Evaluating machine learning algorithms for power demand forecasting.

Yenia Maza Diaz, Patricio G. Donato, Marcos A. Funes, Carlos M. Orallo

Abstract **----The current energy context has increased the interest in research and development of technologies for improving energy efficiency. Demand prediction and optimization of energy systems is crucial to ensure the efficiency and sustainability of the electricity sector. Its impact on infrastructure planning, management of consumption peaks and improved response to emergencies will contribute to a more reliable electricity supply adapted to future needs. Currently, there is great interest in using computational tools and machine learning algorithms for these purposes, due to their great capacity and flexibility to generate models that fit real data. This work shows the results of training and testing different time series forecasting algorithms applied to predicting active power demand in a university building. To do so, the scikit-learn package and the institution's historical power demand records have been used. The results obtained are promising and show different behaviors depending on the type of model, which serves as a basis for further study in multiple databases in the future.**

Index Terms**—Smart Grids, Machine Learning, Demand Forecasting.**

I. Introducción

La predicción de requerimientos futuros de potencia activa [1] es un factor crítico para la planificación y optimización de los sistemas energéticos. La capacidad de anticipar adecuadamente las necesidades futuras de potencia permite a los gestores y planificadores tomar decisiones informadas sobre la asignación y distribución de recursos energéticos a lo largo del tiempo. Si bien la predicción de la demanda de potencia activa es un problema complejo, influenciado por múltiples variables y patrones no lineales, existen diversos modelos de aprendizaje automático que han demostrado su efectividad en este contexto. Estos modelos utilizan datos en tiempo real de consumo, generación y condiciones ambientales para mejorar la precisión de las predicciones [2], identificando patrones complejos y tendencias en la demanda eléctrica.

Este trabajo ha sido realizado gracias al financiamiento de CONICET (PIP 2643) y la Universidad Nacional de Mar del Plata
Y. Maza Díaz es becaria doctoral del CONICET, Mar del Plata, Argentina (e-mail: yenialuna1992@gmail.com)
P. G. Donato es investigador del CONICET y profesor de la UNMDP, Mar del Plata, Argentina (e-mail: donatopg@fi.mdp.edu.ar).
M. A. Funes es investigador del CONICET y profesor de la UNMDP, Mar del Plata, Argentina (e-mail: mafunes@fi.mdp.edu.ar).
C. M. Orallo es profesor de la UNMDP, Mar del Plata, Argentina (e-mail: orallo@fi.mdp.edu.ar).

El procesamiento y análisis de datos relativos a parámetros eléctricos [3] es un área de creciente interés en la comunidad científica. Tradicionalmente, este procesamiento y análisis se ha realizado a través de herramientas de tipo estadísticas u otros enfoques analíticos de tipo lineal, como regresiones, funciones de correlación, etc. En la medida que los sistemas de cómputo han incrementado notoriamente su capacidad de cálculo, han surgido nuevos enfoques en este campo. En los últimos años, una cantidad considerable de trabajos se han centrado en el uso de herramientas de inteligencia computacional para la extracción de información, a partir de los datos crudos [4]. En este contexto, la aplicación de técnicas de aprendizaje automático para la predicción de variables eléctricas ha demostrado ser una herramienta poderosa para optimizar la gestión y control de los sistemas energéticos [5].

El siguiente estudio se realiza con el fin de evaluar diferentes algoritmos de aprendizaje automático que son ampliamente utilizados en el análisis predictivo y la clasificación. La elección entre estos algoritmos dependerá de la naturaleza de los datos, los requisitos computacionales y la interpretabilidad. En este sentido se exploran y evalúan diferentes técnicas de aprendizaje automático para estimar los requerimientos futuros de potencia activa en uno de los edificios perteneciente a la Universidad Nacional de Mar del Plata. La validación del rendimiento de los modelos predictivos implica el cálculo y análisis de métricas de error, lo cual brinda información valiosa para tomar una decisión fundamentada sobre su implementación. Mediante el aprovechamiento de datos históricos de consumo eléctrico y el empleo de técnicas de aprendizaje automático avanzadas, se busca obtener modelos de pronóstico que faciliten la gestión eficiente de los recursos energéticos en el entorno universitario.

II. Análisis de series de datos

El desarrollo de este estudio, se efectúa a partir de un repositorio de datos históricos que registra las mediciones de potencia activa, reactiva y aparente en la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata. Este conjunto de datos abarca el período comprendido entre el 1 de agosto de 2022 y el 6 de mayo de 2024. Los datos representan mediciones de potencia activa, reactiva y aparente tomadas cada 15 minutos.

Previo al entrenamiento de los modelos, se realiza un análisis preliminar de los datos recolectados, con el objetivo

de identificar posibles anomalías, valores faltantes o inconsistencias. La detección y el tratamiento adecuado de estos problemas son fundamentales para garantizar la calidad y confiabilidad de los datos de entrada, lo cual es crucial para el rendimiento óptimo de los modelos predictivos.

En la Fig.1 se muestra la serie temporal de potencia activa durante el rango de tiempo entre el 1 de agosto 2022 y el 6 de mayo 2024. A simple vista se distingue un marcado patrón estacional anual. En concreto se aprecia un pico de consumo centrado en el mes de agosto. Este máximo invernal parece alinearse con el aumento de uso de calefacción eléctrica típico de esa época en el sur. Asimismo, se observan caídas acusadas en la demanda eléctrica durante los meses de verano, enero-marzo, lo que está relacionado con el período vacacional, cuando disminuye la actividad laboral y el flujo de personas. Durante el verano también influye el mayor número de horas de luz solar propio de la época estival, que reduce la necesidad de alumbrado artificial. Un modelo que capture adecuadamente este comportamiento cíclico y sus causas subyacentes será capaz de realizar mejores predicciones de la demanda eléctrica para diferentes períodos del año.

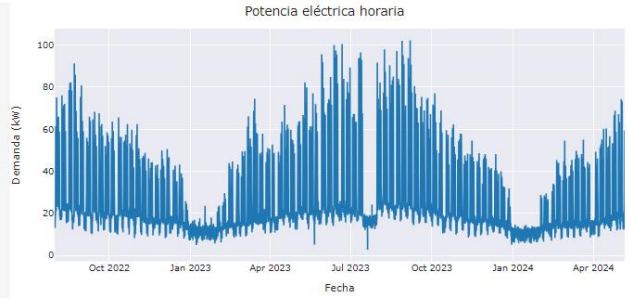

Fig. 1. Visualización de patrones estacionales.

Las series temporales de alta frecuencia como el consumo eléctrico horario suelen presentar variabilidad y ruido que dificultan la identificación de patrones en el corto plazo. Viendo la serie completa no se distingue fácilmente si existen ciclos recurrentes intradiarios.

Al focalizar en un periodo limitado como del 1º de mayo de 2023 a la primera quincena de agosto, podemos apreciar con claridad que si existen fluctuaciones o regularidades a lo largo de los meses (Fig.2). Estos ciclos pasarían desapercibidos en la serie temporal completa.

