Universidad de Cienfuegos “Carlos Rafael Rodríguez”

Facultad de Ingeniería

Departamento de Informática

Título:

Modelo de predicción para parques solares fotovoltaicos usando AutoMachine Learning

Autor:

Adrian Reyes Cabral

Tutores:

MSc. Richard Darian Sánchez Rivero

Ing. Adriamny Gonzalez Gonzáles

Cienfuegos, Cuba

Curso: 2023

AGRADECIMIENTOS

A mis padres, por ser el motor impulsor de mi vida, por guiarme y aconsejarme en cada paso y decisión que tomo, porque sin ellos mi trayectoria no hubiese sido la misma, son el alma de mi ser y mi motivo a seguir. Agradezco a mis profesores por todos los conocimientos que heredo gracias a sus clases. A mis compañeros, porque entre todos supimos crear un fuerte equipo, este recorrido juntos nunca se olvidará. A mis amigos, por darme ánimos en todo momento y estar al pendiente de mis resultados. Por último, agradecer a mis tutores que han sabido tenerme paciencia, ayudarme, aconsejarme, guiarme durante todo el proceso y no dejarme desamparado en ningún momento. ¡Muchas Gracias!

DEDICATORIA

A mis padres, esos que pelean pero que te aman como solo un padre sabe, que sufre contigo cada momento de preocupación y que celebra a tu lado semestre tras semestre culminado. Padre, madre, gracias por las peleas, por las obligaciones, por no permitirme abandonar, por darme fuerzas aún sin saber de dónde ustedes la sacaban, por siempre saber qué decir, cómo decirlo y cuándo decirlo, no hubiese sido la misma sin ustedes a mi lado. Por todo su esfuerzo y amor es a ustedes a quienes va dedicado mi trabajo.

Resumen:

La presente investigación se realizó un la Universidad de Cienfuegos “Carlos Rafael Rodríguez”. Debido a los problemas que enfrenta nuestro país en el sector energético, se hace de vital importancia el uso de la generación eléctrica a través de fuentes de energías renovables. Una de estas es la energía solar fotovoltaica, esta energía posee la característica de ser voluble por lo que una predicción acertada de la generación de esta, ayudaría de manera significativa al despacho de carga nacional y a la economía del país. Existes diferentes maneras de realizar predicciones pero en los últimos años ha surgido una rama de la inteligencia artificial llamada machine learning, la cual presenta buenos resultados en esta área. En esta investigación se llevan a cabo una serie de pasos para lograr la predicción de la potencia generada de un parque de paneles fotovoltaicos con el uso de la herramienta Autokeras, la cual automatiza el flujo de trabajo de machine learning. El objetivo de la presente investigación es el desarrollo de modelos de predicción de generación fotovoltaica para el territorio Cienfueguero, basados en Automachine Learning, con horizontes de tiempo de corto y muy corto plazo.

A través del documento de la investigación se describieron los elementos que conforman el análisis, diseño e implementación del modelo propuesto. Se utilizó como lenguaje de programación Python con las librerías para Machine Learning y Automachine Learning. Para la implementación se utilizó Anaconda. Se craron dos modelos, uno con datos de parques cercanos al parque objetivo para evaluar la incidencia de estos en la predicción del parque objetivo y otro modelo con los datos del objetivo. Se entrenaron los dos modelos y se compararon los resultados definiendo cuál de los dos presenta mejor acierto en la predicción. Se llegó a la conclusión que el modelo entrenado con los datos del parque objetivo presenta mejores indicadores de predicción.

Palabras clave:

energías renovables, predicción, inteligencia artificial, automachine learning, autokeras, modelos

Índice:

Índice de figuras:

Introducción

Todos conocemos la electricidad y la importancia que tiene esta en un mundo donde se ha hecho imposible la vida sin esta. El 80% de la energía mundial se produce actualmente a través de combustibles fósiles. Estos no son infinitos y además contribuyen activamente a la contaminación del medio ambiente. Por lo que se hace de vital importancia ampliar el uso de las llamadas fuentes de energía renovables para la generación eléctrica. Algunas de las fuentes renovables más usadas y conocidas son la energía eólica, la energía solar fotovoltaica, la energía termosolar, y la energía hidráulica.

Las energías renovables han captado la atención tanto de las autoridades como de la opinión publica en los últimos años. El aprovechamiento de las energías renovables implica importantes ventajas ambientales, entre ellas el hecho de que se trata de energías con una contaminación mínima y además no depende de fuerzas energéticas exteriores. Además las instalaciones de generación eléctrica basadas en recursos renovables pueden servir como factor de estabilización de la población en zonas con riesgo de despoblación.[1]

La utilización de la energía solar, el viento, las corrientes de los ríos, entre otras, más que una alternativa, son la única solución posible a las exigencias energéticas de nuestro país y del mundo de cara al desarrollo sostenible. Cada metro cuadrado del territorio recibe diariamente, como promedio anual, 5 kWh de energía solar, equivalente a la energía química acumulada en un litro de petróleo. Con el aprovechamiento tanto directo como indirecto de la energía solar se pueden satisfacer todas nuestras necesidades energéticas.

Anualmente el astro rey derrama sobre muestro planeta 4 000 veces más energía de la que hoy consumimos. El sol es la fuente de vida de nuestro planeta y el origen de las fuentes de energías que la humanidad ha empleado en las distintas etapas de su historia.

El aprovechamiento de la energía solar constituye una alternativa para reducir la dependencia de los combustibles fósiles en la generación de electricidad a nivel mundial. Cuba, por su ubicación en la zona tropical, recibe elevados niveles de

radiación solar que hacen factible el empleo de tecnologías capaces de utilizar este recurso con fines energéticos. Con este objetivo, en la actualidad, se desarrolla el programa “Desarrollo sostenible de las fuentes renovables de energía”, que, entre otros aspectos, incentiva el desarrollo de proyectos de investigación dirigidos al aprovechamiento de las energías renovables y el aporte de las mismas a la matriz energética del país. En el caso particular de la energía fotovoltaica, el país tenía como objetivo producir por esta vía 700MWh para el año 2018. Los niveles de radiación solar que llegan a la superficie terrestre experimentan fluctuaciones, debido a las variaciones en la atenuación atmosférica, ocasionada fundamentalmente por las nubes, la presencia de aerosoles y gases como el vapor de agua y otros componentes menores como es el caso del dióxido de carbono, el ozono, etc., cuyo aporte en la atenuación de la radiación solar resulta de menos envergadura. En consecuencia, la producción de energía eléctrica a partir del recurso solar es fluctuante, lo cual conduce a la necesidad de realizar pronósticos de irradiancia solar para diferentes escalas espaciales y temporales. Estos pronósticos podrían mejorar aspectos como la gestión de las redes eléctricas, la programación de la producción de centrales eléctricas solares o convencionales, el diseño de nuevas políticas de aprovechamiento energético. Generalmente las especificaciones de los sistemas comerciales de aprovechamiento de la energía solar están basadas en una limitada disponibilidad de información del régimen de radiación solar.[2]

Sin embargo, la potencia que entrega un sistema de generación PV es estocástica y voluble por naturaleza, en función de la radiación solar y otras variables climáticas. Este comportamiento estocástico crea algunos problemas para la operación segura en las redes de potencia. Un sistema de pronóstico de la generación PV, que sea preciso, es de mucha utilidad para el despacho de carga de los sistemas de potencia, permitiéndoles que tomen decisiones acertadas sobre aspectos claves tales como el ajuste de las fuentes de generación convencionales, programación de arranques, requerimientos de almacenamiento y planificación en general. Por lo tanto, el pronóstico de la generación eléctrica de los sistemas solares fotovoltaicos, tanto en las grandes redes de potencia como en las microrredes, juega un papel clave para la

operación eficiente, económica, estable y sostenible del suministro de electricidad.

Los pronósticos de irradiancia solar se pueden realizar utilizando diferentes métodos cuya elección depende de los requerimientos en cuanto al horizonte de pronóstico y las resoluciones espacial y temporal. Para pronósticos intra-horarios puntuales con altas resoluciones se utilizan modelos estadísticos de series temporales con mediciones de irradiancia local como datos de entrada. También son empleadas las cámaras de cielo (ground-based sky imagers), que proporcionan información sobre las nubes y su movimiento en los alrededores de un sitio dado, lo cual permite extrapolar las condiciones de nubosidad e irradiancia hacia el futuro con horizontes típicos entre 15 y 30 minutos. Para pronósticos con horizontes entre 3 y 6 horas se utilizan las imágenes de satélite, a partir de las cuales se analizan y extrapolan los campos de nubes para luego estimar la irradiancia que reciben los lugares deseados. Los modelos numéricos de pronostico del tiempo son las herramientas más eficaces para plazos superiores a las 5-6 horas. Con ellos se modela el comportamiento de la física atmosférica utilizando las leyes de conservación de la masa, el momento y la energía, lo cual permite pronosticar numerosas variables meteorológicas incluyendo a los flujos radiactivos.[2]

El Machine Learning (ML) es una rama de la Inteligencia Artificial (IA), que puede ser utilizado en varios dominios y su mayor ventaja es que puede resolver problemas que son imposibles de representar mediante algoritmos explícitos. Los modelos de ML encuentran relaciones entre entradas y salidas, incluso si la representación es imposible. Esta característica permite el uso de modelos de ML en muchos casos, como por ejemplo en el reconocimiento de patrones, problemas de clasificación y problemas de pronóstico. La importancia de poder tener información específica y detallada del potencial solar permite explotar al máximo las capacidades que ofrecen distintas regiones. Con modelos de ML se pueden realizar en este campo estimaciones de potencial del recurso solar para la generación de energía eléctrica a partir de FRV (Fuentes Renovables Variables) con determinados horizontes de tiempo, con el fin de lograr el desarrollo de herramientas decisivas y que sean un componente para integrar

mejor la energía Fotovoltaica(FV) en la red del país según la normativa vigente.[3]

Situación Problemática:

La generación de energía a partir de instalaciones fotovoltaicas (PV) en el sistema eléctrico de potencia cubano se ve afectada por la variabilidad de las condiciones climáticas y la radiación solar, lo que dificulta la predicción precisa de la potencia de salida en el corto y muy corto plazo. Actualmente, en Cuba no existen modelos no lineales que puedan predecir de manera efectiva la potencia de salida de las instalaciones PV en estas escalas de tiempo. La pregunta clave es si es posible desarrollar modelos de inteligencia artificial que utilicen datos históricos para predecir la potencia de salida a corto y muy corto plazo. La resolución de este problema podría mejorar la eficiencia y la confiabilidad del suministro de energía en el país."

Esta situación problemática se centra en la necesidad de desarrollar modelos de pronóstico de potencia de salida para instalaciones PV en Cuba utilizando técnicas de inteligencia artificial y la posterior integración de estos modelos en el sistema eléctrico de potencia cubano. Resolver este problema podría tener un impacto significativo en la gestión de la energía y contribuir a una mayor eficiencia en la generación de energía solar en el país.

Objetivo General:

Desarrollar modelos de predicción de la generación fotovoltaica para el territorio Cienfueguero, basados en Automachine Learning, con horizontes de tiempo de corto y muy corto plazo.

Objetivos específicos:

1- Analizar el estado actual y tendencias en el pronóstico de generación de energías desarrolladas para instalaciones solares fotovoltaicas.
2- Procesar la información existente en el sistema de Supervisión y Adquisición de datos (SCADA) y de la base de datos de las variables meteorológicas disponibles en los parques solares fotovoltaicos seleccionados como casos de estudio.

3- Desarrollar modelos de pronósticos basados en Automachine Learning con horizontes de tiempo corto y muy corto plazo que exhiban indicadores estadísticos de calidad adecuados.

Aporte práctico de la investigación:
Con los modelos de predicción a corto y muy corto plazo se podrá conocer un estimado de la generación futura que tendrá el parque, dándole así al despacho de carga datos que este podrá utilizar para planificar sus acciones en el futuro.

Tareas a desarrollar:

1- Investigación de los proyectos hechos en Cuba sobre la predicción de electricidad usando Machine learning.
2- Estudio las principales tecnologías usadas para la predicción eléctrica
3- Instalación de las librerías autokeras y hacer la preparación del entorno de trabajo.
4- Preprocesamiento de los datos con las herramientas de autokeras y especificar la variable objetivo (Potencia generada).
5- Entrenamiento los modelos.
6- Comparación de los resultados de los modelos definiendo cual presenta mejor acierto en la predicción.

