UNIVERSIDAD TÉCNICA DE COTOPAXI

POSGRADO

UNIVERSIDAD TÉCNICA DE COTOPAXI

DIRECCIÓN DE POSGRADO

MAESTRÍA EN ELECTRICIDAD

MODALIDAD: PROYECTO DE INVESTIGACIÓN

Título:

DISEÑO DE UN ALGORITMO UTILIZANDO MACHINE LEARNING PARA LA PREDICCIÓN DE LA RADIACIÓN SOLAR EN EL SECTOR DE LASSO

Trabajo de titulación previo a la obtención del título de magister en Electricidad mención sistemas eléctricos de potencia

Autor:
Lalaleo Achachi Diego Fernando
Tutor:
Proaño Maldonado Xavier Alfonso Msc

LATACUNGA –ECUADOR
2021

AVAL DEL TUTOR

En mi calidad de Tutor del Trabajo de Titulación "DISEÑO DE UN ALGORITMO UTILIZANDO MACHINE LEARNING PARA LA PREDICCIÓN DE LA RADIACIÓN SOLAR EN EL SECTOR DE LASSO" presentado por Lalaleo Achachi Diego Fernando, para optar por el título magíster en Electricidad mención sistemas eléctricos de potencia

CERTIFICO

Que dicho trabajo de investigación ha sido revisado en todas sus partes y se considera que reúne los requisitos y méritos suficientes para ser sometido a la presentación para la valoración por parte del Tribunal de Lectores que se designe y su exposición y defensa pública.

Latacunga, agosto, 13, 2021

.....

Msc. Xavier Alfonso Proaño Maldonado

CC.: 0502656424

UNIVERSIDAD TÉCNICA DE COTOPAXI

POSGRADO

AVAL DEL TRIBUNAL

El trabajo de Titulación: "DISEÑO DE UN ALGORITMO UTILIZANDO MACHINE LEARNING PARA LA PREDICCIÓN DE LA RADIACIÓN SOLAR EN EL SECTOR LASSO" ha sido revisado, aprobado y autorizado su impresión y empastado, previo a la obtención del título de Magíster en Electricidad; el presente trabajo reúne los requisitos de fondo y forma para que el estudiante pueda presentarse a la exposición y defensa.

Latacunga, octubre, 27, 2021

.....

MSc. Edgar Roberto Salazar Achig
0502847619
Presidente del tribunal

.....

MSc. José Efrén Barbosa Galarza
0501420723
Lector 2

.....

PhD. Secundino Marrero Ramírez
1757107907
Lector 3

DEDICATORIA

El presente trabajo de grado está dedicado a las personas que buscan entender los avances y cambios de la sociedad como es el caso de la aplicación del aprendizaje de máquina que hoy en día se escucha en todos lados, pero poco se conoce sobre el funcionamiento y la aplicación en las actividades diarias.

Diego Lalaleo

AGRADECIMIENTO

En el desarrollo del trabajo y al llegar a la finalización se ha recorrido por varios caminos encontrando a personas que han compartido los conocimientos necesarios para cumplir con el objetivo planteado, por tal motivo el agradecimiento es para las personas que aportaron con la colaboraron, Tutor, Docente, Familia y personas cercanas.

Diego Fernando Lalaleo achachi.

RESPONSABILIDAD DE AUTORÍA

Quien suscribe, declara que asume la autoría de los contenidos y los resultados obtenidos en el presente trabajo de titulación.

Latacunga, agosto, 13, 2021

…. …

Diego Fernando Lalaleo Achachi
1804617841

RENUNCIA DE DERECHOS

Quien suscribe, cede los derechos de autoría intelectual total y/o parcial del presente trabajo de titulación a la Universidad Técnica de Cotopaxi.

Latacunga, octubre 27, 2021

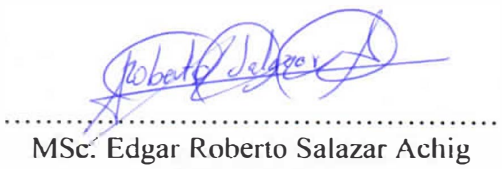
….. ..

Diego Fernando Lalaleo Achachi
1804617841

AVAL DEL PRESIDENTE DEL TRIBUNAL

Quien suscribe, declara que el presente Trabajo de Titulación: Titulación: "DISEÑO DE UN ALGORITMO UTILIZANDO MACHINE LEARNING PARA LA PREDICCIÓN DE LA RADIACIÓN SOLAR EN EL SECTOR LASSO" las correcciones a las observaciones realizadas por los lectores en sesión científica del tribunal.

Latacunga, octubre, 27, 2021



MSc. Edgar Roberto Salazar Achig
0502847619

UNIVERSIDAD TÉCNICA DE COTOPAXI
DIRECCIÓN DE POSGRADO

MAESTRÍA EN ELECTRICIDAD
MENCIÓN SISTEMAS ELÉCTRICOS DE POTENCIA

Título: Diseño de un algoritmo utilizando Machine Learning para la predicción de la radiación solar en el sector de Lasso

Autor: Lalaleo Achachi Diego Fernando
Tutor: Proaño Maldonado Xavier Alfonso Msc

RESUMEN

El diseño del algoritmo para la predicción de la radiación solar mediante técnicas de Machine Learning, se desarrolló aplicando las librerías de keras y tenson Flow para la creación de la red neuronal LSTM modelo secuencial que seleccionó datos anteriores y predijo una semana posterior, se registró con una estación meteorológica la medición de las variables del clima en intervalos de cada minuto para exportar el archivo de medición en formato .CSV, se codificó en Google Colab el procesamiento de datos para que los recursos informativos lo realice el servidor, el lenguaje de programación es Python, los resultados obtenidos son valores de radiación solar en el rango de 6:00 a 18:00 horas, es relevante la secuencia de horas con el tiempo, si un valor fue nulo o se altera el orden de los datos, los resultados son diferentes a los reales por estar aplicado series de tiempo en los datos de predicción y el resultado es la predicción de cada hora de los siguientes siete días continuos.

PALABRAS CLAVE: Machine Learning; Radiación solar; red neuronal; LSTM

UNIVERSIDAD TÉCNICA DE COTOPAXI

MAESTRÍA EN ELECTRICIDAD

MENCIÓN SISTEMAS ELÉCTRICOS DE POTENCIA

Topic: "Design of an algorithm by using Machine Learning for the prediction solar radiation into Lasso sector".

Author: Lalaleo Achachi Diego Fernando
Tutor: Proaño Maldonado Xavier Alfonso Msc

ABSTRACT

The algorithm design for the solar radiation prediction by using Machine Learning techniques, it was developed of applying the keras and tenson Flow libraries for the sequential model LSTM neural network creation, which it was selected previous data and they were predicted a subsequent week, it was recorded with the Weather Station the climate variables measurement into every minute intervals for exporting the measurement file in .CSV format, it was coded in Google Colab data processing for informational resources is performed by the server, the programming language is Python, the got results are solar radiation values into range from 6:00 to 18:00 hours, it is relevant the hours sequence over time, if a value was null or it is altered the data order, the results are different from the real ones, they are applied to time series in the prediction data and the result is the prediction of following each hour at seven continuous days..

KEYWORDS: Machine Learning, solar radiation, neural network, LSTM`.

Yo, Mg. Marco Paúl Beltrán Semblantes con cédula de identidad número 0502666514 Licenciado en Ciencias de la Educación, especialización Inglés con número de registro de la SENESCYT 1020-06-701921; **CERTIFICO** haber revisado y aprobado la traducción al idioma inglés del resumen del trabajo de investigación con el título: **"Diseño de un algoritmo utilizando Machine Learning para la predicción de la radiación solar en el sector de Lasso"** de: Lalaleo Achachi Diego Fernando, aspirante a magister en Electricidad mención Sistemas Eléctricos de Potencia

Latacunga, octubre, 2021

Mg. Marco Paúl Beltrán Semblantes
0502666514

CENTRO DE IDIOMAS

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El interés de conocer el comportamiento y utilización de la radiación solar nace desde tiempo atrás con la concentración de calor para la quema de flota en los enfrentamientos entre ejércitos que la denominaban fuego griego, también con experimentos como el primer motor que usaba la luz de sol para cambiar el estado del agua y producir movimiento, con la concentración del calor se dio inicio para el desarrollo de hornos solares utilizados en la fundición de metales o cocción de verduras.

La presente investigación se realiza en el sector de Lasso perteneciente al cantón Latacunga, con la instalación de una estación meteorológica en un lugar específico para el registro de los parámetros climáticos en intervalos de un minuto y con técnicas de Machine Learning se programe un algoritmo que permita estimar la radiación solar en un tiempo determinado a corto plazo.

La contribución de la investigación es de mejorar la planificación de actividades relacionadas con la incidencia de la radiación solar en diferentes campos de estudio como en la generación de electricidad, instalación de calentadores solares, gestión de cultivos e incluso el cuidado de la salud humana para optimizar los recursos que pueden ser utilizados para solventar terceras necesidades, también con los resultados obtenidos se podrá contrastar con estudios semejantes a la predicción de la irradiancia, analizar las posibles variantes en la planificación diaria o semanales por las instituciones públicas y privadas que pueden estar dentro o fuera del sector.

La investigación buscará obtener los valores de radiación solar predecidos por cada hora mediante la aplicación de aprendizaje de máquina conocido como Machine Learning con un algoritmo que se ajuste a la tendencia estacionaria de la radiación para la planificación a futuro de generación electricidad con paneles fotovoltaicos en minicentrales, considerando que para las planificaciones se utiliza la curva diaria de radiación solar incidente.

Antecedentes:

El tema se encuentra dentro de la línea de investigación de Energías alternativas y renovables correspondiente a la maestría en Electricidad, enfocándose en la sub línea de conversión y uso racional de la energía eléctrica, con el análisis, evalúo y la adecuada utilización de recursos naturales para el desarrollo de proyectos de energía renovable no convencional fortaleciendo el uso racional de la energía.

Planteamiento del problema

La implementación de modelos de predicción aplicando Machine Learning se ha convertido en retos importantes por las empresas, la utilización de modelos de algoritmos se convierte en un problema en la selección de un modelo por la aparición de varios modelos de regresión y clasificación. Los porcentajes de implementación de proyectos de Machine Learning son pequeños por razones como la implementación existen una gran variedad de plataformas y lenguajes de programación que se puede realizar machine Learning y al migrar la codificación de un lenguaje a otro produce errores en la ejecución, también como programas libres y privativos aportan a la incompatibilidad, los programas libres como Python y R son populares para aplicación de machine Learning pero se dificulta al ejecutar en programas licenciados como C++ o Java, el rendimiento es reducido considerando la velocidad, precisión y modelo. Si se utiliza redes neuronales para Machine Learning o Deep Learning en aprendizaje profundo los recursos computacionales son altos que implica el uso de equipos costosos, en la portabilidad carece de capacidad de migrar un componente de software que limita a las organizaciones y podría ser una barrera a la hora de crear modelos y desplegarlo, los modelos de algoritmos con el tiempo presentan problemas de escalabilidad en el que por la configuración inicial con datos estáticos presentan problemas cuando el volumen de los datos aumenta y no se han adaptado un enfoque para el análisis de producción, con los cambios de los datos y nuevos modelos el rendimiento de los algoritmos se reducen por la falta de estrategia de implementación y monitoreo.

Formulación del problema

La necesidad de estimar la radiación solar incidente en el sector de Lasso permitirá la aplicación de técnicas Machine Learning que prediga por cada hora el comportamiento de la variable radiación solar.

Objetivo General

Diseñar un algoritmo mediante la aplicación de técnicas de machine Learning para estimar el comportamiento de la radiación solar en el sector de Lasso

Objetivos Específicos

- Investigar información bibliográfica de Modelos predictivos aplicados en la predicción de energía solar.
- Recopilar los parámetros climáticos mediante la instalación de una estación meteorológicos en el sector de Lasso.
- Programar la red neuronal mediante el desarrollo del algoritmo para predecir el comportamiento de la radiación.
- Comparar a corto plazo el algoritmo utilizando las mediciones de la estación meteorológica.

Sistemas de tareas en relación a los objetivos específicos: estas son actividades que se realizarán para dar cumplimiento a cada objetivo específico planteado.

