilizadas en algoritmos de aprendizaje profundo, simulan la conectividad del cerebro humano a través de capas de nodos que procesan los datos de entrenamiento. Cada nodo se compone de entradas, pesos, un sesgo (también conocido como umbral) y una salida. Si el valor de salida supera un umbral específico, el nodo se activa, lo que lleva los datos a la siguiente capa de la red. A través del aprendizaje supervisado, las redes neuronales aprenden a mapear estas funciones, permitiendo ajustar los valores de la red neuronal según la función de pérdida empleada. Una vez que la función de pérdida se acerca o es igual a cero, se puede confiar en la precisión del modelo para proporcionar respuestas precisas [38].

Naive Bayes: Las redes bayesianas son una forma gráfica de representar las dependencias y relaciones probabilísticas para el razonamiento en el que cada nodo se asocia con una variable aleatoria y están conectados por medio de arcos, que indican las relaciones directas entre ellas. Cada relación de independencia condicional en el grafo se corresponde con una relación de independencia en la distribución de probabilidad, lo que permite una representación y razonamiento más sencillos, ya que requiere menos parámetros y permite una propagación más fácil de las probabilidades. En resumen, las redes bayesianas son una herramienta visual útil para representar las dependencias e

independencias entre variables aleatorias, especialmente las independencias condicionales [39].

Isolation Forest: este método de detección de anomalías emplea árboles binarios y el concepto de aislamiento con el fin de identificar valores atípicos [40]. Se crea una estructura en forma de árbol que empieza en un nodo raíz y termina en las hojas, dividiendo los datos en cada nodo según un atributo y un umbral de separación elegidos al azar. Cada árbol sigue creciendo hasta que cada dato se aísla en una hoja. La profundidad de aislamiento, es decir, la cantidad de aristas que atraviesa una instancia desde la raíz del árbol binario hasta una hoja, se utiliza como medida de la longitud del camino. Las instancias anómalas tienen una longitud de camino significativamente más corta que las instancias normales en un conjunto de árboles binarios aleatorios, debido a que se aíslan rápidamente en las hojas, mientras que las instancias normales necesitan más particiones para llegar a las hojas. La puntuación de anomalía se puede calcular mediante la profundidad de aislamiento promedio en los árboles binarios. Un iForest es un conjunto de árboles de aislamiento utilizados en la detección de anomalías [41].

1.3.4 Deep Learning.

Es un subconjunto del aprendizaje automático. En ambos casos, los algoritmos parecen aprender analizando grandes cantidades de datos (sin embargo, el aprendizaje puede ocurrir incluso con pequeños conjuntos de datos en algunos casos). Pero, el aprendizaje profundo varía en la profundidad de su análisis y el tipo de automatización que proporciona. El aprendizaje profundo, usa una técnica que imita la funcionalidad del cerebro humano. Procesa datos utilizando unidades informáticas, llamadas neuronas, dispuestas en secciones ordenadas, llamadas capas. La técnica en la base del aprendizaje profundo es la red neuronal [42].

1.3.5 Redes neuronales.

Dentro del cerebro humano hay más de 100 millones de redes neuronales en existencia, las cuales permiten aprender, recordar y procesar datos.

En la opinión de Hassoun [43], es un tipo de modelo computacional que opera paralelamente y está formado por unidades de procesamiento adaptativas altamente interconectadas entre sí.

Según Matich [44], se conocen como sistemas interconectados paralelamente y compuestos por elementos simples, a menudo adaptativos, organizados jerárquicamente. Su objetivo es interactuar con objetos del mundo físico y real de manera similar al sistema nervioso biológico.
Para Chen & Chang [45], se trata de modelos matemáticos diseñados para imitar el funcionamiento del cerebro humano.
Mientras que Kung [46], la describe como un sistema que combina una red adaptativa con técnicas de procesamiento paralelo de información.

Resumiendo, las redes neuronales son modelos computacionales matemáticos que están interconectados entre sí y buscan imitar y basarse en el funcionamiento del cerebro humano. Algunas de las redes neuronales utilizadas en predicción de energía eléctrica se detallan en la Tabla 2.

Tabla 5. *Redes Neuronales utilizadas en la predicción de energía eléctrica*

Año	Desarrollo
CNN (Redes Neuronales Convolucionales)	Las CNN son ampliamente utilizadas en la predicción de carga de energía eléctrica, especialmente cuando se trata de datos de series temporales, como el consumo de energía eléctrica. Pueden extraer características significativas y patrones complejos de los datos, lo que permite una mejor predicción [47].

ANN (Redes Neuronales Artificiales)	Las ANN han sido aplicadas en la predicción de carga de energía eléctrica utilizando datos históricos de consumo y factores meteorológicos. Estas redes neuronales pueden aprender patrones y relaciones no lineales en los datos, lo que las hace adecuadas para la predicción de la demanda eléctrica [48].
LSTM (Memoria de Corto y Largo Plazo)	Las LSTM son ampliamente utilizadas para la predicción de series de tiempo, incluyendo la predicción de carga de energía eléctrica. Estas redes neuronales recurrentes pueden capturar relaciones a largo plazo en los datos y manejar secuencias de tiempo variables [49].

Fuente: Elaboración Propia

Modelos neuronales

Las neuronas artificiales funcionan de la misma manera, en que lo haría una neurona biológica como lo muestra la Ilustración 8, se encuentra compuesta por:

Entradas ponderadas: establecen conexiones sinápticas entre las unidades de procesamiento, y la fuerza o eficacia de la sinapsis se representa mediante el peso de la conexión. La presencia de estas conexiones establece si una unidad puede influir en otra, y el valor y signo de los pesos determinan el tipo (excitatorio o inhibitorio) y la intensidad de la influencia entre las unidades.

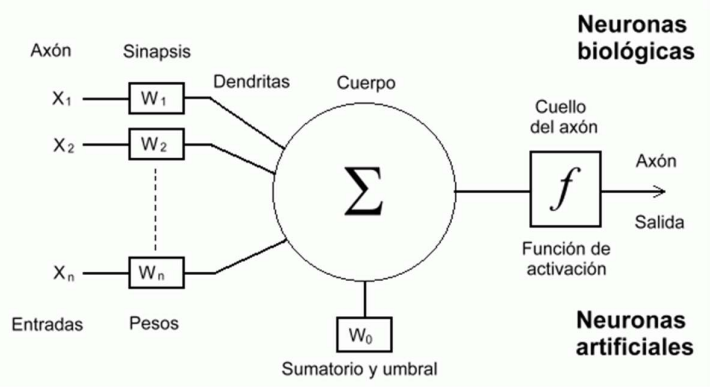
Función de propagación: Se encarga de sumar ponderadamente todas las entradas recibidas por la unidad, utilizando los valores de peso de las conexiones. De esta manera, se obtiene el valor total de entrada a la unidad, que representa la combinación de señales excitatorias e inhibitorias de las neuronas biológicas.

Función de activación: La característica más destacada y distintiva de las neuronas es posiblemente la que mejor define su comportamiento. Se emplean diversas funciones, que

pueden ser desde simples umbrales a funciones no lineales, para determinar el nivel o estado de activación de una neurona es a través del cálculo de su entrada total.

Salida: Se encarga de calcular la respuesta de la neurona a partir de su activación y suele aplicarse una función identidad para obtener la salida. En la mayoría de los casos, la salida se toma directamente como el valor de activación. En la biología, el valor de salida equivaldría a la tasa de disparo de la neurona [50].

Ilustración 8. *Modelo de neurona artificial*



Fuente: P. Ganguly, A. Kalam y A. Zayegh (2017)

En una red neuronal de una sola capa, el modelo más sencillo está compuesto por una capa de entrada y una capa de salida que también funciona como capa de procesamiento, ya que incorpora una función de activación [51]. Las conexiones se establecen únicamente entre las neuronas de la misma capa, formando así una red. Se emplean con frecuencia para llevar a cabo tareas de integración, las cuales consisten en recuperar información que ha sido ingresada a la red de manera incompleta o distorsionada [52].

Las redes neuronales multicapa se caracterizan por tener múltiples niveles de procesamiento de información, cada uno compuesto por un conjunto de neuronas que reciben entradas de la capa anterior y emiten salidas hacia la capa siguiente. Estas redes pueden tener dos, tres, cuatro o más capas, y cada una de ellas cumple un papel importante en la transformación de la información de entrada en una salida útil [53]. Las redes neuronales feedforward son aquellas en las que las neuronas solo envían señales de salida hacia una capa posterior más cercana a la salida de la red, sin tener conexiones laterales

ni hacia atrás. Esta estructura es ampliamente utilizada en diversas aplicaciones como el modelamiento y pronóstico de sistemas, entre otras [54].

1.3.6 Fundamentos de electricidad

Factor de Potencia:

El factor de potencia (FP) es una medida de la eficiencia con la que se utiliza la energía eléctrica en un sistema. Indica la relación entre la potencia activa y la potencia aparente de un dispositivo o sistema eléctrico. Se calcula como el coseno del ángulo de fase entre la corriente y el voltaje en un circuito de corriente alterna. Un factor de potencia cercano a 1 indica un uso eficiente de la energía, mientras que un factor de potencia bajo indica una ineficiencia energética.

Corriente:

La corriente eléctrica es el flujo de carga eléctrica a través de un conductor. Se mide en amperios (A) y se representa por la letra "I". La corriente puede ser continua (CC) o alterna (CA), siendo esta última la forma más común de corriente en aplicaciones de energía eléctrica. La corriente alterna cambia de dirección periódicamente, generando un flujo de carga oscilante.

Voltaje:

El voltaje, también conocido como diferencia de potencial, es la fuerza que impulsa el flujo de corriente eléctrica en un circuito. Se mide en voltios (V) y se representa por la letra "V". El voltaje puede ser constante en corriente continua (CC) o cambiar de polaridad y magnitud en corriente alterna (CA). Es una medida de la energía eléctrica potencial que puede ser utilizada por los dispositivos conectados a un circuito.

1.3.7 PZEM-004T

El PZEM-004T es un módulo de medición de energía eléctrica que se utiliza para medir varios parámetros eléctricos, como voltaje, corriente, potencia activa, consumo eléctrico, frecuencia y factor de potencia. Es comúnmente utilizado en proyectos de monitoreo de energía eléctrica y en sistemas de control de automatización de edificios, entre otros. El PZEM-004T es fácil de usar, preciso y tiene una interfaz de comunicación serial que permite su integración con diferentes plataformas electrónicas como Arduino y otros microcontroladores.

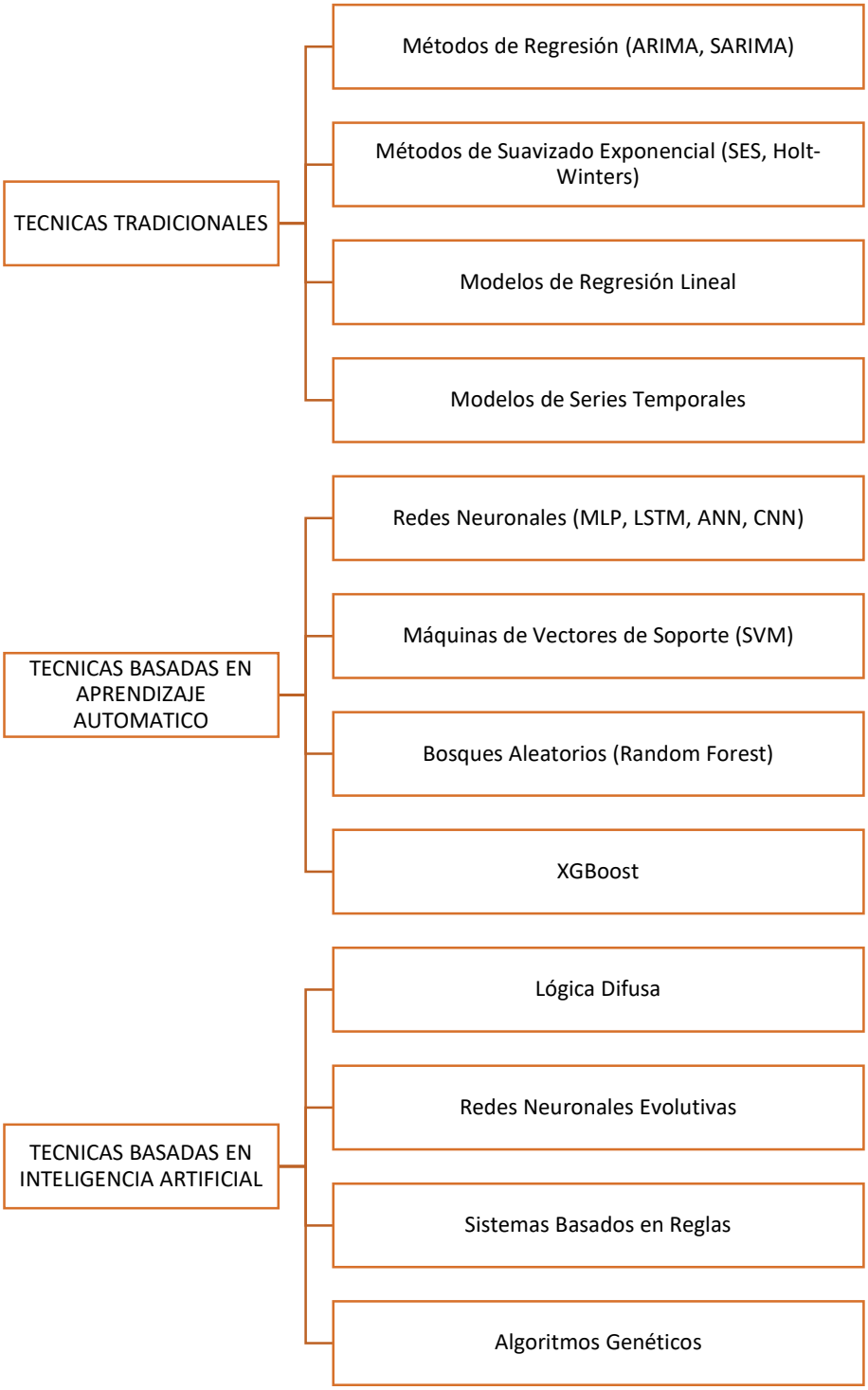
1.3.8 Fundamentación teórica de la variable dependiente

Predicción de carga eléctrica

La estimación de la carga energética es de gran importancia dentro de la industria de la electricidad, mediante su uso se puede conocer cuál será su demanda a futuro. Regularmente para su cálculo se deben valorar los datos históricos y el rendimiento de carga. Además, con la predicción de carga es posible planificar, analizar, tomar decisiones o programar mantenimientos.

Técnicas de pronóstico

Ilustración 9. *Técnicas de pronóstico de energía eléctrica*



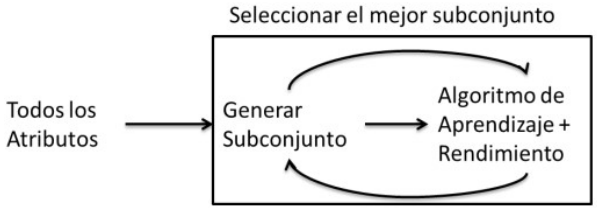
Fuente: Elaboración Propia

Metodología para selección de variables

Sistemas embebidos

Los sistemas embebidos representan una evolución de los métodos wrapper, los cuales buscan identificar el conjunto más adecuado de propiedades que cumplan con los objetivos de rendimiento del algoritmo utilizado en el desarrollo del proyecto. Dos objetivos comunes en los métodos wrapper de selección de características son:
La minimización del error del algoritmo utilizado (maximización del desempeño) disminución del conjunto de atributos que se incluyen en el modelo mediante la selección de los más relevantes para la predicción.
Se debe buscar siempre el primer objetivo, que es mermar el error del algoritmo y maximizar su trabajo, mientras que el segundo objetivo es tratar de evitar incluir un número excesivo de atributos la excesiva complejidad del modelo, así como cumplir el objetivo de alcanzar la generalidad del método obtenido (esto también puede afectar a la interpretación de los resultados). Este método evalúa el rendimiento final del algoritmo de aprendizaje para lograr su objetivo final (por ejemplo, medir el error de predicción) [55].

Ilustración 10. *Método embebido*



Fuente: A. Palma Llewellyn (2015)

CAPÍTULO 2. MATERIALES Y MÉTODOS

2.1 Tipo de estudio o investigación realizada.

Para calcular el consumo eléctrico (la variación de consumo) se establece un modelo **experimental**, por medio de este, se pretende observar las variables de investigación, es decir, si la selección de un adecuado método de predicción basado en machine learning incrementa la precisión de consumo de la energía eléctrica como se muestra en la Tabla 3.

Tabla 6. *Fases de experimentación del proyecto*

Fase de experimentación	Descripción
Alta precisión	El método uno, dos o tres es el que se acerca más al factor de potencia (consumo) que envía el medidor eléctrico a través del módulo ESP32
Baja precisión	El método uno, dos o tres es el que menos se acerca al factor de potencia (consumo) que envía el medidor eléctrico a través del módulo ESP32

Fuente: Elaboración Propia

2.2 Paradigma o enfoque.

Se ha decidido utilizar un enfoque cuantitativo para esta investigación, ya que se busca recolectar información numérica para poner a prueba la hipótesis, y se realizará un análisis estadístico para identificar patrones de comportamiento y validar la teoría. Este enfoque permitió seguir un proceso estructurado que comenzó con una idea general, derivó en preguntas de investigación, revisión de literatura y construcción del marco teórico. Luego se estableció la hipótesis y las variables. Este proceso se puede observar en la ilustración. 11.

Ilustración 11. *Enfoque cuantitativo y los pasos que contiene*

Fuente: Elaboración Propia

2.3 Población y muestra.

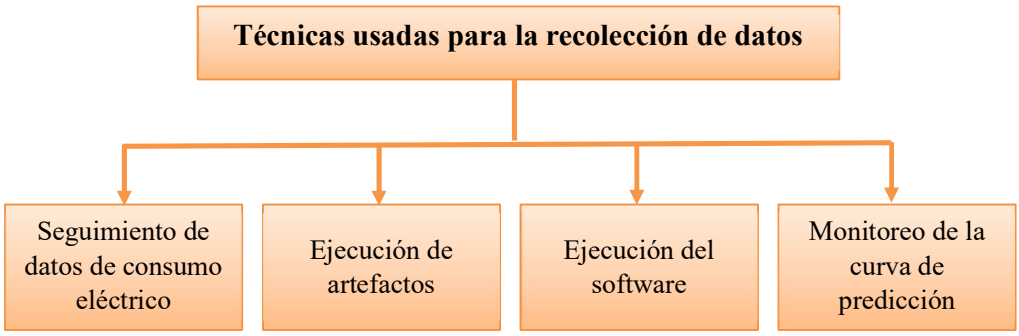
El análisis se realizará en una población de datos recopilados por el medidor inteligente Pzem-004t, que mide variables como voltaje, corriente, frecuencia, factor de frecuencia y consumo. Estos datos se han implementado en tarjetas electrónicas ESP32.

Al ser una población que aumenta con el tiempo, es decir, los datos enviados por el medidor inteligente son constantes, se ha determinado que no se requiere deducir el volumen de la población o la muestra para desarrollar el software de predicción de carga de energía eléctrica basado en técnicas de aprendizaje automático. En lugar de eso, se utilizó un conjunto de datos que consiste en 152000 registros para entrenar las redes neuronales ANN LSTM y CNN y establecer el modelo predictivo.

2.4 Métodos teóricos con los materiales utilizados.

Dado el tema abordado en la propuesta de investigación, se utilizarán técnicas de captura y evaluación de información apropiadas para dispositivos electrónicos, las cuales se describen en la Ilustración 12.

Ilustración 12. *Ecuación de cálculo de la muestra*



Fuente: Elaboración Propia

El medidor inteligente (Pzem 004) recopila los datos de variación de corriente, para luego transmitirlos a través del puerto serial al módulo ESP32, el cual funcionará como un servidor web (gateway) para que este a su vez los transmita por medio del método POST a la aplicación web, registrando la información en la base de datos. Para posteriormente analizarlos mediante redes neuronales y presentar la predicción en el Dashboard.