Descripción de cada capitulo

Resumen del capítulo 1:

En este capítulo se introducirán las tecnologías renovables y en específico en la tecnología solar fotovoltaica. Se detalla la evolución de estas a lo largo de los años y como se han convertido en una opción más que viable para la generación eléctrica. Se realza la importancia que posee la predicción en el sector energético. Se expondrá que es AutoMachine Learning (AutoML) y sus principales variantes, características, ventajas y desventajas. Además, serán definidas las tecnologías que vamos a utilizar a los largo de trabajo.

Resumen del capítulo 2:

En este capítulo se describirá el experimento a realizar y su objetivo. Se realizará la preparación del entorno de trabajo para trabajar con autokeras. Los datos

serán descritos y se explicarán sus principales características. Se entrenarán los modelos con los datos. Se visualizarán las arquitecturas escogidas por los modelos y los valores de sus hiperparámetros.

Resumen del capítulo 3:

En este capítulo serán analizados los resultados de las predicciones. Se graficaran estas en diferentes espacios de tiempo en el futuro con el objetivo de observar cual presenta mejor acierto. Se calcularan los errores cuadráticos medios de ambos modelos para poder concluir cual modelo se ajusta más a los datos reales.

Capítulo 1 Fundamentación Teórica

Introducción

En este capítulo se lleva a cabo la fundamentación teórica del tema que será desarrollado. Se expondrá el comportamiento del uso de las energías renovables a lo largo de la última década y se pondrá especial atención a la energía solar fotovoltaica (FV). Se realiza un estudio de las principales estrategias de América Latina y Cuba con respecto a la fomentación del uso de las energías renovables (ER). Se definen el objeto de estudio y campo de accion. Se aborda la importancia que tienen las predicciones en la generación eléctrica. Se define la técnica que será usada en este trabajo y las tecnologías y tendencias actuales en el campo la Inteligencia Artificial (IA)

Fundamentos teóricos:

En el mundo, el consumo de energías renovables ha presentado un incremento desde el año 2015, lo cual ha reducido la emisión de carbono a la atmosferas asociadas a la producción de energía, en la actualidad se ha creado una conciencia sobre la importancia de las energías renovables y su importancia para frenar el cambio climático, para la creación de nuevas oportunidades económicas y proporcionar acceso a la energía a millones de personas que aún viven sin estos servicios de energía moderna.

La transición hacia fuentes de energía más limpias se ha convertido en una prioridad crítica a nivel global. En este contexto, el Escenario de Emisiones Netas Cero para 2050 de la IEA (Agencia Internacional de Energía) ofrece una visión clara de lo que se necesita para alcanzar cero emisiones netas de CO_2 relacionadas con la energía y los procesos industriales en 2050. Fundamental para este objetivo es el uso de tecnologías de generación de electricidad a partir de fuentes renovables. La participación de las energías renovables en la generación eléctrica ya alcanzó el 29% en 2020 y se prevé que aumente significativamente, llegando al 60% en 2030 y al impresionante 88% en 2050[4]

En la última década, ha sido evidente un crecimiento constante en el desarrollo de tecnologías energéticas renovables, y en este contexto, la energía solar fotovoltaica se erige como un actor esencial, desempeñando un papel aún más crucial.

El año 2022 presenció un aumento del 9% en la capacidad de generación de electricidad a partir de fuentes renovables en comparación con 2021, destacándose la solar fotovoltaica con un impresionante incremento del 18%.

Cuba, una nación insular bañada por el sol durante gran parte del año, se encuentra en una posición envidiable en lo que respecta a la radiación solar, con un promedio de 5 kWh/m2/día, equivalente a 1825 kWh/m2 anual. Esta riqueza natural de radiación solar no solo es un recurso abundante, sino que también es esencial para impulsar la transición hacia fuentes de energía más limpias y sostenibles.[5]

El estado cubano ha demostrado su firme compromiso con la expansión de la energía solar fotovoltaica como parte integral de su estrategia de desarrollo sostenible. A través de iniciativas gubernamentales y políticas de fomento, se ha establecido un marco propicio para la inversión en parques solares y la diversificación de la matriz energética. Esta prioridad se materializa con el Decreto-Ley No. 345 que establece las regulaciones para el desarrollo de las fuentes renovables de energía con el propósito de elevar su participación hasta un 24% para el año 2030.[6], Además, se han tomado medidas como la aprobación de la importación de estas tecnologías con la Resolución 206/2021 y la exención del pago de aranceles de aduanas a las personas naturales mediante la Resolución 319/2021[7]. Estas políticas han resultado en un aumento de la potencia instalada de fotovoltaica del 10% en el año 2021 y un 9.2 % en el año 2022.[8]

Según la International Renewable Energy Agency (IRENA), en el año 2016 las ER proporcionaron un estimado de 19.3% del consumo mundial de energía. Se destaca la producción de energía a partir de la biomasa para calefacción y cocina en las áreas rurales en los países en vía de desarrollo con una representación alrededor del 9.1%, una participación del 10.2% para las ER modernas como la energía solar fotovoltaica, eólica, hidroeléctrica, solar térmica y biocombustibles.

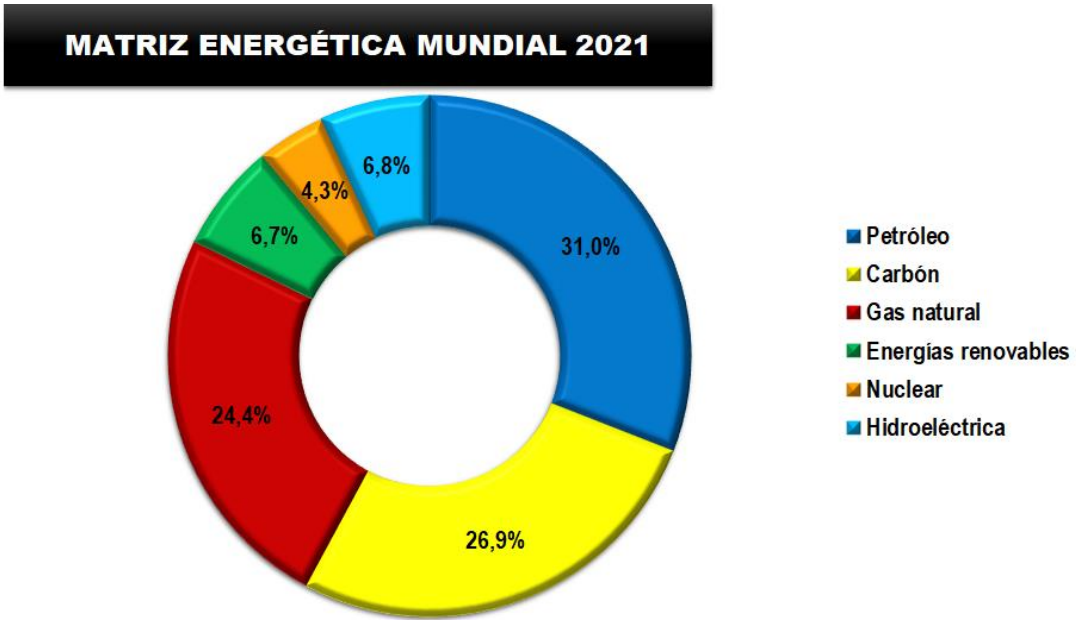

Fig.1 1 Matriz energética mundial del año 2021

Según la International Energy Agency (2014), la eficiencia media de los módulos fotovoltaicos (FV) de silicio comerciales ha mejorado en los últimos diez años en alrededor de 0.3 % por año, llegando a un valor de 16% en 2013. Los módulos comerciales de mejor desempeño, con base en diferentes tecnologías de fabricación alcanzan eficiencias entre un 19% y 21% [9]

Generalmente los módulos FV son garantizados para una vida útil de 25 años, como mínimo, trabajando en el 80% de su potencia nominal, a veces durante 30 años en el 70% de su potencia nominal. Las estadísticas y proyecciones a nivel mundial son: (International Energy Agency, 2014; International Renewable Energy Agency, 2013; REN21, 2015; REN21, 2017; International Energy Agency, 2017). Desde el 2010, en el mundo se ha adicionado más capacidad en energía FV que en las cuatro décadas anteriores. Los nuevos sistemas fueron instalados en 2013 a un ritmo de 100 MW de capacidad por día. A inicios del año 2014 la capacidad global superó los 150 GW. Durante el 2016 se agregaron 75GWdc de capacidad solar fotovoltaica a nivel mundial, lo cual equivale a la instalación de más de 31,000 paneles solares por hora.[9]

Latinoamérica es una región de rápido crecimiento para las ER con un interés creciente para el desarrollo de este tipo de recursos. Los altos precios de la electricidad en la mayor parte de las regiones, la creciente demanda, y en

algunos casos, el potencial de exportación, proporcionan un terreno fértil para el despliegue de tecnologías de energías renovables. Además, en la región existe una larga trayectoria de producción de energía hidroeléctrica lo cual se ha traducido en políticas y legislaciones que fomentan el uso de las ER. La generación de electricidad es el sector que ha atraído la mayor parte de las políticas de ER y la evolución legislativa en América Latina, lo cual abarca la promulgación de leyes de ER, establecimiento de objetivos para generar electricidad renovable, políticas de disminución de aranceles, incentivos fiscales, disposiciones para acceso a la red y servicios de financiación. Actualmente la mayoría de los países de esta región tienen leyes para el campo de las ER, a excepción de Bolivia, Guyana y Surinam que no tienen leyes o programas en este campo. El establecimiento de objetivos nacionales de ER ofrece una clara indicación sobre el nivel de desarrollo de la ER y la línea de tiempo prevista por los gobiernos.

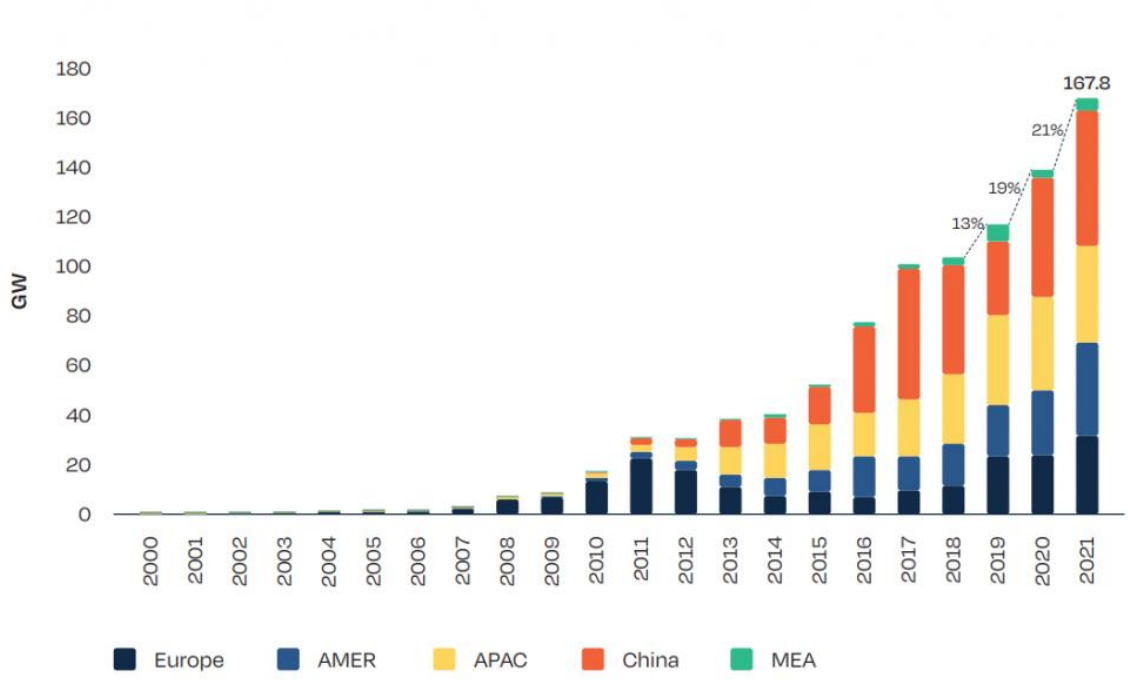

Fig.1 2 Capacidad instalada anual de paneles fotovoltaicos.