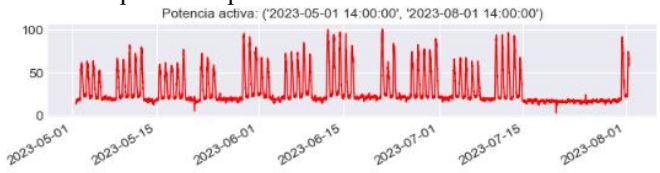

Fig. 2. Visualización de patrones por mes.

Al aplicar este acercamiento sobre la serie de potencia activa, emerge un patrón semanal subyacente. Los días laborables de lunes a viernes exhiben un consumo sistemáticamente superior, mientras que los fines de semana muestran un valle de demanda reducida. Además, se distingue una clara correlación entre la curva de consumo diario y la del día previo. Por ejemplo, los martes suelen seguir la tendencia marcada por los lunes, como se muestra en la Fig.3.

Fig. 3. Visualización de patrones por semanas del mes de junio 2023.

Realizando un análisis de la Fig.3 en la semana del 19 al 23 de junio, se observa un cambio en el patrón cíclico aproximado de las semanas. Esto se debe a que el 19 y 20 de junio fueron feriados, por lo que en el comportamiento de la semana se observan solo 3 picos de consumo.

Un análisis estadístico de los datos de consumo evidencia la componente estacional vinculada a factores climáticos. El uso de aire acondicionado en verano y el periodo vacacional como factores fundamentales. La utilización de la calefacción eléctrica en invierno aumenta notablemente la demanda de energía. El comienzo del curso escolar también parece incidir en la demanda, como se muestra en la Fig. 4.

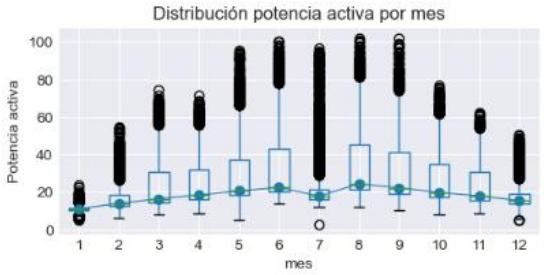

Fig. 4. Distribución de potencia activa anual.

La mediana de potencia activa disminuye notablemente (ver Fig.4) en los meses de invierno. Notándose que los meses de invierno muestran mayor variabilidad (cajas y bigotes más amplios), lo que indica una demanda más fluctuante en esta época. Se nota un aumento en la demanda alrededor de septiembre y octubre, coincidiendo con el inicio de la primavera.

La Fig. 5 revela una estacionalidad semanal, con niveles de consumo que caen durante los fines de semana en comparación a los días laborables. Los ciclos semanales de trabajo y descanso se traducen en fluctuaciones predecibles de la demanda a lo largo de la semana.

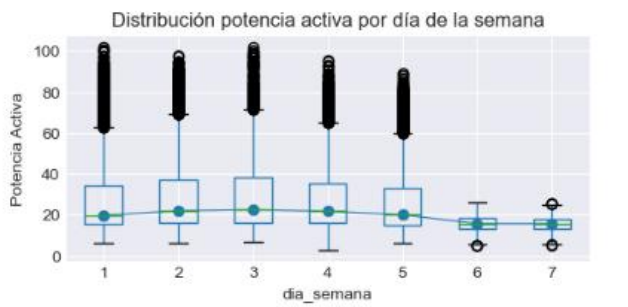

Fig. 5. Distribución de Potencia Activa por día de la semana.

A partir de un análisis de la demanda en función de las horas del día, Fig. 6, se observa en la serie analizada una reducción sistemática de la demanda entre las 17 y 21 horas, correspondiendo a las últimas horas laborales y primeras nocturnas. Este valle se ve contrastado por incrementos matinales y al mediodía, al reanudarse la actividad, como se observa en la Fig.6.

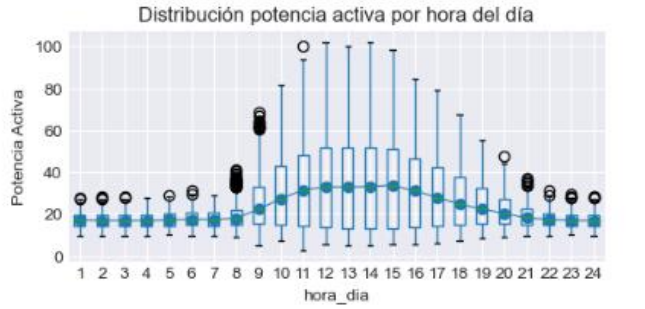

Fig. 6. Distribución de potencia activa por hora del día.

III. Modelos de pronóstico

Para implementar los algoritmos de pronóstico se ha empleado el lenguaje de programación *Python*, debido a que es uno de los de uso más extendido en el campo del análisis de datos [6]. En particular se ha empleado el paquete *Scikit Learn* [7], que se caracteriza por una interfaz consistente que facilita el intercambio de diferentes algoritmos en el código. Este paquete proporciona una amplia gama de implementaciones para los algoritmos de aprendizaje supervisado y no supervisado, incluyendo regresión, clasificación, clustering [8], reducción de dimensionalidad, selección de modelos y preprocesamiento. Se integra bien con otras bibliotecas científicas de Python, estas características hacen que *scikit-learn* sea una excelente opción por su amplia gama de tareas de aprendizaje automático, especialmente para prototipado rápido y para trabajar con conjuntos de datos que caben en la memoria.

En el ámbito del aprendizaje automático supervisado, es común entrenar y evaluar diversos modelos para un mismo problema, con el fin de identificar cuál tiene el mejor rendimiento predictivo. En este caso particular, el objetivo es encontrar el modelo más preciso para predecir la potencia activa a partir del set de datos. El rendimiento de los modelos también dependerá en gran medida de la calidad y la cantidad de datos disponibles para el entrenamiento. Además, como todo algoritmo de aprendizaje automático, su desempeño también dependerá de la configuración de hiperparámetros usada. Éstos son parámetros cuyo valor se utiliza para controlar el proceso de aprendizaje y que no se aprenden a través del mismo.

Al comparar diferentes modelos, se pueden obtener diferentes perspectivas valiosas sobre las fortalezas y debilidades de cada algoritmo, lo que puede guiar la selección del enfoque más adecuado para abordar el problema de manera efectiva. En las siguientes subsecciones se describen los modelos empleados en este trabajo.

A. K-Nearest Neighbors (kNN)

El algoritmo *K-Nearest Neighbors (KNN)* [10] es uno de los métodos de aprendizaje automático más intuitivos. A pesar de su simplicidad, *KNN* puede ser efectivo en una amplia gama de problemas, especialmente cuando se combina con técnicas adicionales. La idea fundamental detrás de *KNN* es encontrar las observaciones de entrenamiento más similares a la instancia que se desea predecir, y utilizar sus valores de respuesta para hacer una predicción.