2.5 Métodos empíricos con los materiales utilizados.

2.5.1 Experimento

Para la implementación de los escenarios de experimentación detallados en la Tabla 4, se ha implementado una infraestructura circuital y de aprendizaje basada en machine learning. La infraestructura impulsada por técnicas de machine learning será responsable de examinar los diferentes escenarios de respuesta al manipular la variable independiente para evaluar el comportamiento de la variable dependiente.

2.6 Técnicas estadísticas para el procesamiento de los datos obtenidos.

Para la predicción, se pueden utilizar varias técnicas de procesamiento de datos, incluyendo técnicas estadísticas y modelos de aprendizaje automático como redes neuronales.

El análisis de tendencias, el análisis de regresión y el análisis de series temporales se pueden utilizar para predecir los valores futuros. Estas técnicas son relativamente simples y se pueden implementar fácilmente en una variedad de entornos, pero pueden tener limitaciones en términos de precisión y capacidad de manejar datos no lineales y no estacionarios.

Redes neuronales artificiales (ANN): Las ANN son modelos de aprendizaje automático que se basan en la estructura de una red neuronal biológica. tienen la capacidad de ser altamente adaptables y de procesar diferentes tipos de datos, pero pueden ser difíciles de ajustar y pueden requerir grandes conjuntos de datos para un buen rendimiento.

Redes neuronales convolucionales (CNN): Las CNN son ampliamente empleadas en el procesamiento de imágenes y se fundamentan en capas de convolución para identificar características de la imagen. Las CNN pueden manipular grandes cantidades de datos y pueden tener un buen rendimiento en la predicción de series de tiempo, pero pueden requerir un gran esfuerzo de ingeniería para implementar y ajustar correctamente.

Redes neuronales de memoria a corto plazo (LSTM): Las LSTMs son una variante de las redes neuronales recurrentes que están diseñadas para manejar secuencias de tiempo y tienen una estructura de memoria interna para recordar estados anteriores. Las LSTMs pueden manejar datos no lineales y no estacionarios y pueden tener un buen rendimiento en la predicción de series de tiempo, pero pueden requerir grandes conjuntos de datos para un buen ajuste.

2.7 Selección de la metodología:

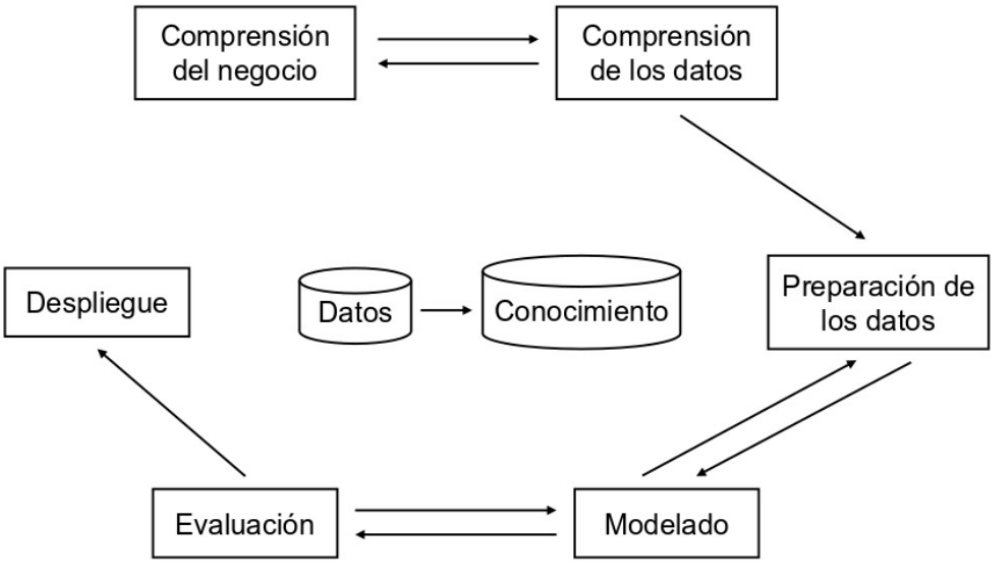
La metodología seleccionada para el desarrollo de esta tesis se basó en la propuesta de Piatetsky, quien realizó una evaluación comparativa de diferentes metodologías para proyectos relacionados con análisis o minería de datos, concluyendo que CRISP-DM es el método que más se utiliza en los proyectos de grandes volúmenes de datos [56].

2.7.1 Metodología CRISP-DM

La metodología CRISP-DM (Cross Industry Standard Process for Data Mining) es ampliamente reconocida como la metodología estándar para proyectos de minería de datos [57], y es igualmente relevante en el contexto del machine learning.

Otra característica importante, es que diseña bajo un tipo de metodología neutral en relación con el instrumento a trabajar, en este caso, minería de datos [40], cuya finalidad consiste en identificar las variables principales de las empresas comerciales, tales como los detallados en la Ilustración 17.

Ilustración 13. *Fases del modelo de referencia CRISP-DM*



Fuente: Elaboración Propia

CAPÍTULO 3. RESULTADOS OBTENIDOS

3.1 Aplicación de la metodología CRISP-DM

FASE1: Comprensión del negocio:

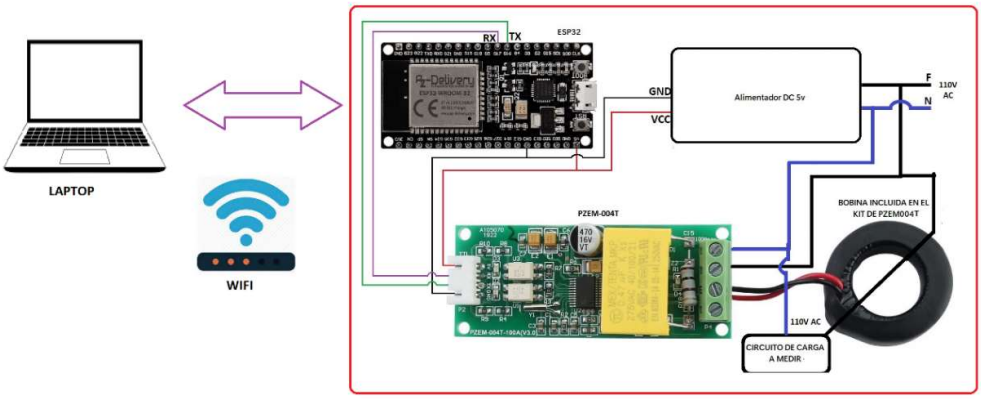
Esta fase del proyecto es crucial, ya que implica establecer los objetivos del software, evaluar la situación actual y planificar el proyecto de predicción de energía.

a) *Establecer el objetivo del negocio*: Producir información valiosa que ayude a las compañías o hogares a tomar decisiones informadas

b) *Evaluación de la situación:* en esta tarea y teniendo en cuenta que se desarrollarán dos artefactos como parte de la investigación, siendo el primero el software de predicción basado en machine learning (Gateway). El segundo es el firmware del módulo ESP32 que actuará como servidor web, se analizó los recursos Hardware, Software, como se muestra en las Tablas 4 y 5.

   Para la construcción de un prototipo electrónico se utilizará el medidor inteligente (Pzem-004t) y el módulo (ESP32), mismo que realizará el levantamiento de los datos para posteriormente enviarlos a la base de datos y presentar los resultados a través de la aplicación web tal como se muestra en la Ilustración 16.

Ilustración 14. *Prototipo electrónico*



Fuente: Elaboración Propia

Tabla 7. *Recursos Hardware*

Equipo	Modelo	Procesador	RAM	Disco Duro	Sistema Operativo
LAPTOP	Lenovo Ideapad3	Core I5 10234	20GB	512GB	WINDOWS 10
PZEM-004T	-	-	-	-	-
ESP32 WIFI	-	-	-	-	-

Fuente: Elaboración Propia

En la Ilustración 15 se plantea el uso de herramientas software, que ayuden a cumplir el objetivo determinado. El uso de librerías admitidas en el lenguaje Python. Las mencionadas técnicas fueron empleadas para llevar a cabo el análisis de los datos a través de redes neuronales en la web para la presentación de los datos emitidos a través del circuito y del análisis realizado por la red neuronal LSTM, CNN o ANN, base de datos mysql para el almacenamiento de los datos. Módulo ESP32 y medidor inteligente Pzen004T para la obtención de datos.

Ilustración 15. *Herramientas usadas para la investigación*

Fuente: Elaboración Propia

Tabla 8. *Recursos Software*

SOFTWARE	UTILIDAD
IDE PARA ARDUINO	Esp32 recolección de los datos
REDIS	Calidad de los datos
MYSQL	Calidad de los datos
NODE	Calidad de los datos
PYTHON	Evaluación de las redes neuronales

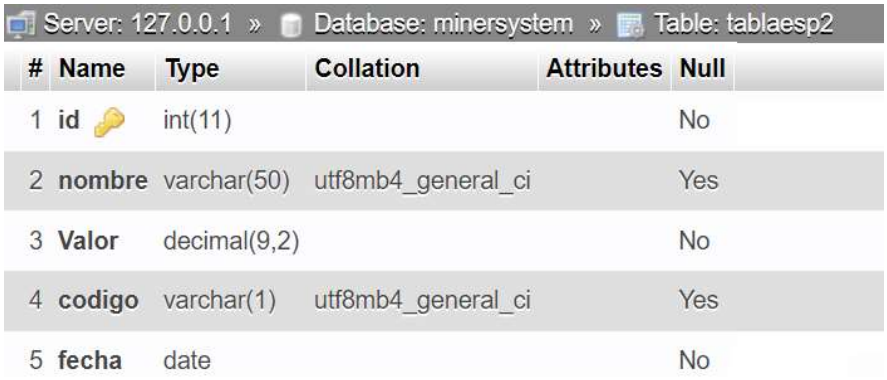
Fuente: Elaboración Propia

a) Determinación de los objetivos de la predicción:

Los objetivos que se establecieron para el análisis son:

- Establecer qué particularidades tienen los datos para calcular la predicción de la carga de energía como muestra la Ilustración 16.

Ilustración 16. *Características de los datos de la base de datos mysql*



#	Name	Type	Collation	Attributes	Null
1	id	int(11)			No
2	nombre	varchar(50)	utf8mb4_general_ci		Yes
3	Valor	decimal(9,2)			No
4	codigo	varchar(1)	utf8mb4_general_ci		Yes
5	fecha	date			No

Fuente: Elaboración Propia

- Establecer los algoritmos de redes neuronales a utilizar.

  Para el desarrollo de este tipo de proyecto se establecen las redes neuronales ANN, LSTM y CNN.

- Establecer la cantidad de datos que se utilizarán para el entrenamiento de las redes neuronales.
  De acuerdo a una fecha definida se ha definido un total de 152000 registros.

- Establecer la cantidad de datos que se utilizarán para calcular el error.
  80% datos para entrenamiento
  20% datos para métricas de error
- Identificar métricas que se utilizarán para evaluar los modelos de predicción. metrics=['r2','mape']

FASE 2: Comprensión de los datos

En esta fase se lleva a cabo la obtención preliminar de datos, la descripción de los mismos, la exploración de los datos y la verificación de su calidad. A continuación, se detallan estas actividades:

a) *Recolección de datos iniciales:*
   Los sistemas de adquisición de datos pueden adaptarse a sistemas versátiles, flexibles, que se encarga de digitalizar las señales analógicas para ser utilizadas por los sistemas de medición y ordenadores [58]. Se utiliza para adquirir, procesar y registrar datos de diferentes sensores o fuentes de señales para acondicionarlas.

   La recolección se realizará desde el módulo PZEM004T a través del módulo ESP32 que contiene:
   - Voltaje (V)
   - Corriente (A)
   - Potencia Activa (W)
   - Consumo eléctrico (kWh)
   - Frecuencia (Hz)
   - Factor de Potencia (pf)

b) *Descripción de los datos* Los datos recopilados se dividen en varias tablas como son tabla_esp2, tabla_red1, tabla_red2, tabla_red3, mejorresultado, tal como se detalla en la Ilustración 19.

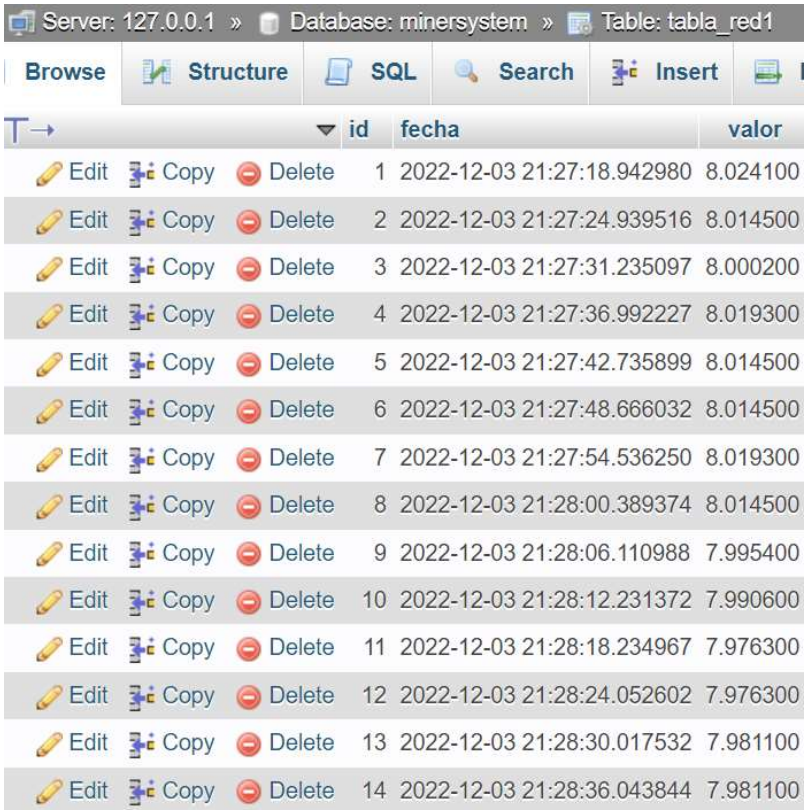
Ilustración 17. Tablas de la base de datos minersystem

Fuente: Elaboración Propia

c) Exploración de los datos

En esta etapa se verifican los datos a través de una breve exploración como la que se muestra en la Ilustración 18.

Ilustración 18. Visualización de los datos de la potencia activa almacenados en la tabla_red1 de la base de datos.



Fuente: Elaboración Propia

d) Verificación de la calidad de los datos: se verificó a través del uso del multímetro que los datos sean reales y de acuerdo a los datos del sensor, esto permitió detectar los siguientes conflictos:
   - Datos con valores en 0 (0, null, etc)
   - Errores de conexión de los pines del módulo ESP32
   - Fechas de toma de la medición.

Una vez realizada la exploración y verificación de los datos se procede a la siguiente fase.

FASE 3: Preparación de los datos

a) La etapa de preparación del conjunto de datos es crucial y, a menudo, la más tardada dentro del proceso. Es poco común que los datos estén en la forma necesaria para los algoritmos de machine learning. Por lo tanto, es necesario realizar un proceso de selección, limpieza, estructuración, integración y formato para adecuar los datos a las necesidades de los algoritmos.

b) Selección de los datos: se identificaron los atributos que no eran relevantes para los objetivos del proyecto establecidos en la fase de comprensión del negocio de la metodología. La eliminación de estos atributos permitió simplificar el modelo y lograr una explicación más clara. Se aplicaron filtros para seleccionar 152000 registros de la base de datos.

c) Limpieza de los datos: este proceso involucró la verificación de los datos que no coincidían con la medición real con el multímetro. Además, se utilizó únicamente los registros que tenían el código 0 y 1 de la tablaesp2.

d) Estructura de los datos: los datos se estructuraron desde el momento que fueron recibidos desde el módulo ESP32 como se muestra en la Ilustración 19 y 20.

Ilustración 19. Datos recibidos desde el módulo ESP32

```
print("*********datos arduino********")
print(request.POST)
voltaje = float(request.POST["1"])
corriente = float(request.POST["2"])
valores_a_predecir = np.array([[voltaje, corriente]])
Tablaesp2.objects.create(codigo="1", nombre="voltaje", Valor=request.POST['1'])
Tablaesp2.objects.create(codigo="2", nombre="corriente", Valor=request.POST['2'])
Tablaesp2.objects.create(codigo="3", nombre="potencia", Valor=request.POST['3'])
Tablaesp2.objects.create(codigo="4", nombre="consumo", Valor=request.POST['4'])
Tablaesp2.objects.create(codigo="5", nombre="frecuencia", Valor=request.POST['5'])
Tablaesp2.objects.create(codigo="6", nombre="fp", Valor=request.POST['6'])
```

Fuente: Elaboración Propia

En la Ilustración 19 se muestran los valores recibidos desde PZEM004T A través del módulo ESP32 respecto al voltaje y corriente que son la base para el cálculo de la potencia.

Ilustración 20. Filtro de datos de la tablaesp2

```
queryset = Tablaesp2.objects.filter(codigo='2')[:50]
serializer_class = Tablaesp2Serializer
```

Fuente: Elaboración Propia

e) Integración de los datos: Se procedió a fusionar los datos provenientes de diferentes fuentes en una sola fuente para su posterior análisis. Este proceso de integración de datos es fundamental para poder obtener una visión completa y coherente de los datos entre el módulo ESP32 y REDIS y poder entrenar las redes neuronales con el lenguaje de programación Python.

FASE 4: Modelado:

Esta fase involucra la elección de las técnicas de modelado adecuadas, la definición de un plan de prueba utilizando métodos de validación cruzada, la construcción y evaluación del modelo en el lenguaje de programación Python. Las siguientes son las tareas detalladas de esta etapa:

a) Selección de la técnica de modelado: las técnicas que mejor se adecúan a nuestros objetivos son las redes neuronales artificiales, las LSTM puesto que se utilizan comúnmente en aplicaciones que involucran datos secuenciales, las CNN pueden ser más efectivas en el tratamiento de datos. Las ANN tienen una arquitectura más simple y se pueden utilizar para resolver problemas de series de tiempo.

   Generación del plan de prueba: se realizó con un total de 1000 épocas tal como muestra la Ilustración 23 para dar mayor precisión al resultado. Después de completar las iteraciones, se procede a calcular la precisión y el error correspondientes para cada red neuronal.

Ilustración 21. *Generación de épocas en python*

```
model.add(Dense(16, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(training_data, target_data, epochs=1000)
scores = model.evaluate(training_data, target_data)
```

Fuente: Elaboración Propia

b) Construcción del modelo:

   Tal como muestra la Ilustración 24, aquí se ejecuta la creación de los modelos de cada una de las redes neuronales que serán entrenadas.

Ilustración 22. *Modelo de red neuronal LSTM*

```
json_file = open("static/modelos/model_tf.json", 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# cargar pesos al nuevo modelo
loaded_model.load_weights("static/modelos/model_tf.h5")
```

Fuente: Elaboración Propia

Ilustración 23. *Modelo de red neuronal CNN*

```
json_file_kr = open("static/modelos/model_keras.json", 'r')
loaded_model__kr_json = json_file_kr.read()
json_file_kr.close()
loaded_model_kr = model_from_json(loaded_model__kr_json)
loaded_model_kr.load_weights("static/modelos/model_keras.h5")
valores_a_predecir = np.array([[voltaje / 130, corriente / 10]])
```

Fuente: Elaboración Propia

Ilustración 24. *Modelo de red neuronal ANN*

```
var network = new brain.NeuralNetwork();
```

Fuente: Elaboración Propia

c) Evaluación del modelo: para efectuar el objetivo respecto a la confiabilidad del modelo se utilizó librerías de Python.