En Cuba es de vital importancia el incremento del uso de las fuentes renovables de energía debido al papel estratégico que para el crecimiento del país tiene el desarrollo del sector energético. Esto se debe no solo a las implicaciones que para el medio ambiente tiene el uso de los combustibles fósiles, sino también para alcanzar la independencia energética. Actualmente en Cuba se satisface

alrededor del 96 % de las necesidades energéticas con combustibles fósiles y aproximadamente un 4 % con fuentes renovables. Constituye un programa de gobierno cambiar la matriz energética y producir alrededor de un 24 % de su energía con fuentes renovables para el año 2030. Para alcanzar esta meta, el país está montando parques eólicos y solares fotovoltaicos, así como plantas bioeléctricas. Ahora bien, dentro de las fuentes renovables, particularmente la solar y la eólica han ganado mayor importancia en los últimos años debido a que son de fácil acceso (disponibles en cualquier lugar), cero emisiones (energías limpias y amistosas con el ambiente) y simples (estructura menos compleja que las fuentes de energía convencional). Esto ha acelerado grandemente la instalación de sistemas solares fotovoltaicos (PV) a nivel global. Sin embargo, la potencia que entrega un sistema de generación PV es estocástica y voluble por naturaleza, en función de la radiación solar y otras variables climáticas. Este comportamiento estocástico crea algunos problemas para la operación segura en las redes de potencia.[9]

1.1 Actualidad y necesidad del trabajo:

La actualidad del trabajo está sustentada en el interés del Estado Cubano y de su Ministerio de Energía y Minas de cambiar la matriz energética del país incrementando la participación de fuentes renovables de energía, especialmente generación fotovoltaica, lo que implica tener capacidad de predecir al corto plazo la producción futura de este tipo de fuentes para garantizar la operación estable del sistema de potencia.

1.2 Importancia de la predicción.

Poder predecir cómo se comportará un fenómeno especifico seria siempre de gran ayuda para la planificación de las medidas a tomar referentes a este fenómeno. En la generación de electricidad esto siempre ha sido un tema a considerar por lo positivo que puede ser conocer los datos futuros de la demanda o de la generación y así poder planificar de manera que se utilice de la manera más óptima posible los recursos.

Conocer la generación fotovoltaica de un parque sería de gran ayuda para la planificación del despacho de carga, esta generación está dada por innumerables variables físicas interrelacionadas entre si destacándose la irradiancia del sol, temperatura de los paneles solares y el clima en general.

El pronóstico de la potencia de salida de los sistemas solares fotovoltaicos también es muy difícil porque está altamente relacionado con las circunstancias externas tales como radiación solar y otros parámetros climáticos que cambian muy rápidamente en una localidad. Un sistema de pronóstico de la generación PV, que sea preciso, es de mucha utilidad para el despacho de carga de los sistemas de potencia, permitiéndoles que tomen decisiones acertadas sobre aspectos claves tales como el ajuste de las fuentes de generación convencionales, programación de arranques, requerimientos de almacenamiento y planificación en general. Por lo tanto, el pronóstico de la generación eléctrica de los sistemas solares fotovoltaicos, tanto en las grandes redes de potencia como en las micro redes, juega un papel clave para la operación eficiente, económica, estable y sostenible del suministro de electricidad.[10]

1.3 Machine Learning tradicional

En una aplicación de ML tradicional, los profesionales deben entrenar un modelo utilizando un conjunto de datos de entrada. Si estos datos no están en la forma adecuada, es posible que un experto deba aplicar algunas técnicas de preprocesamiento de datos, como la extracción de características, la ingeniería de características o la selección de características.

Una vez que los datos están listos y se puede entrenar el modelo, el siguiente paso es seleccionar el algoritmo correcto y optimizar los hiperparámetros para maximizar la precisión de las predicciones del modelo. Cada paso implica desafíos que requieren mucho tiempo y, por lo general, también requiere un científico de datos con la experiencia y el conocimiento para tener éxito. En la siguiente figura, podemos ver los pasos principales representados en una canalización típica de ML.

Fig.1 3 como funciona ML

Entrar los datos:

Como dijimos anteriormente el primer paso sería canalizar los datos entrantes a un almacén de datos. El objetivo es almacenar estos datos sin procesar, sin realizarles ninguna transformación, lo que nos permite tener un registro inmutable del conjunto de datos original. Los datos se pueden obtener de varias fuentes, dígase bases de datos, buses de mensajes, flujos, etc.

Preprocesamiento de datos:

Este segundo paso es fundamental para poder optimizar el uso de nuestros datos aunque también es uno de los pasos que más tiempo consume pues es necesario tener conocimiento de ingeniería de datos y ciencia de datos. Este paso tiene varias subtareas que serán brevemente explicadas a continuación.

Limpieza de datos:

Este paso consiste en detectar y, corregir o eliminar los registros corruptos o incorrectos de nuestra base de datos. Debido a que estos datos no están procesados o estructurados, es raro que se encentren de la forma correcta para ser procesados. Esto implica completar campos faltantes, eliminar filas duplicadas o normalizar o corregir otros errores en los datos.

Extracción de características:

Esta subtarea es un procedimiento para reducir la cantidad de recursos necesarios en un gran conjunto de datos creando nuevas características a partir de dos existentes (y eliminando las originales). Esto presenta el problema que al analizar grandes conjuntos de datos se tienen que tener en cuenta una gran cantidad de variables.

Ingeniería de características

Es el proceso mediante el cual, a través de técnicas de minería de datos, se extraen características de los datos sin procesar utilizando el conocimiento del dominio. Por lo general, esto requiere un experto con conocimientos y se utiliza para mejorar el rendimiento de los algoritmos de ML.

Segregación de datos:

Consiste en dividir el conjunto de datos en dos subconjuntos: un conjunto de datos de tren para entrenar el modelo y un conjunto de datos de prueba para probar el modelo de predicción.

Selección de modelo

Al elegir un modelo candidato para usar, además del rendimiento, es importante considerar varios factores, como la legibilidad (por humanos), la facilidad de depuración, la cantidad de datos disponibles, así como las limitaciones de hardware para entrenamiento y predicción.

Los principales puntos a tener en cuenta para seleccionar un modelo serían los siguientes:

• Interpretabilidad y facilidad de depuración: Cómo saber por qué un modelo tomó una decisión específica. ¿Cómo arreglamos los errores?
• Tipo de conjunto de datos: Hay algoritmos que son más adecuados para tipos de datos específicos.
• Tamaño del conjunto de datos: ¿Cuántos datos hay disponibles? ¿Cambiará esto en el futuro?
• Recursos: ¿De cuánto tiempo y recursos dispone para el entrenamiento y la predicción?[11]

Entrenamiento modelo:

Luego de escoger nuestro modelo candidato llego la hora de entrenar este con nuestros datos ya procesados. En este proceso se utiliza el conjunto de datos de entrenamiento para alimentar el modelo, lo que permite que este aprenda de ellos mediante la aplicación de un algoritmo de retropropagacion que va extrayendo los patrones encontrados en la muestras de entrenamiento.

El modelo se alimenta con los datos de salida del paso de preprocesamiento de datos. Este conjunto de datos se envía al modelo elegido y, una vez entrenado, tanto la configuración del modelo como los parámetros aprendidos se utilizarán en la evaluación del modelo.

Evaluación del modelo

Este paso es responsable de evaluar el rendimiento del modelo utilizando conjuntos de datos de prueba para medir la precisión de la predicción. Este proceso implica ajustar y mejorar el modelo, generando una nueva versión candidata del modelo para ser entrenada nuevamente.

Ajuste del modelo

Este paso implica que los hiperparámetros como la tasa de aprendizaje, el algoritmo de optimización, la cantidad de capas, entre otros sean modificados. En el AutoML estándar estos procedimientos deben ser realizados por un experto. En otras ocasiones se descarta el modelo evaluado y se elige uno completamente nuevo para entrenar.

1.4 Automachine Learning

AutoML es un proceso que automatiza, utilizando algoritmos de la Inteligencia Artificial (IA), cada paso del proceso de ML, desde el preprocesamiento de datos hasta la implementación del modelo ML, Permite que los científicos que no son científicos de datos (como los desarrolladores de software) utilicen técnicas de aprendizaje automático sin necesidad de experiencia en el campo. En la siguiente figura, podemos ver un simple representación de las entradas y salidas de un sistema AutoML

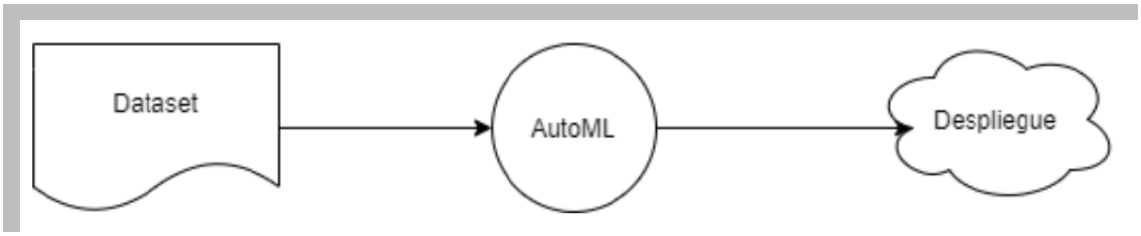

Fig.1 4 Esquema general de AutoML

Las técnicas de ML funcionan muy bien cuando se trata de encontrar patrones en grandes conjuntos de datos. Actualmente, se utilizan estas técnicas para la detección de anomalías, segmentación de clientes, análisis de abandono de clientes, pronósticos de demanda, mantenimiento predictivo y optimización de precios, entre cientos de otros casos de uso[12]. Un ciclo de vida típico de ML se compone de recopilación de datos, tratamiento de datos, gestión del ciclo, reentrenamiento de modelos e implementación de modelos, durante los cuales el tratamiento de datos es normalmente, la tarea que consume más tiempo. El objetivo del AutoML es simplificar y democratizar los pasos de este ciclo.

Originalmente, el enfoque clave del AutoML es la selección de modelos y el ajuste de hiperparámetros, es decir, encontrar el mejor modelo de ejecución para el trabajo y los parámetros correspondientes que funcionan mejor para el problema. En la siguiente figura vemos el esquema del ciclo de vida

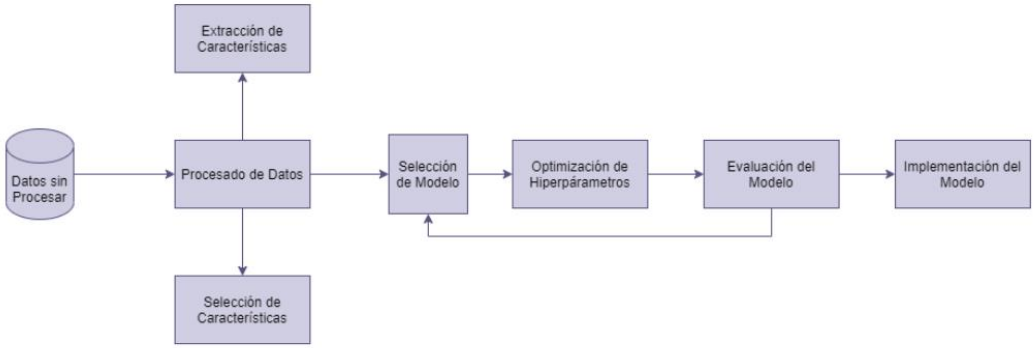

Fig.1 5 Ciclo de vida de AutoML

Tipos de AutoML

Hay tres áreas clave en el proceso del AutoML: la ingeniería de características, la búsqueda de arquitectura neuronal y la optimización de hiperparámetros. Son las más prometedoras para la democratización de la inteligencia artificial y el AutoML. Algunas Técnicas de ingeniería de características automatizadas más comunes son expandir / reducir el conjunto de datos, organizar jerárquicamente transformaciones, meta aprendizaje y aprendizaje reforzado. Para la búsqueda arquitectónica (también conocida como búsqueda de arquitectura neuronal), tenemos algoritmos evolutivos, búsqueda local, meta aprendizaje, refuerzo del aprendizaje, el aprendizaje por transferencia, el morfismo de la red y la optimización continua que suelen ser los más utilizados. Por último, tenemos la optimización de hiperparámetros, que es el arte y la ciencia de encontrar el tipo correcto de parámetros fuera del modelo. Se utilizan una variedad de técnicas aquí, incluida la optimización bayesiana, algoritmos evolutivos, funciones de Lipchitz, búsqueda local, meta aprendizaje, optimización de enjambres de partículas, búsqueda aleatoria y aprendizaje por transferencia, por nombrar unos pocos. En la siguiente figura se tiene un esquema de tipos de AutoML dependiendo de las técnicas usadas.