Objetivos específicos	Actividad (tareas)	Resultado de la actividad	Descripción de la actividad (técnicas e instrumentos)
Investigar información bibliográfica de Modelos predictivos aplicados en la predicción de energía solar.	-Definición de la clase de datasets utilizada -Identificación del método lineal o categórico -Descripción de los modelos aplicados en predicción de datos. -Determinación de los parámetros de entrenamiento.	-Describir los modelos matemáticos aplicados en el análisis de base de datos históricos mediante la aplicación de estadística.	-Recolección de la información de los programas diseñados para análisis de datos -Representaciones gráficas para el análisis de resultados.
Recopilar los parámetros climáticos mediante la instalación de una estación meteorológicos en el sector de Lasso	-Instalación de la estación meteorológica en el sector de Lasso -Configuración del usuario y clave de ingreso al sistema -Registro de los parámetros climáticos por cada mes -Exportación de los datos registrados en archivo Excel	-Instalación de la estación meteorológica configurada para que registre las mediciones del comportamiento del clima.	-Recolección de la información de estaciones meteorológicas utilizadas para la grabación del clima. -Enlazar la estación con la red de internet para acceder al panel de control y monitorear en tiempo real el clima.
Programar un modelo de algoritmo mediante técnicas de Machine	-Definición de la clase de datasets utilizada (database o datasets) -Selección del	-Seleccionar el algoritmo que brinde mejores resultados en la predicción del clima con el software	-Ingresar los datos registrados en el computador mediante software de análisis de datos

Learning para predecir el comportamiento de la radiación.	método lineal o categórico -Elaboración del modelo de algoritmo para la predicción de datos. -Determinación del modelo con mejor ajuste a los resultados.	disponible para la programación.	-Selección de Software de análisis de datos.
Comparar a corto plazo el algoritmo utilizando las mediciones de la estación meteorológica.	-Selección de los datos ingresados al modelo de algoritmo. -Modelación y obtener los resultados predictivos. -Examinación de los datos reales medidos por la estación con los datos precedidos por el algoritmo. -Comparación del modelo con los datos reales medidos	-Comparación del algoritmo seleccionado ingresando los datos registrados de un periodo determinado con los datos registrados.	-Datos de predicción obtenidos del algoritmo de un tiempo determinado -Representación de los resultados mediante gráficas comparativas

Justificación

Las grandes empresas ha utilizado el aprendizaje de máquina para mejorar el rendimiento en la producción de productos y ventas, con el reconocimiento de los patrones de comportamiento de los datos ayudan a utilizar la gran variedad de modelos de regresión y clasificación, la disponibilidad de varias plataformas para realizar Machine Learning permiten la fácil distribución y acogida de los usuarios finales, la facilidad del lenguaje de programación en Python ha ayudado a estandarizar la programación en varios programas que pueden ser ejecutados de forma local en el PC o remota desde el drive reduciendo los altos costos por recursos de los equipos porque el procesamiento lo realiza el servidor, las aplicaciones de algoritmos permite la monitorización de la calidad de diferentes entornos como de ciudades inteligentes, industrias , medioambiente, variables climáticas, detección de objetos y a la ejecución de la industria 4.0, todos las aplicaciones garantiza una calidad de integración de datos almacenados para la obtención de mejores resultados en las predicciones de tendencias y la interpretación de necesidades frente a grandes volúmenes de datos para mejorar la toma de decisiones. Las plataformas automatizadas de machine Learning permiten soportar el desarrollo y comparación de múltiples modelos como en la parte de negocios que se elige el modelo que mejor se adapte a las necesidades precisión y recurso de cálculo, el aumento de la aplicación de técnicas de machine Learning en las empresas pueden ser desarrollados por expertos que trabajan en pequeños proyectos y mejoran notablemente la eficiencia de la empresa trabajando en sectores estratégicos que produce mejores resultados.

Para Solucionar el problema de dinamismo del comportamiento de la radiación solar y el alto nivel de complejidad en la estimación se decide la aplicación de Machine Learning con resultados más rápidos y preciso en un gran volumen de datos, que permite la predicción a una situación futura, tomar decisiones más viables, desarrollar mejoras de acciones en el proceso laboral que influye el sol, también en la actualidad con el avance dela tecnología y equipos informáticos cualquier persona puede hacer uso de machine Learning y puede ingresar los datas a registrados a análisis y obtener buenos resultados, lo que antiguamente solo las grandes empresas utilizaban el aprendizaje de máquina. Por las razones

mencionadas es justificable realizar un algoritmo de predicción de la radiación solar que puede ser aplicado en varios campos.

Hipótesis

El diseño de un algoritmo utilizando Machine Learning permitirá la estimación del comportamiento de la radiación solar en el sector de Lasso

CAPÍTULO I.

1. FUNDAMENTACIÓN TEORICA- METODOLÓGICA

1.1 Antecedentes de la investigación

La investigación con los métodos de aprendizaje automático para la predicción de la radiación solar, recomienda pronosticar la salida de los sistemas solares, es esencial enfocar la predicción en la irradiancia solar. El pronóstico global de la radiación solar se puede realizar por varios métodos; Las dos grandes categorías son las imágenes en la nube combinadas con modelos físicos y los modelos de aprendizaje automático. En este contexto, el objetivo de este documento es proporcionar una visión general de los métodos de pronóstico de la irradiación solar utilizando enfoques de aprendizaje automático, también en muchos documentos describen metodologías como redes neuronales o soporte de regresión vectorial, se mostrará que otros métodos (árbol de regresión, bosque aleatorio, aumento de gradiente y muchos otros) comienzan a usarse en este contexto de predicción [1].

La irradiancia solar es la medida de la cantidad de energía del Sol por unidad de área. La irradiancia solar tiene un grado muy alto de variabilidad debido a muchos factores ambientales, incluida la nubosidad, la humedad relativa y la temperatura del aire. La predicción de la irradiancia solar es muy útil para medir la producción futura de energía solar y la programación de energía. La irradiancia solar en tiempo real se puede pronosticar utilizando modelos de aprendizaje de mecanizado o basados en la física, ambos con sus respectivas compensaciones. Con un modelo estático no puede capturar completamente la dinámica de la irradiancia solar tanto en los días en los que está presente la nubosidad como en los días donde no hay nubosidad.[2]

La razón fue realizar estimaciones de radiación solar utilizando distintas técnicas de aprendizaje automático para regresión y clasificación; el principal objetivo fue evaluar su desempeño. Aunque en la mayoría de los trabajos relacionados los investigadores utilizaron

el aprendizaje profundo para la predicción de la radiación solar, este estudio demostró que, si bien las redes neuronales artificiales son la técnica más utilizada, otros algoritmos de aprendizaje automático como Random Forest, Máquinas de Soporte Vectorial y AdaBoost también proporcionan estimaciones con suficiente precisión para ser utilizados en este campo de estudio.[3]

Al desarrollar los métodos de aprendizaje automático para la predicción de la radiación solar, los autores concluyen que los pronósticos metrológicos no son todavía lo suficientemente buenas porque en los diferentes análisis realizados, las predicciones fallan a la hora de predecir nubes, aunque en días que se hacen bien y otros en los cuales no se predice en ningún momento esa nube. Además de esto la no corrección de la predicción con respecto al día anterior hace seguir cometiendo un error innecesario a la larga sobre todo en los meses de verano [3].

Los métodos de aprendizaje automático para la predicción de la radiación solar, recomienda que se debe aplicar el modelo que presenta el mejor ajuste para predecir la radiación solar para el área de estudio es la regresión múltiple en la que se incluyen todas las variables, no siendo este el modelo más simple para su estimación. Se sugiere realizar los mismos procedimientos aquí presentados con datos multianuales, que permitan identificar con mayor precisión las variaciones multianuales a que tengan lugar las variables meteorológicas; pudiendo establecer una replicabilidad posible de los puntos, que permitan comprobar la influencia de cada uno en las regresiones planteadas [4].

Los métodos de aprendizaje automático para la predicción de la radiación solar, los autores manifiestan que se debe instalar el solarímetro en el INTEC y se realizarán una serie de mediciones que abarcaron todo el mes de febrero luego se decidió volver a calibrar el quipo en el INTEC y el ajuste entre valores medidos y teóricos resultó más apropiado. Se determinó un error porcentual del 7,7 %, algo mayor que en ediciones anteriores. Sin embargo, si se consideran los valores por encima de los 200W/m2 el error se reduce al 4,5

%. Este dato no es menor si tenemos en cuenta que para la aplicación de las mediciones a nuestro proyecto, la zona de interés está por encima de dichos valores [5].

Los métodos de aprendizaje automático para la predicción de la radiación solar, recomienda que se debe realizar el estudio que se enfoque a comprobar los datos del modelo contra los datos experimentales registrados en el suelo por el sensor de radiación ultravioleta para una estación radiométrica localizada en Heredia, Costa Rica. El sistema de datos cubre diferentes horas del día en varios meses del año, período que asegura una variedad de condiciones climáticas de nuestro medio, sin embargo, los datos fueron limitados a condiciones de cielos claros [6].

Los métodos de aprendizaje automático para la predicción de la radiación solar, el autor recomienda que para obtener un resultado claro de debe desarrollar con base en los cálculos de las transmitancias espectrales debido al ozono, dispersión de Rayleigh y del aerosol atmosférico. Se logró obtener separadamente la irradiación ultravioleta- B directa y la irradiancia ultravioleta-B difusa, para luego calcular la irradiación global ultravioleta-B mediante la apropiada suma de ambos componentes [7].

Los métodos de aprendizaje automático para la predicción de la radiación solar, la aplicación de los métodos a las redes neuronales mejora los resultados al agregar datos exógenos al pronóstico, pero se deben probar otros modelos de pronóstico. Específicamente, sería deseable modelar el movimiento y la formación de nubes, ya que uno de los problemas clave es la nubosidad en la parte norte de la isla. El estudio nos permitió determinar diferentes resultados según las estaciones y las épocas del año, por lo que una posible opción sería combinar los modelos óptimos para cada caso [8].

Los métodos de aprendizaje automático para la predicción de la radiación solar, los autores manifiestan que la relación entre mineralización, reciclaje del carbono orgánico edáfico y variables controladas parcialmente por la radiación solar, la inclusión de esta variable en forma de Radiación Solar Potencial Directa mejoró la predicción de COE superficial en

cerca de 14%, respecto al modelo que emplea solamente atributos del terreno y el uso del programa Sol_Rad SIG es una alternativa viable para incluir la radiación solar en el análisis predictivo de COE en países en vías de desarrollo [9].

Los métodos de aprendizaje automático para la predicción de la radiación solar, ayudan a que las redes neuronales mejoren los resultados al agregar datos exógenos al pronóstico, pero se deben probar otros modelos de pronóstico. Específicamente, sería deseable modelar el movimiento y la formación de nubes, ya que uno de los problemas clave es la nubosidad en la parte norte de la isla. El estudio nos permitió determinar diferentes resultados según las estaciones y las épocas del año, por lo que una posible opción sería combinar los modelos óptimos para cada caso [10].

La investigación de los métodos de aprendizaje automático para la predicción de la radiación solar, concluyen que para analizar las redes neuronales de tipo perceptrón multicapa son adecuadas para realizar buenas estimaciones de la radiación solar horaria, aun usando como única variable la temperatura. Los resultados obtenidos muestran que sin datos de heliofanía también es posible estimar la radiación solar horaria satisfactoriamente a partir de variables normalmente provistas por estaciones meteorológicas de uso extendido en Argentina y debido a que se consideraron variables de fácil obtención como datos de entrada de la red, estos modelos podrían aplicarse a cualquier sitio que cuente con una estación meteorológica básica [11].

El estudio de los métodos de aprendizaje automático para la predicción de la radiación solar, se debe aplicar la respuesta a la pregunta de cuál o cuáles son los mejores algoritmos para predecir radiación, no es fácil de responder. Esto en gran parte a los pronósticos resultantes que difieren en muchos puntos geográficos. Así pues, para cada región en particular debe escoger el modelo más apropiado el cual debe seguir una tendencia de aplicación en regiones similares [12].

Los métodos de aprendizaje automático para la predicción de la radiación solar, los autores manifiestan que se debe desarrollar como trabajo futuro, la predicción de radiación solar, con el manejo de los datos registrados por el grupo de investigación en áreas rurales, en cada minuto, entre los que se encuentran: radiación solar horizontal, velocidad del viento, humedad relativa y temperatura del ambiente, para predecir tanto diaria como anualmente la radiación solar en este tipo de áreas donde se puede aprovechar este recurso natural y orientarlo a la población que no posee acceso a la red eléctrica convencional [13].

Los métodos de aprendizaje automático para la predicción de la radiación solar, los autores recomiendan aplicar un modelo híbrido de aprendizaje automático para la electricidad. pronóstico de demanda, basado en Bayesian Clustering by Dynamics (BCD) y Soporte de máquina de vectores (SVM). En el modelo propuesto, un clasificador BCD es primero aplicado para agrupar el conjunto de datos de entrada en varios subconjuntos por la dinámica de series de carga de una manera no supervisada, y luego, grupos de 24 SVM para la curva de demanda de electricidad del día siguiente se utiliza para ajustar los datos de capacitación de cada subconjunto. En el experimento numérico, el modelo propuesto ha sido entrenado y probado en los datos de la carga histórica de la ciudad de Nueva York [14].

La investigación de los métodos de aprendizaje automático para la predicción de la radiación solar, para obtener un mejor resultado se debe aplicar un adecuado método de optimización de los parámetros y con el uso de kernels específicos. Respecto al diseño de los kernels, se ha propuesto una metodología basada en el conocimiento proporcionado por el experto y también se ha propuesto un método automático basado en Programación Genética. Las conclusiones de este estudio han mostrado cómo la tarea del diseño del kernel con Programación Genética es demasiado costosa computacionalmente aun teniendo en cuenta que el método de evaluación de los kernels ha sido optimizado [15].

En la predicción corto plazo de la radiación solar es necesario recoger los datos de la zona de interés, con la utilización de sensores que recepten señales del temperatura y humedad

capturando los parámetros y envíen al servidor para el almacenamiento, actualmente existen gran variedad de modelos de piranómetros pero el costo es elevado lo que limita a la instalación de varios sensores en diferentes lugares estratégicos, por ese motivo una alternativa es la de la creación de un elemento sensor que el soto sea bastante inferior a los tradicionales pero con la misma calidad de mediciones, con lo que ayuda al mejoramiento de la predicción pero es necesario una colaboración de varias partes entre la ciudadanía y lugar de investigación [16].

1.2 Fundamentación Teórica.

1.2.1 El Sol como fuente de energía

Como se ha indicado en la introducción, el Sol es nuestra fuente principal de energía. La luz, el calor y la fuerza gravitatoria del Sol dan lugar a la formación de diversos fenómenos en la naturaleza, que podemos aprovechar por transformación, para la obtención de otras energías, como la electricidad por medio de células fotoeléctricas (paneles fotovoltaicos) agua caliente sanitaria (ACS) y Calefacción [17].

1.2.2 Conceptos básicos

Radiación solar. - Conjunto de radiaciones electromagnéticas emitidas por el Sol. La radiación solar comprende desde las radiaciones infrarrojas hasta las ultravioletas, quedando en el centro, la luz visible para el ojo humano[17].