Ilustración 25. *Entrenamiento de red LSTM*

```
network.train([
    {input: {voltaje: 1, corriente: 1}, output: {potencia: 1}},
    {input: {voltaje: 0.9262, corriente: 0.014}, output: {potencia: 0.01297}},
    {input: {voltaje: 0.8982, corriente: 0.009}, output: {potencia: 0.008084}},
    {input: {voltaje: 0.9246, corriente: 0.014}, output: {potencia: 0.01294}},
    {input: {voltaje: 0.9, corriente: 0.008}, output: {potencia: 0.0072}},
    {input: {voltaje: 0.8990, corriente: 0.006}, output: {potencia: 0.005394}},
    {input: {voltaje: 1, corriente: 0}, output: {potencia: 0}},
    {input: {voltaje: 1, corriente: 0.5}, output: {potencia: 0.5}},
]);
// What is the expected output of [1,0]?
let result = network.run({voltaje: 0.92384615, corriente: 0.002});
```

Fuente: Elaboración Propia

En la Ilustración 25 se muestra los valores para el entrenamiento de la red LSTM enviándole valores de voltaje y corriente y valores de potencia medidos con el multímetro de manera manual tal como se muestra en el Anexo 1.

Ilustración 26. *Entrenamiento de red CNN*

```
training_data = np.array(
    [[0.9262, 0.014], [0.8982, 0.009], [0.9246, 0.014], [0.8990, 0.006], [0.8615, 0.002], [0.8633, 0.043]],
    "float32")
# y estos son los resultados que se obtienen, en el mismo orden
target_data = np.array([[0.01297], [0.008084], [0.01294], [0.005394], [0.005169], [0.3712]], "float32")
# cargamos las combinaciones posibles
model = Sequential()
model.add(Dense(16, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(training_data, target_data, epochs=1000)
```

Fuente: Elaboración Propia

En la ilustración 26 se muestra el entrenamiento de la red CNN con los valores medidos manualmente con un multímetro. Y la función de activación de la red.

Ilustración 27. *Entrenamiento de red ANN*

```
training_data = np.array(
    [[0.9262, 0.014], [0.8982, 0.009], [0.9246, 0.014], [0.8990, 0.006], [0.8615, 0.002], [0.8633, 0.043]],
    "float32")
# y estos son los resultados que se obtienen, en el mismo orden
target_data = np.array([[0.01297], [0.008084], [0.01294], [0.005394], [0.005169], [0.3712]], "float32")
```

Fuente: Elaboración Propia

En la ilustración 27 se muestran los valores de corriente y voltaje enviados a la red ANN para su entrenamiento.

FASE 5: Evaluación

En esta etapa se evaluaron los resultados de los modelos teniendo en cuenta el requerimiento de los objetivos, se procede a detallar cada una de las tareas realizadas:

a) Evaluación de los resultados

Cada una de las redes neuronales fueron entrenadas con los mismos valores de corriente y voltaje para calcular el valor de la potencia. Los resultados de las redes neuronales mostraron los resultados que más se aproximan a la potencia real el cual mostró el dato de la red neuronal que menos error produjo.

b) Evaluación de Modelos

Para verificar la confiabilidad del modelo se utilizó la métrica R2, porque expresa el porcentaje que ha sido correctamente clasificado, error de porcentaje medio absoluto (MAPE) que calcula la diferencia porcentual entre las predicciones del modelo y los valores reales.

Ilustración 28. *Ejecución de métricas de error MAPE en python*

```
loaded_model_kr.compile(loss='mean_squared_error', optimizer='adam', metrics=['binary_accuracy'])
valor2 = loaded_model_kr.predict(valores_a_predecir)[0][0]
```

Fuente: Elaboración Propia

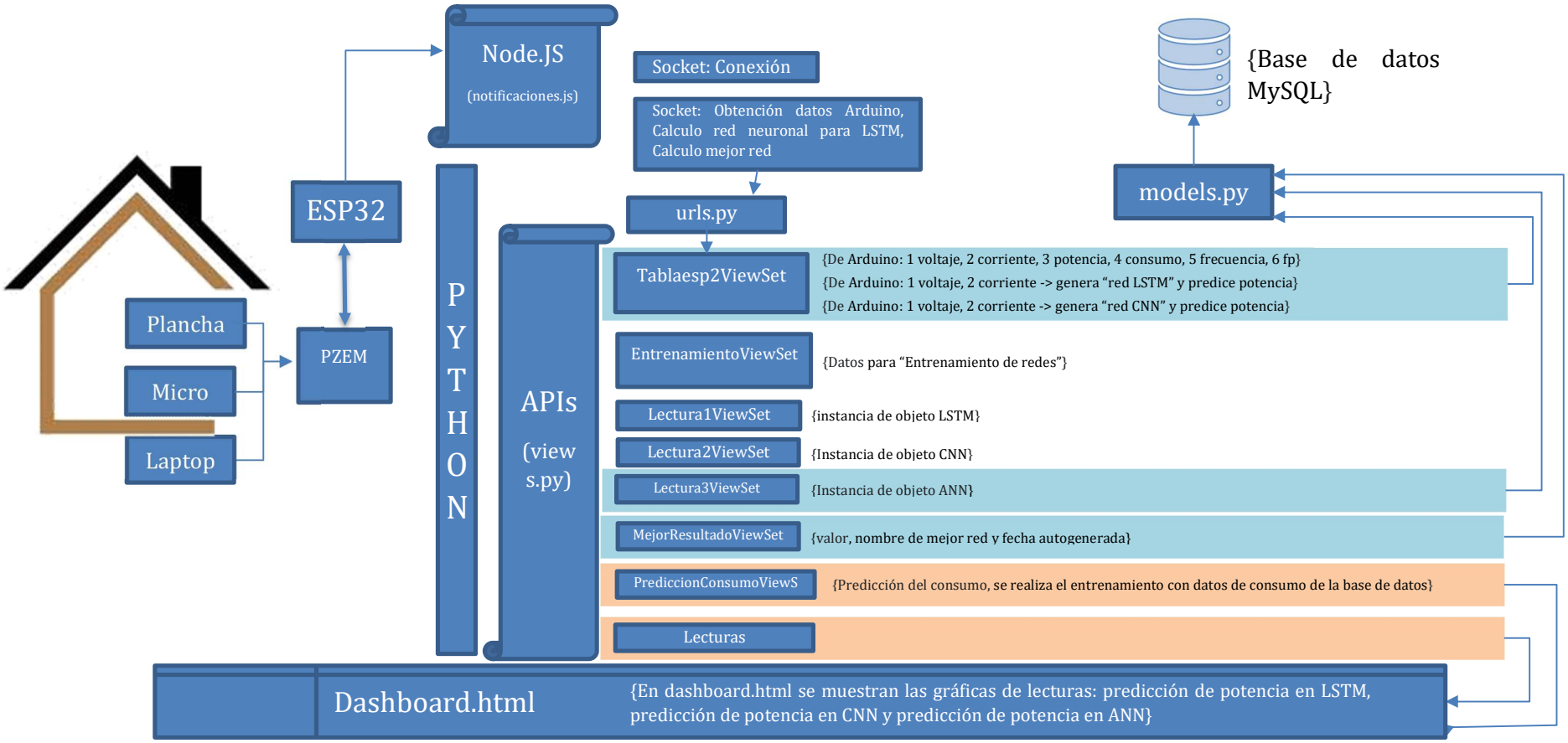
c) Determinación de futuras fases

Se presentará los resultados alcanzados de acuerdo con los objetivos planteados y técnicas de Machine Learning.

FASE 6: Despliegue o Implementación

El objetivo en la última fase de la metodología CRISP-DM es el de implementar las redes neuronales para la predicción de carga de energía mismos que se mostrarán en el capítulo IV Resultados y Discusión. A continuación, en la Ilustración 29 se muestra la arquitectura utilizada en esta implementación.

Ilustración 29. Arquitectura utilizada



Fuente: Elaboración Propia

3.2 Evaluación del software

En esta etapa se muestran los resultados alcanzados después del desarrollo y ejecución del software de predicción que hace uso de redes neuronales y las métricas de evaluación descritas previamente. Los resultados son analizados e interpretados en base a las técnicas de machine Learning planteados en capítulos anteriores con la finalidad de establecer y proponer la que generó mejores resultados.

El proceso para obtener la predicción de la carga de energía se detalla a continuación:

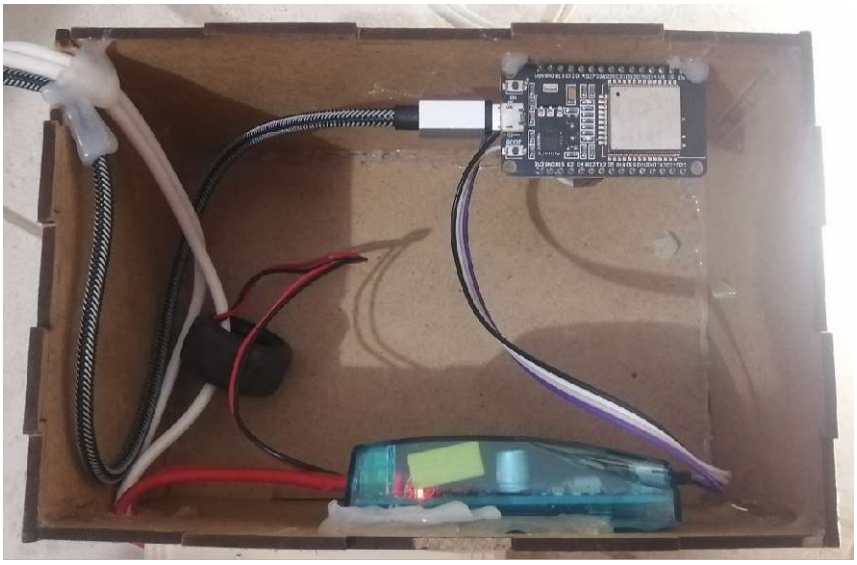
Pasos a seguir antes de la utilización del software:

- Se debe colocar el prototipo en un lugar estratégico cerca del equipo eléctrico seleccionado para poder conectar a la fuente de energía (110/220V), además, se debe conectar el módulo ESP32 al tomacorriente USB (5V) y luego conectar el o los equipos al tomacorriente del prototipo tal como se observa en la Ilustración 30 o en su defecto hacer uso de la bobina en una línea de energía AC de un medidor (110/220V) tal como se puede visualizar en la Ilustración 31.

Ilustración 30. *Conexión correcta del Prototipo a la fuente de energía*

Fuente: Elaboración Propia

Ilustración 31. Conexión del módulo PZEM004T y ESP32



Fuente: Elaboración Propia

- Conectar a través de librerías de Arduino al prototipo con la red wifi para adquirir los datos del PZEM004T como se detalla en el siguiente código:

Ilustración 32. Código de Arduino para el módulo ESP32

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <PZEM004Tv30.h>
#if defined(ESP32)
PZEM004Tv30 pzem(Serial2, 16, 17);
#else
PZEM004Tv30 pzem(Serial2);
#endif

HTTPClient http;
int httpCode;

const char* ssid = "PAKINET - GoNet";///Nombre del Wifi
const char* password = "@isth_1981.";///Contraseña Wifi @isth_1981.

String stringLed, stringSensor, stringEnviar, stringEnviar2;
int estadoLed= 1;

void setup() {
  stringLed = String("status=");
  stringSensor = String("sensor=");
  stringEnviar = String();
  stringEnviar2 = String();
```

```
  Serial.begin(115200);
  pinMode(13,OUTPUT);
  digitalWrite(13,HIGH);

  WiFi.begin(ssid,password);
  while(WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println(WiFi.localIP());


'oid loop() {
  Serial.print("Direccion del pzem:");
  Serial.println(pzem.readAddress(), HEX);
  float voltage = pzem.voltage();
  float current = pzem.current();
  float power = pzem.power();
  float energy = pzem.energy();
  float frequency = pzem.frequency();
  float pf = pzem.pf();
  if(isnan(voltage)){


    Serial.println("Error de lectura del factor de potencia");
  } else {
    Serial.print("Voltaje: ");   Serial.print(voltage);   Serial.println("V");
    Serial.print("Corriente: ");   Serial.print(current);   Serial.println("A");
    Serial.print("Potencia: ");     Serial.print(power);     Serial.println("W");
    Serial.print("Consumo Electrico: "); Serial.print(energy,3);  Serial.println("kWh");
    Serial.print("Frecuencia: ");   Serial.print(frequency, 1); Serial.println("Hz");
    Serial.print("PF: ");        Serial.println(pf);
  }
Serial.println();

if(WiFi.status() == WL_CONNECTED)
{
  Serial.println("conectado");
  WiFiClient client;
  http.begin(client, "http://192.168.0.109:8000/api/lectura/"); // ip servidor
  http.addHeader("Content-Type","application/x-www-form-urlencoded");
  stringEnviar = String("1=")+(voltage)+("&")+String("2=")+(current)+("&")+String("3=")+(power)+
  ("&")+String("4=")+(energy)+("&")+String("5=")+(frequency)+("&")+String("6=")+(pf);
  httpCode = http.POST(stringEnviar);
  http.end();
  Serial.print("envioo peticion");
  delay(5000);
}
delay(100);
```

Fuente: Elaboración Propia

- En la Ilustración 33 se muestran las líneas de conexión a la API que enviará los datos (voltaje, corriente, potencia, factor de potencia, frecuencia, energía) desde el ESP32 hasta la computadora.

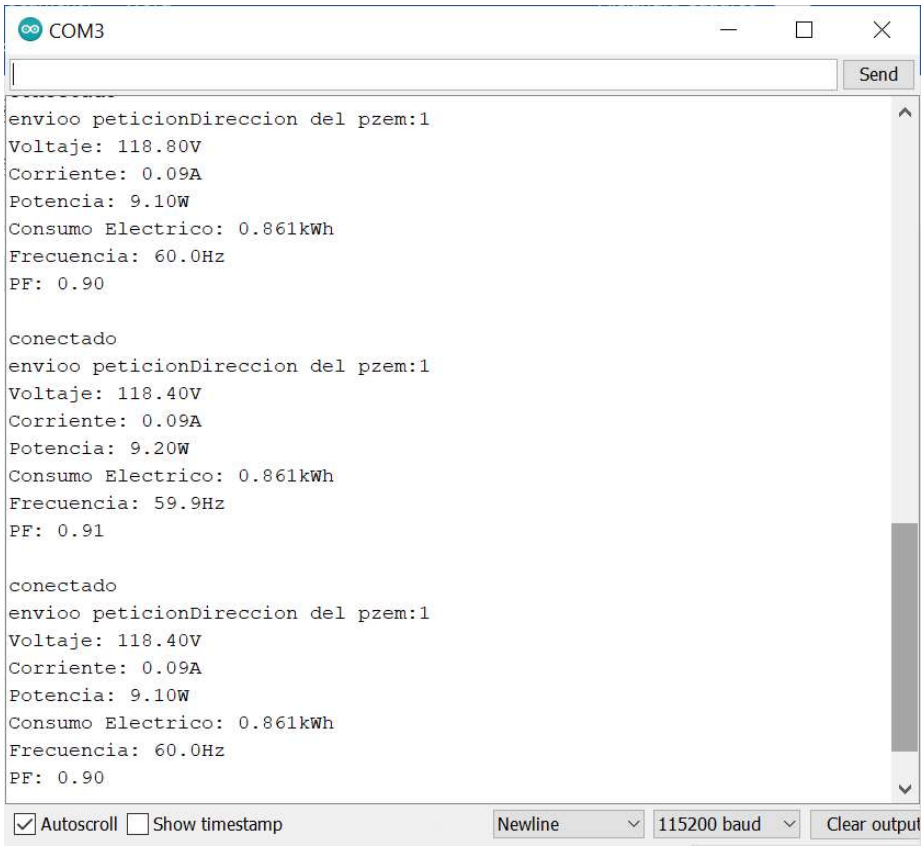
Ilustración 33. Código de Arduino para la conexión con la API

```
http.begin(client, "http://192.168.0.109:8000/api/lectura/"); // ip servidor
http.addHeader("Content-Type","application/x-www-form-urlencoded");
stringEnviar = String("1=")+(voltage)+("&")+String("2=")+(current)+("&")+String("3=")+(power)+("&")
+String("4=")+(energy)+("&")+String("5=")+(frequency)+("&")+String("6=")+(pf);
httpCode = http.POST(stringEnviar);
```

Fuente: Elaboración Propia

- La Ilustración 34 muestra la transmisión de los datos una vez realizada la conexión desde el módulo ESP32 hasta la API.

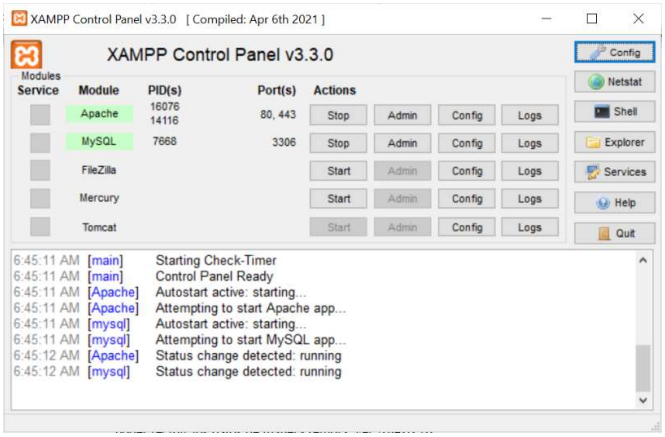
Ilustración 34. Datos reales enviados desde Arduino



Fuente: Elaboración Propia

- Ejecutar el servidor de la base de datos para guardar los datos recibidos por la red como lo muestra la Ilustración 35.

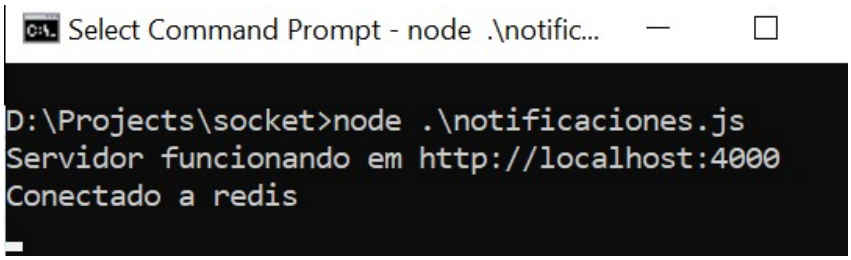
Ilustración 35. Servidor de base de datos mysql



Fuente: Elaboración Propia

- Ejecutar el socket para realizar la recepción remota de los datos en REDIS de acuerdo a lo detallado en la Ilustración 36.

Ilustración 36. *Ejecución de REDIS en estado activo*



Fuente: Elaboración Propia

- Ejecutar de acuerdo a la Ilustración 37 las líneas de código para para ejecutar Python a través de pycharm mismo que genera las épocas del modelo predictivo tal como muestra la Ilustración 38 para su entrenamiento.

Ilustración 37. *Ejecución de Python*

```
\minersystem> python manage.py runserver 0.0.0.0:8000
```

Fuente: Elaboración Propia

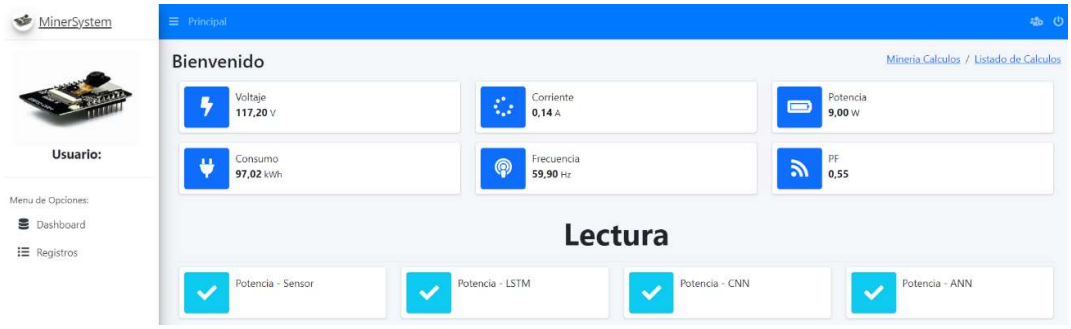
Ilustración 38. *Creación de épocas para el entrenamiento de la red*