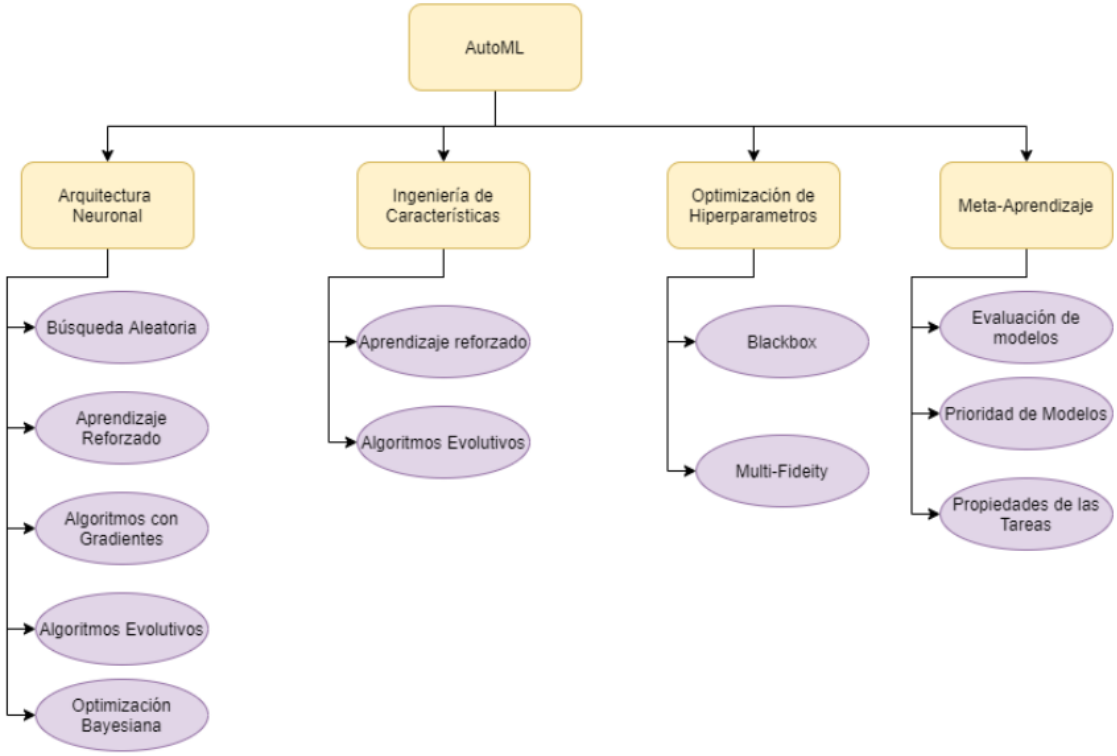

Fig.1 6 Tipos de AutoML

Ventajas y desventajas de AutoML

Ventajas:

1. Se obtienen resultados más rápidos. Con AutoML, puedes omitir gran parte del trabajo del aprendizaje automático y, como resultado, ahorrar algo de tiempo. Es perfecto si se está construyendo un prototipo para probar un producto que se ajusta al mercado u obtener una experiencia temprana de un modelo de machine learning.

2. Es menos probable que esté desactualizado. Un problema muy común en el desarrollo de la IA es que los modelos envejecen casi a medida que se desarrollan. La tecnología de la inteligencia artificial se mueve tan rápido que lo que hoy podría ser de vanguardia y con mucho trabajo detrás, mañana podría quedarse atrás de sus competidores. Con AutoML, esa labor recae en los grandes proveedores de tecnología AutoML que tienen la economía de escala para que puedan invertir en mantenerse a la vanguardia.

3. Con una solución alojada en AutoML, ahorrará mucho tiempo y código al no tener que construir la infraestructura circundante. En promedio, el 95 % del código en las soluciones de aprendizaje automático es un código adhesivo que

construye la infraestructura en torno a los modelos. El código de aprendizaje automático real es solo del 5 %.

4. Se necesita menos experiencia. Un problema común en la IA es que requiere mucha experiencia y, por lo general, involucra a varios tipos diferentes de expertos. Eso es extremadamente caro y los expertos son difíciles de encontrar. Se pueden ahorrar muchos de esos problemas con el AutoML.

Desventajas:

1. Se obtiene menos información sobre los datos. Una de las grandes ventajas de realizar un modelo de aprendizaje automático a mano es que obtienes muchos conocimientos sobre por qué tus modelos no funcionan como se esperaba. Esto le brinda información valiosa, para saber qué datos se podrían necesitar recopilar para obtener los resultados necesarios.

2. AutoML es inflexible. Es posible que obtenga resultados rápidamente, pero como ocurre con la mayoría de los equipos de IA, no pasará mucho tiempo antes de que los requisitos cambien y, con una solución de AutoML, corre el riesgo de que estos requisitos queden repentinamente fuera del alcance de los que AutoML puede manejar. Si eso sucede, se debe comenzar de nuevo con un modelo personalizado.

3. El coste de funcionamiento puede ser demasiado alto cuando escala. Una herramienta de pago es ideal para soluciones de pequeña a mediana escala. Pero si se esperan volúmenes muy altos de uso, es posible que se termine gastando demasiado dinero en los distintos proveedores. En este caso, una mejor solución sería crear y alojar los modelos en local.

4. La calidad probablemente no será la más alta. Si la característica más importante de su solución es la más alta calidad en comparación con los competidores, AutoML probablemente se quede corto. AutoML son modelos generalizados y, en la mayoría de los casos, no pueden competir con modelos verdaderamente especializados que han sido cuidadosamente construidos y adaptados al problema específico desde cero.[13]

Autokeras

Auto-Keras, como su nombre indica, es una herramienta basada en el framework de redes neuronales de Keras, que a su vez está basado en Tensorflow [24]. Por ello, todos sus modelos están basados en redes neuronales. La diferencia con utilizar simplemente Keras, como era de esperar, es que no se requiere ningún tipo de conocimiento previo en redes neuronales, pudiéndose utilizar la herramienta sin siquiera saber que se estén aplicando este tipo de modelos. La librería es bastante completa, e incluye modalidades para las principales tareas en las que tradicionalmente han funcionado bien las redes neuronales, como clasificación o regresión de imágenes o texto, o predicción de series temporales. Donde falla es en su facilidad de uso, por ejemplo, en la Figura 3.3se aprecia que se deben aportar parámetros a la hora de lanzar las predicciones, como la cantidad de datos en el pasado que se utilizan para predecir el siguiente lookback. También deja en manos del usuario la importación de datos y su estructuración correcta, tarea que puede no ser trivial para modelos como los de redes LSTM.

```
predict_until = 10
lookback = 3
clf = ak.TimeseriesForecaster(
    lookback=lookback,
    predict_from=predict_from,
    predict_until=predict_until,
    max_trials=1,
    objective="val_loss",
)
# Train the TimeSeriesForecaster with train data
clf.fit(
    x=data_x,
    y=data_y,
    validation_data=(data_x_val, data_y_val),
    batch_size=32,
    epochs=10,
)
# Predict with the best model(includes original training data).
predictions = clf.predict(data_x_test)
print(predictions.shape)
# Evaluate the best model with testing data.
print(clf.evaluate(data_x_val, data_y_val))
```

Fig.1 7 Ejemplo de entrenamiento y predicción de serie temporal con autokeras

Un punto fuerte de autokeras es que permite la exportación fácil de los modelos, y al estar basada en Tensorflow, una de las librerías más utilizadas para redes neuronales, estos son fácilmente adaptables a un entorno de producción con pocas dependencias y en casi cualquier tipo de arquitectura de máquina.

1.5 Tendencias, metodologías y tecnologías actuales

Lenguaje de Programación utilizado: Python versión 3.11.3

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes.
Python fue concebido a finales de los años 80 como un lenguaje para ser interpretado orientado a la enseñanza. Con el paso del tiempo, Python se ha convertido en una herramienta esencial para todo tipo de programadores, ingenieros e investigadores, tanto en el ámbito académico como industrial.
El éxito de Python reside no solo en su simplicidad, sino que sobre él se ha construido una enorme cantidad de herramientas disponibles para todo tipo de dominios de aplicación.
Gran parte de los programas escritos en Python en computación científica y ciencia de datos está codificado utilizando el siguiente grupo de paquetes, consolidados casi como estándar:
NumPy, permite el almacenamiento y computación eficiente de matrices multidimensionales.
SciPy, contiene una enorme colección de herramientas de cálculo numérico.
Pandas, permite manipular, filtrar, agrupar y transformar datos, así como el sencillo acceso a las bases de datos que eventualmente puedan contenerlos.
Matplotlib, dispone de un conjunto de funciones para la creación de figuras y gráficos de gran calidad.
Scikit-Learn, proporciona un paquete de herramientas con los algoritmos más usuales de aprendizaje automático.[14]

Interfaz gráfica Anaconda3 2022.5:

Anaconda es una distribución de los lenguajes de programación Python y R para computación científica (ciencia de datos, aplicaciones de Machine Learning, procesamiento de datos a gran escala, análisis predictivo, etc.)
Tiene como ventaja simplificar la gestión e implementación de paquetes. La distribución incluye paquetes de “data science” adecuados para Windows, Linux y macOS.[7]

Jupyter Notebooks:

Son una aplicación web, también de código abierto que nos va a permitir crear y compartir documentos con código en vivo, ecuaciones, visualizaciones y texto explicativo. Estos documentos registran todo el proceso de desarrollo y, lo más interesante, pueden ser compartidos fácilmente con otras personas a través de correo electrónico, Dropbox, sistemas de control de versiones como git/GitHub y Entre sus usos están:

- la limpieza y transformación de datos
- la simulación numérica
- el modelado estadístico
- el aprendizaje automático
- y mucho más.[7]

Librerías de Python para Machine Learning:

Tensorflow es una de las librerías open sources más importantes de Deep Learning y ha sido creada por Google. Está formada por un ecosistema flexible de herramientas, librerías y recursos de la comunidad y ayuda a los investigadores a innovar aplicando técnicas de Machine Learning. Además, permite a los desarrolladores incorporar este tipo de tecnologías en sus aplicaciones. Además permite la compilación y entrenamiento de modelos de ML de una forma sencilla utilizando sus API. Entre estas destaca Keras, API de alto nivel utilizada para protipado rápido, investigación de vanguardia y soluciones productivizadas. Entre las características de Keras destacan su interfaz simple y optimizada para casos comunes, el modularidad mediante el uso de bloques y la facilidad de adaptar estos bloques para aplicar nuevos descubrimientos estado del arte

Pandas:

La primera librería de Python para Machine Learning a considerar sería Pandas, una de las más utilizadas para el tratamiento de datos en Python. Una de las principales virtudes que tiene la librería es la carga de datos, la cual permite realizar la carga desde distintos orígenes. Entre los orígenes que acepta encontramos archivos de texto plano como CSV, ficheros en el extendido formato Excel y cargas directas desde bases de datos SQL, entre otros orígenes de datos. Todas estas fuentes de datos contienen la información en formato tabular y pandas permite representar este tipo de datos a la perfección mediante el uso de su estructura principal, el DataFrame.[15]

Keras:

Es un interfaz de alto nivel para trabajar con redes neuronales. La interfaz de Keras es mucho más fácil de usar que el de TensorFlow. Esta facilidad de uso es su principal característica.

Con Keras es muy fácil comprobar si nuestras ideas tendrán buenos resultados rápidamente. Keras utiliza otras librerías de deep learning (TensorFlow, CNTK o Theano) de forma transparente para hacer el trabajo que le digamos.[15]

1.6 Conclusiones

El mundo está apostando por las energías renovables ya que estas son el único camino para la preservación del medio ambiente. Pero aún no tenemos la capacidad de aprovechar al máximo estas energías, por tanto la utilización de pronósticos utilizando AutoMachine learning nos dará una vista al futuro cercano que nos permitirá utilizar el recurso solar lo más eficientemente posible.