Irradiancia. - Es la potencia incidente por unidad de superficie, y se mide en W/m2 (valor medido en una hora) [17].

Irradiación. - Es la energía incidente por unidad de superficie en un determinado período de tiempo y se mide en J/m2. No confundir irradiancia con irradiación, aunque coincidan numéricamente los valores, cuando la unidad de tiempo considerada sea una hora [17].

Radiación directa. - Es la radiación que corresponde al ángulo sólido limitado por el disco solar sin tener en cuenta la dispersión atmosférica [17].

Radiación difusa.- Corresponde a la radiación solar dispersada por los diferentes componentes de la atmósfera [17]

Radiación reflejada. - Corresponde a la radiación reflejada por el suelo (albedo)[17]

Radiación total. - Corresponde a la radiación total de todas ellas (radiación directa y difusa), si la medida de radiación se realiza sobre una superficie horizontal, no teniendo en cuenta la radiación reflejada. Fig. 1.

Radiación solar global= Radiación directa + radiación difusa.

La constante solar se define como la cantidad de energía que recibe la atmósfera sobre una superficie perpendicular a los rayos del Sol, y cuyo valor es: 1,92 cal · cm-2 · min-1 = 1.353 W · m-2[17]

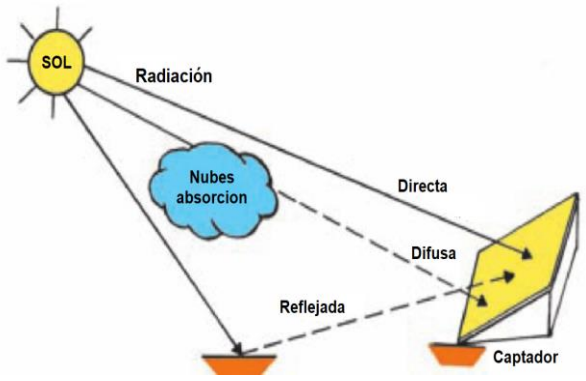


Fig. 1. Representación de las formas de radiación

El sol

Es una estrella, emite luz y calor, elementos imprescindibles para la vida sobre la tierra, es la principal fuente de energía y almacena el 99% del total del sistema solar, la superficie total es de 510.065.284,702 km^2, El 71% del a superficie terrestre está recubierta de agua,

de la que el 97% es agua salada y el 3%, agua dulce. El agua es básica para la vida sobre la Tierra.[17]

Azimut, altura

El giro de la tierra alrededor del sol para un observador situado en la tierra le produce la sensación de que es el centro solar el que gira alrededor de él, recorriendo una trayectoria que denominaremos aparente, la cual queda definida por el ángulo (α) que forman los rayos solares con el plano que pasa por el observador y es tangente a la estera terrestre (horizonte astronómico) y azimut (Az) [18]. Fig. 2. [17].

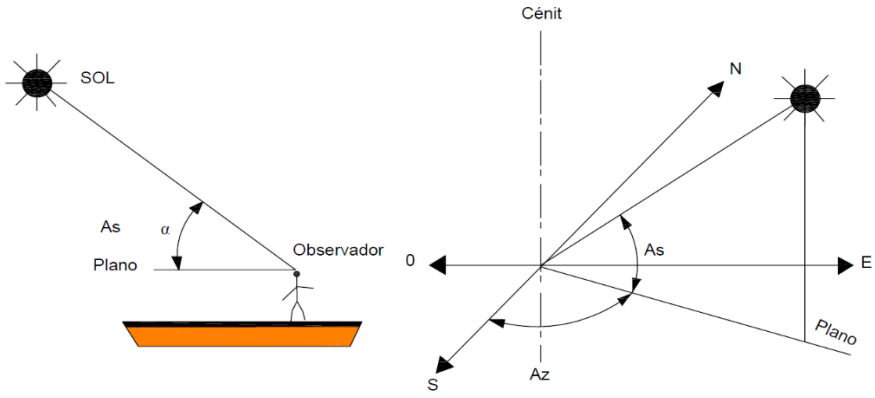


Fig. 2. Altura solar medida en grados y azimut

1.2.2 Machine Learning

Es utilizar algoritmos para que puedan decir algo interesante en base a un conjunto de datos sin tener que escribir ningún código específico para el problema[19].

1.2.3 Criterios para la aplicación de machine Learning

Se muestra un diagrama de fases para el proceso de aplicación de Machine Learning basados en CRISP-DM (Cross Industry Standard Process for Data Mining).

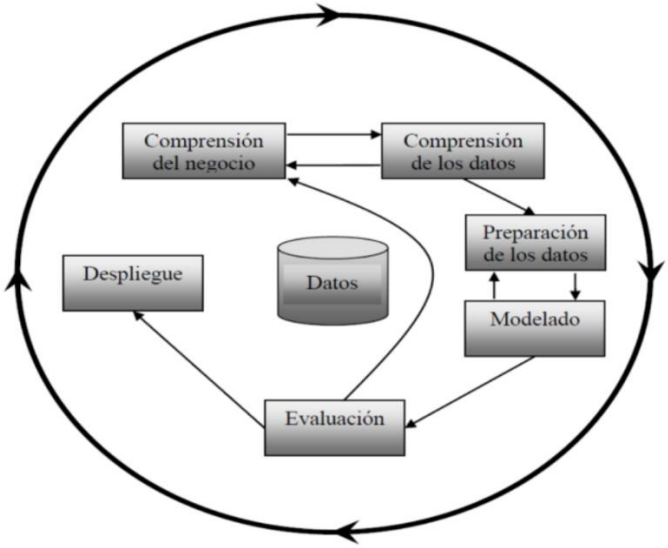


Fig. 3. Fase de metodología CRISP-DM

Fase I – Compresión del negocio. – de forma general conocer el negocio y las necesidades del mismo, que ayuda a delimitar un problema a ser resuelto con el procesamiento de datos[20]

Fase II – Exploración. – Exploración de la información para simplificar el problema y así optimizar la eficiencia del modelo. Se puede aplicar la visualización o técnicas estadísticas para conocer la relación entre variables que serán las entradas. [20]

Fase III – Modificación. – Se modifica y manipulan los datos en función de la exploración realizada, de forma que sean correctamente definidos y con el formato adecuado para introducirlos en el modelo[20]

Fase IV – Modelado. – Se identifican las relaciones entre las diferentes variables a fin de obtener conclusiones validas que aporten a los datos estudiados. Se puede aplicar modelos estadísticos tradicionales, arboles de decisiones, redes neuronales, etc[20]

Fase V – Valoración. – Determinación de la bondad del proyecto, contrastando los datos obtenidos con otros métodos estadísticos o con otras poblaciones muestrales[20]

Fase VI– Implementación. Ultima fase que aplica los resultados obtenidos, puede ser sencillo como redactar un informe o complejo como el desarrollo de un nuevo modelo de gestión[20]

1.2.4 Principios del diseño de algoritmos

Los algoritmos, en su forma más simple, son solo una secuencia de acciones, una lista de instrucciones, Los algoritmos son la base de toda la informática. Si Pensamos en una computadora como una pieza de hardware, un disco duro, chips de memoria, procesadores, etc. Sería imposible la tecnología moderna, sin los algoritmos.[22]

Razones generales para estudiar algoritmos:

1. Son esenciales para la informática y los sistemas inteligentes.
2. Son importantes en muchos otros dominios (biología computacional, economía, ecología, comunicaciones, ecología, física, etc.).
3. Desempeñan un papel en la innovación tecnológica.
4. Mejoran la resolución de problemas y el pensamiento analítico.

1.2.5 Paradigmas de diseño de algoritmos

Tres enfoques amplios para el diseño de algoritmos. [22]

-Divide y conquistaras. - Dividir el problema subcategorías para resolver.

-Algoritmos codiciosos. – El enfoque codicioso siempre elige primero el destino más cercano.

-Programación dinámica. – Los resultados intermedios se almacenan en caché y se pueden utilizar en operaciones posteriores.

1.2.6 Recurrencia y retroceso

La recursividad es particularmente útil para dividir y vencer problemas, pero es difícil entender exactamente lo que está sucediendo, porque recursiva está girando de otras

llamadas recursivas, por facilidad lo calcula mediante factoriales que define dos casos: el caso base cuando *n* es cero, y el caso recursivo cuando *n* es mayor que cero. [22]

TABLA 1.
COMPARACIÓN DE RECURSIVIDAD E ITERACION

Recursividad	Iteración
-Termina cuando se alcanza un caso base	-Termina cuando se cumple una condición definida
-Cada recursivo no se almacena en la memoria	-Cada llamada de iteración requiere espacio en la memoria
-Una recursividad infinita da como resultado un error de desbordamiento de tachuela	-Se ejecutará una iteración infinita mientras el hardware esté encendido
-Algunos problemas, naturalmente, se adaptan mejor a las soluciones recursivas.	-Las soluciones iterativas pueden no siempre ser obvias

1.2.7 Retroceso

El retroceso es un método de divide y vencerás para una búsqueda exhaustiva, el retroceso poda las ramas que no pueden dar resultado, es particularmente útil para tipos de problemas como atravesando estructuras de árbol, donde se presenta una serie de opciones en cada nodo, del cual debemos elegir uno. [22]

1.2.8 Clasificación del algoritmo por implementación

Recursividad. - Los algoritmos recursivos son los que se llaman a sí mismos hasta que se cumple una determinada condición.

Lógico. - Una implementación de un algoritmo lo expresa como una deducción lógica controlada. Esta El componente lógico está compuesto por los axiomas que se utilizarán en el cálculo.

Serie o paralelo. - Los algoritmos seriales, también conocidos como algoritmos secuenciales, son algoritmos que se ejecutan secuencialmente. La ejecución comienza de principio a fin sin ninguna otra ejecución. Los algoritmos paralelos / distribuidos dividen un problema en subproblemas entre sus procesadores para recopilar los resultados. [22]

1.2.9 Técnicas de predicción

Predecir no es tarea fácil en ningún campo científico, los físicos o los matemáticos nos hablan de que el crecimiento de errores, o caos, impide una predicción con certeza de un sistema dinámico. Los meteorólogos o los economistas reconocen frecuentemente los riesgos de sus predicciones por el nivel de exactitud [21].

Es necesario definir los conceptos fundamentales para el estudio de técnicas de predicción.

- Predicción
- Sistemas

Predicción

Es anunciar por revelación, ciencia o conjetura, algo que ha de suceder. Aplicados en la ciencia y en particular en la ingeniería para buscar la tendencia [22].

Sistemas

Es un conjunto de cosas que ordenadamente relacionadas entre si contribuyen a determinado objeto. O también un sistema es un objeto formado de relación que articula en la unidad que es el sistema [22].

1.2.10 Características generales de los modelos neuronales artificiales

Los modelos neuronales asumen muchas simplificaciones del modelo biológico para poder plantear su desarrollo matemático, así en esta línea, el primer modelo artificial se conseguía un output que se transmitía a lo largo de la estructura vinculada a la red neuronal, pero con la limitación que sólo permitían computar funciones booleanas. La fig. 3 compara una neurona biológica con el modelo matemático homólogo [23].

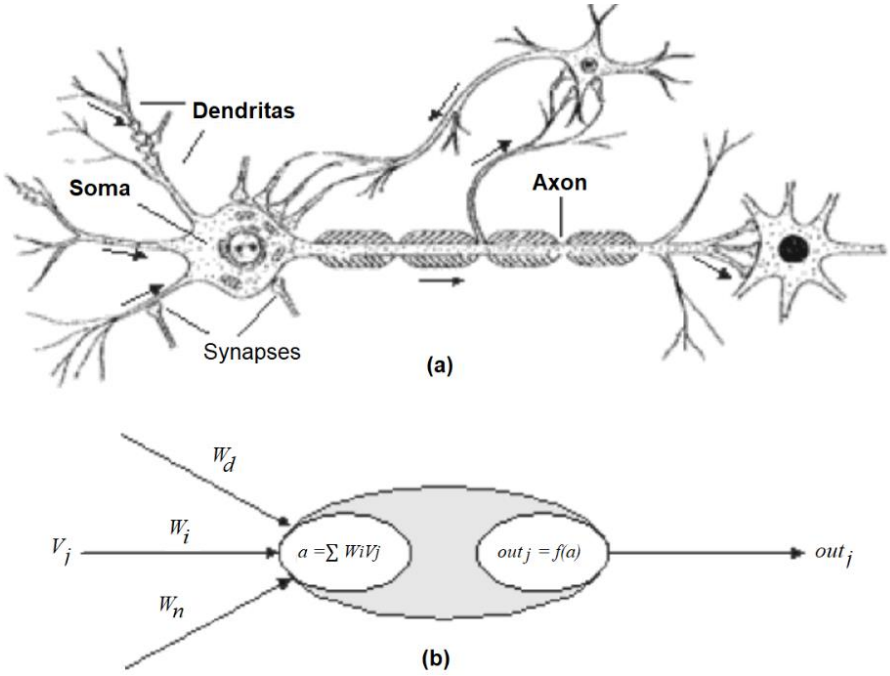


Fig. 4. Paralelismo entre el Modelo Biológico (a) y el Modelo McCulloch-Pitts (b) (Terminología original en inglés)

Red neuronal artificial

Son conocidas como RNAs, formadas por una serie de procesadores elementales denominados neuronas artificiales que constituyen dispositivos simples de cálculo, que a partir de entradas provenientes del mundo exterior, bien a partir de estímulos recibidos de otras neuronas, proporciona una respuesta única (salida). Se caracterizan tres tipos de neuronas artificiales: unidades de entrada, de salida y unidades ocultas, se observa en la Fig. 4 [24].