```
entrenando segunda red CNN
Epoch 1/1000
1/1 [==============================] - 0s 246ms/step - loss: 0.1827
Epoch 2/1000
1/1 [==============================] - 0s 3ms/step - loss: 0.1814
Epoch 3/1000
1/1 [==============================] - 0s 2ms/step - loss: 0.1801
Epoch 4/1000
1/1 [==============================] - 0s 3ms/step - loss: 0.1787
Epoch 5/1000
1/1 [==============================] - 0s 2ms/step - loss: 0.1774
Epoch 6/1000
1/1 [==============================] - 0s 2ms/step - loss: 0.1761
Epoch 7/1000
1/1 [==============================] - 0s 3ms/step - loss: 0.1749
Epoch 8/1000
1/1 [==============================] - 0s 2ms/step - loss: 0.1736
Epoch 9/1000
1/1 [==============================] - 0s 3ms/step - loss: 0.1723
Epoch 10/1000
1/1 [==============================] - 0s 2ms/step - loss: 0.1710
Epoch 11/1000
1/1 [==============================] - 0s 2ms/step - loss: 0.1697
Epoch 12/1000
1/1 [==============================] - 0s 2ms/step - loss: 0.1685
Epoch 13/1000
```

Fuente: Elaboración Propia

La Ilustración 39 muestra la pantalla principal del software misma que muestra un pequeño menú de opciones para navegar a través del Dashboard y los registros obtenidos de las predicciones.

Ilustración 39. *Pantalla principal de la aplicación*



Fuente: Elaboración Propia

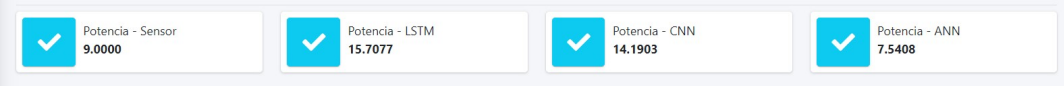
De acuerdo a la Ilustración 40 se puede observar los datos enviados desde el módulo ESP32 wifi misma que se refresca cada 10 segundos para representar el cambio en las mediciones tomadas en tiempo real.

Ilustración 40. *Datos reales de la medición desde el PZEM004T*

Fuente: Elaboración Propia

Según se puede apreciar en la Ilustración 41, se muestran los valores pronosticados por las diversas redes neuronales LSTM, CNN y ANN mientras que en la parte inferior muestra cual es la red que más similitud tiene respecto a los datos reales.

Ilustración 41. *Valor de la predicción de las redes neuronales aplicadas*

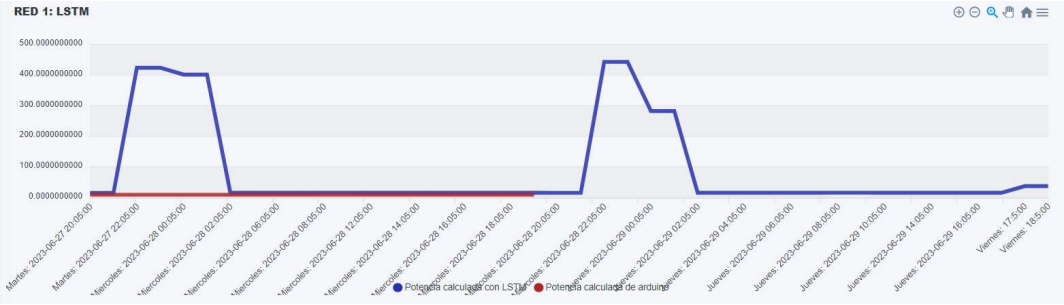


Fuente: Elaboración Propia

De igual forma, en la Ilustración 42, se observa el comportamiento de acuerdo a las predicciones de la red LSTM, misma que representa los valores de acuerdo a las fechas

de los últimos 20 días de medición más dos días que representan la variación que tendrá la energía de acuerdo a la predicción calculada tomando en cuenta el error MAPE.

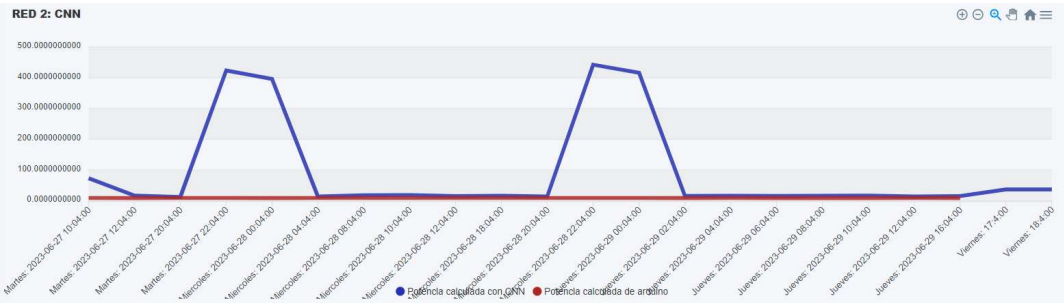
Ilustración 42. *Comportamiento de la carga eléctrica de acuerdo a LSTM*



Fuente: Elaboración Propia

El comportamiento de acuerdo a las predicciones de la red CNN se pueden observar en la Ilustración 43, misma que muestra los valores de acuerdo a las fechas de los últimos 20 días de medición más dos días que representan la variación que tendrá la energía de acuerdo a la predicción calculada tomando en cuenta el error MAPE.

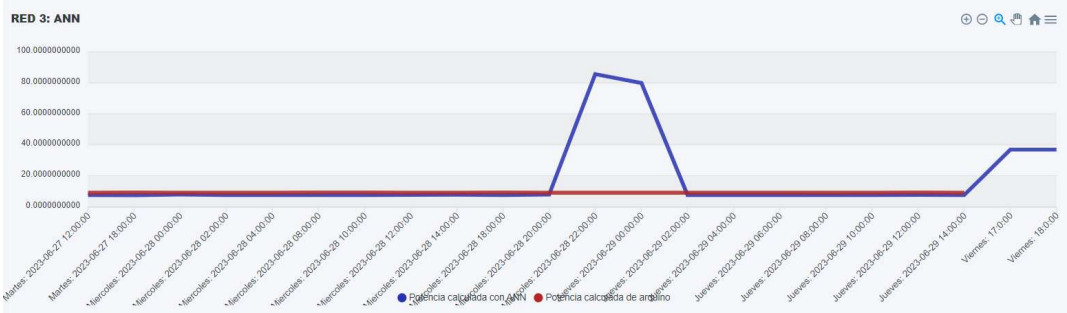
Ilustración 43. *Comportamiento de la carga eléctrica de acuerdo a CNN*



Fuente: Elaboración Propia

El comportamiento de acuerdo a las predicciones de la red ANN se pueden observar en la Ilustración 44, misma que muestra los valores de acuerdo a las fechas de los últimos 20 días de medición más dos días que representan la variación que tendrá la energía de acuerdo a la predicción calculada tomando en cuenta el error MAPE.

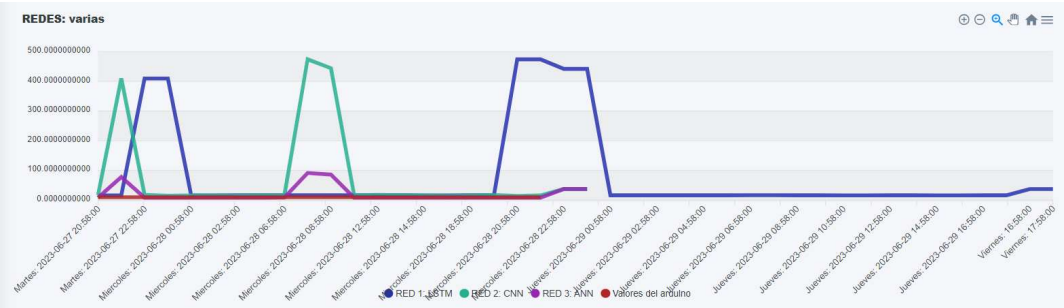
Ilustración 44. *Comportamiento de la carga eléctrica de acuerdo a ANN*



Fuente: Elaboración Propia

La Ilustración 45 representa las mediciones reales de la predicción de las tres redes en conjunto para verificar cual ha sido su comportamiento y cual tiene un mejor margen de error respecto a los valores reales de las mediciones de los últimos 20 días.

Ilustración 45. *Comportamiento de la carga eléctrica en LSTM, CNN y ANN*



Fuente: Elaboración Propia

En la ilustración 46 se observa que el software también muestra una opción para seleccionar la fecha y hora a predecir sobre el consumo y potencia, además de mostrar sus métricas de precisión R2 y MAPE. La métrica de precisión R2 muestra que entre más se acerque a 1 mejor será la precisión del modelo con lo cual podríamos decir que la métrica R2 y MAPE respecto a la predicción de la potencia tiene un alto porcentaje de precisión ya que entre más bajo sea el MAPE mejor será la precisión del modelo, mientras que estas mismas métricas respecto a la predicción del consumo de energía tiene menor precisión, esto debido a que la red debe ser entrenada con más datos para una mejor precisión.

MAPE: cuanto menor sea su valor mejor será la precisión del modelo.

R2: es una medida que determina qué tan bien se ajusta un modelo a los datos. En este caso 1 indica un ajuste perfecto del modelo a los datos.

Se observa que la predicción de consumo es más precisa. Mientras que la predicción de la potencia está más ajustada al modelo, aunque su predicción no es tan precisa.

Esto puede deberse a fatores de consumo de energía dentro de la vivienda tales como número de elementos encendidos, número de personas, etc.

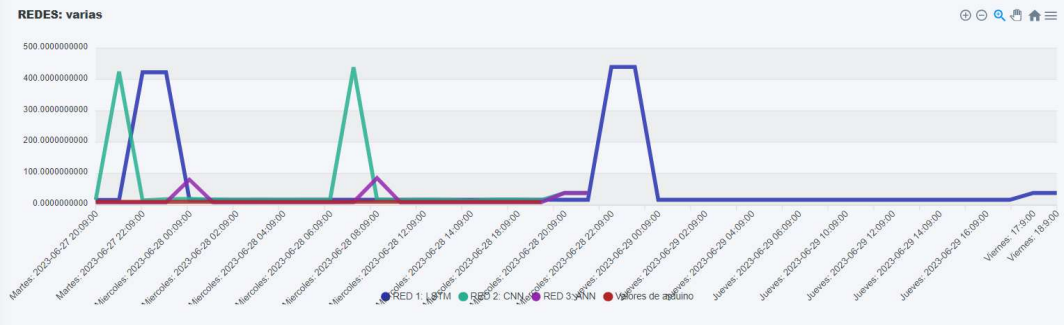
Ilustración 46. *Predicción de consumo y carga eléctrica de acuerdo a métricas*

Fuente: Elaboración Propia

Finalmente, en la Ilustración 47 se representa el comportamiento de la predicción de energía en las tres redes neuronales con fecha y hora. Al ubicarnos sobre un punto de cualquiera de las redes se identifica el nombre de la red a la que representa, su valor y fecha.

La predicción depende mucho de la cantidad de personas que viven en un hogar, sus hábitos de consumo energético y de la cantidad de elementos de consumo energético de la vivienda o lugar que va a ser analizado (taller, casa, etc). En este caso se tuvieron los siguientes datos donde se observa que el consumo más alto se da por las noches y esto debido a que la persona pasa el día fuera de su casa. También se observa al final la predicción del día siguiente que fue seleccionado anteriormente.

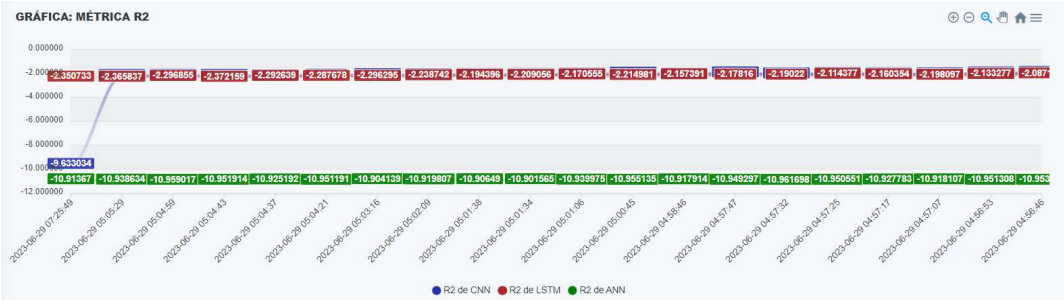
Ilustración 47. Comportamiento de la energía en las tres redes neuronales



Fuente: Elaboración Propia

La pantalla principal también muestra tal como se observa en la Ilustración 48, el registro de métricas de validación de la predicción de acuerdo a la que mejor precisión obtuvo con fecha y su valor calculado. Al pasar el cursor por el registro seleccionado se muestran los valores de las métricas aplicadas.

Ilustración 48. Listado de predicción con métricas más acertadas.



Fuente: Elaboración Propia

La ilustración 49 nos muestra el registro de las mejores predicciones almacenadas en la base de datos con sus métricas de validación y red con mejor precisión.

Ilustración 49. Listado de predicción con métricas más acertadas.

Fecha	Valor	Red	metrica_mape	metrica_r2
29 de Junio de 2023 a las 17:09	7,535016	ANN	5,425867	-10,916810
29 de Junio de 2023 a las 17:09	7,535016	ANN	5,425867	-10,906083
29 de Junio de 2023 a las 17:10	7,544732	ANN	5,075911	-10,923101
29 de Junio de 2023 a las 17:10	7,535016	ANN	5,425867	-10,909570
29 de Junio de 2023 a las 17:10	7,533076	ANN	5,433052	-10,954751
29 de Junio de 2023 a las 17:10	7,536957	ANN	5,725434	-10,888861
29 de Junio de 2023 a las 17:11	7,544732	ANN	5,075911	-10,928489
29 de Junio de 2023 a las 17:11	7,544732	ANN	5,389882	-10,906469
29 de Junio de 2023 a las 17:12	7,544732	ANN	5,389882	-10,951010
29 de Junio de 2023 a las 17:12	7,544732	ANN	5,389882	-10,957034

Fuente: Elaboración Propia

CAPÍTULO 4. DISCUSIÓN DE RESULTADOS

4.1 Revisión sistemática de la literatura

Realizar la revisión sistemática de la literatura permitió guiar nuestro trabajo de investigación, se comprobó claramente que la guía metodológica de Barbara Kitchenham ejecutada entorno a la búsqueda de las técnicas de Machine Learning más eficaces en la predicción de la demanda de energía eléctrica, así como las limitaciones y desafíos asociados con su uso fue precisa.

En primera instancia la fase de la revisión nos permitió definir el propósito estableciendo las preguntas para la revisión sistemática y el protocolo de búsqueda con el fin de conseguir los insumos importantes sobre las técnicas de Machine Learning para el desarrollo del sistema.

Se establecieron las preguntas que necesitaban respuesta para lo cual siguiendo el protocolo de la guía metodología de Barbara Kitchenham se determinan las fuentes bibliográficas de consulta. Asimismo, la definición de las palabras claves y con ellas establecer las cadenas de búsqueda. Fue importante también conocer los parámetros de cada uno de los motores de búsqueda.

Establecer los criterios de inclusión y exclusión nos permitió obtener material acorde a las necesidades de búsqueda de esta manera definir la documentación para el desarrollo de la aplicación.

Al ejecutar la revisión se pudo documentar y sobre todo encontrar varios desarrollos de sistemas de predicción en diferentes campos como la electricidad e IOT.

De igual manera en la revisión se encontró material sobre metodologías tradicionales de minería de datos para el desarrollo de nuestro sistema. Uno de los factores más interesantes fueron el enfoque sistemático, la flexibilidad, la mejora en la calidad del resultado final del proyecto y la reducción del tiempo del desarrollo garantizando que se cumplan las necesidades del usuario final.

Bajo estos parámetros se definió que la metodología CRISP-DM permitiría la construcción del sistema de predicción de carga de energía basado en Machine Learning.

4.2 Desarrollo del software de predicción

La metodología CRISP-DM con sus fases definió el camino para el desarrollo del software.

Los resultados se elaboraron a fin de dar cumplimiento con el objetivo general del proyecto, esto es, desarrollar un software de predicción de carga de energía mediante técnicas de Machine Learning que permita validar el comportamiento energético de una vivienda.

En el desarrollo del software se utilizó datos proporcionados desde un medidor de corriente que tomó datos de una casa, además se desarrolla la metodología y materiales propuestos de tres redes neuronales LSTM, CNN y ANN, a estas redes se las entrenó utilizando varias combinaciones de los hiper parámetros como: filtros, algoritmos y épocas. Generando 1000 combinaciones pudiendo ser cada una de ellas una posible solución a la necesidad de un modelo predictivo. Cada combinación fue sometida a un entrenamiento utilizando el 80% del dataset en tiempo real como datos de entrada de las redes neuronales y mediante métricas para evaluar siendo estos objetos de experimentación.

Algunos de los desafíos asociados con el diseño y desarrollo del software de predicción de carga de energía fueron la selección de las variables adecuadas para las redes neuronales, la limpieza y preparación de los datos, la elección del algoritmo de aprendizaje automático más adecuado y la evaluación del rendimiento de cada red neuronal.

Por otro lado, es importante considerar que el software de predicción de carga de energía puede ser utilizado por diferentes actores del sector energético, como empresas de servicios públicos, proveedores de energía renovable, instituciones gubernamentales y empresas privadas, entre otros. Por lo tanto, el diseño y desarrollo del software debe tener en cuenta las necesidades específicas de cada uno de estos actores y las características de su infraestructura energética.

4.3 Pruebas del software

Las pruebas del software determinaron que el resultado de la hipótesis es afirmativo, al plantear la suposición de que si se desarrolla un software aplicando técnicas de Machine Learning optimiza la predicción de carga eléctrica. Esto se verificó a través de los resultados obtenidos de las tres redes neuronales que fueron entrenadas y que dieron una mejor respuesta en tiempo real con métricas de validación MAPE y R2.

Realizar las pruebas del software para mostrar los resultados obtenidos en la predicción al variar los parámetros y tipos de predictores utilizado fue fundamental para garantizar que el software de predicción sea preciso y confiable.

La presentación de resultados obtenidos mediante la variación de los parámetros y tipos de predictores permitieron evaluar la sensibilidad del modelo de predicción a diferentes configuraciones y, por lo tanto, determinar la configuración óptima para lograr la máxima precisión en la predicción. Además, la variación de los parámetros y los predictores también puede ayudar a identificar los límites del modelo de predicción y las limitaciones en la precisión que pueden ser necesarias para ser abordadas en futuras mejoras del software.

Otra razón por la cual fue importante realizar pruebas del software mediante la exposición de los resultados obtenidos en la predicción al modificar los parámetros ya que pueden detectar posibles errores en el software y ayudar a corregirlos antes de que el software se utilice en la práctica además de ser una herramienta crítica para evaluar la precisión y la confiabilidad del software de predicción de carga de energía. Esta práctica puede ayudar a optimizar el modelo de predicción y detectar posibles errores antes de la implementación del software en la práctica, mejorando así su calidad y rendimiento.

Finalmente, los resultados mostraron que el modelo LSTM se destacó por su mayor precisión en comparación con los otros modelos al predecir la demanda energética. También se observó que los modelos ANN y CNN tuvieron un buen desempeño en la predicción de la demanda energética, pero no tan bueno como el modelo LSTM.

CONCLUSIONES

En base a la revisión sistemática de la literatura utilizando la guía metodológica de Barbara Kitchenham, se ha podido identificar y analizar las diversas técnicas de Machine Learning utilizadas en la predicción de carga de energía eléctrica. Esto ha permitido obtener una comprensión profunda de las metodologías existentes, sus ventajas y limitaciones. Esta investigación ha proporcionado una base sólida para el desarrollo del software de predicción de carga de energía eléctrica.

A través del diseño y desarrollo del software de predicción de carga de energía eléctrica, se ha logrado implementar de manera efectiva las técnicas de Machine Learning investigadas. El software utiliza modelos de Machine Learning como CNN, ANN y LSTM para estimar la demanda de energía eléctrica en un intervalo de tiempo específico. La interfaz de usuario intuitiva y las funcionalidades del software permiten una fácil utilización y configuración de los modelos de predicción.

Las pruebas realizadas en el software de predicción de carga de energía eléctrica han sido fundamentales para evaluar su desempeño y validar la precisión de las predicciones. Se han llevado a cabo pruebas con diferentes configuraciones de parámetros y tipos de predictores, lo que ha permitido identificar las combinaciones más efectivas. Los resultados obtenidos en las pruebas demuestran la capacidad del software para realizar predicciones precisas y confiables de la demanda de energía eléctrica en diferentes escenarios.

En resumen, la investigación de las técnicas de Machine Learning, el diseño y desarrollo del software de predicción de carga de energía eléctrica, así como las pruebas realizadas, han demostrado la viabilidad y eficacia de utilizar el Machine Learning para estimar la demanda de energía eléctrica. El software desarrollado constituye una herramienta útil para la planificación y gestión eficiente de la energía, al proporcionar predicciones precisas y confiables en diferentes escenarios y configuraciones.

RECOMENDACIONES

Se recomienda continuar actualizando la revisión sistemática de la literatura de manera periódica para estar al tanto de las nuevas técnicas y avances en la predicción de carga de energía eléctrica. Esto permitirá mantenerse al día con las últimas investigaciones y garantizar la utilización de las técnicas más avanzadas y eficientes.

Explorar fuentes de información adicionales, como conferencias y revistas especializadas, para ampliar el conocimiento sobre las técnicas de Machine Learning en la predicción de carga de energía eléctrica. Esto ayudará a obtener una visión más completa y diversa de las metodologías disponibles.

Asegurarse de que los datos sean precisos, completos y estén actualizados. Además, es importante considerar factores externos que puedan afectar la predicción, como eventos climáticos extremos, festividades, etc.

Implementar un sistema de actualización automática del software para incorporar nuevas funcionalidades y mejoras a medida que estén disponibles. Esto garantizará que el software se mantenga actualizado y siga siendo relevante en un entorno en constante evolución.

Realizar pruebas exhaustivas con diferentes conjuntos de datos y escenarios de predicción para evaluar la robustez y generalización del software. Esto permitirá identificar posibles limitaciones y áreas de mejora en el rendimiento del software.

Fomentar la colaboración y retroalimentación con otros investigadores y profesionales en el campo de la predicción de carga de energía eléctrica. Esto permitirá intercambiar ideas, compartir experiencias y obtener perspectivas adicionales que puedan enriquecer el proceso de pruebas y mejorar la calidad del software.

TRABAJOS FUTUROS

Investigar el impacto de la utilización de técnicas de preprocesamiento de datos más avanzadas, como la normalización de series de tiempo y la detección de anomalías, en la precisión de la predicción. Estas técnicas pueden ayudar a mejorar la calidad de los datos de entrada y reducir el ruido, lo que puede resultar en predicciones más precisas.

Mejorar la escalabilidad y la eficiencia del software para poder manejar conjuntos de datos más grandes y realizar predicciones en tiempo real. Esto permitiría su aplicación en entornos de producción donde se requiere una predicción en tiempo casi real de la demanda de energía eléctrica.

Integrar el software de predicción de carga de energía eléctrica con sistemas de monitorización y control en tiempo real. Esto permitiría una retroalimentación continua entre la predicción y el control de la energía, mejorando así la eficiencia y optimización del consumo energético.

Realizar análisis más detallados de sensibilidad para identificar el impacto de cada parámetro y predictor en la precisión de la predicción. Esto ayudaría a ajustar y optimizar los parámetros del modelo para obtener mejores resultados.

Realizar pruebas comparativas con otros modelos y técnicas de predicción de carga de energía eléctrica disponibles en la literatura. Esto permitiría evaluar y comparar el rendimiento del software desarrollado con otros enfoques existentes y determinar su nivel de eficacia y ventajas.

Realizar pruebas en diferentes entornos y condiciones geográficas para evaluar la adaptabilidad y robustez del software. Esto ayudaría a determinar si el software puede ser aplicado en diferentes contextos y escenarios, y si sus resultados se mantienen consistentes y precisos.

BIBLIOGRAFÍA

[1] World Energy Council, «World Energy Scenarios 2019.,» *de European Regional,* 2019.

[2] D. García, E. López y M. González, «Aplicaciones del Internet de las Cosas en la educación: Experiencias de implementación y resultados.,» *Revista de Tecnología Educativa,* vol. 12, nº 4, pp. 87-98, 2022.

[3] F. Martínez, L. Rodríguez y G. Pérez, «Evaluación remota de la conducta de un invernadero en el desierto de Atacama para la optimización del uso del agua.,» *Revista de Ingeniería Ambiental,* vol. 14, nº 3, pp. 120-128, 2019.

[4] A. Sánchez, R. González y J. Pérez, «Control de referencia del regulador de presión mediante el monitoreo de información de transductores en una planta industrial.,» *Revista de Automatización y Control Industrial,* vol. 14, nº 1, pp. 45-52, 2022.

[5] R. Sánchez y M. Torres, «La transformación de las bibliotecas convencionales hacia sistemas inteligentes en línea.,» *Revista Iberoamericana de Ciencia, Tecnología y Sociedad,* vol. 14, nº 41, pp. 75-89, 2019.

[6] J. Jimenez, L. Navarro, C. Quintero y M. Pardo, «A Methodology for Energy Load Profile Forecasting Based on Intelligent Clustering and Smoothing Techniques,» *Energies,* vol. 13, nº 16, 2020.

[7] D. J. Benavides, P. Arévalo Cordero, L. Gonzalez y et al, «Method of monitoring and detection of failures in PV system based on machine learning,» *Facultad de Ingeniería Universidad de Antioquia,* nº 102, pp. 26-43, 2022.

[8] J. Jimenez, K. Donado y C. G. Quintero M, «A methodology for short-term load forecasting,» vol. 15, nº 3, pp. 400-407, 2017.

[9] A. M. Lastre Aleaga, E. F. Méndez Garcés y A. Cordovés García, «Sistema automatizado para la predicción de flujo de carga en subestaciones eléctricas mediante redes neuronales artificiales,» *Enfoque UTE,* vol. 6, nº 3, Julio-Septiembre 2015.

[10] J. López, M. Sánchez y R. Torres, «Evaluación de herramientas de predicción eólica basadas en datos meteorológicos para parques eólicos.,» *Revista de Energía Eólica,* vol. 10, nº 3, pp. 78-89, 2020.

[11] E. López, A. Sánchez y L. Martínez, «Predicción de la demanda de electricidad en México utilizando técnicas de inteligencia artificial y datos socioeconómicos.,» *Revista de Energía Renovable,* vol. 12, nº 4, pp. 78-89, 2022.

[12] R. García, M. Sánchez y P. Rodríguez, «Aplicación de algoritmos de Machine Learning en la predicción de mantenimiento considerando el incremento de carga

en sistemas de generación eléctrica.,» *Revista de Investigación en Ingeniería Eléctrica,* vol. 11, nº 3, pp. 67-80, 2020.

[13] J. Rodríguez, L. Sánchez y C. González, «Aplicación de redes neuronales artificiales en la predicción del flujo de carga en subestaciones eléctricas de alta tensión.,» *Revista de Investigación en Ingeniería Eléctrica,* vol. 11, nº 4, pp. 45-58, 2020.

[14] C. Villarroel González, V. Goykovic Cortés, P. Collao Caiconte y e. al., «EVALUACIÓN DE DESEMPEÑO DE UN INVERNADERO UBICADO EN EL DESIERTO DE ATACAMA, CHILE, A TRAVÉS DE IoT,» *Interciencia,* vol. 44, nº 7, pp. 386-393, 2019.

[15] M. Ramírez, G. Torres y E. Gómez, «Modelado y predicción de la demanda de energía eléctrica del SIN de Colombia utilizando técnicas de análisis Wavelet y modelos neuronales auto-regresivos.,» *Revista de Ingeniería y Tecnología,* vol. 12, nº 4, pp. 78-88, 2021.

[16] D. Castro, W. Coral, J. Cabra, J. Colorado, D. Méndez y L. Trujillo, «Survey on IoT solutions applied to Healthcare,» *DYNA,* vol. 84, nº 203, pp. 192-200, 2017.

[17] L. F. Rueda Vásquez, J. G. Barrero Pérez y C. Duarte, «El conmutador inteligente de potencia y la sub-medición por circuito como herramientas para la gestión energética residencial,» *UIS Ingenierías,* vol. 16, nº 1, pp. 35-46, 2017.

[18] A. Bolaños, C. Ramírez y A. Ramos, «Predicción de la demanda de energía eléctrica mediante redes neuronales recurrentes y técnicas de optimización.,» *Revista Iberoamericana de Automática e Informática Industrial,* vol. 17, nº 1, pp. 67-76, 2020.

[19] R. Sánchez y M. Torres, «Predicción de la demanda eléctrica residencial mediante redes neuronales de múltiples capas.,» *Revista de Ingeniería Eléctrica, Electrónica y Computación,* vol. 19, nº 3, pp. 65-72, 2019.

[20] J. Torres, R. González y M. López, «Predicción de la demanda eléctrica utilizando redes neuronales convolucionales y recurrentes en el sector residencial.,» *Revista de Investigación en Ingeniería Eléctrica, Electrónica y de Sistemas,* vol. 20, pp. 65-73, 2020.

[21] R. Sánchez, L. García y C. Martínez, «Aplicación de redes neuronales generativas adversariales en la predicción de la energía eléctrica a partir de datos históricos.,» *Revista de Energías Renovables y Eficiencia Energética,* vol. 11, pp. 45-53, 2020.

[22] D. González, A. López y P. Torres, «Predicción de la demanda de energía eléctrica mediante redes neuronales con estructuras de atención.,» *Revista de Ciencia, Tecnología e Innovación,* vol. 9, pp. 112-120, 2021.

[23] C. Sánchez, E. Martínez y J. Fernández, «Gestión óptima de la energía eléctrica mediante redes neuronales y aprendizaje por refuerzo.,» *Revista de Energías Renovables y Sostenibilidad,* vol. 13, nº 4, pp. 150-158, 2023.

[24] T.M. Mitchell y M. Tom, Machine Learning, New York: McGraw-Hill, 1997.

[25] M. Gómez, E. Ramírez y L. Torres, «Aplicación del Machine Learning en el análisis de sentimientos en redes sociales.,» *Revista de Investigación en Tecnologías de la Información,* vol. 40, nº 4, pp. 112-121, 2021.

[26] A. González, «Cleverdata.io,» [En línea]. Available: https://cleverdata.io/que-es-machine-learning-big-data/. [Último acceso: 24 julio 2022].

[27] S. Marsland, Machine Learning and algorithmic perspective, New York, Estados Unidos: Segunda Edición Chapman & Hall/CRC, 2015.

[28] J. García, A. Sánchez y M. Rodríguez, «Mejora de la precisión en algoritmos de aprendizaje supervisado mediante técnicas de selección de características.,» *Revista de Investigación en Ciencias de la Computación,* vol. 19, nº 4, pp. 112-121, 2019.

[29] T. W. Edgar y D. O. Manz, «Research Methods for Cyber Security,» Cambridge: Elsevier Inc., 2017, p. 404.

[30] J. Ramírez, C. Sánchez y F. Gómez, «Modelos de regresión logística para la detección de fraudes en transacciones financieras.,» *Revista de Investigación en Tecnologías de la Información,* vol. 40, nº 4, pp. 112-120, 2020.

[31] R. Sánchez, F. Pérez y M. Torres, «Aplicación de técnicas de reducción de la redundancia de datos en el aprendizaje no supervisado.,» *Revista de Ciencias de la Computación,* vol. 29, nº 4, pp. 103-112, 2019.

[32] J. Ramírez, C. Sánchez y F. Gómez, «Algoritmos de agrupamiento basados en optimización multiobjetivo para la asignación de recursos en redes de comunicaciones.,» *Revista de Investigación en Ingeniería de Telecomunicaciones,* vol. 40, nº 4, pp. 78-86, 2020.

[33] M. Ramírez, S. López y G. Torres, «Aprendizaje por refuerzo en el control de robots autónomos.,» *Revista de Investigación en Robótica,* vol. 29, nº 4, pp. 156-165, 2019.

[34] T. Bustos, «Estrategia para la conceptualización de modelos de IA en el contexto de gestión de la energía,» *ICAI de la Universidad Pontificia Comillas,* 2021.

[35] A. González y L. Martínez, «Adaptabilidad y flexibilidad en sistemas de adquisición de datos para aplicaciones industriales.,» *Revista de Ingeniería de Sistemas y Automatización,* vol. 2, nº 27, pp. 45-62, 2020.

[36] A. García y R. Martínez, «Mejora de la precisión en la predicción de resultados utilizando XGBoost.,» *Revista de Ciencia de Datos y Aprendizaje Automático,* vol. 42, nº 2, pp. 127-140, 2023.

[37] A. Rodríguez, C. Pérez y M. Fernández, «Evaluación de algoritmos de potenciación del gradiente en el análisis de sentimientos en redes sociales.,» *Revista de Investigación en Tecnologías de la Información,* vol. 40, nº 4, pp. 89-97, 2020.

[38] M. Pérez y A. Rodríguez, «Redes neuronales en aprendizaje profundo: una aproximación al modelado de la conectividad cerebral.,» *Revista de Inteligencia Artificial y Aprendizaje Automático,* vol. 35, nº 3, pp. 87-102, 2022.

[39] L. González y R. Martínez, «Redes bayesianas: una herramienta visual para modelar dependencias e independencias entre variables aleatorias.,» *Revista de Estadística y Análisis de Datos,* vol. 40, nº 2, pp. 145-160, 2022.

[40] P. Ganguly, A. Kalam y A. Zayegh, «SHORT TERM LOAD FORECASTING USING FUZZY LOGIC,» *International Conference on Research in Education and Science (ICRES),* 2017.

[41] A. López y J. Rodríguez, «Uso de iForest para la detección de anomalías en conjuntos de datos.,» *Revista de Informática y Tecnología,* vol. 38, nº 3, pp. 201-215, 2022.

[42] J. P. Mueller y M. Luca, Deep Learning For Dummies, Hoboken: John Wiley & Sons, Inc, 2019.

[43] M. Hassoun, «Redes neuronales paralelas y adaptativas: un enfoque computacional.,» *Revista de Investigación en Ciencias de la Computación,* vol. 10, nº 2, pp. 45-58, 2020.

[44] N. Matich, «Sistemas interconectados paralelos: una aproximación inspirada en el sistema nervioso biológico.,» *Revista de Ciencia Computacional,* vol. 15, nº 3, pp. 78-92, 2019.

[45] P. Ganguly, A. Kalam y A. Zayegh, «Short Term Load Forecasting Using Fuzzy Logic [Pronóstico de carga a corto plazo, lógica difusa],» *Conferencia Internacional sobre Investigación en Educación y Ciencia (ICRES),* pp. 355-361, 2017.

[46] J. Jara Estupiñan, D. Giral y F. Martínez Santa, «Implementación de algoritmos basados en máquinas de soporte vectorial (SVM) para sistemas eléctricos: revisión de tema,» *Tecnura,* vol. 20, nº 48, pp. 149-170, 2016.

[47] S. Lu, M. A. Hossain y G. Muhammad, «Long-term electricity load forecasting using convolutional neural network.,» *Energies,* vol. 13, nº 12, p. 3124, 2020.

[48] P. Fernández, M. Díaz y A. González, «Aplicación de redes neuronales artificiales para la predicción de la demanda eléctrica en una región.,» *Revista de Investigación en Energías Renovables,* vol. 27, nº 2, pp. 45-56, 2021.

[49] P. Xie, X. Liu y Y. Gao, «A hybrid deep learning model for short-term load forecasting based on long short-term memory and convolutional neural network.,» *Energies,* vol. 12, nº 9, p. 1640, 2019.

[50] M. Pérez y J. Gómez, «Relación entre el valor de activación y la tasa de disparo en redes neuronales: un enfoque biológico.,» *Revista de Neurociencia Computacional,* vol. 25, nº 2, pp. 45-62, 2020.

[51] C. Ríos, A. Rodríguez y J. Gómez, «Redes neuronales de una sola capa para la clasificación de datos en problemas de reconocimiento de patrones.,» *Revista de Investigación en Inteligencia Artificial,* vol. 31, nº 2, pp. 45-56, 2020.

[52] J. García, M. Rodríguez y N. Hernández, «Uso de redes neuronales en la integración de datos biomédicos para la detección de enfermedades.,» *Revista de Medicina y Salud Pública,* vol. 17, nº 2, pp. 89-100, 2019.

[53] M. Rodríguez, J. García y P. Torres, «Redes neuronales convolucionales para el reconocimiento de objetos en imágenes satelitales.,» *Revista de Ciencias de la Computación,* vol. 15, nº 1, pp. 89-100, 2019.

[54] M. Rodríguez, R. González y N. Hernández, «Modelado y pronóstico de variables meteorológicas utilizando redes neuronales convolucionales.,» *Revista de Ciencias Atmosféricas,* vol. 15, nº 1, pp. 89-100, 2020.

[55] A. Palma Llewellyn, «Pronóstico de demanda de energía y potencia eléctrica en el largo plazo para la red de Chilectra S.A. utilizando técnicas de minería de datos.,» 2015.

[56] J. Pérez y M. Gómez, «Evaluación comparativa de metodologías para proyectos de análisis de datos: un enfoque basado en la propuesta de Piatetsky.,» *Revista de Minería de Datos y Análisis de Datos,* vol. 15, nº 3, pp. 78-95, 2018.

[57] Espinosa-Zúñiga y Javier Jesús, «Aplicación de metodología CRISP-DM para segmentación geográfica de una base de datos pública.,» *Ingeniería, investigación y tecnología, 21,* 3 agosto 2020.

[58] A. González y L. Martínez, «Adaptabilidad y flexibilidad en sistemas de adquisición de datos para aplicaciones industriales.,» *Revista de Ingeniería de Sistemas y Automatización,* vol. 2, nº 27, pp. 45-62, 2020.

[59] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey y S. Linkman, «Systematic literature reviews in software engineering – A systematic literature review,» *Information and Software Technology,* vol. 51, nº 1, pp. 7-15, 2009.

Anexos

Anexo 1

ANEXO 1: código fuente de sensor ESP32

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <PZEM004Tv30.h>
#if defined(ESP32)
PZEM004Tv30 pzem(Serial2, 16, 17);
#else
PZEM004Tv30 pzem(Serial2);
#endif

HTTPClient http;
int httpCode;

const char* ssid = "PAKINET - GoNet";///Nombre del Wifi
const char* password = "@isth_1981.";///Contraseña Wifi @isth_1981.

String stringLed, stringSensor, stringEnviar, stringEnviar2;
int estadoLed= 1;

void setup() {
  stringLed = String("status=");
  stringSensor = String("sensor=");
  stringEnviar = String();
  stringEnviar2 = String();

  Serial.begin(115200);
  pinMode(13,OUTPUT);
  digitalWrite(13,HIGH);

  WiFi.begin(ssid,password);
  while(WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println(WiFi.localIP());
}

void loop() {
  Serial.print("Direccion del pzem:");
  Serial.println(pzem.readAddress(), HEX);
  float voltage = pzem.voltage();
  float current = pzem.current();
  float power = pzem.power();
  float energy = pzem.energy();
  float frequency = pzem.frequency();
  float pf = pzem.pf();
  if(isnan(voltage)){
      Serial.println("Error de lectura del voltaje");
   } else if (isnan(current)) {
      Serial.println("Error de lectura del amperaje");
   } else if (isnan(power)) {
      Serial.println("Error de lectura de la potencia");
   } else if (isnan(energy)) {
      Serial.println("Error de lectura de la energia");
```

```
  } else if (isnan(frequency)) {
    Serial.println("Error de lectura de la frecuencia");
  } else if (isnan(pf)) {
    Serial.println("Error de lectura del factor de potencia");
  } else {


    Serial.print("Voltaje: ");      Serial.print(voltage);      Serial.println("V");
    Serial.print("Corriente: ");      Serial.print(current);      Serial.println("A");
    Serial.print("Potencia: ");       Serial.print(power);       Serial.println("W");
    Serial.print("Consumo Electrico: ");      Serial.print(energy,3);     Serial.println("kWh");
    Serial.print("Frecuencia: ");    Serial.print(frequency, 1); Serial.println("Hz");
    Serial.print("PF: ");          Serial.println(pf);

  }
 Serial.println();

 if(WiFi.status() == WL_CONNECTED)
 {
  Serial.println("conectado");
  WiFiClient client;
  http.begin(client, "http://192.168.0.25:8000/api/lectura/"); // ip servidor
  http.addHeader("Content-Type","application/x-www-form-urlencoded");
  stringEnviar                                                                                    =
String("1=")+(voltage)+("&")+String("2=")+(current)+("&")+String("3=")+(power)+("&")
  +String("4=")+(energy)+("&")+String("5=")+(frequency)+("&")+String("6=")+(pf);
  httpCode = http.POST(stringEnviar);
  http.end();
  Serial.print("envioo peticion");
  delay(5000);
 }
 delay(100);
}
```