Capítulo 2: Descripción del experimento:

Introducción

En este capítulo se dará a conocer el experimento a realizar, sus características y su objetivo. Se describe nuestro conjunto de datos, su origen y características principales así como el preprocesamiento que este recibió previo a utilizarlo. Se comenzará la preparación de nuestros datos y el entrenamiento de los modelos y una vez entrenados se observará las arquitecturas y valores de hiperparámetros encontrados este para nuestros datos.

2.1 Descripción del experimento a realizar

Este experimento tiene el objetivo de encontrar el mejor modelo de predicción para un parque del territorio cienfueguero utilizando la tecnología de autokeras. El parque en cuestión es el parque de paneles fotovoltaicos “Yaguaramas”, ubicado cerca del poblado con el mismo nombre. En nuestra base de datos contamos con las mediciones de la potencia generada de 8 parques ubicados dentro de la misma zona climática, los parques son Caguaguas, Frigorífico, Guasimal, Marrero, Mayagigua l, Mayagigua ll, Venegas y Yaguaramas. Atendiendo a los dos escenarios presentes en la investigación se entrenaran dos modelos, un primer modelo alimentado solamente por los datos del parque Yaguaramas (Modelo Univariado) y un segundo modelo alimentado por los datos de los 8 parques (modelo multivariado). Los modelos serán identificados como modelo univariado y modelo multivariado. Esto se hará con el fin de evaluar la incidencia que tienen los otros parques en la predicción de generación del parque Yaguaramas. Una vez entrenados los dos modelos se realizarán las determinadas predicciones y se compararan los resultados.

2.2 Descripción de los datos

Nuestra base de datos se basa en los valores de la irradiancia del sol, lo cual es la potencia por unidad de área recibida del Sol en forma de radiación electromagnética, medida en el rango de longitud de onda del instrumento de medición. Estos valores pertenecen a un total de 8 parques de paneles fotovoltaicos distribuidos en el territorio cienfueguero. Las mediciones fueron tomadas en intervalos de 5 minutos cada una dando un total de 135 743

mediciones siendo un total de 808 días. La base de datos se posee 8 columnas, cada una contiene los valores de la irradiancia de un parque específico. El documento se encuentra en formato xlsx. Estos datos han sido procesados previamente en el entorno de programación MATrix LABoratory (MATLAB), donde a este conjunto de datos se le aplicaron varias fórmulas con el fin de completar los datos faltantes.

2.3 Preprocesamiento de los datos.

Con el fin de completar los datos faltantes. Se utilizó el método de ponderación de distancia inversa (IDW por sus siglas del inglés *Inverse Distance Weighting*). Este es un método determinista de interpolación espacial basado en la distancia entre la ubicación para la cual se debe interpolar un valor y las ubicaciones de las observaciones registradas

. Los valores faltantes en la serie de potencia de un parque se sustituyen por valores calculados por la siguiente expresión:

$$P_E = \frac{\sum_{i=1}^{n} W(r_{i,E}) P_i}{\sum_{i=1}^{n} W(r_{i,E})} \quad (2.2)$$

$$W(r_{i,E}) = \frac{1}{r_{i,E}^{p}} \quad (2.3)$$

Donde:

P_i es el registro de la potencia en el sitio i, con i = 1,2,…,n

$W(r_{i,E})$ es la función de ponderación del i-ésimo sitio

$r_{i,E}$ es la distancia en km entre el parque fotovoltaico i-ésimo y el parque fotovoltaico de estimación

p es el exponente utilizado en la interpolación

En la Fig2.1 se muestra un ejemplo de la curva de potencia generada para el parque de Yaguaramas en la que se aprecia la construida a partir de datos originales y la que se ha completado imputando los datos por el método de ponderación de distancia inversa.

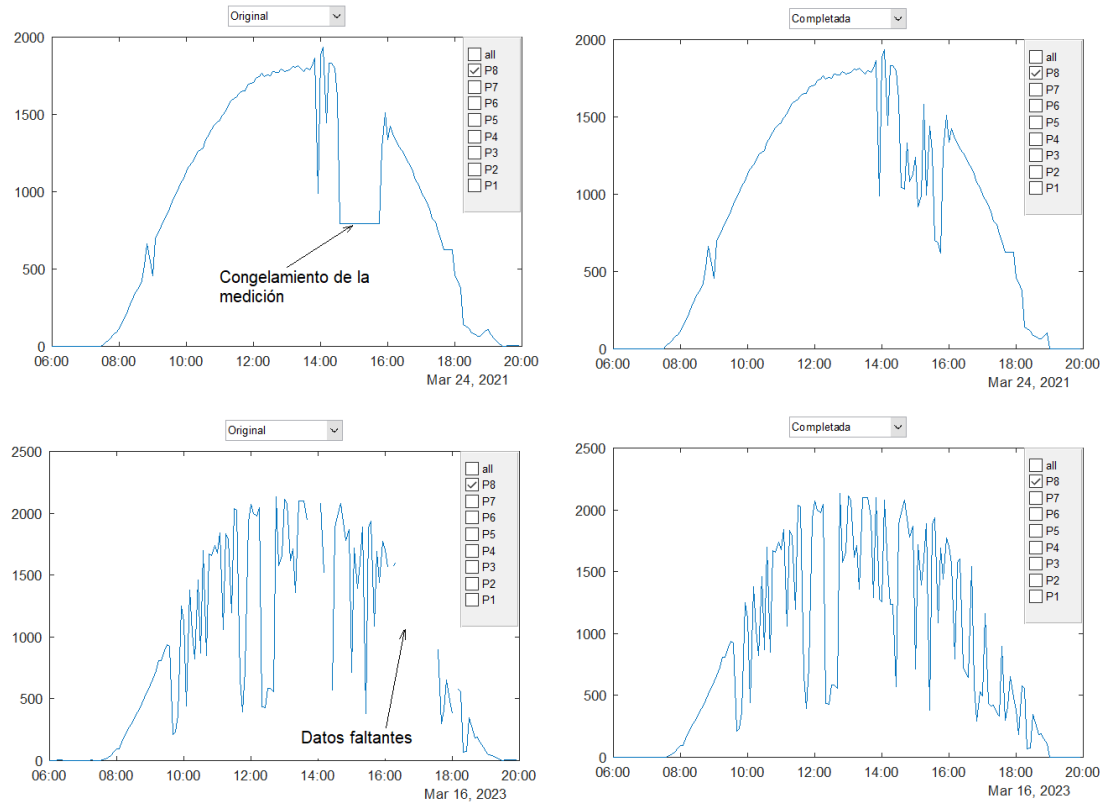

Fig. 2 1 Ejemplo de datos originales e imputados

2.4 Importación de librerías y cargado de datos

El primer paso para comenzar el proceso de AutoML es importar las librerías que utilizaremos, ya sea para entrenar modelos o para graficar las predicciones.

```
In [1]:
1 import pandas as pd
2 import tensorflow as tf
3 import numpy as np
4 import autokeras as ak
5 import matplotlib.pyplot as plt

Using TensorFlow backend
```

Fig. 2 2 Importación de librerías para ML

En nuestro caso vamos a importar las principales librerías que nos van a servir para tratar nuestros datos, entrenarlos y graficarlos.

Pandas. La librería Pandas en Python es ampliamente utilizada en el ámbito de la ciencia de datos y el aprendizaje automático. Sirve para gestionar y manipular cualquier tipo de información sin importar el formato, ofreciendo estructuras poderosas y flexibles como Series y DataFrame. Entre sus características,

Pandas permite leer y escribir fácilmente archivos en formatos como CSV, Excel y bases de datos SQL, acceder a los datos mediante índices o nombres, y trabajar con series temporales.

TensorFlow es una herramienta de software de código abierto para el aprendizaje automático y la inteligencia artificial. Se puede utilizar en una variedad de tareas, pero se enfoca particularmente en el entrenamiento y la inferencia de redes neuronales profundas. TensorFlow sirve como una plataforma central y una biblioteca para el aprendizaje automático, lo que permite a los usuarios crear y entrenar sus propios modelos de aprendizaje automático, cargar los datos para entrenar el modelo y desplegarlo utilizando TensorFlow Serving.

Numpy se utiliza para realizar operaciones matemáticas en arreglos multidimensionales. Es fundamental para la computación científica y el aprendizaje automático, ya que proporciona estructuras de datos eficientes y herramientas para trabajar con estas estructuras.

Matplotlib es una herramienta integral para crear visualizaciones estáticas, animadas e interactivas. Se utiliza para generar una amplia variedad de gráficos y trazados, lo que la hace esencial en el ámbito de la ciencia de datos y la visualización de información. Matplotlib facilita la creación de gráficos de líneas, barras, dispersión, histogramas, entre otros, y ofrece la flexibilidad para personalizar estos gráficos según las necesidades específicas del usuario

AutoKeras es una biblioteca que forma parte de AutoML (Machine Learning Automatizado) y está diseñada para automatizar el proceso de creación de modelos de aprendizaje profundo. Permite a los usuarios crear modelos de aprendizaje profundo sin necesidad de codificar la arquitectura ellos mismos, ya que AutoKeras elige la estructura de capas, el número de neuronas y otros hiperparámetros. Además, se encarga de la preparación de datos, lo que incluye la selección del modelo y el manejo de la preparación de datos, ya sea numéricos, de texto o gráficos.

Ahora podemos cargar nuestros datos, esto lo haremos utilizando el método read_csv(), de pandas, este nos permite cargar nuestros datos desde un fichero .csv.

```
In [2]:
1 dataset = pd.read_csv('Parques.csv')
2 dataset
```

Out[2]:

	Columna1	Columna2	Columna3	Columna4	Columna5	Columna6	Columna7	Columna8
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
135739	0	0	0	0	0	0	0	0
135740	0	0	0	0	0	0	0	0
135741	0	0	0	0	0	0	0	0
135742	0	0	0	0	0	0	0	0
135743	0	0	0	0	0	0	0	0

135744 rows × 8 columns

Fig. 2 3 Cargado del fichero Parques

Como vemos el método read_csv guarda todos los valores que tiene el fichero Parques dentro de la variable dataset que nos hemos creado. En la segunda línea de código al escribir el nombre de la variable podemos visualizar el principio y el final de los datos. También nos muestra informaciones como son los nombres de las columnas que poseen nuestros datos, la cantidad de filas de datos y la cantidad de columnas.

El proceso de cargado de datos es el mismo en los dos modelos a excepción de que el modelo univariado solo trabajara con los datos del parque Yaguaramas, que se encuentran en la columna 8 de nuestros datos. Por lo que necesitamos asignar solo esa columna para el modelo univariado. Lo haremos de la siguiente manera:

```
1 dataset = datos.iloc[:, -1]
2 dataset
```

```
0         0
1         0
2         0
3         0
4         0
         ..
135739    0
135740    0
135741    0
135742    0
135743    0
Name: Columna8, Length: 135744, dtype: int64
```

Fig. 2 4 Asignación de los datos del parque Yaguaramas para el modelo univariado

Como podemos ver en Fig 2.4, con el método iloc sobrescribimos los datos del dataset dejando solamente los datos de la Columna8, pertenecientes al parque Yaguaramas.

En este momento ya tenemos los datos cargados y listos para comenzar a prepararlos para que alimenten nuestro modelo.

2.5 Segregación de los datos y preparación de variables para el entrenamiento

Ya teniendo los datos cargados, el siguiente paso es segregarlos. Aunque hay varias formas y métodos para realizar esta acción, en este trabajo preferimos hacerlo manual para poder observar mejor el proceso.

Antes de dividir los datos cabe recalcar que este proceso depende en gran parte del problema que se está estudiando. Los datos pueden ser divididos por cantidades y por características. En nuestro caso al tener las mediciones diarias de los Parques y al no contar con una característica asociada al tiempo en los datos, decidimos segregar los datos por días enteros, evitando así que en los extremos de las variables que vamos a crear con los fragmentos de los datos, existan días incompletos. Este proceso de seccionamiento por días se llevará a cabo sabiendo que cada dia cuenta con un total de 168 mediciones (1 cada 5 minutos) y que el conjunto de datos cuenta con las mediciones de 808 días se calculará de la siguiente forma

Datos de entrenamiento = desde 0 hasta int(808*0.7) * 168 dando como resultado desde 0 hasta 94 920 con un total de 94920 filas.