La elección del valor de K es un hiperparámetro crucial que influye en el rendimiento del modelo. Un valor pequeño de K puede hacer que el modelo sea demasiado sensible al ruido, mientras que un valor grande de K puede suavizar demasiado los patrones subyacentes. A pesar de su simplicidad conceptual, *KNN* puede ser computacionalmente costoso, especialmente con conjuntos de datos grandes y de alta dimensionalidad.

B. Ridge y Lasso

Las técnicas de regresión como *Ridge y Lasso* [11] se caracterizan por prevenir el sobreajuste, mejorando la generalización al penalizar la complejidad del modelo. La regresión *Ridge* es una variante de la regresión lineal que agrega una penalización a la suma de los cuadrados de los coeficientes del modelo durante el proceso de entrenamiento. Esta penalización, es controlada por el hiperparámetro alfa (α), que permite ajustar el nivel de regularización, influyendo directamente en el equilibrio entre el ajuste del modelo y su capacidad de generalización. Tiene el efecto de reducir la magnitud de los coeficientes, lo que a su vez disminuye la varianza del modelo y lo hace menos propenso al sobreajuste.

La regresión *Lasso* (*Least Absolute Shrinkage and Selection Operator*) es otra técnica de regularización que agrega una penalización diferente en el proceso de entrenamiento. En lugar de penalizar la suma de los cuadrados de los coeficientes, Lasso penaliza la suma de los valores absolutos de los coeficientes. Esta penalización fuerza a que algunos coeficientes sean exactamente cero, eliminando así ciertas variables del modelo. Esta simplificación del modelo puede reducir el sobreajuste del mismo.

C. Random Forest

Random Forest [12] construye una gran cantidad de árboles de decisión individualmente entrenados en diferentes submuestras aleatorias de los datos originales. Durante el proceso de entrenamiento de cada árbol, se selecciona aleatoriamente un subconjunto de las variables predictoras disponibles. Este enfoque de muestreo aleatorio de datos y variables introduce aleatoriedad en el proceso de construcción de los árboles, lo que reduce la correlación entre ellos.

Para realizar una predicción en un nuevo punto de datos, se obtiene la predicción de cada árbol individual en el bosque. En el caso de problemas de clasificación, se asigna la clase más votada por los árboles. En problemas de regresión, se calcula el promedio de las predicciones de los árboles.

Random Forest es ampliamente utilizado debido a su

capacidad para manejar conjuntos de datos de alta dimensión, su robustez al ruido y su interpretabilidad relativamente alta en comparación con otros métodos de aprendizaje automático. Sin embargo, es importante ajustar adecuadamente los hiperparámetros mediante técnicas como la validación cruzada para obtener un rendimiento óptimo.

D. Gradient Boosting Trees

Gradient Boosting Trees [13] es un algoritmo de aprendizaje automático que se basa en la técnica de boosting [12] para mejorar iterativamente el rendimiento del modelo al combinar múltiples árboles de decisión. El proceso de *boosting* en este algoritmo se caracteriza por los siguientes aspectos: aprendizaje secuencial, optimización de la función de pérdida, contribución ponderada, regularización, manejo de errores residuales, capacidad de capturar relaciones no lineales y adaptabilidad.

Comienza ajustando un árbol de decisión inicial a los datos de entrenamiento. Luego, se ajusta un segundo árbol que intenta predecir los residuos (errores) del primer árbol. Este proceso se repite iterativamente, donde cada nuevo árbol aprende a predecir los errores residuales del modelo del conjunto anterior.

En cada iteración, se agrega un nuevo árbol débil al modelo general, con un factor de ponderación que controla la influencia del nuevo árbol en el modelo. Este factor de ponderación, conocido como tasa de aprendizaje, es un hiperparámetro clave que ayuda a evitar el sobreajuste.

E. Stacking

Stacking [14] es una técnica de aprendizaje automático que combina múltiples modelos individuales para mejorar el rendimiento predictivo general. Se basa en la premisa de que diferentes modelos pueden identificar patrones complementarios en los datos, y al combinarlos de manera adecuada, se pueden obtener predicciones más precisas que lograría con cualquiera de los modelos individuales.

Una vez entrenado, el modelo puede utilizarse para realizar predicciones en nuevos datos. Para hacer una predicción, primero se aplican los modelos base que son los modelos individuales que se combinan para formar el modelo final a los nuevos datos, obteniendo sus predicciones individuales. Luego, estas predicciones se utilizan como entrada a un modelo de nivel superior, que combina las predicciones de los modelos base para generar la predicción final.

Stacking puede ser especialmente efectivo cuando los modelos base capturan diferentes aspectos del problema y sus errores no están correlacionados. Al combinar sus predicciones, el modelo puede aprovechar las fortalezas individuales de cada modelo base y compensar sus debilidades.

Sin embargo, Stacking también tiene algunas desventajas, como un mayor costo computacional debido a la necesidad de entrenar múltiples modelos, y la posibilidad de sobreajuste si no se implementa correctamente.

IV. Selección de hiperparámetros

La selección de hiperparámetros es crucial para obtener un buen desempeño en los modelos de aprendizaje automático. Los hiperparámetros son valores que se establecen antes del entrenamiento y que influyen en el comportamiento del modelo. Ejemplos comunes de hiperparámetros son la tasa de aprendizaje, la regularización, el número de capas ocultas en una red neuronal, entre otros.

Desafortunadamente, no existe una fórmula exacta para determinar los valores óptimos de los hiperparámetros. A pesar de que los expertos en aprendizaje automático desarrollan cierta intuición con la experiencia, la forma más efectiva de encontrar la configuración ideal es mediante prueba y error. Este proceso, conocido como ajuste de hiperparámetros, implica evaluar sistemáticamente el desempeño del modelo con diversas combinaciones de valores de hiperparámetros.

Una técnica habitual para el ajuste de hiperparámetros es la búsqueda en cuadrícula, donde se define un rango de valores para cada hiperparámetro y se evalúa el modelo con todas las posibles combinaciones. Aunque exhaustiva, esta técnica puede ser computacionalmente costosa, especialmente cuando hay muchos hiperparámetros involucrados.

El hiperparámetro clave para todos los algoritmos analizados es alpha, también conocido como el parámetro de regularización. Una forma efectiva de encontrar el valor óptimo de alpha es mediante validación cruzada repetida. Este método divide los datos en múltiples particiones, entrenando y evaluando el modelo en cada una de ellas. Luego, se promedian los resultados para obtener una estimación más confiable del desempeño. Este valor controla la intensidad de la penalización aplicada a los coeficientes del modelo durante el entrenamiento.

El hiperparámetro α en la regresión *Ridge* controla la magnitud de la penalización aplicada a los coeficientes del modelo. A medida que α aumenta, se impone una mayor restricción sobre los coeficientes, lo que conduce a una reducción de la varianza, una atenuación del efecto de la multicolinealidad entre predictores y una minimización del riesgo de sobreajuste (overfitting). Con el objetivo de optimizar el rendimiento del modelo, se procedió a reajustar un modelo de regresión *Ridge* utilizando diversos valores de α.