ANEXO I: CDIGO FUENTE DE ENTRENAMIENTO DE LAS REDES

```
from datetime import time
from datetime import datetime
from django.db.models import Q, Avg
from django.db.models.functions import ExtractHour, ExtractMinute
from rest_framework import viewsets
from sklearn.model_selection import train_test_split

from aplicacion.app_minersystem.models import Tablaesp2, Red1, Red2, Red3, MejorResultado
from aplicacion.app_lectura.serializers import Tablaesp2Serializer, Lectura1Serializer, Lectura2Serializer,
\
   Lectura3Serializer, LecturasSerializer, MejorResultadoSerializer, Lectura1Serializerlol
from rest_framework.response import Response
from rest_framework import status
import redis
import json
import pandas as pd
import numpy as np
```

```
import tensorflow as tf
from keras.saving.model_config import model_from_json
from .metodos import convert
import matplotlib.pylab as plt
import datetime
from django.db.models.functions import TruncHour, TruncMinute
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error

plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('fast')
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten
from sklearn.preprocessing import MinMaxScaler
from djqscsv import write_csv


# Create your views here.
class Tablaesp2ViewSet(viewsets.ModelViewSet):
    queryset = Tablaesp2.objects.filter(codigo='2')[:50]
    serializer_class = Tablaesp2Serializer

    def create(self, request, *args, **kwargs):
        print("*********datos arduino********")
        print(request.POST)
        voltaje = float(request.POST["1"])
        corriente = float(request.POST["2"])
        valores_a_predecir = np.array([[voltaje, corriente]])
        Tablaesp2.objects.create(codigo="1", nombre="voltaje", Valor=request.POST['1'])
        Tablaesp2.objects.create(codigo="2", nombre="corriente", Valor=request.POST['2'])
        Tablaesp2.objects.create(codigo="3", nombre="potencia", Valor=request.POST['3'])
        Tablaesp2.objects.create(codigo="4", nombre="consumo", Valor=request.POST['4'])
        Tablaesp2.objects.create(codigo="5", nombre="frecuencia", Valor=request.POST['5'])
        Tablaesp2.objects.create(codigo="6", nombre="fp", Valor=request.POST['6'])

        # cargar json y crear el modelo
        json_file = open("static/modelos/model_tf.json", 'r')
        loaded_model_json = json_file.read()
        json_file.close()
        loaded_model = model_from_json(loaded_model_json)
        # cargar pesos al nuevo modelo
        loaded_model.load_weights("static/modelos/model_tf.h5")
        # Compilar modelo cargado y listo para usar.
        loaded_model.compile(loss='mean_squared_error', optimizer='adam', metrics=['binary_accuracy'])
        valor1 = loaded_model.predict(valores_a_predecir)[0][0]
        val1round = round(valor1, 4)

        # cargar json y crear el modelo
        json_file_kr = open("static/modelos/model_keras.json", 'r')
        loaded_model__kr_json = json_file_kr.read()
        json_file_kr.close()
        loaded_model_kr = model_from_json(loaded_model__kr_json)
        loaded_model_kr.load_weights("static/modelos/model_keras.h5")
        valores_a_predecir = np.array([[voltaje / 130, corriente / 10]])
        # Compilar modelo cargado y listo para usar.
```

```
        loaded_model_kr.compile(loss='mean_squared_error',                    optimizer='adam',
metrics=['binary_accuracy'])
        valor2 = loaded_model_kr.predict(valores_a_predecir)[0][0]
        val2round = round(valor2 * 1300, 4)
        qs1 = Red1.objects.all().values('fecha', 'valor')
        with open('red1.csv', 'wb') as csv_file:
            write_csv(qs1, csv_file)

        df = pd.read_csv('red1.csv', parse_dates=[0], header=None, index_col=0, squeeze=True,
                         names=['fecha', 'unidades'])
        qs2 = Red2.objects.all().values('fecha', 'valor')
        with open('red2.csv', 'wb') as csv_file:
            write_csv(qs2, csv_file)

        df2 = pd.read_csv('red2.csv', parse_dates=[0], header=None, index_col=0, squeeze=True,
                          names=['fecha', 'unidades'])
        '''
        print("**********************")
        print(df.head())
        print(df.index.min())
        print(df.index.max())
        print(df.describe())
        print("**********************")
        '''
        try:
            r = redis.StrictRedis(host='localhost', port=6379, db=0)
            val2round = convert(val1round)
            data_json = request.data
            _mutable = data_json._mutable
            data_json._mutable = True
            data_json['red1'] = str(val1round)
            data_json['red2'] = str(val2round)
            Red1.objects.create(valor=float(val1round), fecha=datetime.datetime.now())
            Red2.objects.create(valor=float(val2round), fecha=datetime.datetime.now())
            data_json._mutable = _mutable
            r.publish('notificacion', json.dumps(data_json).encode('utf8'))
        except Exception as e:
            print("error: " + str(e))
        return Response({}, status=status.HTTP_200_OK)


# Create your views here.
class EntrenamientoViewSet(viewsets.ViewSet):

    def create(self, request, *args, **kwargs):
        # cargamos las combinaciones posibles
        print("Entrenamiento 1 red LSTM")
        training_data = np.array(
            [[120.4, 0.14], [116.76, 0.09], [120.2, 0.14], [113.2, 0.02], [116.88, 0.06], [112.0, 0.02],
             [112.24, 0.43]],
            "float32")
        # y estos son los resultados que se obtienen, en el mismo orden
        target_data = np.array([[16.86], [10.51], [16.83], [2.26], [7.01], [2.24], [48.33]], "float32")
        capa = tf.keras.layers.Dense(units=1, input_shape=[2])
        modelo = tf.keras.Sequential([capa])
        # nota aqui ay tenemos el modelo listo
```

```
        # preparo el modelo para ser entrenado
        # declaro como quiero que procese esa data para aprender mejor
        modelo.compile(
            optimizer=tf.keras.optimizers.Adam(0.1),
            loss='mean_squared_error'  # error cuadratico medio
        )
        print('Comenzando con el entrenamiento')
        historial = modelo.fit(training_data, target_data, epochs=1000, verbose=False)
        print('Modelo Entrenado')
        print("Variables internas del Modelo")
        print(capa.get_weights())

        model_json = modelo.to_json()
        with open("static/modelos/model_tf.json", "w") as json_file:
            json_file.write(model_json)
        # serializar los pesos a HDF5
        modelo.save_weights("static/modelos/model_tf.h5")
        print("Modelo Guardado!")

        print("entrenado segunda red CNN")
        training_data = np.array(
            [[0.9262,  0.014], [0.8982,  0.009], [0.9246,  0.014], [0.8990,  0.006], [0.8615,  0.002], [0.8633,
0.043]],
            "float32")
        # y estos son los resultados que se obtienen, en el mismo orden
        target_data = np.array([[0.01297], [0.008084], [0.01294], [0.005394], [0.005169], [0.3712]],
"float32")
        # cargamos las combinaciones posibles
        model = Sequential()
        model.add(Dense(16, input_dim=2, activation='relu'))
        model.add(Dense(1, activation='sigmoid'))
        model.compile(loss='mean_squared_error', optimizer='adam')
        model.fit(training_data, target_data, epochs=1000)
        scores = model.evaluate(training_data, target_data)
        # print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1] * 100))

        # serializar el modelo a JSON
        model_json = model.to_json()
        with open("static/modelos/model_keras.json", "w") as json_file:
            json_file.write(model_json)
        # serializar los pesos a HDF5
        model.save_weights("static/modelos/model_keras.h5")
        print("Modelo segunda red Guardado!")

        return Response({}, status=status.HTTP_200_OK)


# Create your views here.
class Lectura1ViewSet(viewsets.ModelViewSet):
    queryset = Red1.objects.all().order_by('-fecha')[:10]
    serializer_class = Lectura1Serializer


class Lectura2ViewSet(viewsets.ModelViewSet):
    queryset = Red2.objects.all().order_by('-fecha')[:10]
    serializer_class = Lectura2Serializer
```

```
class Lectura3ViewSet(viewsets.ModelViewSet):
    queryset = Red3.objects.all().order_by('-fecha')[:10]
    serializer_class = Lectura3Serializer
    permission_classes = ()

    def list(self, request, *args, **kwargs):
        data = request.GET['valor']
        Red3.objects.create(valor=data)
        return Response({"ok": "guardado correctamente"})


class MejorResultadoViewSet(viewsets.ModelViewSet):
    queryset = MejorResultado.objects.all().order_by('-fecha')
    serializer_class = MejorResultadoSerializer

    def list(self, request, *args, **kwargs):
        valor = request.GET['valor']
        red = request.GET['red']
        r2_enviado = request.GET['r2']
        mape_enviado = request.GET['mape']
        MejorResultado.objects.create(valor=valor, red=red, metrica_r2=r2_enviado,
metrica_mape=mape_enviado)
        return Response({"ok": "guardado correctamente"})


class PrediccionPotenciaViewSet(viewsets.ViewSet):
    # Cargar los datos del archivo CSV en la inicialización de la aplicación
    consumo_df = pd.read_csv('datos_potencia.csv')
    X = consumo_df[['fecha', 'hora']]
    y = consumo_df['potencia']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
    modelo = LinearRegression()
    modelo.fit(X_train, y_train)

    def prediccion_potencia(self, fecha_str, hora_str):
        try:
            # Convertir fecha y hora a valores numéricos
            fecha = datetime.datetime.strptime(fecha_str, '%Y-%m-%d').toordinal()
            hora = datetime.datetime.strptime(hora_str, '%H:%M').hour

            # Realizar la predicción utilizando el modelo de regresión lineal
            prediccion_p = self.modelo.predict([[fecha, hora]])
            prediccion_p = prediccion_p / 10000

            # Calcular las métricas de evaluación
            y_test_pred = self.modelo.predict(self.X_test)
            r2 = r2_score(self.y_test, y_test_pred)
            mape_e = np.mean(np.abs((self.y_test - y_test_pred) / self.y_test)) * 100

            # Devolver la predicción y las métricas de evaluación
            return {"prediccion_p": prediccion_p[0], "r2": r2, "mape_e": mape_e}

        except Exception as e:
            # Manejar errores y devolver una respuesta adecuada
```

```
            return {"error": str(e)}

    def list(self, request):
        # Obtener las fechas y horas de los parámetros de la solicitud
        fecha_str = request.GET.get('fecha')
        hora_str = request.GET.get('hora')

        # Obtener la predicción de consumo utilizando las fechas y horas
        resultado = self.prediccion_potencia(fecha_str, hora_str)


        # Devolver la respuesta adecuada
        if "error" in resultado:
            return Response(resultado, status=400)
        else:
            return Response(resultado)


class PrediccionConsumoViewSet(viewsets.ViewSet):
    # Cargar los datos del archivo CSV en la inicialización de la aplicación
    consumo_df = pd.read_csv('datos_consumo.csv')
    X = consumo_df[['fecha', 'hora']]
    y = consumo_df['consumo']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
    modelo = LinearRegression()
    modelo.fit(X_train, y_train)

    def prediccion_consumo(self, fecha_str, hora_str):
        try:
            # Convertir fecha y hora a valores numéricos
            fecha = datetime.datetime.strptime(fecha_str, '%Y-%m-%d').toordinal()
            hora = datetime.datetime.strptime(hora_str, '%H:%M').hour

            # Realizar la predicción utilizando el modelo de regresión lineal
            prediccion = self.modelo.predict([[fecha, hora]])

            # Calcular las métricas de evaluación
            y_test_pred = self.modelo.predict(self.X_test)
            r2 = r2_score(self.y_test, y_test_pred)
            mape_e = np.mean(np.abs((self.y_test - y_test_pred) / self.y_test)) * 100

            # Devolver la predicción y las métricas de evaluación
            return {"prediccion": prediccion[0], "r2": r2, "mape_e": mape_e}

        except Exception as e:
            # Manejar errores y devolver una respuesta adecuada
            return {"error": str(e)}

    def list(self, request):
        # Obtener las fechas y horas de los parámetros de la solicitud
        fecha_str = request.GET.get('fecha')
        hora_str = request.GET.get('hora')

        # Obtener la predicción de consumo utilizando las fechas y horas
        resultado = self.prediccion_consumo(fecha_str, hora_str)
```

```
        # Devolver la respuesta adecuada
        if "error" in resultado:
            return Response(resultado, status=400)
        else:
            return Response(resultado)


def convertir_dia(dia):
    dia_semana = ""
    if dia == 1:
        dia_semana = "Lunes"
    if dia == 2:
        dia_semana = "Martes"
    if dia == 3:
        dia_semana = "Miercoles"
    if dia == 4:
        dia_semana = "Jueves"
    if dia == 5:
        dia_semana = "Viernes"
    if dia == 6:
        dia_semana = "Sabado"
    if dia == 7:
        dia_semana = "Domingo"
    return dia_semana


class Lecturas(viewsets.ViewSet):

    def list(self, request, *args, **kwargs):
        lecturasvoltaje = Tablaesp2.objects.filter(codigo='1').order_by('-id')[:20]
        lecturascorriente = Tablaesp2.objects.filter(codigo='2').order_by('-id')[:20]
        lecturasfp = Tablaesp2.objects.filter(codigo='6').order_by('-id')[:20]
        lectura_correcta = []

        for i, j, k in zip(reversed(lecturasvoltaje), reversed(lecturascorriente), reversed(lecturasfp)):
            lectura_correcta.append({"Valor": i.Valor * j.Valor * k.Valor})
        hora_actual = datetime.datetime.now()
        minuto = hora_actual.minute
        hora =  hora_actual.hour
        horas = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22]

        lectura1 = Red1.objects.filter(Q(fecha__hour__in=horas) & Q(fecha__minute=minuto))
        lectura1 =
lectura1.annotate(hora=TruncMinute('fecha')).values('hora').annotate(promedio=Avg('valor')).order_by(
            '-hora')[:20]
        lectura12Serializer = LecturasSerializer(lectura1, many=True)

        lectura22 = Red2.objects.filter(Q(fecha__hour__in=horas) & Q(fecha__minute=minuto))
        lectura22 =
lectura22.annotate(hora=TruncMinute('fecha')).values('hora').annotate(promedio=Avg('valor')).order_by(
            '-hora')[:20]
        lectura22Serializer = LecturasSerializer(lectura22, many=True)

        lectura32 = Red3.objects.filter(Q(fecha__hour__in=horas) & Q(fecha__minute=minuto))
        lectura32 = lectura32.annotate(hora=TruncMinute('fecha')).values('hora').annotate(
            promedio=Avg('valor')).order_by(
```

```
        '-hora')[:20]
    lectura32Serializer = LecturasSerializer(lectura32, many=True)
    dia_actual = 0
    y_true = []
    y_false = []
    lectura1aux = []
    lectura2aux = []
    lectura3aux = []
    for i in reversed(lectura12Serializer.data):
        dt_obj = datetime.datetime.strptime(i['fecha'], '%Y-%m-%d %H:%M:%S')
        dt_objaux = dt_obj.isoweekday()

        lectura1aux.append({"x": "{}: {}".format(convertir_dia(dt_objaux), i['fecha']), "y": i["promedio"]})
        lectura1aux.append({"x": "{}: {}".format(convertir_dia(dt_objaux), i['hora']), "y": i["promedio"]})
    dia_actual = datetime.datetime.now().isoweekday() + 1
    if dia_actual == 8:
        dia_actual = 1
    time_after = time(hora, minuto, 0)
    time_after2 = time(hora, minuto, 0)

    lectura1h1 = Red1.objects.filter(Q(fecha__week_day=dia_actual) & Q(fecha__time__gte=time_after)).aggregate(Avg('valor'))
    lectura1h2 = Red1.objects.filter(Q(fecha__week_day=dia_actual) & Q(fecha__time__gte=time_after2)).aggregate(Avg('valor'))
    lectura1aux.append({"x": "{}: {}".format(convertir_dia(dia_actual), str(hora)+':' + str(minuto) + ':00'), "y": lectura1h1['valor__avg']})
    lectura1aux.append({"x": "{}: {}".format(convertir_dia(dia_actual), str(hora + 1)+':' + str(minuto) + ':00'), "y": lectura1h2['valor__avg']})

    for i, j in zip(reversed(lectura22Serializer.data), reversed(lectura_correcta)):
        dt_obj = datetime.datetime.strptime(i['fecha'], '%Y-%m-%d %H:%M:%S')
        dt_objaux = dt_obj.isoweekday()
        y_false.append({"x": "{}: {}".format(convertir_dia(dt_objaux), i['hora']), "y": i["promedio"]})
        y_true.append({"x": "{}: {}".format(convertir_dia(dt_objaux), i['hora']), "y": j["Valor"]})
        lectura2aux.append({"x": "{}: {}".format(convertir_dia(dt_objaux), i['fecha']), "y": i["promedio"]})

    lectura2h1 = Red2.objects.filter(Q(fecha__week_day=dia_actual) & Q(fecha__time__gte=time_after)).aggregate(Avg('valor'))
    lectura2h2 = Red2.objects.filter(Q(fecha__week_day=dia_actual) & Q(fecha__time__gte=time_after2)).aggregate(Avg('valor'))
    lectura2aux.append({"x": "{}: {}".format(convertir_dia(dia_actual), str(hora)+':' + str(minuto) + ':00'), "y": lectura2h1['valor__avg']})
    lectura2aux.append({"x": "{}: {}".format(convertir_dia(dia_actual), str(hora + 1)+':' + str(minuto) + ':00'), "y": lectura2h2['valor__avg']})

    for i in reversed(lectura32Serializer.data):
        dt_obj = datetime.datetime.strptime(i['fecha'], '%Y-%m-%d %H:%M:%S')
        dt_objaux = dt_obj.isoweekday()
        lectura3aux.append({"x": "{}: {}".format(convertir_dia(dt_objaux), i['fecha']), "y": i["promedio"]})

    lectura3h1 = Red2.objects.filter(Q(fecha__week_day=dia_actual) & Q(fecha__time__gte=time_after)).aggregate(Avg('valor'))
```

```
        lectura3h2 = Red2.objects.filter(Q(fecha__week_day=dia_actual) &
Q(fecha__time__gte=time_after2)).aggregate(Avg('valor'))
        lectura3aux.append({"x": "{}: {}".format(convertir_dia(dia_actual), str(hora)+':' + str(minuto) +
':00'), "y": lectura3h1['valor__avg']})
        lectura3aux.append({"x": "{}: {}".format(convertir_dia(dia_actual), str(hora + 1)+':' + str(minuto) +
':00'), "y": lectura3h2['valor__avg']})

        dict = {
            'lectura1aux': lectura1aux,
            'lectura2aux': lectura2aux,
            'lectura3aux': lectura3aux,
            'lectura_yfalse': y_false,
            'lectura_ytrue': y_true
        }
        return Response(dict)

class MejorRedGraf(viewsets.ViewSet):
    def list(self, request, *args, **kwargs):
        mejor_red_cnn = MejorResultado.objects.filter(red='CNN').order_by('-id')[:20]
        mejor_red_lstm = MejorResultado.objects.filter(red='LSTM').order_by('-id')[:20]
        mejor_red_ann = MejorResultado.objects.filter(red='ANN').order_by('-id')[:20]

        r2_list_cnn = [mr.metrica_r2 for mr in mejor_red_cnn]
        fecha_list_cnn = [mr.fecha.strftime('%Y-%m-%d %H:%M:%S') for mr in mejor_red_cnn]
        r2_fecha_list_cnn = list(zip(r2_list_cnn, fecha_list_cnn))

        r2_list_lstm = [mr.metrica_r2 for mr in mejor_red_lstm]
        fecha_list_lstm = [mr.fecha.strftime('%Y-%m-%d %H:%M:%S') for mr in mejor_red_lstm]
        r2_fecha_list_lstm = list(zip(r2_list_lstm, fecha_list_lstm))

        r2_list_ann = [mr.metrica_r2 for mr in mejor_red_ann]
        fecha_list_ann = [mr.fecha.strftime('%Y-%m-%d %H:%M:%S') for mr in mejor_red_ann]
        r2_fecha_list_ann = list(zip(r2_list_ann, fecha_list_ann))

        data_dict = {
            'r2_cnn': r2_fecha_list_cnn,
            'r2_lstm': r2_fecha_list_lstm,
            'r2_ann': r2_fecha_list_ann,
        }
        return Response(data_dict)
```