Datos de validación = desde int(808*0.7) * 168 hasta int(808*0.9) dando como resultado desde 94 920 hasta 122 136 con un total de 27 216 filas.

Datos de prueba = desde int(808*0.9) hasta el final dando como resultado desde 122 136 hasta el final con un total de 12 608 filas.

```
n = len(dataset)

# Split 70:20:10 (train:validation:test)
data_train = dataset[0:int(565*168)]
validation_data = dataset[int(565*168):int(727*168)]
test = dataset[int(727*168):]

data_train.shape, validation_data.shape, test.shape

```

```
((94920,), (27216,), (13608,))
```

Fig. 2 5 Segregación de datos del modelo univariado.

```
n = len(dataset)

# Split 70:20:10 (train:validation:test)
data_train = dataset[0:int(565*168)]
validation_data = dataset[int(565*168):int(727*168)]
test = dataset[int(727*168):]

data_train.shape, validation_data.shape, test.shape
```

```
((94920, 8), (27216, 8), (13608, 8))
```

Fig. 2 6 Segregación de datos del modelo multivariado.

Como podemos apreciar en los dos casos el código es prácticamente el mismo, solo existe la diferencia de la cantidad de columnas que poseen las variables, existiendo 0 en el caso del modelo univariado y 8 columnas en el modelo multivariado.

Grafiquemos nuestros datos de entrenamiento, validación y prueba

```
# Dibujemos los subsets
fig, ax = plt.subplots(figsize = (16,5))
ax.plot(train_df, label='Train')
ax.plot(val_df, label='Val')
ax.plot(test_df, label='Test')
plt.legend();
```

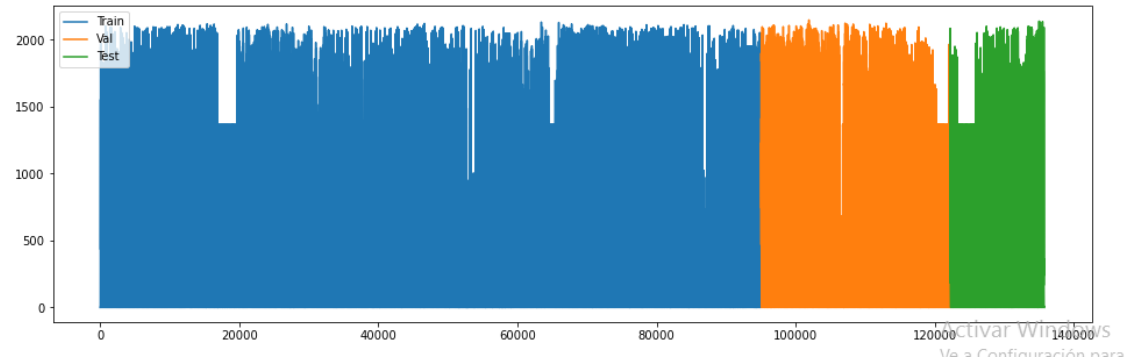

Fig. 2 7 Grafica de entrenamiento, validación y prueba.

Preparación de variables

Ya divididos nuestros datos hay que saber que la clase TimeSerieForecaster de Autokeras, con la cual vamos a realizar el entrenamiento de nuestros modelos, requiere que los datos estén divididos en datos de entrenamiento y objetivos, en el caso del modelo univariado esto tiene poca relevancia ya que al presentar una sola característica(columna), esta misma será su objetivo. Pero en el caso del modelo multivariado se busca saber si a través de los datos de los parques circundantes al parque objetivo, se pueden predecir los valores de este. Por lo que se hace necesario dividir los datos del modelo multivariado en datos de entrenamiento y datos objetivos. Esto lo haremos de la siguiente manera.

En el caso del modelo univariado se hará de la siguiente forma:

```
data_x = data_train.astype("float64")
data_x.shape
```

```
(94920,)
```

```
data_x_val = validation_data.astype("float64")
data_x.shape

```

```
(94920,)
```

```
data_x_test = dataset.astype("float64")
data_x_test.shape
```

```
(135744,)
```

```
data_y = data_train.astype("float64")
data_x.shape
```

```
(94920,)
```

```
data_y_val = validation_data.astype("float64")
data_x.shape
```

```
(94920, 1)
```

Fig. 2 8 Creación de variables para alimentar el modelo univariado

En este caso nos creamos varias variables, las cuales tendrán papeles diferentes en el entrenamiento.

data_x contendrá el 70% de los valores para el entrenamiento.

data_y contendrá el 70% de los datos como objetivos de entrenamiento

data_x_val contendrá el 20% siguiente como datos de validación de entrenamiento

data_y_val contendrá el 20% siguiente como datos de validación objetivos de entrenamiento

data_x test contendrá todos los datos

Como se puede apreciar, al visualizar las dimensiones de nuestras variables podemos percatarnos de que no poseen columnas, esto debido al guardarlas con el método .iloc anteriormente. Para solucionar esto ejecutaremos la siguiente sección de código:

```
data_x = pd.DataFrame(data_x)
data_x_val = pd.DataFrame(data_x_val)
data_x_test = pd.DataFrame(data_x_test)
data_y = pd.DataFrame(data_y)
data_y_val = pd.DataFrame(data_y_val)
```

```
print(data_x.shape)          #94920,1
print(data_y.shape)          #94920,1
print(data_x_val.shape)      #27216,1
print(data_x_test.shape)     #135744,1
print(data_y_val.shape)      #27216,1
```

```
(94920, 1)
(94920, 1)
(27216, 1)
(135744, 1)
(27216, 1)
```

Fig. 2 9 Conversión de variables a DataFrames del modelo univariado

En la Fig 2.9, al reasignar los datos con el método pd.DataFrame, volvemos nuestras variables al tipo DataFrames, lo que les proporciona la columna que no tenían antes.

En el caso del modelo multivariado se hará:

```
#Datos de entrenamiento para el modelo(todas las columnas menos la objetivo)70% de los datos desde el principio
data_x = data_train[["Columna1","Columna2","Columna3","Columna4","Columna5","Columna6","Columna7",]
                    ].astype("float64")
```

```
#Datos de validaacion para el modelo(todas las columnas menos la objetivo)20% de los datos después de train
data_x_val=validation_data[["Columna1","Columna2","Columna3","Columna4","Columna5","Columna6","Columna7",]
                    ].astype("float64")
```

```
#Datos de prueba para nuestro modelo(todas las columnas menos la objetivo)100% de los datos despues de val
data_x_test = test[["Columna1","Columna2","Columna3","Columna4","Columna5","Columna6","Columna7",]
                    ].astype("float64")
```

```
data_y = data_train["Columna8"].astype("float64")
```

```
data_y_test = test["Columna8"].astype("float64")
```

```
data_y_val = validation_data["Columna8"].astype("float64")
```

```
data_y =pd.DataFrame(data_y)
data_y_val =pd.DataFrame(data_y_val)
data_y_test =pd.DataFrame(data_y_test)
```


Fig. 2 10 Creación de variables para alimentar el modelo multivariado

En el modelo multivariado utilizamos las mismas variables pero con la diferencia de que las variables de entrenamiento contendrán los datos de los parques circundantes al objetivo y las variables objetivas tendrán los datos del parque

objetivo (Columna8). Por lo que se realiza una asignación por nombres donde las variables quedan almacenando los siguientes datos:

data_x contendrá el 70% de los valores para el entrenamiento de las primeras 7 columnas

data_y contendrá el 70% de los datos como objetivos de entrenamiento de la columna 8(columna objetivo).

data_x_val contendrá el 20% siguiente como datos de validación de entrenamiento de las primeras 7 columnas

data_y_val contendrá el 20% siguiente como datos de validación objetivos de entrenamiento de la columna 8(columna objetivo).

data_x test contendrá el 10% siguiente como datos de validación de entrenamiento de las primeras 7 columnas

data_y_val contendrá el 10% siguiente como datos de validación objetivos de entrenamiento de la columna 8(columna objetivo).

Luego se realiza el mismo procedimiento mencionado para convertir a DataFrames las variables que no poseen columnas.

Ya con nuestras variables listas tanto en el modelo univariado como en e multivariado estamos listos para crear nuestra clase encargada de realizar el entrenamiento de los datos.

2.6 Entrenamiento de los modelos

Ahora si estamos listos para comenzar el entrenamiento de datos. Autokeras dispone para nosotros clases ya diseñadas para diferentes tareas específicas como la clasificación y la regresión ya sea de imágenes, texto o datos estructurados, además cabe recalcar que existe una clase de autokeras llamada AutoModel, la cual decide que por nosotros cuál de las técnicas es la mejor para nuestros datos. En nuestro caso vamos a trabajar con la clase TimeSerieForecaster, idónea para trabajar con series de tiempo, pero antes de declararla necesitamos crear tres variables que van a ser importantes para el entrenamiento.

predict_from = 1

predict_until = 12

lookback = 8

Como podemos observar en sus traducciones predict_from es un valor que nos va a especificar desde que posición va a comenzar la predicción, en nuestro caso está definida como 1 ya que queremos que comience a predecir desde el principio.
predict_until es el valor hasta el cual vamos a predecir y lo fijamos como 12. Esto significa que nuestros modelos predecirán los siguientes 12 espacios de tiempo.
lookback es la variable que le va a especificar al modelo cuantos espacios de tiempo del pasado va a tener en cuenta para predecir el siguiente y está definido como 8
Teniendo ya creadas estas variables vamos a declarar nuestra clase TimeSerieForecaster

```
predict_from = 1
predict_until = 12
lookback = 8
clf = ak.TimeseriesForecaster(
    lookback=lookback,
    predict_from=predict_from,
    predict_until=predict_until,
    max_trials=5,
    overwrite=True,
    project_name='Entrenamiento'
)
# Train the TimeSeriesForecaster with train data
clf.fit(
    x=data_x,
    y=data_y,
    validation_data=(data_x_val, data_y_val),
    batch_size=256,
    epochs=10,
)
```

Fig. 2 11 Creación de la clase TimeSerieForecaster

Como se puede evidenciar en la figura 2.11, la clase TimeSerieForecaster acepta varios parámetros, estos parámetros son opcionales, por ejemplo, si no especificamos el parámetro max_trials, la clase tomará su valor por defecto, o el parámetro metrics, el cual tiene de valor por defecto la métrica mse(mean_square_error) la cual nos mostrara el error cuadrático medio de nuestro modelo. Para conocer los valores por defecto de todos los parámetros que pueden ser pasados a la clase TimeSerieForecaster, desde donde este ejecutando el código, bastara con poner la línea de código help(clf) si ya creamos la clase o help(ak.TimeSerieForecaster) luego de importar Autokeras, este mostrara la documentación correspondiente a la versión de autokeras instalada.

A continuación de la creación de la clase TimeSerieForecaster, como se aprecia en la figura 2.11, utilizamos el método fit(), este método es el que dará inicio al entrenamiento de los datos. El método fit() posee argumentos obligatorios

```
fit(x=None, y=None, validation_split=0.2, validation_data=None, **k
wargs)
```

Como se aprecia en la Fig. 2.11 estos argumentos son :

X serán los datos de entrenamiento previamente guardados en train_data.

Y serán los objetivos de entrenamiento previamente guardados en train_targets.

Validation_data serán los datos de validación que hemos guardado previamente en las variables data_x_val y dat_y_val.

Validation_split será la cantidad de datos que serán separados para la validación, este parámetro tiene la característica de no ser obligatorio, su valor por defecto es 0.2, lo cual significa que el 20% de los datos serán de validación.