En la gráfica izquierda de la Fig.7 se observa la evolución del error del hiperparámetro α, que a medida que este aumenta, los errores medio de prueba y entrenamiento tienden a disminuir (los valores se vuelven menos negativos) hasta cierto punto, después del cual el error de prueba comienza a aumentar nuevamente. En la gráfica derecha se observa la variabilidad en los resultados de diferentes particiones de datos en la validación cruzada.

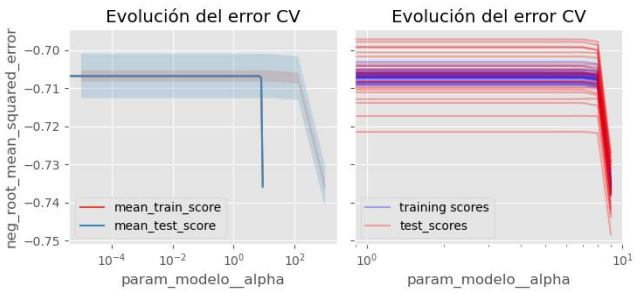

Fig.7. Evolución del error del parámetro α.

El modelo resultante de este proceso de reajuste se almacena en el atributo best_*estimator*_, facilitando así su acceso directo para posteriores predicciones o análisis sin necesidad de un entrenamiento adicional por parte del usuario. Esta funcionalidad no solo optimiza la eficiencia computacional, sino que también asegura que el modelo final capitalice toda la información disponible en los datos de entrenamiento, potencialmente mejorando su capacidad de generalización.

Respecto al algoritmo *K-Nearest Neighbors (kNN)*, en la Fig.8 se muestra el resultado de un análisis del efecto del hiperparámetro n_neighbors. Teniendo en cuenta que no hay forma de saber de antemano los mejores valores para los hiperparámetros, es necesario probar todos los valores posibles para conocer los valores óptimos.

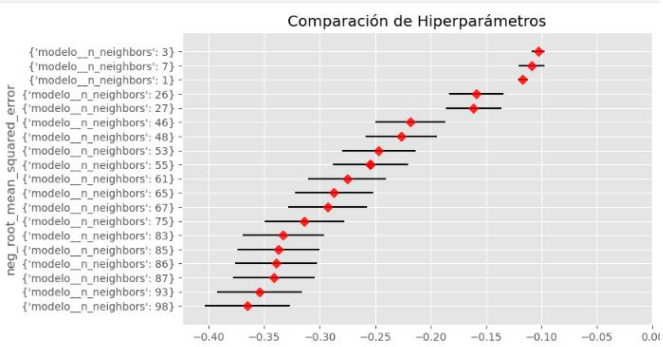

Fig.8. Comparación de hiperparámetros para el modelo K-Nearest Neighbors (kNN).

La Fig.8 presenta una serie de modelos *KNN* con diferentes valores de n_neighbors, que van desde 1 hasta 98. Para cada modelo, se muestra un intervalo de confianza (representado por una línea horizontal) y un valor medio (indicado por un punto rojo). Los resultados se muestran en una escala de error negativo (el error cuadrático medio) indicando que los valores más cercanos a cero indican un mejor rendimiento del modelo.

Se observa una tendencia general donde los modelos con valores más bajos de n_neighbors (entre 1 y aproximadamente 20) muestran un rendimiento más variable y potencialmente peor, con intervalos de confianza más amplios. A medida que aumenta el número de vecinos, el rendimiento tiende a estabilizarse, con intervalos de confianza más estrechos.

Los modelos con valores de n_neighbors entre aproximadamente 20 y 60 muestran el mejor rendimiento, con valores de error más cercanos a cero e intervalos de confianza relativamente estrechos. Después de este punto, se observa una ligera tendencia hacia un peor rendimiento a medida que n_neighbors sigue aumentando.

V. Resultados de los pronósticos

En esta sección, se presentan los resultados obtenidos al aplicar los algoritmos al set de datos. Para evaluar y comparar el rendimiento de estos métodos, se utilizaron diversas métricas y visualizaciones que proporcionan una visión comprehensiva de su eficacia predictiva.

A. Métricas de rendimiento.

Para evaluar y comparar el rendimiento de los algoritmos presentados en la sección III, se utilizaron diversas métricas de evaluación. La Tabla 1 resume las métricas clave para cada algoritmo en unidades, incluyendo el Error Cuadrático Medio (MSE), la Raíz del Error Cuadrático Medio (RMSE), el Error Absoluto Medio (MAE) y el coeficiente de determinación (R^2).

Tabla 1. Comparación de métricas de rendimiento entre Algoritmos.

Algoritmos	MSE	RMSE	MAE	R^2
Ridge y Lasso	1.4308	0.7509	0.0541	0.008
KNN	0.0062	0.0841	0.0416	1.000
Gradient Boosting	0.0444	0.0233	0.1540	0.998
Random Forest	0.0012	0.0324	0.0132	1.000
Stacking	0.0010	0.0322	0.0143	1.000

Los resultados revelan variaciones significativas en el rendimiento entre los algoritmos analizados. *Stacking* demuestra el mejor desempeño general, con el error cuadrático medio más bajo y un coeficiente de determinación perfecto, sugiriendo una excelente capacidad predictiva. *Random Forest* muestra un rendimiento muy cercano a *Stacking*, con un MSE ligeramente superior y también un R^2 de 1.000, indicando una alta precisión en sus predicciones.

K- Nearest Neighbors presenta un desempeño notable, con un MSE de 0.0062 y un R^2 de 1.000, situándose como una alternativa robusta. Mientras que *Gradient Boosting*, con un MSE más alto, mantiene un R^2 muy alto, demostrando una buena capacidad de generalización.

Sin embargo, *Ridge* y *Lasso* muestran un rendimiento significativamente inferior en particular, exhiben el MSE más alto y el R^2 más bajo, indicando una capacidad predictiva limitada en este contexto.

Es importante notar la disparidad entre los valores de MSE y R^2 para algunos algoritmos, especialmente en el caso de *Ridge* y *Lasso*. Esto podría sugerir particularidades en la distribución de los datos que afectan de manera diferente a las distintas métricas de evaluación. Stacking y Random Forest

emergen como los métodos más efectivos para este conjunto de datos particular, seguidos de cerca por KNN.

Teniendo en cuenta que Gradient Boosting ofrece un rendimiento sólido, los métodos de regularización *Ridge* y *Lasso* parecen menos adecuados para este problema específico.

B. Gráficos de residuos.

El análisis de los gráficos de residuos proporciona perspectivas sobre el rendimiento de los diferentes algoritmos de aprendizaje automático empleados en este estudio. A continuación, se presenta una interpretación detallada de cada gráfico.