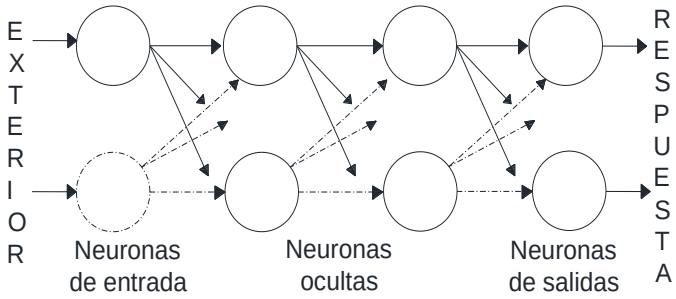


Fig. 5. Unidades de la neurona artificial

1.2.11 Modelos de redes neuronales

La gran variedad de modelos de redes neuronales existentes en la actualidad obliga en cierta medida a la realización de clasificaciones o taxonomías. De esta forma se pueden clasificar desde triple óptica: en función de la forma de aprendizaje ("learning paradigm"), en función de la arquitectura ("network architecture") y la tercera en las aplicaciones, se describe en la fig. 5 [23].

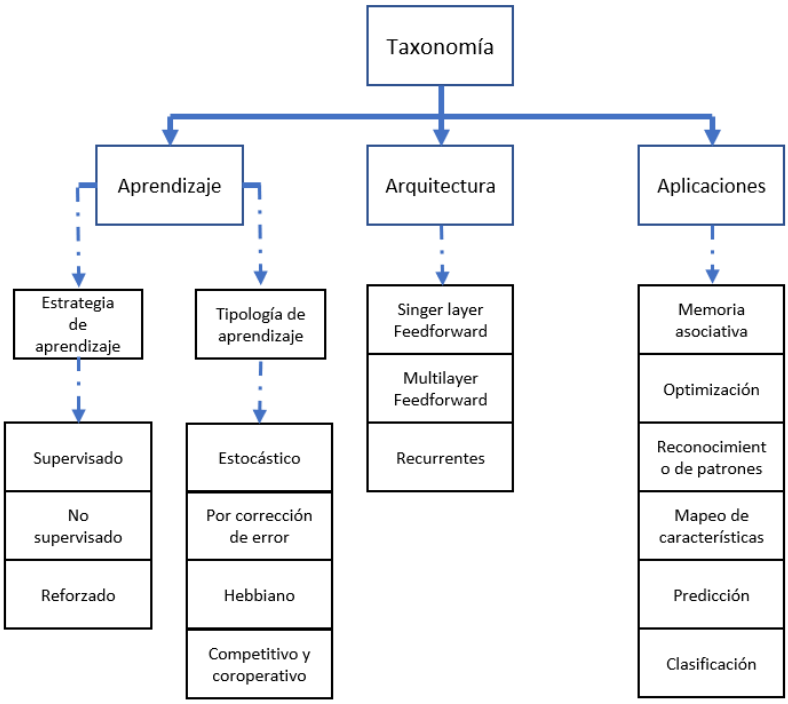


Fig. 6. Clasificación de redes neuronales

Fase de aprendizaje o entrenamiento.

Se parte de un modelo determinado de neurona y de una determinada arquitectura de red, estableciéndose pesos iniciales aleatorios o nulos. A partir de este modelo es necesario entrenar la red para solucionar el problema objeto de estudio. El ajuste puede ser mediante el modelado de las sinapsis (pesos) de la red, a través de una regla de aprendizaje y mediante la creación y/o destrucción de neuronas en la red [24].

Aprendizaje supervisado

Se caracteriza por la presencia de un agente externo (supervisor o maestro) que controla el proceso de entrenamiento, estableciendo la respuesta que debería generar la red (output del sistema) a partir de una entrada determinada. El tipo de algoritmo de aproximación permite distinguirse tres tipos de aprendizajes supervisado fig. 6: por corrección de error, por refuerzo o de tipo estocástico.

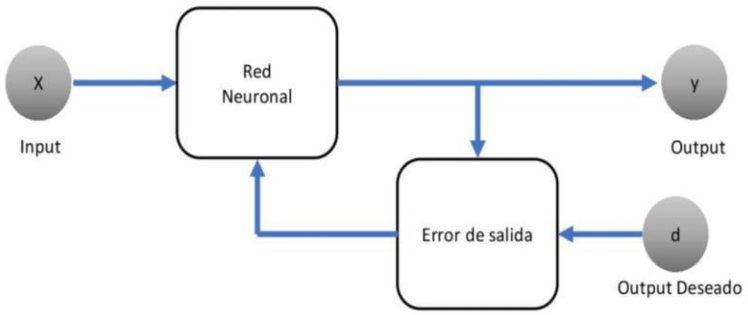


Fig. 7. Estructura de modelo supervisado

Aprendizaje por corrección de error

Constituye el tipo de aprendizaje supervisado más utilizado en la práctica. Su funcionamiento se basa en el ajuste de los pesos de las conexiones de la red a partir de la diferencia entre valores deseados y los obtenidos por el sistema, esto es, en función del error cometido en la salida. Una de las reglas más sencillas de aprendizaje por corrección de error es la siguiente:

$$\Delta W_{ij} = \propto * x_i(t_j - y_j)$$

Donde ΔW_{ij} la variación en el peso de la conexión entre las neuronas i y j, x_i la i-ésima entrada a la j-ésima neurona, t_j el valor de salida deseado para la neurona j, y_j el valor de salida obtenido en la j-ésima neurona y $\propto$ el factor o tasa de aprendizaje que regula su velocidad [24].

Aprendizaje por refuerzo

La tarea del supervisor se limita a indicar, mediante una señal de refuerzo (éxito = +1 o fracaso = -1), si la salida obtenida por la red se ajusta o no a la deseada y, en función de ello, se procede al ajuste de los pesos utilizando un mecanismo basado en probabilidades, también es denominado aprendizaje de "premio-castigo" fig. 7.[24]

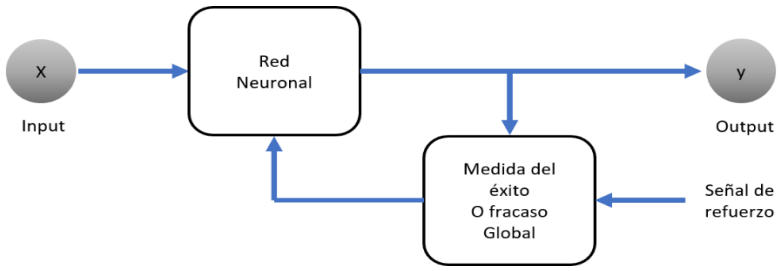


Fig. 8. Estructura de aprendizaje por refuerzo

Aprendizaje estocástico

Se establece una analogía de términos termodinámicos, asociado la red con un sólido físico que presenta cierto estado energético (grado de estabilidad de la red), por lo que aprendizaje consiste en minimizar la energía del sistema a través del ajuste de los pesos. Para ello se realizan cambios estocásticos de los valores de los pesos, determinándose la energía de la red tras estas modificaciones; si la energía no es menor la aceptación del cambio depende de una distribución de probabilidad determinada y preestablecida.[24]

Aprendizaje no supervisado

Redes con aprendizaje no supervisado (o "autosupervisado") no requieren información externa para ajustar los pesos de las conexiones neuronales. Se presenta a la red un conjunto de patrones sin adjuntar la respuesta deseada por lo que la red, por medio del algoritmo de aprendizaje fig. 8, estima la función de densidad probabilística p(x) que describe la distribución de patrones x, $x \in \Re^p$ [24].

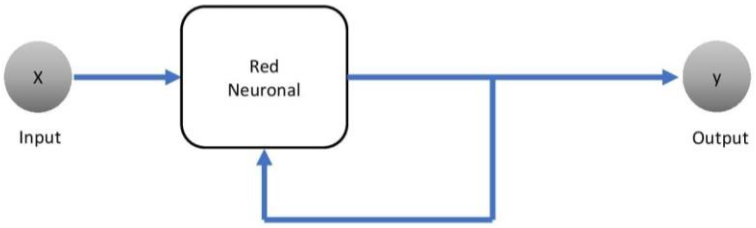


Fig. 9. Estructura aprendizaje no supervisado

Asociada a las anteriores tipologías de aprendizaje están las diferentes estrategias para poder conseguirlas, así la fig. 9 las recoge de forma esquemática. En primer lugar, para el caso supervisado diferenciamos entre un aprendizaje por corrección de error y uno de carácter estocástico. En segundo lugar, para el caso supervisado, tenemos el aprendizaje hebbiano y competitivo. En último lugar están las estrategias reforzadas, donde encontramos un aprendizaje por refuerzo.

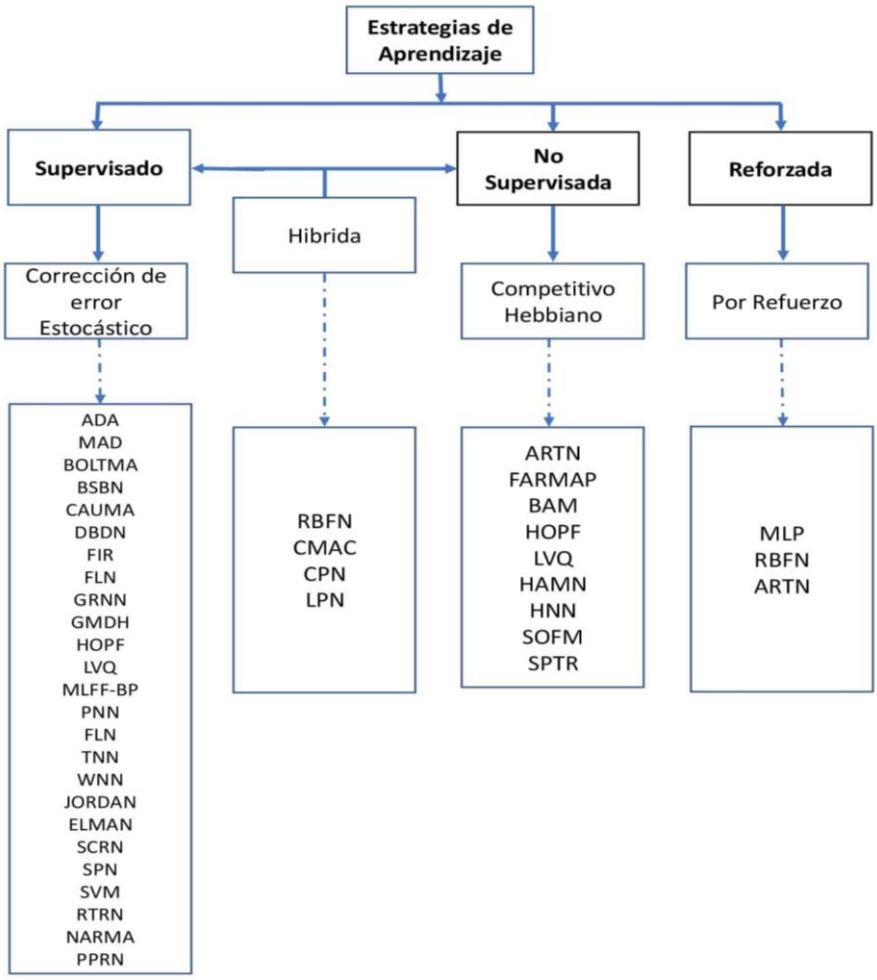


Fig. 10. Tipos de estrategias de aprendizaje

1.2.12 Descripción de los Modelos Neuronales

Modelos feed-forward y aprendizaje supervisado

La red cómoda como perceptrón simple es una red neuronal tipo feed-forward supervisada, sin capa oculta, constituida por un vector de "p" inputs,

$$X = (x_1, x_2, \dots, x_p)^!$$

Un vector "n" outputs deseados,

$$Y = (y, y_2, \dots, y_n)^!$$

La relación entre ambos vectores, (inputs ; outputs) se obtiene mediante la regla de aprendizaje, perceptrón learning rule. Se demuestra que converge de forma correcta en un número finito de iteraciones (perceptrón convergence theorem). Si adicionalmente las clases son linealmente separables, permite su utilización en problemas de clasificación con más de una categoría fig. 10 [23].

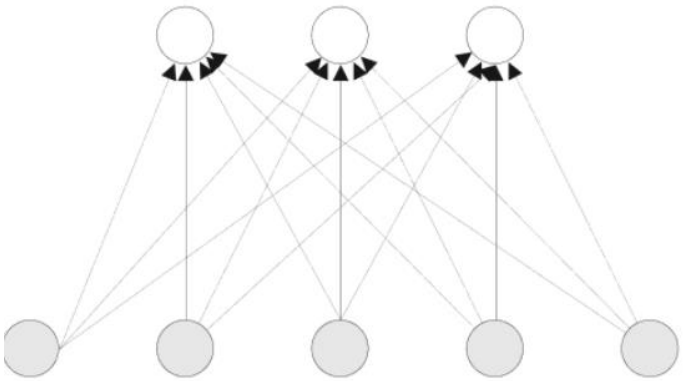


Fig. 11 .Ejemplo de percepción simple (5 entradas y 3 salidas)

Modelos Neuronales Recurrentes

Las redes neuronales recurrentes también definidas como, feedback network, permiten conexiones de las neuronas consigo mismas y con neuronas en capas inferiores, dotando a las mismas de una dinámica de comportamiento que no es posible ser replicada con las redes feed-forward. Fig 11.