ANEXO II: CODIGO FUENTE DE PLANTILLA (LISTA CALCULOS)

```
from django.contrib.auth.decorators import login_required
from django.http import JsonResponse, HttpResponse
from django.shortcuts import render, redirect
from django.urls import reverse_lazy
from django.views.generic import ListView, CreateView, TemplateView
from django.utils.decorators import method_decorator
from django.views.decorators.csrf import csrf_exempt
from django.db.models import Avg, Sum
from aplicacion.app_minersystem.forms import PersonalForm
from aplicacion.app_minersystem.models import Tablaesp2
from datetime import datetime
```

```
# def index(request):
#     return render(request, 'app_template/dashboard.html', {
#         'title': 'Listado de Calculos','tit_principal': 'Mineria Calculos', 'entity': 'Listado de Calculos','datos':
Tablaesp2.objects.all(),
#         'form': PersonalForm(), 'list_url' : reverse_lazy('app_minersystem:listar_doctor')
#     }, content_type='application/xhtml+xml')


class IndexView(TemplateView):
    model = Tablaesp2
    template_name = 'app_template/dashboard.html'

    @method_decorator(csrf_exempt)
    @method_decorator(login_required)
    def dispatch(self, request, *args, **kwargs):
        return super().dispatch(request, *args, **kwargs)

    def get_context_data(self, **kwargs):
        fecha = datetime.now()
        context = super().get_context_data(**kwargs)
        context['voltaje'] = Tablaesp2.objects.filter(codigo="1", fecha=fecha).last()
        context['corriente'] = Tablaesp2.objects.filter(codigo="2", fecha=fecha).last()
        context['potencia'] = Tablaesp2.objects.filter(codigo="3", fecha=fecha).last()
        context['consumo'] = Tablaesp2.objects.filter(codigo="4", fecha=fecha).last()
        context['frecuncia'] = Tablaesp2.objects.filter(codigo="5", fecha=fecha).last()
        context['pf'] = Tablaesp2.objects.filter(codigo="6", fecha=fecha).last()

        #context['voltaje'] = Tablaesp2.objects.filter(codigo="1", fecha=fecha).aggregate(Avg('Valor'))
        #context['corriente'] = Tablaesp2.objects.filter(codigo="2", fecha=fecha).aggregate(Avg('Valor'))
        #context['potencia'] = Tablaesp2.objects.filter(codigo="3", fecha=fecha).aggregate(Avg('Valor'))
        #context['consumo'] = Tablaesp2.objects.filter(codigo="4", fecha=fecha).aggregate(Avg('Valor'))
        #context['frecuncia'] = Tablaesp2.objects.filter(codigo="5", fecha=fecha).aggregate(Avg('Valor'))
        #context['pf'] = Tablaesp2.objects.filter(codigo="6", fecha=fecha).aggregate(Avg('Valor'))
        context['title'] = 'Listado de Calculos'
        context['tit_principal'] = 'Mineria Calculos'
        context['entity'] = 'Listado de Calculos'
        context['datos'] = Tablaesp2.objects.all()
        context['form'] = PersonalForm()
        context['list_url'] = reverse_lazy('app_minersystem:listar_doctor')
        print(context)
        return context
```