Al ejecutar este bloque de comandos nuestro modelo comenzara a entrenarse, este entrenamiento puede demorar y esta demora depende en gran medida del hardware en el que se está corriendo el entrenamiento.

```
Best val_loss So Far: 3809916.5
Total elapsed time: 00h 20m 11s
INFO:tensorflow:Oracle triggered exit
Epoch 1/10
371/371 [==============================] - 49s 77ms/step - loss: 4429841.0000 - mean_squared_error: 4429841.0000 - val_loss: 38
48275.7500 - val_mean_squared_error: 3848275.7500
Epoch 2/10
371/371 [==============================] - 20s 54ms/step - loss: 4425233.0000 - mean_squared_error: 4425233.0000 - val_loss: 38
43668.5000 - val_mean_squared_error: 3843668.5000
Epoch 3/10
371/371 [==============================] - 22s 59ms/step - loss: 4420449.0000 - mean_squared_error: 4420449.0000 - val_loss: 38
39588.0000 - val_mean_squared_error: 3839588.0000
Epoch 4/10
371/371 [==============================] - 19s 51ms/step - loss: 4416171.5000 - mean_squared_error: 4416171.5000 - val_loss: 38
35749.2500 - val_mean_squared_error: 3835749.2500
Epoch 5/10
371/371 [==============================] - 23s 61ms/step - loss: 4412300.5000 - mean_squared_error: 4412300.5000 - val_loss: 38
32300.7500 - val_mean_squared_error: 3832300.7500
Epoch 6/10
371/371 [==============================] - 15s 41ms/step - loss: 4408661.0000 - mean_squared_error: 4408661.0000 - val_loss: 38
28963.5000 - val_mean_squared_error: 3828963.5000
Epoch 7/10
371/371 [==============================] - 18s 48ms/step - loss: 4405103.0000 - mean_squared_error: 4405103.0000 - val_loss: 38
25688.0000 - val_mean_squared_error: 3825688.0000
Epoch 8/10
371/371 [==============================] - 19s 51ms/step - loss: 4401599.5000 - mean_squared_error: 4401599.5000 - val_loss: 38
```

Fig. 2 12 Ejemplo de entrenamiento del modelo univariado

```
Epoch 1/10
371/371 [==============================] - 36s 60ms/step - loss: 4415608.0000 - mean_squared_error: 4415608.0000 - val_loss: 38
23287.5000 - val_mean_squared_error: 3823287.5000
Epoch 2/10
371/371 [==============================] - 20s 54ms/step - loss: 4390226.5000 - mean_squared_error: 4390226.5000 - val_loss: 38
04331.7500 - val_mean_squared_error: 3804331.7500
Epoch 3/10
371/371 [==============================] - 19s 52ms/step - loss: 4371070.5000 - mean_squared_error: 4371070.5000 - val_loss: 37
87418.7500 - val_mean_squared_error: 3787418.7500
Epoch 4/10
371/371 [==============================] - 19s 52ms/step - loss: 4353515.5000 - mean_squared_error: 4353515.5000 - val_loss: 37
71215.2500 - val_mean_squared_error: 3771215.2500
Epoch 5/10
371/371 [==============================] - 19s 51ms/step - loss: 4336255.0000 - mean_squared_error: 4336255.0000 - val_loss: 37
55317.0000 - val_mean_squared_error: 3755317.0000
Epoch 6/10
371/371 [==============================] - 19s 50ms/step - loss: 4319501.5000 - mean_squared_error: 4319501.5000 - val_loss: 37
39627.0000 - val_mean_squared_error: 3739627.0000
Epoch 7/10
371/371 [==============================] - 19s 50ms/step - loss: 4302238.0000 - mean_squared_error: 4302238.0000 - val_loss: 37
24019.7500 - val_mean_squared_error: 3724019.7500
Epoch 8/10
```


Fig. 2 13 Ejemplo de entrenamiento del modelo multivariado

Como se puede apreciar en la Fig2.12 y 2.13, al realizarse el entrenamiento del modelo este nos arrojara datos, cada renglón de datos pertenecerá a una época del entrenamiento de un modelo, el cual ha sido elegido por nuestra clase, la variable loss es la métrica por defecto que tiene la clase, la cual coincide con la que le hemos especificado (mean_square_error) por lo que se aprecia que tienen los mismos valores, las demás nos proporcionan otros datos en los cuales no vamos a hacer énfasis debido a que nuestro objetivo es el mse. Es posible controlar la cantidad de datos que nos va a mostrar el entrenamiento con el parámetro verbose, el cual se lo podemos pasar a la clase fit().

Una vez terminado el entrenamiento, el mejor modelo encontrado por este será guardado en la variable clf, desde la cual podremos acceder a él para futuros usos ya sea para evaluar su rendimiento o para predecir espacios de tiempo en

el futuro, pero antes de pasar a eso vamos a ver la arquitectura que encontró nuestro mejor modelo y los valores de los hiperparametros.
El primer paso es exportar nuestro modelo. Esto lo haremos con el método export_model() y de la siguiente forma:

keras_model = clf.export_model()

Es importante mencionar que el modelo es exportado como un modelo de Keras, por lo cual ahora debe ser tratado como tal.
Ahora sí, vamos a ver que arquitectura e hiperparametros encontró para nuestros datos, primero veremos el mejor modelo univaiado y luego el mejor modelo multivariado.

```
# Primero exportamos el modelo a un modelo keras.
keras_model = clf.export_model()

# Ahora solicitamos el resumen del modelo:
keras_model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 8, 1)]            0

 bidirectional (Bidirectiona  (None, 8, 2)             24
 l)

 bidirectional_1 (Bidirectio  (None, 2)                32
 nal)

 regression_head_1 (Dense)   (None, 1)                 3

=================================================================
Total params: 59
Trainable params: 59
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2 14 Mejor modelo univariado

Como se ve en la fig2.13 El modelo tiene varias capas, incluyendo capas de entrada, capas bidireccionales y una capa densa de regresión. Cada capa tiene

un cierto número de parámetros entranbles que se utilizan para ajustar el modelo durante el entrenamiento. En total, el modelo tiene 59 parámetros entrenables.

Ahora veamos el modelo multivariado

```
# Primero exportamos el modelo a un modelo keras.
keras_model = clf.export_model()

# Ahora solicitamos el resumen del modelo:
keras_model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 8, 7)]            0

 bidirectional (Bidirectiona  (None, 8, 14)            840
 l)

 bidirectional_1 (Bidirectio  (None, 14)               1232
 nal)

 dropout (Dropout)           (None, 14)                0

 regression_head_1 (Dense)   (None, 1)                 15

=================================================================
Total params: 2,087
Trainable params: 2,087
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2 15 Mejor modelo mutivariado

Como se ve en la fig2.13 El modelo tiene varias capas, incluyendo capas de entrada, capas bidireccionales, capas de abandono(dropout) y una capa densa de regresión. Cada capa tiene un cierto número de parámetros entranbles que se utilizan para ajustar el modelo durante el entrenamiento. En total, el modelo tiene 2 087 parámetros entrenables.

Como podemos observar, los modelos poseen varias capas en común, aunque los valores de los hiperparámetros de estas capas sean diferentes. Podemos apreciar que el modelo univariado presenta una arquitectura ocupando dos

capas bidireccionales y una capa densa mientras que el modelo multivariado presenta además de las dos capas bidireccionales y la densa, agrega una capa de dropout(abandono), lo cual es una medida contra el sobreentrenamiento del modelo.

Ahora estamos listos para realizar las predicciones y evaluar que modelo presenta el mejor acierto.

2.7 Conclusiones:

Como vimos anteriormente AutoML nos permite realizar las mismas acciones que el Machine Learning tradicional con poco conocimiento acerca del tema. Los modelos de AutoML nos ahorran grandes cantidades de tiempo de entrenamiento y permite que sus modelos puedan ser exportados para su uso. Además la herramienta Autokeras posee una api fácil de usar para cualquier persona con conocimientos básicos de programación, además de poseer ejemplos en su página oficial que nos permiten reutilizarlos adecuandolos a nuestras propias necesidades. Con el entrenamiento de los datos, encontramos los mejores modelos que nos proporcionó autokeras, y estamos listos para realizar la predicción donde definiremos que modelo presenta emjor desempeño en las predicciones.

Capítulo 3 Descripción de la solución propuesta

Introducción:

En este capítulo se expondrán los resultados del trabajo, se realizará la evaluación del mejor modelo que nos brindó Autokeras, se realizarán las predicciones correspondientes y se graficarán para comparar estos valores con los valores de prueba. Además se calculará el por ciento de acierto de nuestras predicciones con el objetivo de presentar un valor expresado en porciento para calificar la calidad de nuestra predicción. Por último se expondrán las conclusiones generales de este trabajo.

3.1 Predicción de los modelos

Ya con nuestro modelo entrenado podemos realizar las predicciones observar la precisión de los modelos. Este proceso de predicción se ejecuta a través del método predict(), al que le vamos a pasar una variable que contenga todos los datos de entrenamiento originales y los datos que se desean predecir. El método predict() se ejecutara y luego de unos segundos tendrá listos los 12 valores predichos para los 12 siguientes espacios de tiempo. Estos valores serán las predicciones que ocupan los lugares justo después de los datos de validación de las variables objetivos.

```
# Predecir con el mejor modelo (incluye datos de entrenamiento originales).
predictions = clf.predict(data_x_test)
print(predictions.shape)
```

```
531/531 [==============================] - 31s 53ms/step
531/531 [==============================] - 27s 51ms/step
(12, 1)
```

Fig. 3 1 Predicción del modelo univariado

```
# Predecir con el mejor modelo (incluye datos de entrenamiento originales).
predictions = clf.predict(prueba)
print(predictions.shape)
```

```
531/531 [==============================] - 39s 67ms/step
531/531 [==============================] - 35s 66ms/step
(12, 1)
```

Fig. 3 2 Predicción del mejor modelo multivariado

Como podemos apreciar en la Fig3.1, la variable pasada al método es data_x_test, la cual contiene el dataset completo(los 808 dias de la columna objetiva).

En la Fig3.2 vemos una variable prueba la cual hemos creado y le asiganmos todas las mediciones de la columna objetivo.

En ambos casos las variables pasadas como parámetros al método predict() contienen los mismos datos y la misma cantidad de ellos. Ya que pertenecen a la columna objetivo de ambos modelos. Ahora grafiquemos las predicciones

```
import matplotlib.ticker as ticker
ig, ax = plt.subplots()
ax.plot(data_x_test.iloc[122136:122148], color = 'tab:red')
ax.plot(data['predicciones'], color = 'tab:green')
ax.xaxis.set_major_formatter(ticker.FormatStrFormatter('%d'))
plt.xticks(rotation=45)
plt.show()

```

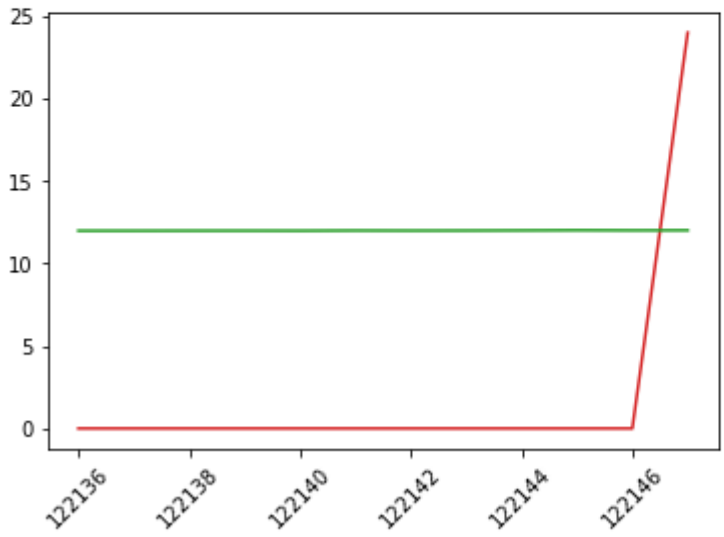

Fig. 3 3 Gráfica de la predicción univariada en 12 espacios de tiempo.

```
import matplotlib.ticker as ticker
ig, ax = plt.subplots()
ax.plot(data_x_test.iloc[122124:122148], color = 'tab:red')
ax.plot(data['predicciones'], color = 'tab:green')
ax.xaxis.set_major_formatter(ticker.FormatStrFormatter('%d'))
plt.xticks(rotation=45)
plt.show()

```

Fig. 3 4 Gráfica de la predicción univariada en 24 espacios de tiempo

```
import matplotlib.ticker as ticker
ig, ax = plt.subplots()
ax.plot(data_x_test.iloc[122124:122184], color = 'tab:red')
ax.plot(data['predicciones'], color = 'tab:green')
ax.xaxis.set_major_formatter(ticker.FormatStrFormatter('%d'))
plt.xticks(rotation=45)
plt.show()

```

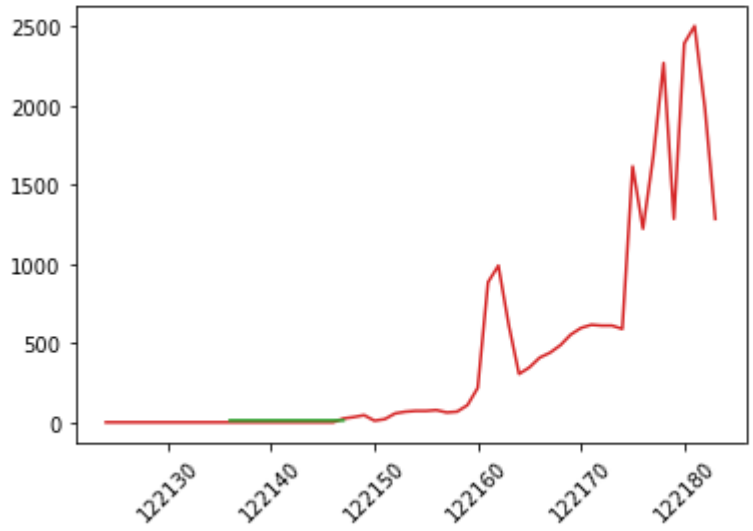

Fig. 3 5 Gráfica de la predicción univariada en 60 espacios de tiempo.