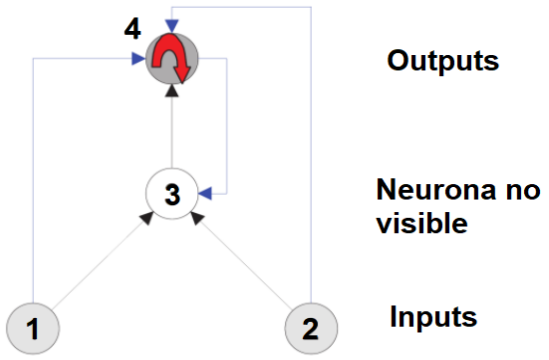


Fig. 12. Ejemplo de Recurrent Back Propagation (RBP) con 2 entradas

Los modelos neuronales recurrentes son muy adecuados para entender el comportamiento de las series espacio-temporales fig. 11 Poseen un número importante de aplicaciones, desde memorias asociativas, clasificación de patrones espacio-temporales, optimización, control, predicción y generación de secuencias [23].

1.2.13 Modelos híbridos

Redes neuronales Radial Basis Function Las redes defunción de base radial (RBF)

Son en este momento uno de los modelos más utilizados debido a su rápida formación, generalidad y simplicidad. Se trata de un modelo híbrido, ya que incorpora tanto el aprendizaje supervisado como el no supervisado. La parte no supervisada se sitúa desde el vector de inputs a la capa oculta, donde debe decidirse el número de nodos radiales que cubran buena parte del espacio definido por los inputs o entradas y la parte supervisa desde la capa oculta y la capa del output.

Primeramente, las relaciones existentes entre los modelos RBF y otras formas de aproximar funciones, como por ejemplo, la regresión de kernel fig. 12 [23].

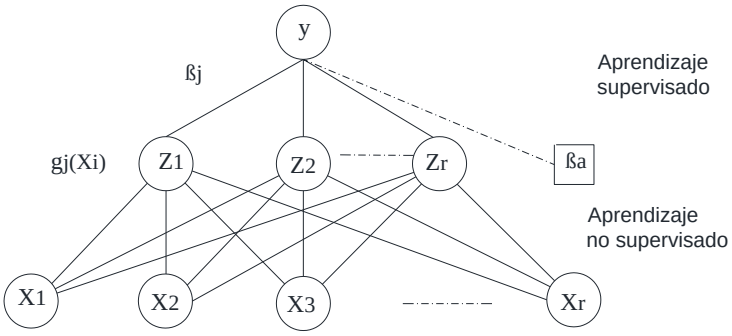


Fig. 13. Modelo neuronal RBF

Redes neuronales Probabilisticas (PNN)

Las PNN poseen por su parte tres limitaciones. Su lentitud en clasificar, un alto coste computacional y una elevada necesidad de memoria de sistema. Pero a su favor juega su alta resistencia y estabilidad frente la presencia de outliers. Los modelos probabilísticos, tal y como se ha comentado, utilizan la metodología de Bayes90 para su proceso de clasificación, pero de forma habitual se desconocen las fi1nciones de densidad necesarias para su aplicación. En este entorno los métodos de estimación existentes son los siguientes, fig. 13 [23].

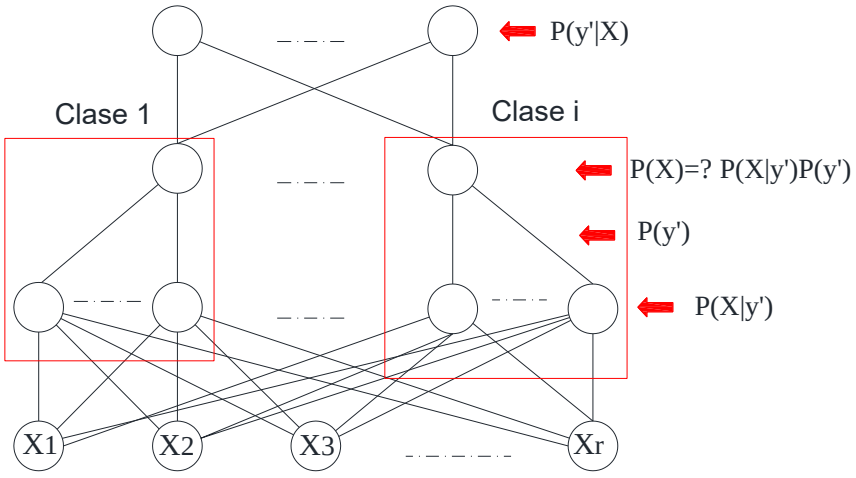


Fig. 14. Topología de un modelo neuronal PNN

1.2.14 Fundamentación metodológica

- Enfoque.

La investigación tiene enfoque cuantitativo con la descripción del comportamiento de la radiación solar en un sector de Lasso, con el registro de las variables climáticas de la radiación sola directa, difusa, reflejada, velocidad del viento, temperatura y humedad, con intervalos de medición de un minuto cada día por cada mes desde el año 2019.

- Tipo de investigación.

Investigación descriptiva. - con la cuantificación de los parámetros climáticos que son receptados en los sensores de la estación y con el uso de programas computacionales se pretende entender el comportamiento del parámetro climático con mayor énfasis en la radiación solar total.

Investigación correlacional. - Se pretende encontrar la relación que existe en las variables climáticas con el tiempo determinado a corto plazo con la ayuda de programas estadísticas que analicen los registros del año de medición.

Investigación experimental. – Se desarrollará algoritmos de análisis de grandes de cantidades de datos en busca de encontrar el patrón característico entre los parámetros climáticos y el las tiempos establecidos.

- Técnica de recolección de información.

Se aplicará la técnica de mediciones convencionales utilizando la estructura de la estación meteorológica para el registro diario en el sistema conectado mediante ethernet a la nube de almacenamiento y clasificando los parámetros por las unidades de medida.

- Procedimientos empleados para la obtención y análisis de la información.

Para la obtención de la información se iniciará con la instalación de la estación meteorológica en un lugar fijo libre de manipulación, con la conexión vía ethernet a la red de internet para el ingreso al sistema de monitoreo en línea, los datos son registrados en una nube de datos autorizada por la licencia y software, para la extracción de los registros en el panel de control del software seleccionar en intervalos de un minuto para la exportación del registro en un formato .CSV por cada mes, con la ayuda de software computacionales se realiza varios diseños de prueba para el procesamiento de los registros obtenidos, se revisara el modelo que con los resultados generados, caso contrario se debe reajusta el modelo diseñado, si ya no se tiene más reajustes

Del modelo se validará los datos generados con los datos medidos, pero no ingresados al algoritmo.

1.1 Conclusiones Capítulo I

La radiación solar es una fuente de energía natural que es variable durante el día y responsable de las actividades biológicas y químicas del planeta, por tal razón el interés de predecir el comportamiento de la radiación total incidente en un sector determinado con técnicas informáticas como es el Machine Learning o aprendizaje de máquina, para la aplicación se ha descritos 6 pasos basados en CRISP-DM (Cross Industry Standard Process for Data Mining) para el procesamiento de datos y las técnicas comunes aplicadas al aprendizaje supervisado, como la regresión lineal simple, regresión múltiple, arboles de decisiones, máquina de vectores que son las tradicionales, y de aprendizaje no supervisado a las agrupaciones conocidas como clustering, para trabajos que requieran más análisis se aplica el aprendizaje profundo con neuronas de diferentes configuraciones y procesamiento de datos

CAPÍTULO II.

PROPUESTA

2.1 Título del proyecto.

Modelamiento del algoritmo de series temporales para la predicción de la radiación incidente en el sector de Lasso.

2.2 Objetivo del proyecto.

Modelar el algoritmo de serie temporal con la aplicación de Machine Learning en Python para la predicción de la curva daría por horas de radiación del sector de Lasso

2.3 Descripción de la propuesta.

En la aplicación de Machine Learning se encuentran varias técnicas, dependiendo del comportamiento de los datos a analizar se elige un modelo, en el caso de la radiación solar se trata de valores que están variando en el tiempo desde las 6:00 horas hasta las 18:00 horas comúnmente, en la noche no se presentan valores por no existir la presencia del sol, se pretende realizar el algoritmo con una configuración de predicción por cada hora y generar la curva diaria de radiación de un tiempo establecido dependiendo de los datos medidos ingresados, se describe la estructura que se configurará para el funcionamiento del algoritmo.

2.4 Metodología para el cumplimiento de los objetivos planteados

El machine Learning ha evolucionado a pasos agigantados creando casos de estudios con el ayuda personal colaborador con variedades de áreas de la inteligencia artificial con campos como la teoría del control, la teoría de juegos, la economía y la neurociencia, en las últimas dos décadas ha tomado fuerza el aprendizaje automático en aplicaciones teóricas como prácticas [25].

La idea de que hay que ser una persona de matemáticas para aprender a programar está bastante extendida, es evidente incluso en el sistema educativo, en varias universidades, los cursos avanzados de matemáticas aparecen como prerrequisitos para las clases de programación, pero en realidad no es cierto. Un estudio de 2020 publicado en Nature descubrió que ser un genio de las matemáticas no ayudaba realmente a las personas a aprender Python en línea, las habilidades de resolución de problemas, las habilidades lingüísticas y la capacidad de memoria de trabajo eran mejores predictores del éxito que la aptitud matemática, esto no quiere decir que las habilidades matemáticas no sean importantes, por supuesto. Pero, si puede aprender a codificar, o a aprender la Ciencia de los Datos, si no es una persona matemática, recordando que el mayor predictor del éxito en el aprendizaje de la programación no es ningún talento o habilidad innata si no es el hábito [26].

Con lo expuesto se realizará la aplicación de herramientas computacionales para el procesamiento de datos entendiendo el comportamiento de los resultados sin ser necesario la explicación detallada de las expresiones matemáticas desarrolladas por el lenguaje de programación, enfocándose en los resultados comparados con valores reales.

2.4.1 Flujograma y pseudocódigo del procedimiento aplicado para el análisis de datos

El procedimiento que se aplicara para la predicción de la radiación solar se describe la figura, considerando que se iniciara desde los registros exportados de la estación, revisado la estructura del dataset se iniciara cada uno de los pasos del flujo de proceso, cuando cumpla con el requerimiento se continúa al siguiente paso, caso contrario de reconfigura la configuración inicial hasta obtener los indicadores que se determinen válidos.