1) Ridge y Lasso:

Ambos algoritmos muestran patrones similares en sus gráficos de residuos. Se observa una forma de abanico, con una mayor dispersión de residuos a medida que aumentan los valores predichos. Este fenómeno se denomina heterocedasticidad [16] lo cual podría indicar que estos modelos lineales no capturan completamente la complejidad de los datos.

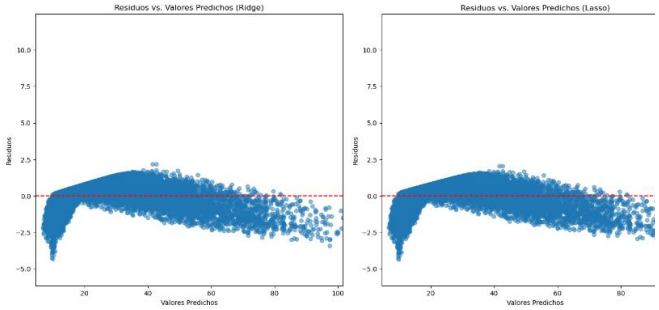

Fig. 9. Gráfico de Residuos Ridge y Lasso

2) K-Nearest Neighbors (KNN):

La Fig.10 de *KNN* muestra una distribución de residuos más uniforme en comparación con *Ridge* y *Lasso*. La ausencia de patrones claros o tendencias en la distribución de los residuos sugiere que el modelo no presenta sesgos sistemáticos en sus predicciones a lo largo del rango de valores predichos, lo que podría indicar que el modelo no está capturando completamente algunas relaciones no lineales de los datos.

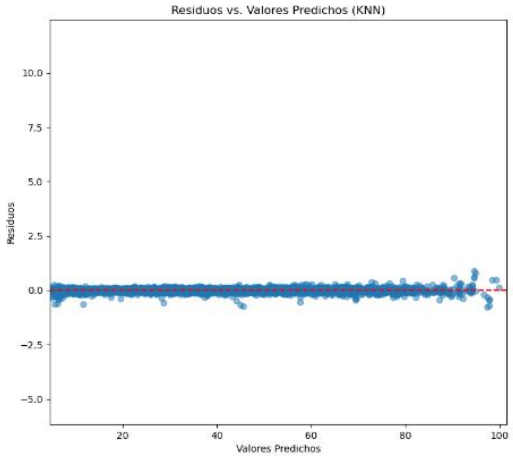

Fig. 10. Gráfico de Residuos K-Nearest Neighbors (KNN)

3) Gradient Boosting Trees:

Este algoritmo presenta una distribución de residuos más compacta y centrada alrededor de cero, especialmente para valores predichos medios. No obstante, se observa en la Fig.11 una expansión de los residuos en los extremos, lo que sugiere que el modelo podría tener dificultades con valores atípicos o extremos.

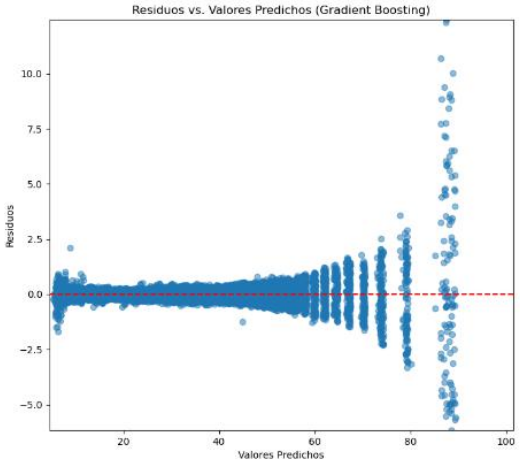

Fig. 11. Gráfico de Residuos Gradient Boosting Trees.

4) Random Forest:

El gráfico de *Random Forest* muestra una de las distribuciones de residuos más uniformes y centradas alrededor de cero. La consistencia en la dispersión de los residuos a lo largo de todo el rango de valores predichos indica un buen rendimiento general del modelo y sugiere que captura eficazmente tanto relaciones lineales como no lineales como se observa en la Fig.12.

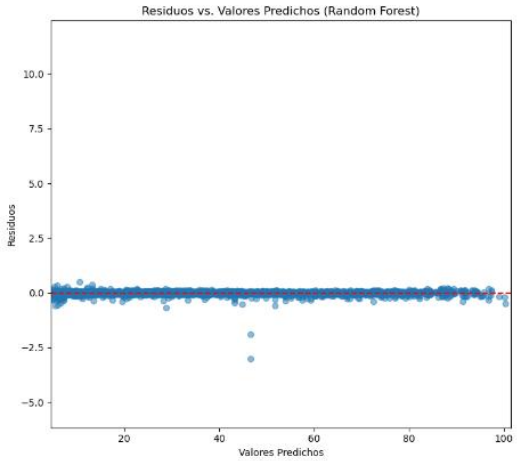

Fig. 12. Gráfico de Residuos Random Forest.

5) Stacking:

El modelo de *Stacking* exhibe la distribución de residuos más compacta y uniforme de todos los algoritmos analizados (ver Fig.13). La consistencia en la dispersión de los residuos y su concentración cercana a cero a lo largo de todo el rango de valores predichos sugiere que este enfoque de *Stacking* ha logrado combinar eficazmente las fortalezas de los modelos individuales, resultando en predicciones más precisas y robustas.

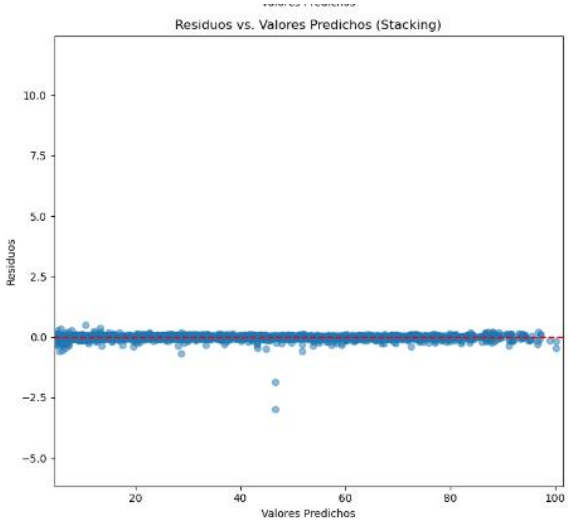

Fig. 13. Gráfico de Residuos Stacking.

Estos gráficos de residuos revelan que los modelos basados en árboles como son *Gradient Boosting* y *Random Forest*, y el método de ensemble Stacking muestran un mejor rendimiento general en términos de la distribución de residuos, en comparación con los métodos lineales *Ridge* y *Lasso*. Sin embargo, desde el punto de vista del comportamiento de los residuos *KNN* parece funcionar tan bien como *Stacking* y *Gradient Boost*. Mientras Stacking, en particular, ofrece las predicciones más consistentes y precisas en todo el rango de valores, lo que corrobora los resultados cuantitativos presentados anteriormente en las métricas de evaluación.