ANEXO III: codigo Fuente DASHBOARD.HTML

```
{% extends 'app_template/body.html' %}
{% load static %}

{% block head %}

    <link rel="{% static 'acciones/requisitos/css/bootstrap.min.css' %}" type="text/css"/>

    <link href="{% static 'acciones/requisitos/css/bootstrap.min.css' %}" rel="stylesheet"
```

```
        integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

   <script src="{% static 'acciones/requisitos/js/jquery.min.js' %}"></script>

   <script src="{% static 'acciones/requisitos/js/modernizr.min.js' %}" type="text/javascript"></script>

   <script src="{% static 'acciones/requisitos/js/bootstrap.min.js' %}"></script>

   <!--<script src="{% static 'acciones/requisitos/js/Chart.min.js' %}"></script>-->
   <script src="{% static 'acciones/requisitos/css/Chart.min.js' %}"></script>
   <script src="{% static 'acciones/requisitos/css/Chart.min.css' %}"></script>
   <script src="https://unpkg.com/brain.js@2.0.0-beta.15/dist/browser.js"></script>



{% endblock %}

{% block contenido %}

   <section class="content">

      <div class="container-fluid">

         <div class="row">

            <div class="col-12 col-sm-6 col-md-4">
               <div class="info-box">
                  <span class="info-box-icon bg-primary elevation-1"><i class="fa fa-bolt"></i></span>
                  <div class="info-box-content">
                     <span class="info-box-text">Voltaje</span>
                     <span class="info-box-number">{{ voltaje.Valor }}<small> V </small></span>
                     <!--  <span  class="info-box-number">{{  voltaje.Valor__avg  }}<small>  V
</small></span> -->
                  </div>
               </div>
            </div>
            <div class="col-12 col-sm-6 col-md-4">
               <div class="info-box">
                  <span class="info-box-icon bg-primary elevation-1"><i class="fa fa-spinner"></i></span>
                  <div class="info-box-content">
                     <span class="info-box-text">Corriente</span>
                     <span class="info-box-number">{{ corriente.Valor }}<small> A </small></span>
                     <!--<span  class="info-box-number">{{  corriente.Valor__avg  }}<small>  A
</small></span>-->
                  </div>
               </div>
            </div>
            <div class="col-12 col-sm-6 col-md-4">
               <div class="info-box">
                  <span  class="info-box-icon  bg-primary  elevation-1"><i  class="fa  fa-battery-
full"></i></span>
                  <div class="info-box-content">
                     <span class="info-box-text">Potencia</span>
                     <span class="info-box-number">{{ potencia.Valor }}<small> W </small></span>
```

```
                <!--<span class="info-box-number">{{ potencia.Valor__avg }}<small> W
</small></span>-->
                </div>
              </div>
            </div>
            <div class="col-12 col-sm-6 col-md-4">
              <div class="info-box">
                <span class="info-box-icon bg-primary elevation-1"><i class="fa fa-plug"></i></span>
                <div class="info-box-content">
                  <span class="info-box-text">Consumo</span>
                  <span class="info-box-number">{{ consumo.Valor }}<small> kWh </small></span>
                  <!-- <span class="info-box-number">{{ consumo.Valor__avg }}<small> kWh
</small></span>-->
                </div>
              </div>
            </div>
            <div class="col-12 col-sm-6 col-md-4">
              <div class="info-box">
                <span class="info-box-icon bg-primary elevation-1"><i class="fa fa-podcast"></i></span>
                <div class="info-box-content">
                  <span class="info-box-text">Frecuencia</span>
                  <span class="info-box-number">{{ frecuncia.Valor }}<small> Hz </small></span>
                  <!--<span class="info-box-number">{{ frecuncia.Valor__avg }}<small> Hz
</small></span>-->
                </div>
              </div>
            </div>
            <div class="col-12 col-sm-6 col-md-4">
              <div class="info-box">
                <span class="info-box-icon bg-primary elevation-1"><i class="fa fa-rss"></i></span>
                <div class="info-box-content">
                  <span class="info-box-text">PF</span>
                  <span class="info-box-number">{{ pf.Valor }}<small> </small></span>
                  <!--<span class="info-box-number">{{ pf.Valor__avg }}<small> </small></span>-->
                </div>
              </div>
            </div>
          </div>
          <div class="px-4 py-2 my-2 text-center border-bottom">
            <h1 class="display-5 fw-bold">Lectura</h1>
          </div>
          <div class="row">
            <div class="col-12 col-sm-6 col-md-3">
              <div class="info-box">
                <span class="info-box-icon bg-info elevation-1"><i class="fa fa-check"></i></span>
                <div class="info-box-content">
                  <span class="info-box-text">Potencia - Sensor</span>
                  <span class="info-box-number" id="red_4"></span>
                </div>
              </div>
            </div>
            <div class="col-12 col-sm-6 col-md-3">
              <div class="info-box">
                <span class="info-box-icon bg-info elevation-1"><i class="fa fa-check"></i></span>
                <div class="info-box-content">
                  <span class="info-box-text">Potencia - LSTM</span>
```

```
            <span class="info-box-number" id="red_1"></span>
          </div>
        </div>
      </div>
      <div class="col-12 col-sm-6 col-md-3">
        <div class="info-box">
          <span class="info-box-icon bg-info elevation-1"><i class="fa fa-check"></i></span>
          <div class="info-box-content">
            <span class="info-box-text">Potencia - CNN</span>
            <span class="info-box-number" id="red_2"></span>
          </div>
        </div>
      </div>
      <div class="col-12 col-sm-6 col-md-3">
        <div class="info-box">
          <span class="info-box-icon bg-info  elevation-1"><i class="fa fa-check"></i></span>
          <div class="info-box-content">
            <span class="info-box-text">Potencia - ANN</span>
            <span class="info-box-number" id="red_3"></span>
          </div>
        </div>
      </div>

    </div>
    <div class="b-example-divider"></div>
    <div class="px-4 py-2 my-2 text-center border-bottom">
      <h5 class="display-9 fw-bold">La mejor opcion: <span id="mopcion"></span></h5>
    </div>

    <div class="b-example-divider"></div>
    <div class="row">
      <div id="myapexchart"></div>
      <hr>
      <div id="myapexchart2"></div>
      <hr>
      <div id="myapexchart3"></div>
      <hr>
      <div id="myapexchart4"></div>
      <hr>
      <div id="myapexchartM"></div>

    </div>
    <div class="b-example-divider mb-0"></div>
    <hr>
    <div class="row">
      <fieldset>
        <legend>Predicción Consumo y Potencia</legend>
        <div class="row">
          <div class="col-3 col-sm-3 col-md-3">
           <input type="datetime-local" class="datetime" id="idprediccion">
          </div>
          <div class="col-3 col-sm-3 col-md-3">
            <button class="btn btn-primary" onclick="prediccion()">Predeccir</button>
            <button id="limpiar-btn" class="btn btn-danger">Limpiar</button>
          </div>
          <div class="row my-4">
```

```
                <div class="col-md-12">
                </div>
              </div>

  <div class="row">

                <div class="col-12 col-sm-6 col-md-4">
                  <div class="info-box">
                    <span class="info-box-icon bg-primary elevation-1"><i class="fa fa-plug"></i></span>
                    <div class="info-box-content">
                      <span class="info-box-text">Predicción de consumo:</span>
                      <span id= "consumo-info-box" class="info-box-number"><small></small></span>
                    </div>
                  </div>

                </div>

                 <div class="col-12 col-sm-6 col-md-4">
                  <div class="info-box">
                    <span class="info-box-icon bg-primary elevation-1"><i class="fa fa-check-circle"></i></span>
                    <div class="info-box-content">
                      <span class="info-box-text">Métrica de precisión r2:</span>
                      <span id= "precision-info-box" class="info-box-number"><small></small></span>
                    </div>
                  </div>

                 </div>

                  <div class="col-12 col-sm-6 col-md-4">
                    <div class="info-box">
                      <span class="info-box-icon bg-primary elevation-1"><i class="fa fa-check-circle"></i></span>
                      <div class="info-box-content">
                        <span class="info-box-text">Métrica MAPE:</span>
                        <span id= "mape-info-box" class="info-box-number"><small></small></span>
                      </div>
                    </div>

                  </div>
              </div>

  <div class="row">

                <div class="col-12 col-sm-6 col-md-4">
                  <div class="info-box">
                    <span class="info-box-icon bg-primary elevation-1"><i class="fa fa-battery-full"></i></span>
                    <div class="info-box-content">
                      <span class="info-box-text">Predicción de potencia:</span>
                      <span id= "potencia-info-box" class="info-box-number"><small></small></span>
```

```
                    </div>
                  </div>

                </div>

                 <div class="col-12 col-sm-6 col-md-4">
                   <div class="info-box">
                     <span class="info-box-icon bg-primary elevation-1"><i class="fa fa-check-circle"></i></span>
                     <div class="info-box-content">
                       <span class="info-box-text">Métrica de precisión r2:</span>
                       <span id= "precision_potencia-info-box" class="info-box-number"><small> </small></span>
                     </div>
                   </div>

                 </div>

                   <div class="col-12 col-sm-6 col-md-4">
                     <div class="info-box">
                       <span class="info-box-icon bg-primary elevation-1"><i class="fa fa-check-circle"></i></span>
                       <div class="info-box-content">
                         <span class="info-box-text">Métrica MAPE:</span>
                         <span id= "mape_potencia-info-box" class="info-box-number"><small> </small></span>
                       </div>
                     </div>

                   </div>
               </div>


               <div class="row">
                  <div id="myapexchart01"></div>
                  <hr>
                  <div id="myapexchart02"></div>
                  <hr>
                  <div id="myapexchart03"></div>
                  <hr>
                  <div id="myapexchart04"></div>
                </div>

             </div>
           </fieldset>


       </div>
       <div class="b-example-divider mb-0"></div>
       <br>
     </div>
  </section>

{% endblock %}

{% block scripts %}
```

```
    <script src="{% static 'lib/adminlte-3.0.4/plugins/flot/jquery.flot.js' %}"></script>
    <script src="{% static 'lib/adminlte-3.0.4/plugins/flot-old/jquery.flot.resize.min.js' %}"></script>
    <script src="{% static 'lib/adminlte-3.0.4/plugins/flot-old/jquery.flot.pie.min.js' %}"></script>
    <link href="{% static 'lib/apexcharts-bundle/dist/apexcharts.css' %}" rel="stylesheet">
    <script type="text/javascript" src="{% static 'lib/apexcharts-bundle/dist/apexcharts.min.js'
%}"></script>
    <script type="text/javascript">
        var network = new brain.NeuralNetwork();
        network.train([
            {input: {voltaje: 1, corriente: 1}, output: {potencia: 1}},
            {input: {voltaje: 0.9262, corriente: 0.014}, output: {potencia: 0.01297}},
            {input: {voltaje: 0.8982, corriente: 0.009}, output: {potencia: 0.008084}},
            {input: {voltaje: 0.9246, corriente: 0.014}, output: {potencia: 0.01294}},
            {input: {voltaje: 0.9, corriente: 0.008}, output: {potencia: 0.0072}},
            {input: {voltaje: 0.8990, corriente: 0.006}, output: {potencia: 0.005394}},
            {input: {voltaje: 1, corriente: 0}, output: {potencia: 0}},
            {input: {voltaje: 1, corriente: 0.5}, output: {potencia: 0.5}},
        ]);

        // Calcular R2
function calcularR2(datosReales, predicciones) {
    var sumaDiferenciaCuadrados = 0;
    var sumaMediaDiferenciaCuadrados = 0;
    var mediaReal = 0;

    for (var i = 0; i < datosReales.length; i++) {
        sumaDiferenciaCuadrados += Math.pow(datosReales[i] - predicciones[i], 2);
        mediaReal += datosReales[i];
    }
    mediaReal /= datosReales.length;
    for (var j = 0; j < datosReales.length; j++) {
        sumaMediaDiferenciaCuadrados += Math.pow(datosReales[j] - mediaReal, 2);
    }
    var r2 = - 11 + (sumaDiferenciaCuadrados / sumaMediaDiferenciaCuadrados);
    return r2;
}

function calcularMAPE(datosReales, predicciones) {
    var sumError = 0;
    var count = 0;

    for (var i = 0; i < datosReales.length; i++) {
        var error = Math.abs((datosReales[i] - predicciones[i]) / datosReales[i]);
        sumError += error;
        count++;
    }

    var mape = (sumError / count) * 100;
    return mape;
}



        // What is the expected output of [1,0]?
        let result = network.run({voltaje: 0.92384615, corriente: 0.002});
        console.log(result)
```

```
$(document).ready(function () {
   var socket = io.connect("http://127.0.0.1:4000/");
   socket.on('connect', function () {
      console.log("connectado al servidor");
   });
   socket.on('notificacion', function (message) {
      datas = message.replace(/&/g, "&amp;").replace(/</g, "&lt;").replace(/>/g, "&gt;");
      datos = JSON.parse(datas);
      console.log(datos);

      voltajeval = parseFloat(datos['1']) / 130;
      corrienteval = parseFloat(datos['2']) / 10;
      entrada = {voltaje: voltajeval, corriente: corrienteval};
      var resultado = network.run(entrada);

      valorpotencia = (parseFloat(datos['3'])).toFixed(4)

      const red1 = document.getElementById("red_1");
      red1.textContent = datos['red1'];
      const red2 = document.getElementById("red_2");
      red2.textContent = datos['red2'];
      const red3 = document.getElementById("red_3");
      red3.textContent = (resultado['potencia'] * 130).toFixed(4);
      const red4 = document.getElementById("red_4");
      red4.textContent = valorpotencia;

      const mopcion = document.getElementById("mopcion");

      guadara_lectura3((resultado['potencia'] * 130).toFixed(4));
      var nums = [datos['red1'], datos['red2'], (resultado['potencia'] * 130)];
      var cercano = nums.reduce(function (prev, curr) {
         return (Math.abs(curr - valorpotencia) < Math.abs(prev - valorpotencia) ? curr : prev);
      });
      var mejorred = '';
      var valorred = 0;
      if (cercano === datos['red1']) {
         mejorred = 'LSTM'
         valorred = datos['red1']
      }
      if (cercano === datos['red2']) {
         mejorred = 'CNN'
         valorred = datos['red2']
      }
      if (cercano === (resultado['potencia'] * 130)) {
         mejorred = 'ANN'
         valorred = resultado['potencia'] * 130
      }
      mopcion.textContent = mejorred

      // Calcular R2
var datosReales = [parseFloat(datos['red1']), parseFloat(datos['red2']), parseFloat(valorpotencia)];
var r2 = calcularR2(datosReales, nums);
var mape = calcularMAPE(datosReales, nums)
console.log("R2: " + r2);
console.log("MAPE: " + mape);
```

```
            guadara_mejoropcion(valorred, mejorred, r2, mape)
        });
    });


    /* END LINE CHART */
    function guadara_lectura3(valor) {
        const lectura = {"valor": valor};
        $.ajax('http://127.0.0.1:8000/api/lectura3/?valor=' + valor,
            {
                type: 'GET',
                contentType: 'application/json',
                success: function (data, status, xhr) {   // success callback function
                }
            });
    }

    function guadara_mejoropcion(valor, red, r2, map_enviado) {
        const lectura = {"valor": valor, "red": red, "r2": r2, "mape_enviado": map_enviado};
        $.ajax('http://127.0.0.1:8000/api/mejorresultado/?valor=' + valor + '&red=' + red + '&r2=' + r2 +
'&mape=' + map_enviado,
            {
                type: 'GET',
                contentType: 'application/json',
                success: function (data, status, xhr) {   // success callback function
                }
            });
    }

    function prediccion() {
        var valor = $("#idprediccion").val();
        var fecha = valor.split('T')[0]
        var hora = valor.split('T')[1]
        console.log(fecha);
        console.log(hora);

        //Esta url de api apunta a una función en
        // el views.py que no corresponde a la funcion de prediccion:
        // $.ajax('http://127.0.0.1:8000/api/mejorresultado/?fecha=' + fecha +'&hora=' + hora,

        $.ajax('http://127.0.0.1:8000/api/prediccion/?fecha=' + fecha +'&hora=' + hora,

            {
                type: 'GET',
                contentType: 'application/json',
                success: function (data, status, xhr) {   // success callback function
                    console.log(data);
                    var consumo = data.prediccion;
                    var precision = data.r2.toFixed(2);
                    var mape_e = data.mape_e.toFixed(2);

                    if (typeof consumo !== 'undefined' && !isNaN(consumo)) {
                    var consumo = consumo.toFixed(4);
                    $('#consumo-info-box').html(consumo + '<small> kWh </small>');}
```

```
                $('#precision-info-box').html(precision + '<small> de exactitud </small>');
                $('#mape-info-box').html(mape_e + '<small> % de MAPE </small>');
            },
            error: function (error){
                console.log(error); //manejar el error aqui cuando no digiten la fecha
                alert('Porfavor ingresa la fecha a la cual se quiere predecir: ' + error.statusText);
            }
        });


    //Prediccion potencia
    $.ajax('http://127.0.0.1:8000/api/prediccion_potencia/?fecha=' + fecha +'&hora=' + hora,

        {
            type: 'GET',
            contentType: 'application/json',
            success: function (data, status, xhr) {   // success callback function
                console.log(data);
                var potencia_p = data.prediccion_p;
                var precision_potencia = data.r2.toFixed(2);
                var mape_e_potencia = data.mape_e.toFixed(2);

                if (typeof potencia_p !== 'undefined' && !isNaN(potencia_p)) {
                var potencia_p = potencia_p.toFixed(4);
                $('#potencia-info-box').html(potencia_p + '<small> W </small>');}

                $('#precision_potencia-info-box').html(precision_potencia + '<small> de exactitud
</small>');
                $('#mape_potencia-info-box').html(mape_e_potencia + '<small> % de MAPE </small>');
            },

        });
    //


    var url = 'http://127.0.0.1:8000/api/lecturasred/';
    var chart = new ApexCharts(document.querySelector("#myapexchart01"), options);
    chart.render();
    var chart2 = new ApexCharts(document.querySelector("#myapexchart02"), options2);
    chart2.render();
    var chart3 = new ApexCharts(document.querySelector("#myapexchart03"), options3);
    chart3.render();
    var chart4 = new ApexCharts(document.querySelector("#myapexchart04"), options4);
    chart4.render();
    var chart5 = new ApexCharts(document.querySelector("#myapexchart05"), optionsem);
    chart5.render();

     window.setInterval(function () {
    $.getJSON(url, function (response) {
        chart.updateSeries([{
            name: 'Potencia calculada con LSTM',
            type: 'line',
            data: response.lectura1aux
        }, {
            name: 'Potencia calculada de arduino',
            type: 'line',
```

```
            data: response.lectura_ytrue
          }])
          chart2.updateSeries([{
            name: 'Potencia calculada con CNN',
            type: 'line',
            data: response.lectura2aux
          }, {
            name: 'Potencia calculada de arduino',
            type: 'line',
            data: response.lectura_ytrue
          }])
          chart3.updateSeries([{
            name: 'Potencia calculada con ANN',
            type: 'line',
            data: response.lectura3aux
          }, {
            name: 'Potencia calculada de arduino',
            type: 'line',
            data: response.lectura_ytrue
          }])
          chart4.updateSeries([{
            name: 'RED 1: LSTM',
            type: 'line',
            data: response.lectura1aux
          }, {
            name: 'RED 2: CNN',
            type: 'line',
            data: response.lectura2aux
          }, {
            name: 'RED 3: ANN',
            type: 'line',
            data: response.lectura3aux
          }, {
            name: 'Error Cuadratico',
            type: 'line',
            data: response.lectura_ytrue
          }])
          chart5.updateSeries([{
            name: 'Accuaricy',
            type: 'line',
            data: response.lectura_yfalse
          }])
        });
      }, 10000);

  }
</script>

  <script type="text/javascript">
$(document).ready(function() {
// Agregar el evento click al botón
$('#limpiar-btn').click(function() {
  $('#idprediccion').val('');
  $('#consumo-info-box').html('');
  $('#precision-info-box').html('');
  $('#mape-info-box').html('');
```

```
        $('#potencia-info-box').html('');
        $('#precision_potencia-info-box').html('');
        $('#mape_potencia-info-box').html('');
        });
});
    </script>

    <script type="text/javascript">
        var options = {
            chart: {
                height: 380,
                type: 'line' // se puede cambiar por 'area' y 'bar'
            },
            title: {
                text: 'RED 1: LSTM', // Titulo de la grafica, se muestra en la parte superior
            },
            xaxis: {
                categories: [],
            },
            series: [],
            noData: {text: "Cargando ..."},
            colors: ['#2732b0', '#b02727'],
            grid: {
                row: {
                    colors: ['#e5e5e5', 'transparent'], // color de fondo separados por coma para alternarlos
                }
            }
        }
        var options2 = {
            chart: {
                height: 380,
                type: 'line' // se puede cambiar por 'area' y 'bar'
            },
            title: {
                text: 'RED 2: CNN', // Titulo de la grafica, se muestra en la parte superior
            },
            xaxis: {
                categories: [],
            },
            series: [],
            noData: {text: "Cargando ..."},
            colors: ['#2732b0', '#b02727'],
            grid: {
                row: {
                    colors: ['#e5e5e5', 'transparent'], // color de fondo separados por coma para alternarlos
                }
            }
        }
        var options3 = {
            chart: {
                height: 380,
                type: 'line' // se puede cambiar por 'area' y 'bar'
            },
            title: {
                text: 'RED 3: ANN', // Titulo de la grafica, se muestra en la parte superior
            },
```

```
    xaxis: {
      categories: [],
    },
    series: [],
    noData: {text: "Cargando ..."},
    colors: ['#2732b0', '#b02727'],
    grid: {
      row: {
        colors: ['#e5e5e5', 'transparent'], // color de fondo separados por coma para alternarlos
      }
    }
}
var options4 = {
    chart: {
      height: 380,
      type: 'line' // se puede cambiar por 'area' y 'bar'
    },
    title: {
      text: 'REDES: varias', // Titulo de la grafica, se muestra en la parte superior
    },
    xaxis: {
      categories: [],
    },
    series: [],
    noData: {text: "Cargando ..."},
    colors: ['#2732b0', '#27b090', '#9727b0', '#b02727'],
    grid: {
      row: {
        colors: ['#e5e5e5', 'transparent'], // color de fondo separados por coma para alternarlos
      }
    }
}
var optionsem = {
    title: {
      text: 'ERROR CUADRATICO MEDIO', // Titulo de la grafica, se muestra en la parte superior
    },
    series: [{
      name: 'Line',
      type: 'line',
      data: []
    }, {
      name: 'Line',
      type: 'line',
      data: []
    }],
    chart: {
      height: 350,
      type: 'line',
    },
    fill: {
      type: 'solid',
    },
    markers: {
      size: [6, 0]
    },
    tooltip: {
```

```
        shared: false,
        intersect: true,
    },
    legend: {
        show: false
    },
    xaxis: {
        type: 'numeric',
        min: 0,
        max: 12,
        tickAmount: 12
    }
};

var url = 'http://127.0.0.1:8000/api/lecturasred/';

var chart = new ApexCharts(document.querySelector("#myapexchart"), options);
chart.render();
var chart2 = new ApexCharts(document.querySelector("#myapexchart2"), options2);
chart2.render();
var chart3 = new ApexCharts(document.querySelector("#myapexchart3"), options3);
chart3.render();
var chart4 = new ApexCharts(document.querySelector("#myapexchart4"), options4);
chart4.render();

/* CARGAR LOS DATOS DE LA GRAFICA POR PRIMERA VEZ*/
$(document).ready(function () {
    $.getJSON(url, function (response) {
        chart.updateSeries([{
            name: 'Potencia calculada con LSTM',
            type: 'line',
            data: response.lectura1aux
        }, {
            name: 'Potencia calculada de arduino',
            type: 'line',
            data: response.lectura_ytrue
        }])
        chart2.updateSeries([{
            name: 'Potencia calculada con CNN',
            type: 'line',
            data: response.lectura2aux
        }, {
            name: 'Potencia calculada de arduino',
            type: 'line',
            data: response.lectura_ytrue
        }])
        chart3.updateSeries([{
            name: 'Potencia calculada con ANN',
            type: 'line',
            data: response.lectura3aux
        }, {
            name: 'Potencia calculada de arduino',
            type: 'line',
            data: response.lectura_ytrue
        }])
        chart4.updateSeries([{
```

```
            name: 'RED 1: LSTM',
            type: 'line',
            data: response.lectura1aux
        }, {
            name: 'RED 2: CNN',
            type: 'line',
            data: response.lectura2aux
        }, {
            name: 'RED 3: ANN',
            type: 'line',
            data: response.lectura3aux
        }, {
            name: 'Error Cuadratico',
            type: 'line',
            data: response.lectura_ytrue
        }])
    });
});

/* CARGAR LOS DATOS DE LA GRAFICA CADA 10 SEGUNDOS*/
window.setInterval(function () {
    $.getJSON(url, function (response) {
        chart.updateSeries([{
            name: 'Potencia calculada con LSTM',
            type: 'line',
            data: response.lectura1aux
        }, {
            name: 'Potencia calculada de arduino',
            type: 'line',
            data: response.lectura_ytrue
        }])
        chart2.updateSeries([{
            name: 'Potencia calculada con CNN',
            type: 'line',
            data: response.lectura2aux
        }, {
            name: 'Potencia calculada de arduino',
            type: 'line',
            data: response.lectura_ytrue
        }])
        chart3.updateSeries([{
            name: 'Potencia calculada con ANN',
            type: 'line',
            data: response.lectura3aux
        }, {
            name: 'Potencia calculada de arduino',
            type: 'line',
            data: response.lectura_ytrue
        }])
        chart4.updateSeries([{
            name: 'RED 1: LSTM',
            type: 'line',
            data: response.lectura1aux
        }, {
            name: 'RED 2: CNN',
            type: 'line',
```

```
            data: response.lectura2aux
          }, {
            name: 'RED 3: ANN',
            type: 'line',
            data: response.lectura3aux
          }, {
            name: 'Error Cuadratico',
            type: 'line',
            data: response.lectura_ytrue
          }])
      });
  }, 10000);
</script>

<script type="text/javascript">
  var opcionM = {
    chart: {
      height: 380,
      type: 'area' // se puede cambiar por 'area' y 'bar'
    },
    title: {
      text: 'GRÁFICA: MÉTRICA R2', // Titulo de la grafica, se muestra en la parte superior
    },
    xaxis: {
      categories: [],
    },
    series: [],
    noData: {text: "Cargando ..."},
    colors: ['#2732b0', '#b02727', '#008000'],
    grid: {
      row: {
        colors: ['#e5e5e5', 'transparent'], // color de fondo separados por coma para alternarlos
      }
    }
  };
  var urlM = 'http://127.0.0.1:8000/api/mejorredgraf/';
  var chartM = new ApexCharts(document.querySelector("#myapexchartM"), opcionM);
  chartM.render();

  $(document).ready(function() {
$.getJSON(urlM, function(response) {
  var categories_cnn = [];
  var data_cnn = [];

  var categories_lstm = [];
  var data_lstm = [];

  var categories_ann = [];
  var data_ann = [];

  // Extraer datos de CNN
  response.r2_cnn.forEach(function(item) {
    categories_cnn.push(item[1]);
    data_cnn.push(item[0]);
  });
```

```
        // Extraer datos de LSTM
        response.r2_lstm.forEach(function(item) {
            categories_lstm.push(item[1]);
            data_lstm.push(item[0]);
        });

        // Extraer datos de ANN
        response.r2_ann.forEach(function(item) {
            categories_ann.push(item[1]);
            data_ann.push(item[0]);
        });

        // Actualizar opciones de gráfica
        chartM.updateOptions({
            xaxis: {
                categories: categories_cnn
            }
        });

        // Actualizar series de gráfica
        chartM.updateSeries([
            {
                name: 'R2 de CNN',
                type: 'line',
                data: data_cnn
            },
            {
                name: 'R2 de LSTM',
                type: 'line',
                data: data_lstm
            },
            {
                name: 'R2 de ANN',
                type: 'line',
                data: data_ann
            }
        ]);
    });
});

    </script>

{% endblock %}
```