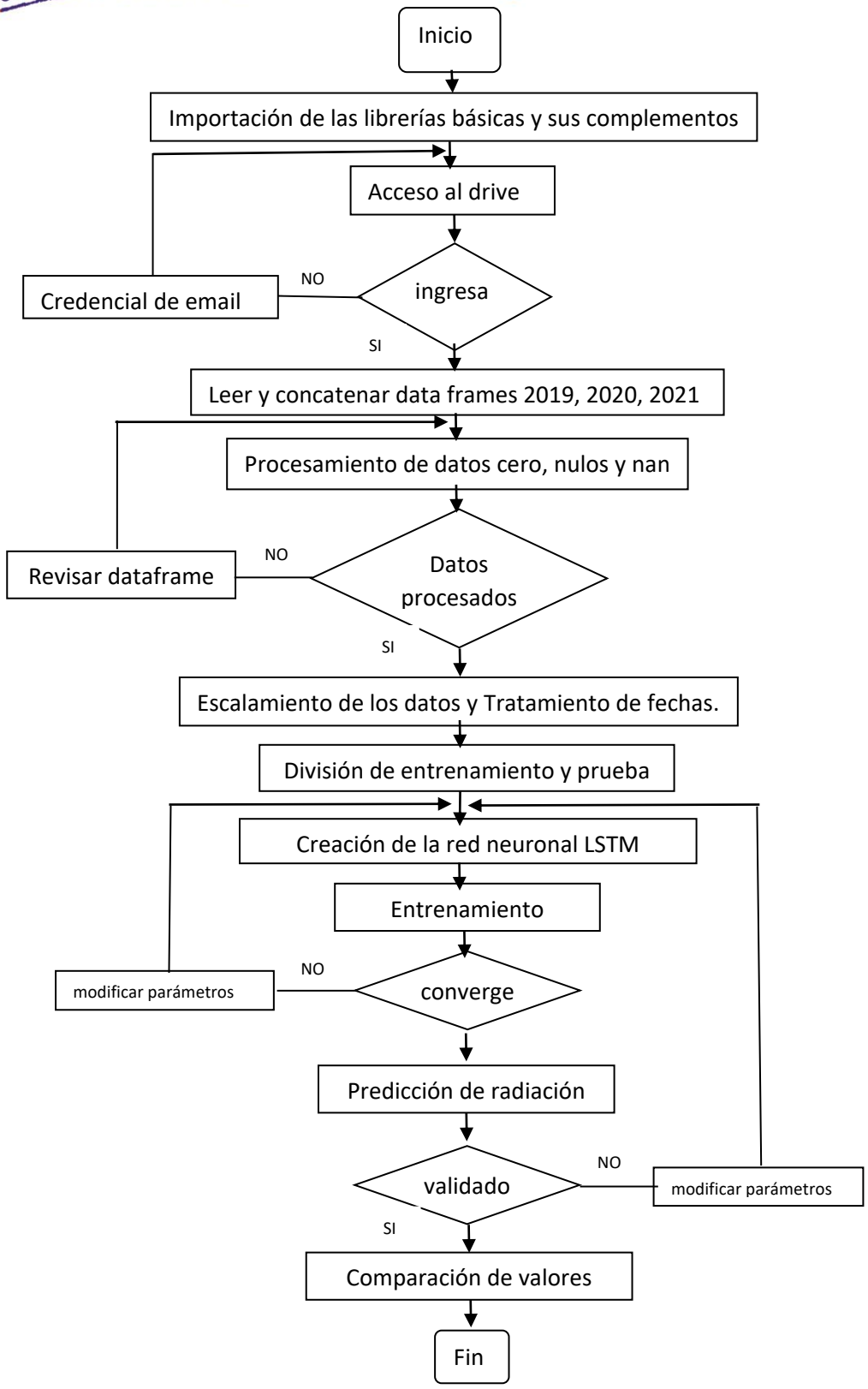


Fig. 15. Flujograma del algoritmo

INICIO

IMPORTACIÓN DE LAS LIBRERÍAS BÁSICAS PARA: procesamiento, lectura, gráficas dataframe
CONFIGURACIÓN DEL ACCESO AL REPOSITORIO DEL DRIVE DE GOOGLE DRIVE
LECTURA DE LOS DATOS DE: AÑO_2019,2020, 2021_depurado
DEFINIR VARIABLES: df2019, df2020, df2021
PASAR A HORAS LAS MEDICIONES DE VARIABLES: df2019, df2020, df2021
SELECCIONAR LA COMPONENTE "Solar Radiation Avg"
PRESENTAR UN GRÁFICO DE BARRAS DE LA COMPONENTE: "Solar Radiation Avg"
DEFINICIÓN VARIABLE: enero2019, enero 2020 y enero 2021 PARA LOS PRIMEROS 10 DÍAS
PRESENTAR UN GRÁFICO DE BARRAS DE LA COMPONENTE: "Solar Radiation Avg" DE LOS PRIMERO 10 DÍAS DEL AÑO
REALIZAR EL ANÁLISIS CORRELACIONAL DE LAS COMPONENTES
SELECCIONAR LAS COMPONENTES: "Solar Radiation Avg", "Baro Avg", "Hum Avg", "Windspeed", "Temp Avg"
CONCATENAR EN UN "df" LOS DATAFRAME: df2019, df2020, df2021
ELIMINAR LOS VALORES MENORES A CERO, NULOS Y VACÍOS DE LA COMPONENTE 'Solar Radiation Avg'
FIJAR LA COLUMNA TIME COMO ÍNDICE EN EL DATAFRAME
SELECCIONAR LAS COMPONENTES: "Solar Radiation Avg", "Baro Avg", "Hum Avg", "Windspeed", "Temp Avg" del "df"
CREAR LAS FECHAS DE 6:00 A 18:00 HORAS CON TIMESTAMP
ORDENAR LOS DATOS EN UNAMATRIZ CON LAS FECHAS CREADAS
ESCALAR LOS VALORES DEL DATAFRAME
SEPARACIÓN DE DATOS DE ENTRENAMIENTO Y PRUEBA CON 180 HORAS A REVISAR DEL PASADO PARA PREDECIR 91 DATOS HORAS
IMPORTACIÓN DE LAS LIBRERÍAS LSTM PARA PROCESAMIENTO, LECTURA, Y ENTRENAMIENTO CON DATOS DEL DATAFRAME
DEFINICIÓN DE LA LSTM CON 64 NEURONAS DE ENTRADA, 32 NEURONAS EN CAPA OCULTA, UNA SALIDA, DROPUOT DE 0.25, ACTIVACIÓN TAG HIPERBÓLICA Y OPTIMIZADOR ADANS CON PASOS DE 0.000045
CONFIGURACIÓN DE UN Callbacks CON UNA PACIENCIA DE 15 PARA EVITAR MEMORIZACION
ENTRENAMIENTO DEL ALGORITMO CON 500 EPOCAS 20% DE VALIDACIÓN, CANTIDAD DE LA MUESTRA DE 150.
GRÁFICAR LAS PÉRDIDAS DE ENTRENAMIENTO Y PÉRDIDAS DE PRUEBA.
SELECCIÓN DEL RANGO DE FECHAS QUE NO HA VISTO EL ALGORITMO ['2021-08-27':'2021-09-23'] PARA REALIZAR LA PREDICCIÓN
TRATAMIENTO DE FECHAS CON Timestamp PARA MATENER DE 6:00 A 18:00 HORAS
PREDICCIÓN DE LOS 91 VALORES POSTERIORES A LA FECHA INGRESADA
CARGAR LOS DATOS REALES MEDIDOS POR LA ESTACIÓN METEOROLÓGICA EN LA FECHA SELECCIONADA.
GRAFICAR LOS VALORES PREDECIDOS JUNTO A LOS VALORES MEDIDOS POR LA ESTACIÓN
COMPARAR LAS GRÁFICAS DE VALORES PREDECIDOS Y MEDIDOS POR LA ESTACIÓN.

FIN.

Fig. 16. Pseudocódigo del algoritmo

2.4.2 Registro y visualización de Dataset

Los datos climáticos son registrados por equipos especializados como es la estación meteorológica con dispositivos de medición para las variables tradicionales climáticas como son radiación solar, velocidad del viento, temperatura, humedad [27]. El modelo de la estación es "MK-III Weather Station" se describe las características de medición y el modelo utilizado hace que la instalación sea rápida y fácil [28]. El registro de la estación se monitorea mediante la interfaz gráfica accediendo a la página oficial de Rainwise https://rainwise.net/index.php con el nombre de usuario y credenciales, también se exporta los datos registrados en un archivo CSV para analizar fuera de la página oficial.

Fig. 17. Características de la estación meteorológica MK-III

2.4.3 Limpieza y Preprocesamiento de datos

La mayor parte del tiempo se ha dedicado a la limpieza de los datos, casi el 70% del tiempo, con los datos registrados se detecta e identifica los valores incompletos vacíos y Null para a posterior la sustitución, modificación o eliminación de datos.

Importación de las librerías

Numpy.- Librerías para todas cosas matemáticas
Pandas. - Herramienta disponible para importación y gestión de conjunto de datos
Matplotlib. - Librería para graficas

Cargar los datos en pandas

Con la ayuda de la librería pandas se carga el conjunto de datos de formato .CSV para convertirse en DataFrame de Pandas para la revisión.

Datos=pd.read_csv('data/')

Exploración del conjunto de datos

Con los datos cargados se procede a la revisión manual del conjunto de datos importados, se revisa el Dataframe con la cantidad y nombres de las columnas, revisando que no esté mal el formado, que estén completos y que sea información útil. Para el procesamiento en las librerías posteriores como en Scikit Learn con un valor Null se producirá un error en el procesamiento de entrenamiento.

Manejo de datos faltantes

Los datos faltantes, vacío, Null o ceros se procedió a la revisión visual con la gráfica de cada mes del comportamiento de la variable a predecir, identificando los datos faltantes se aplica la media aritmética para completarlos e incluso la copia del valor del último dato de la columna, considerando la secuencia de los valores, en la revisión del tipo de fecha se aplicó el paquete de Datetime para mantener la secuencia de datos.

Elegir la columna objetivo

Con los datos procesados se identifican la variable independiente y variable dependiente, la variable a predecir es la radiación solar en función de los parámetros climáticos con mayor correlación.

Guarda en archivo CSV

Revisado los datos se guarda los datos en un archivo .CSV para la posterior cargada y procesamiento, se guarda con la ayuda de la librería de pandas data.to_CSV (“datos _ depurados”)

2.4.4 Selección de la población y muestra.

Se denominan población al conjunto de elementos que cumplen ciertas propiedades y entre los que se desea estudiar el fenómeno en cuestión, con distinta naturaleza[20] con el registro de los datos de la estación metrológica con los sensores instalados se determinan las variables a ser analizadas: temperatura, humedad, velocidad del viento, dirección del viento, presión barométrica y radicación solar. La muestra es cualquier subconjunto de individuos perteneciente a una población determinada[20] considerando la metodología experimental la muestra se estableció los registros que se realizaron desde el 2019 y 2020 iniciando con la adquisición de la licencia de la estación y el tiempo de registro determinado es de 10 minutos las 24 horas, solo en ocasiones que presente fallas no se guardan los datos como el corte de energía que evita el acceso a internet.

2.4.5 Visualización de los registros

Los archivos CSV denominados valores separados por coma son archivos de texto cuyos valores están separados por un delimitador[29] presentado las ventajas de estar soportado por muchos programas d análisis como Python, permite representar los datos de las hojas de cálculo y puede ser visualizado por editores de texto [30] los registros realizados por la estación son guardados en formato CSV por ser un formato estándar en los equipos de medición, para exportar e importar en programas de analices, se utiliza Python por ser un software libre y presentar varias alternativas en lenguaje de programación como Jupyter netbook o Google Colab que para la lectura del archivo CSV se inicia conociendo la ruta exacta y con la importación de librerías específicas de Python como Pandas que facilita la lectura escribiendo el código pd.read_csv(ubicación archivo-nombre_archivo.CSV) [31]

con los datos en el programa y con la ayuda de la librería de matplolib se importa pylab [32] que permite la visualización de los datos cargados para analizar el comportamiento.

2.4.6 Correlación de datos

El coeficiente de correlación es identificado como r o R, es una medida de asociación entre variables aleatorias X y Y [33], el grado de correlación se mide por el coeficiente que va +1 para series positivas perfectas, a -1 para series con correlación negativa perfecta, es positiva cuando van en la misma dirección y negativa cuando es dirección opuesta [34], en la dependencia entre dos variables, se analiza la correlación de todo los datos registrados en el Dataframe en función de la radiación solar para identificar las variables que estén más relacionadas y aporten a la predicción, se encuentran registrados valores máximos, mínimos y promedios, para el análisis se usara los valores promedios. Con la ayuda de Pandas que es un paquete que facilita la importación y análisis de datos se encuentra la correlación de las variables entre columnas numéricas del marco de datos[35]

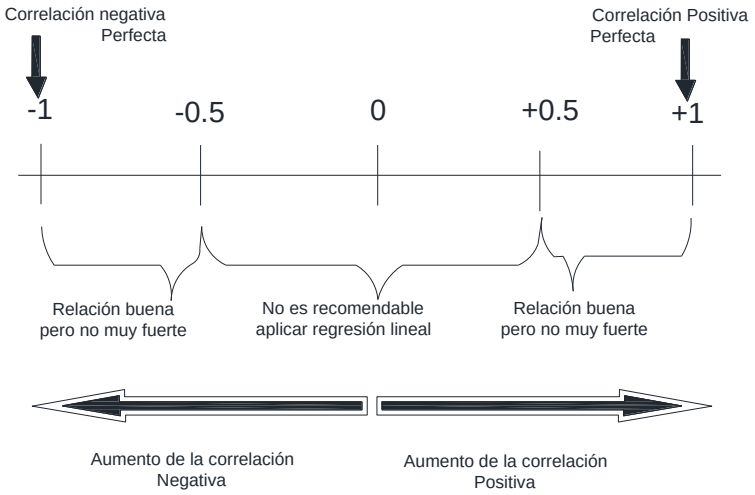


Fig. 18. Coeficiente de correlación r o R [33].

2.4.7 Amanecer y atardecer en Latacunga

El amanecer y el atardecer no difieren significativamente a lo largo del año en Ecuador, los días tiene una duración de aproximadamente 12 horas con una diferencia de 2 minutos en

los meses de diciembre al ser los días más largos [36] El amanecer es cuando aparece la luz del día, situación o condición al aparecer la luz del día [37], con la revisión de las horas que aparece el sol se identifica que amanece a partir de las 6:00 horas que varían en algunos minutos[38]. El anochecer es la falta de luz, situación o condición determinados al pesar la noche[39], se determina que la falta de luz inicia a partir de las 18:00 horas con variación de algunos minutos[38], se establece como estacionario para el comportamiento del día en todo el año considerando los rangos máximos y mínimos de tiempo de análisis de 6:00 a 18:00 horas, en la siguiente grafica se observa las 24 horas del día con pequeñas variaciones de minutos en las horas del amanecer y atardecer.

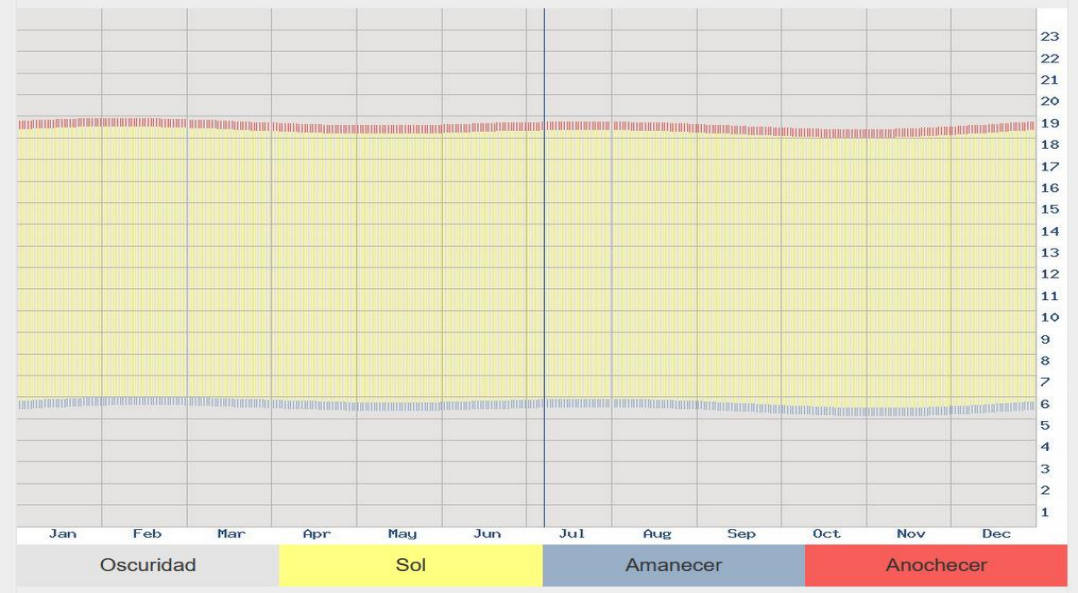


Fig. 19. Salida del sol, puesta de sol, amanecer, atardecer para el año - Latacunga[38]

2.4.8 Horas de radiación solar

La radiación solar es recibida un módulo fotovoltaico registrando la radiación directa, difusa y reflejada o de albedo, la suma de las tres radiaciones genera la radiación solar global, la representación de la radiación solar global se realiza por cada hora, en la noche no se registra radiación por tal razón solo se considera las horas del día que varía dependiendo del día si se encuentra despejado o nublado [40], con la librería de Pandas de Python aplicada al análisis de datos se aplica la función de dataframe.resample para

modificar los datos en una misma serie de tiempo sucesivas [41], los registros de la estación se guardan por default en minutos que aplicando la librería se analiza por cada hora para la interpretación de la curva diaria.