C. Gráficos de dispersión.

Los gráficos de dispersión ofrecen una perspectiva visual sobre el rendimiento de los diferentes algoritmos de aprendizaje automático, comparando los palores predichos versus valores reales de potencia activa en kW, empleados en este estudio.

1) Ridge y Lasso:

Ambos algoritmos muestran patrones muy similares, con una línea de dispersión que se ajusta estrechamente a la diagonal ideal. Esto indica una fuerte correlación entre los valores predichos y los valores reales, sugiriendo un buen rendimiento general. Sin embargo, se observa en la Fig.14 una ligera dispersión en los valores más altos, lo que podría indicar una pequeña tendencia a subestimar en ese rango.

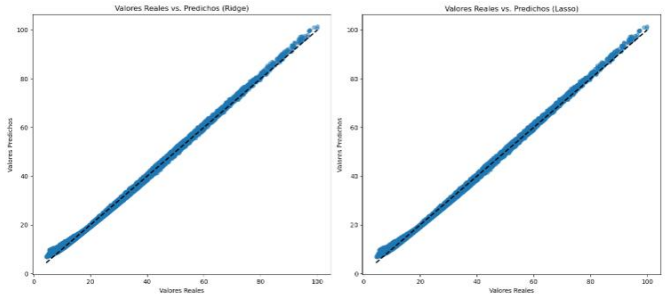

Fig. 14. Gráfico de dispersión de Ridge y Lasso.

2) K-Nearest Neighbors (KNN):

La Fig.15 presenta una línea de dispersión que se adhiere muy cerca de la diagonal ideal en todo el rango de valores. La consistencia en la alineación sugiere que *KNN* logra predicciones precisas tanto para valores bajos como altos, demostrando su capacidad para capturar eficazmente las relaciones en los datos.

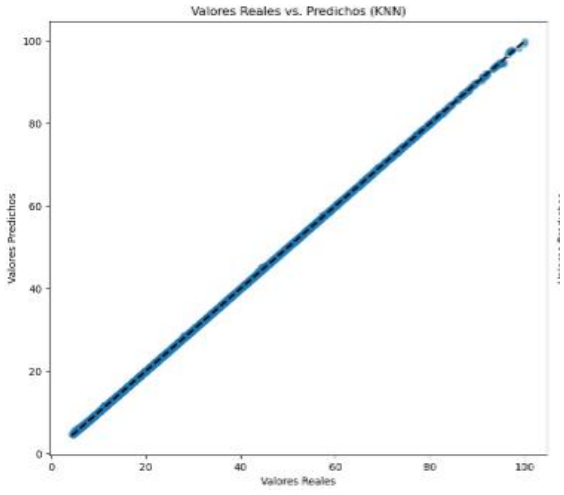

Fig.15. Gráfico de dispersión K-Nearest Neighbors.

3) Gradient Boosting Trees:

Este algoritmo muestra una excelente alineación con la diagonal ideal, con puntos agrupados muy cerca de la línea perfecta de predicción. Se observa en la Fig.16, una dispersión en los valores más altos, pero en general, el modelo demuestra

un rendimiento robusto en todo el rango de valores.

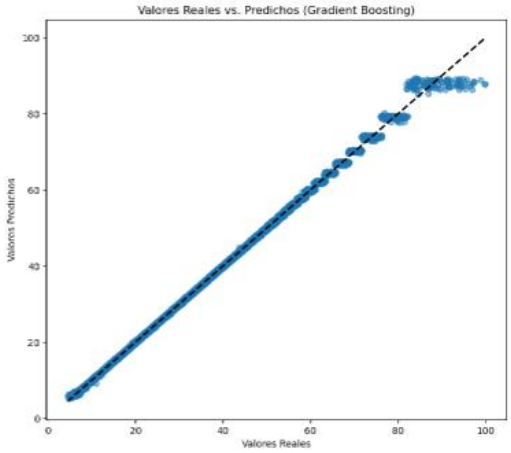

Fig. 16. Gráfico de dispersión Gradient Boosting Trees.

4) Random Forest:

La Fig.17, exhibe una de las mejores alineaciones con la diagonal ideal entre todos los algoritmos. La dispersión es mínima a lo largo de todo el rango de valores, indicando predicciones precisas y consistentes. Este rendimiento sugiere que Random Forest captura eficazmente tanto las relaciones lineales como no lineales en los datos.

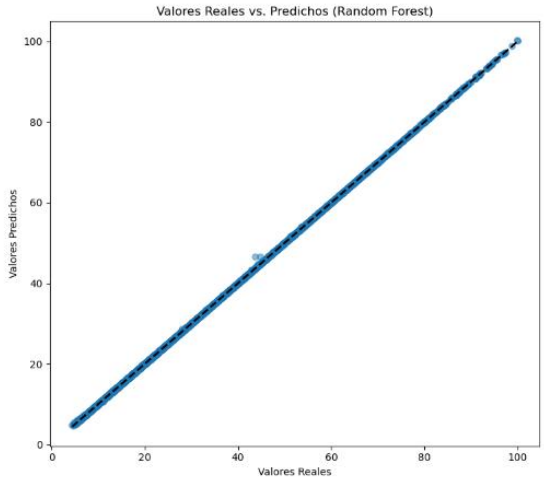

Fig.17. Gráfico de dispersión Random Forest.

5) Stacking:

El modelo Stacking (ver Fig.18) presenta una alineación más ajustada con la diagonal ideal. Además, la dispersión es mínima en todo el rango de valores, demostrando una capacidad predictiva buena. Este rendimiento superior sugiere que el enfoque de ensemble ha logrado combinar eficazmente las fortalezas de los modelos individuales, teniendo como resultado predicciones más precisas y robustas.

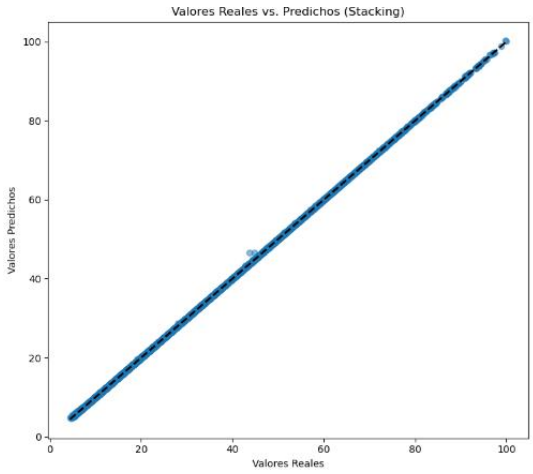

Fig. 18. Gráfico de dispersión Stacking.

Los algoritmos implementados muestran un buen rendimiento general, como lo evidencia la estrecha alineación de los puntos con la diagonal ideal en sus respectivos gráficos de dispersión. Destacando que *Ridge* y *Lasso* muestran un rendimiento sólido que implica que estos métodos son confiables, versátiles y efectivos en situaciones prácticas de modelado estadístico y machine learning, con una ligera dispersión en valores altos y bajos.