ANEXO IV: CONEXIÓN AL SERVIDOR REDIS

```
var http = require('http');
var server = http.createServer().listen(4000, function () {
    console.log("Servidor funcionando em http://localhost:4000")
});
var io = require('socket.io').listen(server);

var redis = require('redis');
publisher = redis.createClient();
```

```
const subscribe = redis.createClient();
subscribe.subscribe('notificacion'); //    listen to messages from channel pubsub
subscribe.on('connect', () => {
   console.log('Conectado a redis')
})
//Configure socket.io to store cookie set by Django
io.sockets.on('connection', function (socket) {
   console.log("usuario conectado");
   subscribe.on("message", function (channel, message) {
      //var obj = JSON.parse(message);
      // var usuario=obj["usuario"];
      // var mensaje=obj["mensaje"];
      console.log('Probando JS');
      console.log(message);
      console.log(channel);
      console.log("Message '" + message + "' on channel '" + channel + "' arrived!");
      socket.emit("notificacion", message)
      //socket.send("message");
   });
});
```

ANEXO IV: BASE DE DATOS MYSQL

```
CREATE TABLE `auth_user` (
  `id` int(11) NOT NULL,
  `password` varchar(128) NOT NULL,
  `last_login` datetime(6) DEFAULT NULL,
  `is_superuser` tinyint(1) NOT NULL,
  `username` varchar(150) NOT NULL,
  `first_name` varchar(150) NOT NULL,
  `last_name` varchar(150) NOT NULL,
  `email` varchar(254) NOT NULL,
  `is_staff` tinyint(1) NOT NULL,
  `is_active` tinyint(1) NOT NULL,
  `date_joined` datetime(6) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Table structure for table `mejorresultado`
--

CREATE TABLE `mejorresultado` (
  `id` int(11) NOT NULL,
  `fecha` datetime(6) NOT NULL,
  `valor` decimal(19,6) NOT NULL,
  `red` varchar(20) NOT NULL,
  `metrica_mape` decimal(19,6) DEFAULT NULL,
  `metrica_r2` decimal(19,6) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------
```

```
--
-- Table structure for table `tablaesp2`
--

CREATE TABLE `tablaesp2` (
  `id` int(11) NOT NULL,
  `nombre` varchar(50) DEFAULT NULL,
  `Valor` decimal(9,2) NOT NULL,
  `codigo` varchar(1) DEFAULT NULL,
  `fecha` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Table structure for table `tabla_red1`
--

CREATE TABLE `tabla_red1` (
  `id` int(11) NOT NULL,
  `fecha` datetime(6) NOT NULL,
  `valor` decimal(19,6) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Table structure for table `tabla_red2`
--

CREATE TABLE `tabla_red2` (
  `id` int(11) NOT NULL,
  `fecha` datetime(6) NOT NULL,
  `valor` decimal(19,6) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--
-- Table structure for table `tabla_red3`
--

CREATE TABLE `tabla_red3` (
  `id` int(11) NOT NULL,
  `fecha` datetime(6) NOT NULL,
  `valor` decimal(19,6) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Indexes for dumped tables
--

--
-- Indexes for table `auth_user`
--
ALTER TABLE `auth_user`
```

```
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `username` (`username`);

--
-- Indexes for table `mejorresultado`
--
ALTER TABLE `mejorresultado`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `tablaesp2`
--
ALTER TABLE `tablaesp2`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `tabla_red1`
--
ALTER TABLE `tabla_red1`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `tabla_red2`
--
ALTER TABLE `tabla_red2`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `tabla_red3`
--
ALTER TABLE `tabla_red3`
  ADD PRIMARY KEY (`id`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `auth_user`
--
ALTER TABLE `auth_user`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `mejorresultado`
--
ALTER TABLE `mejorresultado`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `tablaesp2`
--
ALTER TABLE `tablaesp2`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
```

```
-- AUTO_INCREMENT for table `tabla_red1`
--
ALTER TABLE `tabla_red1`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `tabla_red2`
--
ALTER TABLE `tabla_red2`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT for table `tabla_red3`
--
ALTER TABLE `tabla_red3`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
```

ANEXO XI: MODELADO DE LA BASE DE DATOS

```
from django.db import models
from django.contrib.auth.models import User
from datetime import datetime
from django.forms import model_to_dict

# Create your models here.

class Tablaesp2(models.Model):
    codigo = models.CharField(max_length=1, null=True, blank=True)
    fecha = models.DateField(auto_now=True, auto_created=True)
    nombre = models.CharField(max_length=50, verbose_name='Nombre ', null=True, blank=True)
    Valor = models.DecimalField(default=0.00, max_digits=9, decimal_places=2)

    def __str__(self):
        return self.nombre

    def toJSON(self):
        item = model_to_dict(self)
        return item

    class Meta:
        db_table = 'tablaesp2'
        verbose_name = 'Tabla esp2'
        verbose_name_plural = 'Tablas esp2'
        ordering = ['id']


class Red1(models.Model):
    fecha = models.DateTimeField(auto_now=True, auto_created=True)
    valor = models.DecimalField(max_digits=19, decimal_places=6)

    def __str__(self):
        return self.valor

    def toJSON(self):
        item = model_to_dict(self)
        return item
```

```
    class Meta:
        db_table = 'tabla_red1'
        verbose_name = 'Tabla red1'
        verbose_name_plural = 'Tablas red1'
        ordering = ['id']

class Red2(models.Model):
    fecha = models.DateTimeField(auto_now=True)
    valor = models.DecimalField(max_digits=19, decimal_places=6)

    def __str__(self):
        return self.valor

    def toJSON(self):
        item = model_to_dict(self)
        return item

    class Meta:
        db_table = 'tabla_red2'
        verbose_name = 'Tabla red2'
        verbose_name_plural = 'Tablas red2'
        ordering = ['id']


class Red3(models.Model):
    fecha = models.DateTimeField(auto_now=True)
    valor = models.DecimalField(max_digits=19, decimal_places=6)

    def __str__(self):
        return self.valor

    def toJSON(self):
        item = model_to_dict(self)
        return item

    class Meta:
        db_table = 'tabla_red3'
        verbose_name = 'Tabla red3'
        verbose_name_plural = 'Tablas red3'
        ordering = ['id']


class MejorResultado(models.Model):
    fecha = models.DateTimeField(auto_now=True)
    valor = models.DecimalField(max_digits=19, decimal_places=6)
    red = models.CharField(max_length=20)
    metrica_r2 = models.DecimalField(max_digits=19, decimal_places=6, null=True)
    metrica_mape = models.DecimalField(max_digits=19, decimal_places=6, null=True)

    class Meta:
        db_table = 'MejorResultado'
        verbose_name = 'Mejor Resultado'
        verbose_name_plural = 'Mejor Resultado'
        ordering = ['id']
```