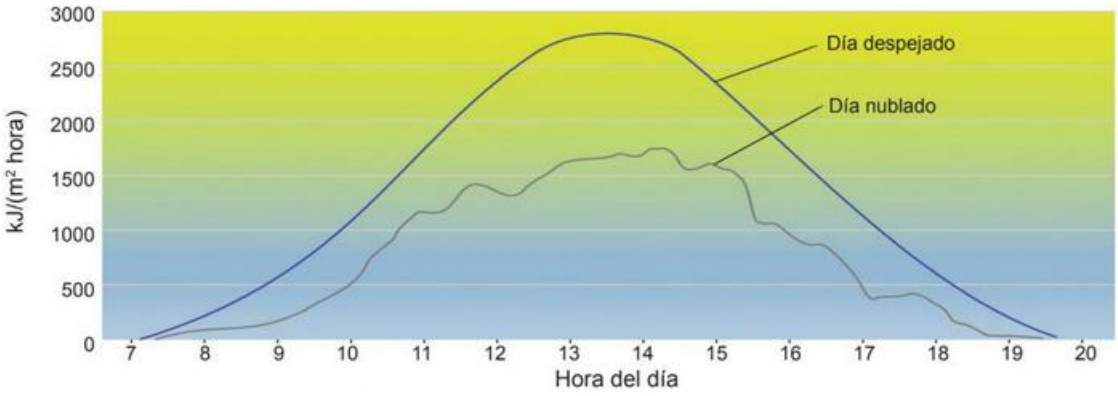


Fig. 20. Radiación solar global incidente en un día.

2.4.9 Normalización del DataFrame y datos estadísticos

Los datos registrados en la estación meteorológica tiene valores diferentes entre cero y mayores de 500 en todas las variables [38], los algoritmos de machine Learning tienen mejor rendimiento cuando las variables de entrada son numéricas y a un rango estándar por tal motivo es necesario la normalización, lo tradicional a convertir los valores es de -1 a 1 [42], se aplica la librería de *MinMaxScaler y* A los valores aplican la siguiente expresión para normalizarlos [43], una escala mínima-máxima se realiza normalmente mediante la siguiente ecuación:[44]

$$m = \frac{(X - Xmin)}{(Xmax - Xmin)}$$

2.4.10 Separación de DataFrame de entrenamiento, validación y prueba

Para conocer si el algoritmo está funcionando de forma correcta es necesario separa los registros en una parte de entrenamiento y otra parte de validación, no se presentan normativas sobre el porcentaje a dividir el Dataframe, solo por experiencias de programadores recomiendan 80% entrenamiento y 20% validación en algoritmos de aprendizaje automático aplicados en Python ML [45], también otra recomendación en

análisis de datos es entrenar y validar los modelos, utilizando el 70 % de todos los datos para entrenamiento y el 30% restante para evaluarlos con los criterios de calidad [46], los valores establecidos se caracterizarán para entrenamiento como: X_train, Y_train, y para la validación se expresa como X_test , Y_test, los porcentajes para la separación de los datos se realiza considerando a los dos años ingresados, separando el 80% de los datos para el entrenamiento – validación y para la prueba de test el 20 %, el comportamiento de los datos de entrenamiento, validación y prueba queda separada como la siguiente gráfica.

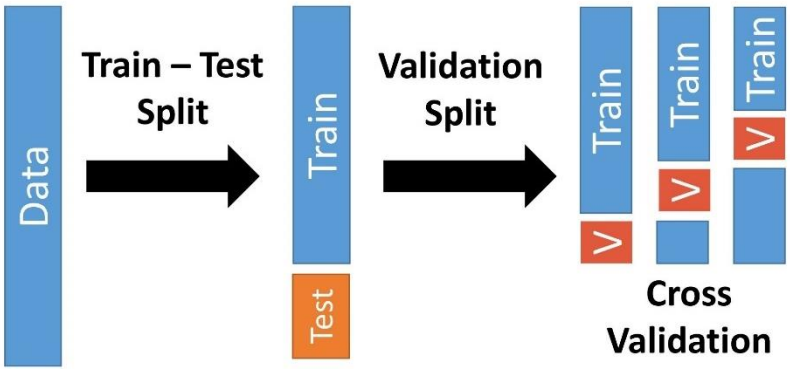


Fig. 21. Datos de entrenamiento, validación y prueba.

Los datos de entrenamiento escalados ingresados corresponden a la radiación solar junto con los parámetros climáticos con mayor correlación presentada y recomendada en otras bibliografías, para los datos de prueba está considerada solo los valores de la radiación solar, el rango de elección de los datos lo realiza el algoritmo separando automáticamente los datos para procesarlos.

2.4.11 Creación del modelo de la red Neuronal

Existen varios modelos de aprendizaje en machine Learning pero al tratarse de datos con secuencia de registro se opta por revisar las redes neuronales recurrentes (RNN, *Recurrent Neutral Network)* que tiene la capacidad de almacenar información en "memoria", por lo tanto para unidad cualquiera en una red recurrente se puede definir que su salida en el instante *t* es[47]:

$$y_i(t) = f_i(net_i(t-1))$$

Las redes neuronales básicas son útiles para el reconocimiento de patrones en reconocimiento de voz, análisis de escritura, etc, pero no dan buen resultado para el análisis con una gran dependencia temporal debido al problema de desviamiento del gradiente [47], por esa razón las RNN son consideradas con una memora a corto plazo con la secuencia de procesamiento relativamente corta para que las predicciones sean efectivas[48]

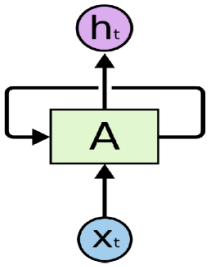


Fig. 22. Ejemplo básico de red neuronal recurrente.

Para resolver el problemas del enviciamiento de la memoria de las RNN se aplica las redes LSTM (*Long Short-Term Memory*) que es un tipo especial de red neuronal recurrente diseñada con celdas de memoria que mantiene el estado a largo plazo [47], la red LSTM tiene forma de cadena, pero el módulo repetidor en vez de tener una sola capa tiene cuatro permite a la información fluir por toda la red con solo algunas pequeñas interacciones, sin ser apenas alterada. La red también tiene la habilidad de borrar o añadir información al estado de la celda mediante estructuras llamadas puertas.[49]

2.4.12 Entradas y salidas de celda LSTM

Una celda de LSTM tiene (4 x n x m) + (4 x m x m) pesos y (4 x m) sesgos. Los pesos y los sesgos son constantes.

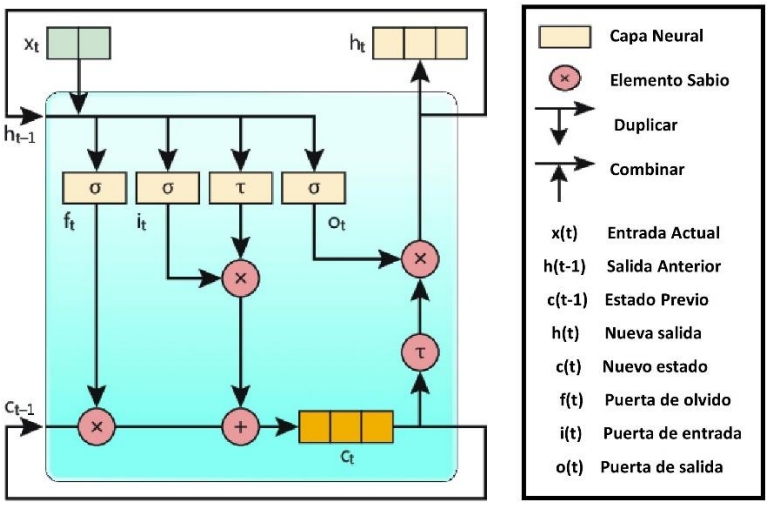


Fig. 23. Arquitectura de celda de LSTM[50]

Las ecuaciones matemáticas de la celda LSTM se presentan a continuación[50]:

$$(1)\ f_t = \sigma(W_f X_t + U_f h_{t-1} + b_f)$$

$$(2)\ i_t = \sigma(W_i X_t + U_i h_{t-1} + b_i)$$

$$(3)\ o_t = \sigma(W_o X_t + U_o h_{t-1} + b_o)$$

$$(4)\ c_t = f_t . c_{t-1} + i_t . \tau(W_c X_t + U_c h_{t-1} + b_c)$$

$$(5)\ h_t = o_t . \tau(c_t)$$

Donde:

n: $tamaño\ de\ entrada$

m: $tamaño\ del\ estado\ de\ la\ celda\ y\ salida$

x_t: $vector\ de\ entrada, tiempo\ t, tamaño\ nx1$

f_t: $vector\ de\ puerta\ de\ olvido, tamaño\ mx1$

i_t: $vector\ de\ puerta\ de\ entrada, tamaño\ mx1$

o_t: $vector\ de\ puerta\ de\ salida, tamaño\ mx1$

h_t: $vector\ de\ salida, tamaño\ mx1$

C_t: $vector\ de\ celda\ de\ estado, tamaño\ mx1$

$W_f,\ W_i,\ W_o,\ W_c$: $matrices\ de\ peso\ de\ la\ puerta\ de\ entrada, tamaño\ mxn$

$U_f,\ U_i,\ U_o,\ U_c$: $matrices\ de\ peso\ de\ la\ puerta\ de\ salida, tamaño\ mxm$

$b_f,\ b_i,\ b_o,\ b_c$: $vectores\ de\ sesgo, tamaño\ mx1$

σ: $función\ de\ activación\ logística\ sigmoide$

τ: $funcion\ de\ activación\ tangente\ hiperbólica$

Las ecuaciones (1), (2) y (3) definen tres puertas: una puerta de olvido, una de entrada y una de salida. Cada puerta es un vector de valores entre 0,0 y 1,0, que se usan para determinar cuánta información se olvida (o, equivalentemente, se recuerda) en cada ciclo de entrada-salida. La ecuación (4) calcula el nuevo estado de celda, mientras que la ecuación (5) calcula el nuevo resultado.

La función sigmoidal es precisamente que da a la compuerta el comportamiento de válvula, funcionando como numero binario de 0 y 1 que permite pasar o anular los valores en la entrada [51] es el comportamiento que permite aplicar la memoria a largo plazo.

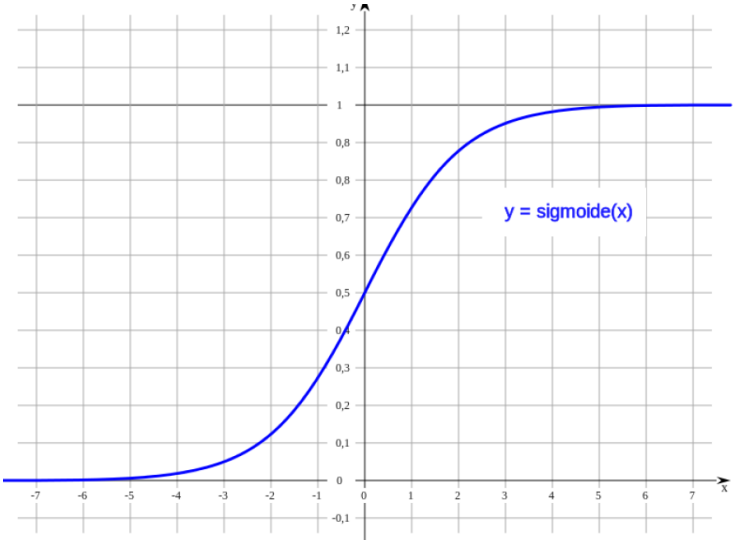


Fig. 24. Gráfica de una función sigmoide [47]

La salida de la red LSTM de la celda cj en el instante t se escribe como[47]:

$$y^{cj}(t) = y^{out_i}(t)h(Sc_i(t))$$

Donde:

$y^{out_i}(t)$ función de activación de la puerta de salida

En el estado interno se define como [47]:

$$Sc_j(0) = 0$$

$$Sc_j(t) = Sc_j(t-1) + y^{in_j}(t)g\left(net_{cj}(t)\right) \quad para\ t > 0$$

La LSTM determina si el estado anterior es relevante o no relevante para la activación de la puerta de salida permitiendo recordar periodos de tiempos arbitrarios.

La creación del modelo de la red neuronal se debe mantener el orden en la separación de los datos por tratarse de serie temporal influye el orden de selección de los datos, al tratarse de una predicción por horas la configuración se realizará con 1 entrada que representará cada hora y una salida por hora con las siguientes configuraciones [52]:

-Entradas 64 neuronas

-Una capa oculta 32 neuronas

-Salida 1 neurona

-Activación tangente hiperbólico valores -1 y 1.

-Optimizador Adam

-Error de predicción mse.

2.4.13 Librería keras

Keras es una librería aplicada en redes neuronales basada en lenguaje Python y puede ejcutarse en TensorFlow como theano, las principales características son fácil de usar por ser modular, soporta redes convucionales como recurrentes, esquema arbitrario, corre en CPU y GPU.