K-Nearest Neighbors demuestra una consistencia notable en todo el rango de valores. Mientras que, *Gradient Boosting Trees* ofrece predicciones precisas y una dispersión en los valores más altos.

Random Forest presenta uno de los mejores rendimientos, con una alineación muy estrecha con respecto a la diagonal ideal. Por otra parte, Stacking sobresale con la alineación más ajustada, sugiriendo el mejor rendimiento general.

D. Curvas ROC.

El gráfico de Curvas ROC (Receiver Operating Characteristic), ofrece una comparación visual del rendimiento de varios algoritmos de clasificación. Este análisis permite evaluar la capacidad de discriminación de los modelos en un problema de clasificación binaria.

En la Fig.19 se muestran las curvas ROC para los seis algoritmos. La línea diagonal punteada representa el rendimiento de un clasificador aleatorio, el desempeño de un modelo mejora a medida que su curva se aleja de esta línea hacia la esquina superior izquierda, indicando una mayor capacidad discriminativa.

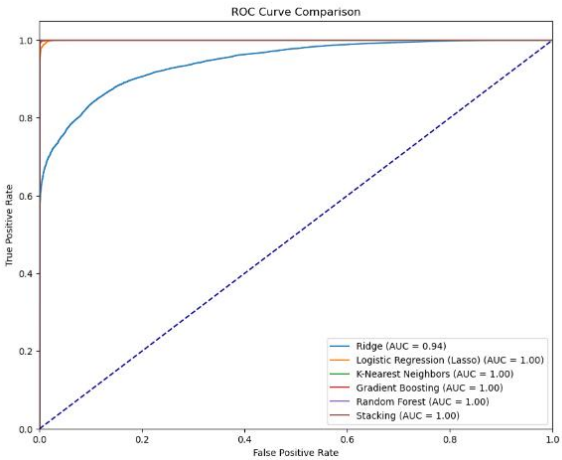

Fig.19. Comparación de curvas ROC.

Las curvas ROC de *Ridge* y *Lasso*, muestran un rendimiento efectivo, aunque ligeramente inferior a los otros modelos. Su curva se eleva rápidamente, indicando una buena capacidad para discriminar entre clases, pero no alcanza el nivel de perfección de los otros algoritmos. Con un área bajo la curva (AUC) con un valor aproximado de 0.94, indicando un rendimiento óptimo en la clasificación. Esto sugiere que los modelos logran una separación perfecta entre las clases en el conjunto de datos evaluado.

Por otra parte, los algoritmos *K-Nearest Neighbors (KNN)* y *Gradient Boosting* también exhiben un rendimiento sobresaliente, alcanzando un área bajo la curva (AUC) cercana a 1.00. Esto indica una buena capacidad de clasificación para ambos modelos. No obstante, se observa cierta dispersión en los valores más elevados, como se evidencia en las Figuras 11 y 16. Esta ligera variabilidad en el extremo superior de la distribución sugiere una potencial variabilidad en la precisión de las predicciones para los casos más extremos. Sus curvas se superponen con las de los otros modelos de alto rendimiento, indicando que logra una discriminación ideal entre las clases. La curva de *Gradient Boosting* sigue el patrón ideal, maximizando la tasa de verdaderos positivos mientras minimiza la de falsos positivos.

Random Forest y *Stacking* también alcanzan un AUC de 1.00, demostrando una capacidad de clasificación óptima. Esto sugiere que la combinación de modelos en el enfoque de *Stacking* y *Random Forest* ha mantenido o incluso mejorado el rendimiento ya excelente de los modelos individuales.

Este análisis de curvas ROC revela un rendimiento excepcional de casi todos los algoritmos evaluados, con AUC de 1.00. Esto indica una buena capacidad de clasificación en el conjunto de datos utilizados. Es importante notar que un rendimiento tan uniformemente alto podría sugerir un conjunto de datos particularmente bien separable o posiblemente un sobreajuste de los modelos.

VI. Ejemplo de predicción probabilísticas de demanda

En esta sección se muestran algunos resultados de predicciones realizadas sobre los datos históricos de demanda de potencia activa de la Facultad de Ingeniería. Se evaluó el algoritmo *Random Forest*, obteniéndose una predicción como la que se muestra en la Fig. 21.

La predicción sigue de cerca la tendencia general de los valores reales y los predichos de potencia activa, indicando un buen rendimiento del modelo. El periodo de tiempo abarca desde marzo hasta julio de 2024. La línea punteada roja en la parte inferior indica el error entre la predicción y el valor real. Se observan fluctuaciones significativas en la demanda a lo largo del período, con picos y valles que siguen patrones semanales y mensuales. El modelo de predicción captura la tendencia general, aunque hay períodos donde el error es más pronunciado, especialmente durante cambios abruptos en la demanda.

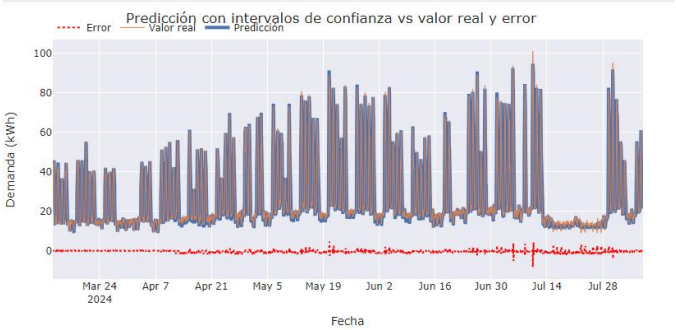

Fig.21. Curva de predicción *Random Forest* vs valor real de potencia activa.

Los errores tienden a ser más grandes durante los períodos de alta demanda, presentando patrones repetitivos que sugieren un comportamiento estacional o cíclico de la serie de potencia activa. El error medio es de 362 W, lo que sugiere que existen discrepancias en ciertos períodos, posiblemente debido a días feriados.

Al realizar un acercamiento a la Fig. 21, que corresponde a los primeros 15 días del mes de julio, se observan períodos donde el error es más consistente (puntos más agrupados) y otros donde es más variable (puntos más dispersos) (ver Fig. 22). La desviación estándar es de 0.6388, indicando una ligera tendencia a sobreestimar la demanda, pero con una precisión razonable considerando la variabilidad de los datos. Esto sugiere que las predicciones son consistentemente cercanas al error medio.

Fig.22. Curva de error de predicción *Random Forest* vs valor real de potencia activa (Comportamiento el 10 de julio del 2024).

Las predicciones a lo largo de la primera semana de julio

2024 siguen de cerca los valores reales, con el error fluctuando ligeramente. Sin embargo, en la segunda semana existe una variabilidad acentuada en el error, ya que hubo un día feriado de por medio, destacando áreas de menor precisión predictiva. Las bandas de confianza reflejan la incertidumbre en las predicciones.