Como se puede apreciar en las Fig de la 2.3 a la 3.5, la predicción del modelo no es mala. En el tramo representado en la fig3.5 podemos apreciar que prácticamente pasa por encima de los valores reales.

Hora veamos las predicciones de nuestro modelo multivariado

```
import matplotlib.ticker as ticker
ig, ax = plt.subplots()

ax.plot(prueba2.iloc[122136:122148], color = 'tab:red')
ax.plot(otro['predictions'], color = 'tab:green')
ax.xaxis.set_major_formatter(ticker.FormatStrFormatter('%d'))
plt.xticks(rotation=45)
plt.show()

```

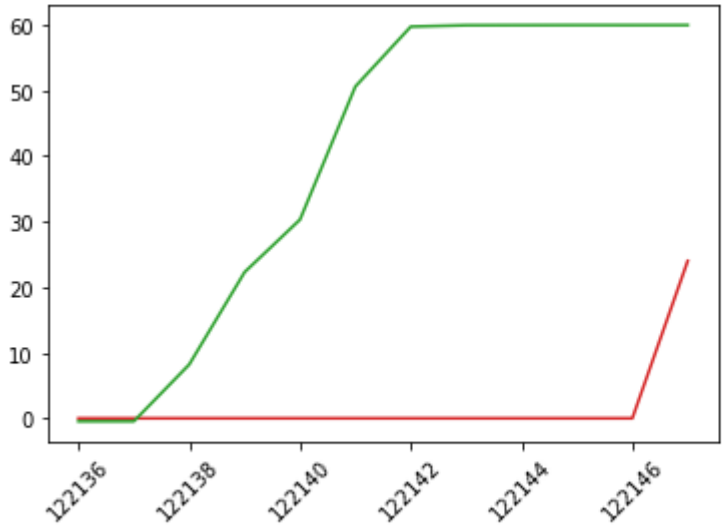

Fig. 3 6 Gráfica de la predicción multivariada en 12 espacios de tiempo

```
import matplotlib.ticker as ticker
ig, ax = plt.subplots()

ax.plot(prueba2.iloc[122136:122160], color = 'tab:red')
ax.plot(otro['predictions'], color = 'tab:green')
ax.xaxis.set_major_formatter(ticker.FormatStrFormatter('%d'))
plt.xticks(rotation=45)
plt.show()

```

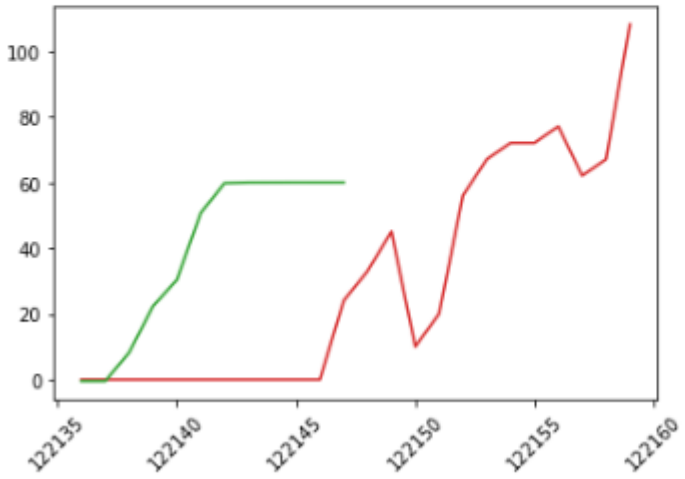

Fig. 3 7 Gráfica de la predicción multivariada en 24 espacios de tiempo.

```
import matplotlib.ticker as ticker
ig, ax = plt.subplots()

ax.plot(prueba2.iloc[122124:122184], color = 'tab:red')
ax.plot(otro['predictions'], color = 'tab:green')
ax.xaxis.set_major_formatter(ticker.FormatStrFormatter('%d'))
plt.xticks(rotation=45)
plt.show()

```

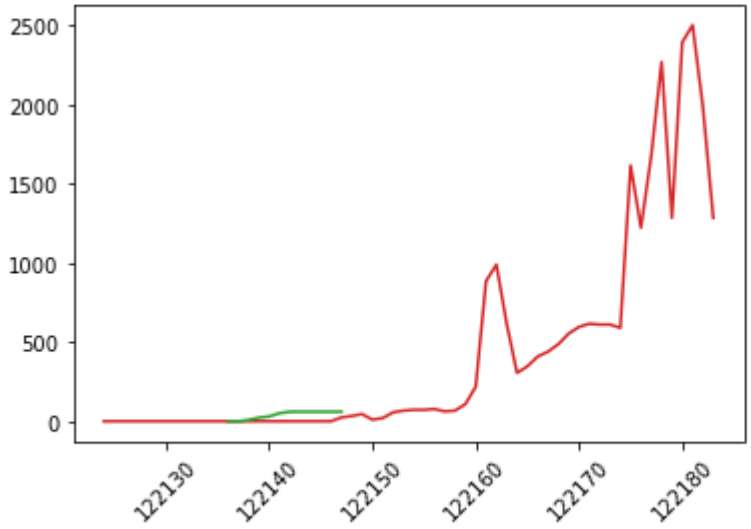

Fig. 3 8 Gráfica de la predicción multivariada en 60 espacios de tiempo.

Como podemos apreciar desde la fig. 3.6 a la 3.8 la predicción del modelo multivariado se encuentra a mayor distancia de los valores reales que la predicción del modelo univariado. Por lo que podríamos concluir que el modelo univariado presenta mejor acierto en los 12 pasos de tiempo futuro que el modelo multivariado

Ahora calculemos los errores cuadráticos medios de las predicciones

Para el caso del modelo univariado seria:

```
suma = 0
for i in range(len(predictions)):
    suma+=(predictions[i]- data_x_test.iloc[0:12].values[i] )**2
suma / len(predictions)
import math
math.sqrt(suma)
```

```
41.55809566392777
```

Fig. 3 9 Calculo del mse de la predicción en el modelo univariado.

Y en el caso del modelo multivariado seria:

```
suma = 0
for i in range(len(predictions)):
    suma+=(predictions[i]- data_test.iloc[0:12].values[i] )**2
suma / len(predictions)
import math
math.sqrt(suma)
```

```
152.60806624404728
```

Fig. 3 10 Calculo del mse de la predicción en el modelo multivariado.

Como podemos apreciar al comparar los cálculos del mse de las predicciones de los dos modelos, el modelo univariado presenta menor error. Esto significa que el modelo univariado se desempeña mejor que el multivariado para la predicción de potencia generada del parque Yaguaramas.

3.2Conclusiones

En este capítulo pudimos apreciar que al realizar las predicciones de los modelos univariado y multivariado, el primero presenta mejor acierto en la predicción. Por lo que llegamos a la conclusión de que los parques circundantes al parque objetivo tienen menor influencia a la hora de predecir la potencia generada del parque Yaguaramas.

Recomendaciones

Autokeras no es la única herramienta existente para el AutoML. Por lo que se recomienda probar el entrenamiento de los modelos en otras herramientas y comparar los resultados para así tener una mejor visión de lo eficaz que es autokeras para la predicción de series temporales univariadas y multivariadas Algunas de estas son H2O AutoML, TPOT, Auto-Sklearn y Autogoal.

Referencias bibliográficas

[1] A. M. Jiménez, «Modelos de predicción a corto plazo de la generación eléctrica en instalaciones fotovoltaicas», p. 286, 2014.

[2] I. Borrajero Montejo, A. Baró Pérez, J. C. Peláez Chávez, y M. Hinojosa Fernández, «Pronóstico de la radiación solar y potencia a generar en las plantas fotovoltaicas conectadas a la red el éctrica nacional.» Centro de Física de la Atmósfera Instituto de Meteorología, octubre de 2016.

[3] J. E. Alfonso Ortiz y D. J. Bogoya Mora, «Predicción del recurso solar diario mediante técnicas de machine learning para la proyección de generación de energía eléctrica», nov. 2021, Accedido: 15 de junio de 2023. [En línea]. Disponible en: http://repository.udistrital.edu.co/handle/11349/28947

[4] I. The IEA examines the full spectrum of energy issues including oil, gas and coal supply and demand, renewable energy technologies, electricity markets, energy efficiency, access to energy, demand side management and much more. Through its work, the IEA advocates policies that will enhance the reliability, affordability and sustainability of energy in its 30 member countries, 8 association countries and beyond., «Net Zero by 2050 - A Roadmap for the Global Energy Sector», p. 224.

[5] R. Delgado, R. Jiménez Borges, y J. Yanes, *Anteproyecto de sistema solar fotovoltaico en la Delegación Provincial de Materiales de la Construcción. Cienfuegos.* 2022.

[6] M. de justicia Republica de Cuba, «Gaceta Oficial», Gaceta Oficial. Accedido: 19 de octubre de 2023. [En línea]. Disponible en: https://www.gacetaoficial.gob.cu/es

[7] «¿Qué es Anaconda? -Escuela Internacional de Posgrados». Accedido: 15 de junio de 2023. [En línea]. Disponible en: https://eiposgrados.com/blog-python/que-es-anaconda/

[8] «ONEI | Oficina nacional de estadísticas e información». Accedido: 19 de octubre de 2023. [En línea]. Disponible en: https://www.onei.gob.cu/

[9] C. R. Algarin y O. R. Álvarez, «Un panorama de las energías renovables en el Mundo, Latinoamérica y Colombia», p. 15.

[10] M. A. GómezRodríguez *et al.*, «Pronóstico de la generación eléctrica de sistemas fotovoltaicos. Un inicio en Cuba desde la universidad», *Rev. Univ. Soc.*, vol. 13, n.º 1, pp. 253-265, feb. 2021.

[11] L. Sobrecueva, *Automated machine learning with AutoKeras: deep learning made accessible for everyone with just few lines of coding*. Birmingham: Packt, 2021.

[12] «The new era of automl - ProQuest». Accedido: 26 de noviembre de 2023. [En línea]. Disponible en: https://www.proquest.com/openview/a3de9c2fd93296a61dc656829a64617f/1?pq-origsite=gscholar&cbl=48426

[13] C. Castilla Gil, «Identificación y análisis de las principales tecnologías de AutoML», 2021, Accedido: 26 de noviembre de 2023. [En línea]. Disponible en: https://riull.ull.es/xmlui/handle/915/25418

[14] «Presentación — Fundamentos de Programación en Python». Accedido: 12 de octubre de 2023. [En línea]. Disponible en: https://www2.eii.uva.es/fund_inf/python/notebooks/00_Introduccion/Introduccion.html#Introducci%C3%B3n

[15] «6 Librerías imprescindibles de Python para Machine Learning», IA SOLVER. Accedido: 15 de junio de 2023. [En línea]. Disponible en: https://iasolver.es/6-librerias-de-python-para-machine-learning/