VII. Conclusiones

Este estudio ha demostrado la eficacia de diversos algoritmos de aprendizaje automático en la predicción de requerimientos futuros de potencia activa, específicamente en el contexto de la Universidad Nacional de Mar del Plata. La investigación subraya la importancia de estas predicciones para la planificación y optimización de sistemas energéticos, permitiendo una toma de decisiones más informada en la gestión de recursos.

Los resultados obtenidos corroboran la creciente relevancia del procesamiento y análisis de datos de parámetros eléctricos mediante técnicas de inteligencia computacional, superando las limitaciones de los enfoques estadísticos tradicionales.

Este análisis, que incluye el cálculo y la interpretación de métricas de error, proporciona una base para la selección e implementación de modelos predictivos en otros entornos universitarios con comportamientos de potencia diferente.

Futuros estudios podrían expandir estos hallazgos, explorando la integración de variables adicionales o la aplicación de estos modelos en contextos más amplios y diversos dentro del ámbito de la gestión energética institucional.

VIII. REFERENCIAS.

[1] Herrera Granda, Dayana Estefanía. Predicción de demanda eléctrica mediante la aplicación de modelos ARIMA y SARIMA en lenguaje de programación R–caso de estudio en la Empresa Eléctrica Quito. 2019. Tesis de Licenciatura. Quito, 2019.

[2] Lara, Jorge; Samper, Mauricio; Colomé, Graciela. Predicción a corto plazo de sistemas de medición inteligentes mediante arquitecturas de aprendizaje profundo multivariable y multipaso. Revista Técnica" energía", 2024, vol. 21, no 1, p. 153-164.

[3] Herrero Arnedo, Marcos. Obtención de métricas sobre la actividad cerebral de usuarios a través de señales EEG y Machine Learning. 2023. Tesis Doctoral. Universitat Politècnica de València.

[4] De Leceta, Aitor Moreno Fdez. Inteligencia Computacional en Sistemas de Tele-asistencia en Domicilios. 2018. Tesis Doctoral. Universidad del País Vasco-Euskal Herriko Unibertsitatea.

[5] Piedrahita Vasco, Manuela; Llerena Riascos Camilo. Modelo analítico de aprendizaje automático para la predicción en corto plazo del sentido del S&P 500. 2023.

[6] Francesca Lazzeri, “Machine learnign for time series forecasting with Python”, Wiley, 2021.

[7] Pedregosa, Fabian, et al. Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 2011, vol. 12, p. 2825-2830.

[8] Agapito,Giuseppe; Milano, Marianna; Cannataro, Mario. A python clustering analysis protocol of genes expression data sets. Genes, 2022, vol. 13, no 10, p. 1839.

[9] OLIPHANT, Travis E., et al. Guide to numpy. USA: Trelgol Publishing, 2006.

[10] SzymaĹ, Piotr, et al. scikit-multilearn: A Python library for Multi-Label Classification. Journal of Machine Learning Research, 2019, vol. 20, no 6, p. 1-22.

[11] Imandoust, Sadegh Bafandeh, et al. Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background. International journal of engineering research and applications, 2013, vol. 3, no 5, p. 605-610.

[12] Melkumova, L. E.; Shatskikh, S. Ya. Comparing Ridge and LASSO estimators for data analysis. Procedia engineering, 2017, vol. 201, p. 746-755.

[13] Cutler, Adele; Cutler, D. Richard; Stevens, John R. Random forests. Ensemble machine learning: Methods and applications, 2012, p. 157-175.

[14] Natekin, Alexey; Knoll, Alois. Gradient boosting machines, a tutorial. Frontiers in neurorobotics, 2013, vol. 7, p. 21.

[15] Xu, Mijian; HE, Jing. Seispy: Python module for batch calculation and postprocessing of receiver functions. Seismological Society of America, 2023, vol. 94, no 2A, p. 935- 943.

[16] Agrawal, Tanay. Hyperparameter Optimization in Machine Learning.

[17] Alhakeem, Zaineb M., et al. Prediction of ecofriendly concrete compressive strength using gradient boosting regression tree combined with GridSearchCV hyperparameter-optimization techniques. Materials, 2022, vol. 15, no 21, p. 7432.

[18] Arauna, Kevin Victor. Comparison Between Stochastic Gradient Descent And Support Vector Machine On Ecommerce Data Using Randomizedsearchcv. 2023. Tesis Doctoral. Universitas Katholik Soegijapranata Semarang.

[19] Hyndman, Rob J.; Athanasopoulos, George. Forecasting: principles and practice. OTexts, 2018.

IX. Biografías

Yenia Maza Díaz nació en La Habana, Cuba en 1992. Recibió el grado de Ingeniera Eléctrica por parte de la Universidad Tecnológica de la Habana (La Habana, Cuba), en 2018. Actualmente es becaria doctoral del CONICET en el ICYTE. Su trabajo de investigación está dedicado a las Redes Eléctricas Inteligentes, incluyendo temas como inteligencia computacional e implementación de algoritmos de aprendizajes automático para predicción de variables eléctricas.

Patricio G. Donato nació en Puerto Madryn, Argentina, en 1975. Recibió el grado de Ingeniero

Electrónico por parte de la Universidad Nacional de la Patagonia San Juan Bosco (Comodoro Rivadavia, Argentina), en 2000, y el grado de Doctor en Electrónica por parte de la Universidad de Alcalá, (Alcalá de Henares, España), en 2005. Es investigador independiente del CONICET y profesor asociado en la UNMDP. Actualmente trabaja en el ICYTE, instituto de doble dependencia CONICET-UNMDP. Su trabajo de investigación está dedicado a las Redes Eléctricas Inteligentes, incluyendo temas específicos como procesamiento de señales, inteligencia computacional y calidad de la energía eléctrica.

Marcos A. Funes nació en Mar del Plata, Argentina, en 1974. Recibió el grado de Ingeniero Electrónico por parte de la Universidad Nacional de Mar del Plata en 1999, y el grado de Doctor en Electrónica por parte de la misma universidad en 2007. Es investigador independiente del CONICET y profesor asociado en la UNMDP. Actualmente trabaja en el ICYTE, instituto de doble dependencia CONICET-UNMDP y tiene el cargo de director del Laboratorio de Instrumentación y Control (LIC). Su trabajo de investigación está dedicado a procesamiento de señales, calidad de la energía eléctrica, microrredes de corriente continua y control de convertidores electrónicos de potencia.

Carlos M. Orallo nació en Mar del Plata, Argentina, en 1982. Recibió el grado de Ingeniero Electrónico por parte de la Universidad Nacional de Mar del Plata en 2011, y el grado de Doctor en Electrónica por parte de la misma universidad en 2015. Es profesor adjunto en la UNMDP y actualmente trabaja en el ICYTE, instituto de doble dependencia CONICET-UNMDP. Su trabajo de investigación está dedicado al procesamiento de señales y la calidad de la energía eléctrica en el marco de las redes eléctricas inteligentes.