El modelo secuencial *fit(self, x, y, batch_size=32, nb_epoch=10, verbose=1, callbacks=[ ], validation_split=0.0,validation_data=None, shuffle=True, class_weight=None, sample_weight=None)* : Entrena el modelo para un numero fijado de ciclos *(nb_epoch)*, con actualizaciones de gradiente cada *batch_size.*

En el diseño de la red neuronal se consideran las siguientes funciones:

Dense: Para capas regulares totalmente conectadas.
Activation: Se aplica una función de activación a una salida. Las principales funciones que ofrece Keras son:
• **tanh:** función tangente hiperbólica
• **sigmoid:** función sigmoide

Se aplica la capa recurrente LSTM: Redes Long-Short Term Memory. Este tipo de red ha sido descrito en el apartado, con el optimizador Adam: algoritmo basado en el gradiente de primer orden de funciones estocásticas objetivas. Este método es sencillo de implementar, es eficiente computacionalmente hablando, necesita poca capacidad de memoria, y es muy bueno en problemas con una gran cantidad de datos y/o parámetros Es utilizado por defecto en los modelos de Keras.[49]

2.4.14 Error de predicción

Existen distintos criterios para analizar la capacidad para responder correctamente ante la presentación de patrones nuevos en los datos analizados, se define la función de perdida siendo los más habituales el error absolutos o cuadrático y el error total [53], los estadísticos utilizan una estimación del error cuadrático medio respecto al valor verdadero[54], el error cuadrático medio MSE es una segunda forma de medir el error global del pronóstico. El MSE es el promedio de los cuadrados en las diferencias entre los valores pronosticados y observados, la fórmula es la siguiente[55]:

$$MSE = \frac{\sum(Errores\ de\ pronostico\ )}{n} =$$

2.4.15 Entrenamiento del algoritmo

En el entrenamiento es necesario estimar un punto óptimo para evitar el overfitting y underfitting, por lo que se deberá modificar varias variantes hasta la obtención de un resultado optimo, no se ha encontrado normativa para la fijación de valores, pero se debe realizar varias veces el mismo procedimiento variando cada característica como el número de épocas, las neuronas, las capas ocultas.

Entrenamiento=model.fit(x_train,y_train,epochs=EPOCHS,validation_data=(x_val,y_val), batch_size=saltos)

Se busca el resultado optimo guiado con la gráfica de resultado de los datos como la representación siguiente:

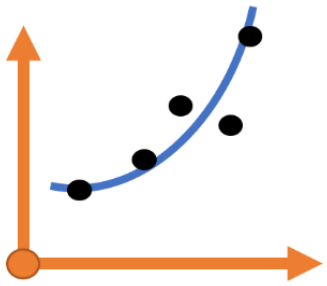


Fig. 25. Optimización del algoritmo

2.4.16 Preparación de los datos para la Predicción

El algoritmo se encuentra entrenado con los datos de 2 años en cada hora, revisado que no exista malos ajustes se ingresará los datos medidos de un mes de prueba para predecir la primera semana del mes siguiente, se procederá a cargar una nueva dataset con la ayuda pandas fijando el valor inicial y final considerando que es importante mantener la secuencia por tratarse de serie de tiempo, la estructura es:

Dataframe_Prueba = Dataframe['año-mes-día': 'año-mes-día']

Los datos cargados se encuentran en valores de diferente escala, para evitar errores de interpretación se normalizan con *scaler = MinMaxScaler()* entre valores de -1 y 1 según como está configurado para el procesamiento

2.4.17 Predicción

Los valores de prueba se encuentran en las mismas características para que se cargado al algoritmo, lo que se debe tener en cuenta es que se ingresara datos de un mes y se busca predecir los valores de 7 días consecutivos, como los valores son normalizados el resultado será valores entre -1 y 1, para interpretar los valores se invertirá a valores reales aplicando el código:

inverted = scaler.inverse_transform(adimen)

2.4.18 Comparación de la predicción con valores reales

El modelo entrenado y con el ingreso de datos que no han sido visto por el algoritmo se procede a predecir un rango de 7 días, aparte de las métricas generadas en el algoritmo del entrenamiento se busca revisar que la predicción se ajuste a los datos reales medidos por la estación, considerando que el rango de fechas sea el mismo.

Los resultados obtenidos de las predicciones se graficarán con un plot y juntamente con los valores reales médicos por la estación para comprobar el acierto de las predicciones y llenar la tabla comparativa de resultados estructurado de la siguiente forma

Hora	Predicción	Medición real	% Diferencia
1	X	X'	(X/X')*100%

Fig. 26. Comparación de valores

2.5 Conclusiones Capítulo II.

Python es un lenguaje de código abierto para los programadores, de fácil sintaxis y flexibilidad, con varias librerías de aprendizaje automático, las librerías incluyen varios modelos matemáticos que desarrollarlos es complicado pero existe la comunidad desarrolladora que generan librerías libres para la aplicación, las librería permiten la exploración de datos y manejo de datos faltantes para depúralos con facilidad, manejo de matrices con Dataframe, visualización de los datos en gráficas, escalar los valores para el entrenamiento del algoritmo, dependiendo de los datos a procesar se identifica si se aplicara clasificación o regresión para seleccionar la estructura de la red neuronal, al tratarse de la variable de radiación solar que depende con el tiempo se ha optado aplicar una LSTM por la ventaja que tiene una memoria interna y permite obtener mejores resultados en predicciones relacionados con el tiempo, los datos ingresados se encuentran en forma consecutiva medidos desde el 2019 hasta el 2021, los datos deben estar completos de todos los días para no alterar la secuencia del comportamiento de la radiación.

CAPÍTULO III.

MODELAMIENTO DEL ALGORITMO DE SERIES TEMPORALES PARA LA PREDICCIÓN DE LA RADIACIÓN INCIDENTE EN EL SECTOR DE LASSO.

3.1 Objetivo de la propuesta.

Modelar el algoritmo de serie temporal con la aplicación de Machine Learning en Python para la predicción de la curva daría por horas de radiación del sector de Lasso

3.2 Desarrollo de la propuesta

La investigación se desarrolló en el sector de Lasso de la ciudad de Latacunga, las actividades diarias en el campo dependen de la variación del clima para la ejecución de actividades, al tratarse de zonas de grandes extensiones de terreno la cobertura de las redes eléctricas no cubren a todos los usuarios por lo que es importante la estimación de parámetros climáticos como la radiación solar para la generación de pico centrales fotovoltaicas y calentadores solares.

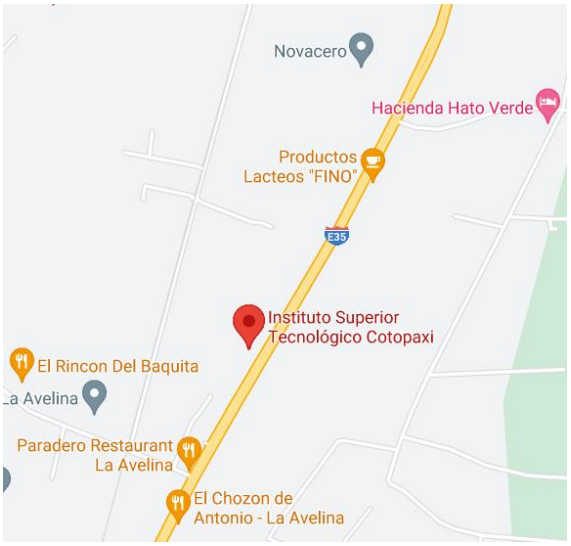


Fig. 27. Ubicación Instituto Superior Tecnológico Cotopaxi

Se realiza la adquisición de una estación meteorológica semiprofesional “Rainwise” para el monitoreo y registro de los parámetros climáticos, la estación cuenta con una comunicación de inalámbrica de 2.4 GHz desde la matriz hasta el modem receptor con alcance de 1 km de distancia, el almacenamiento de las mediciones se realiza mediante la nube que sincronizada en la estación, los datos son monitoreados en tiempo real al encontrase conectado al internet, también para evitar daños en los sensores por tormentas eléctricas la fuente de energía se alimenta por el panel solar y la batería de autonomía, todo el hardware está diseñado de acero inoxidable que garantiza la durabilidad como en los registros gravados.

Fig. 28. Estación Meteorológica

La instalación se realizó en el Instituto Superior Tecnológico Cotopaxi con coordenadas Latitud: -0.800396, Longitud -78.619555, msnm: 2929, por la infraestructura actual se ubicó en la terraza de la zona administrativa considerando la dirección de salida y puesta del sol, el envío de los datos por radio frecuencia al modem receptor que se instaló en área de TICs a 100 metros de distancia dentro del alcance, la sincronización de mediciones y el monitoreo se realiza por la interfaz de página web https://rainwise.net con el nombre de

usuario, contraseña se accede al panel de control, las opciones se encuentran vigentes al renovar la licencia de funcionamiento por cada año.

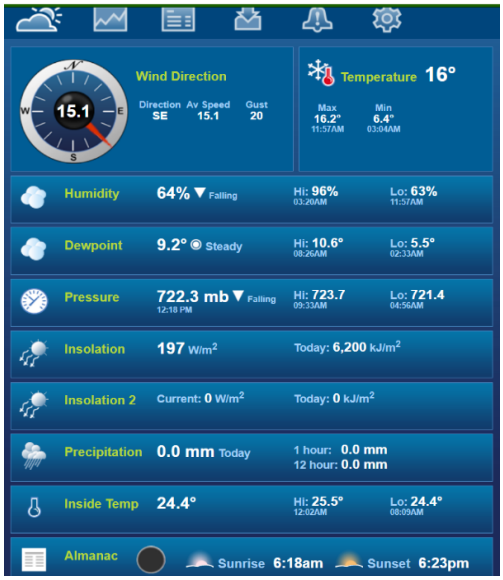


Fig. 29. Panel de control estación

Los registros por la cantidad de datos almacenados son exportados en formato valores separados por coma .CSV que los procesara Python, los registros son interpretados como Dataset en Jupyter Notebooks, se realizó los Dataset por año de medición 2019, 2020 y comparación del 2021, con el uso de la librería Pandas, Sickitlearn, Matplotlib pemite la tabulación de datos, gráficos, manipulación de series temporales, en la técnica de Machine Learning se aplican aprendizaje supervisado y no supervisado,

3.2.1 Selección e Inicio del programa

La lectura de los registros de los datos de la estación se almacenan todos los días generando una gran cantidad de información que con herramientas tradicionales como Excel no se pueden analizar porque saturan el procesamiento, alternativas para procesar grandes cantidades de información se encuentran licenciadas como Matlab y también software libre como Python , R studio, por tal motivo para el análisis se ha optado con la utilización del software libre de Python, al no tener costo de adquisición se ha descargado desde la página oficial la distribución de Anaconda que contiene una colección de paquetes

de código abierto para análisis de datos, la aplicación a utilizar es Jupyter Notebook disponibles para los sistemas operativos Windows, MacOS y Linux, para no tener incompatibilidades solo es necesario instalar la versión actualizada en Python, el código de programación se inicia en el servidor del equipo mediante el navegador de preferencia, una alternativa si el equipo a utilizar no tiene buenas características para el procesamiento de datos se puede utilizar Google Colab que es procesamiento en la nube basado en Jupyter Notebooks.

TABLA 2.
COMPARACIÓN DE LENGUAJE DE PROGRAMACIÓN

Lenguaje de programación	Observación
R	Velocidad de desarrollo rápida (60% menos código que Python)
Python	La velocidad general es la más rápida.
MATLAB	Licencia alrededor de US $ 1,000, paquete adicional requiere un pago adicional de US $ 50 +

El tamaño de los archivos de registro de la estación es de aproximadamente 7 Mb de cada mes registrados, se guarda en una sola carpeta el registro de los 24 meses que se sincroniza con la nube Google drive, la carpeta total supera los 300 Mb y para facilidad en el procesamiento de datos se decidió desarrollar en Colab accediendo al drive cargado de la cuenta personal, el código desarrollado está basado en Python 3 con una Ram disponible de 12 Gb y un disco virtual de 100Gb, de las características del servidor dependerá la velocidad procesamiento

3.2.2 Limpieza y Preprocesamiento de datos

Importación de librerías.

Las librerías a utilizar son: Pandas para modificar dataframe de las tablas de datos, Numpy para la facilidad de realizar ecuaciones matemáticas, Matplolib para visualización, Tensor Flow y keras para la estructuración de la rede neuronal, Datetime para leer los datos en

formato caso contrario las leería como dato numérico, Miximarscaler para normalizar las mediciones y aplicar en series de tiempo, son las convencionales para el desarrollo del algoritmo.

```
#Importación de las librerías básicas y sus complementos
import pandas as pd              # Lectura y procesamiento del Dataframe
import numpy as np               # Realizar ecuaciones matemáticas
import matplotlib.pyplot as plt  # Visualización de gráficas
import tensorflow as tf          # Aplicacion de tensores junto con keras
import datetime as dt            # Trabajar con fechas
from datetime import datetime    # Trabajar con fechas
import tensorflow.keras as keras # Aplicación de tensores junto con keras
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint, TensorBoard
%matplotlib inline
from pandas import DataFrame     # Procesamiento del Dataframe
from sklearn.preprocessing import MinMaxScaler # Escalar los valores
```

Fig. 30. Importación de librerías en Google Colab.

Carga y lectura de los datos

El desarrollo se realizó en Google Colab, las mediciones se almacenaron en un drive personal, cada mes de medición se exporta del programa de la estación meteorológica que pesa un promedio de 10mb en formato .CSV, desde enero 2019 hasta agosto 2021, con la importación de la información del drive personal se carga cada mes con nombre de “df5” iniciando desde el número cinco para el mes de enero 2019 y finalizando en “df36” correspondiente al agosto 2021, con todos los archivos ingresados se procede a concatenar los frames identificando por cada año independiente y se presentan los valores por cada hora, los datos iniciales están conformados por 41 diferentes variables climáticas medidas, siendo la primera columna de fechas de registro, para conocer el comportamiento de la radiación solar se gráfica cada mes de los años registrados verificando datos faltantes como se muestra en la figura.

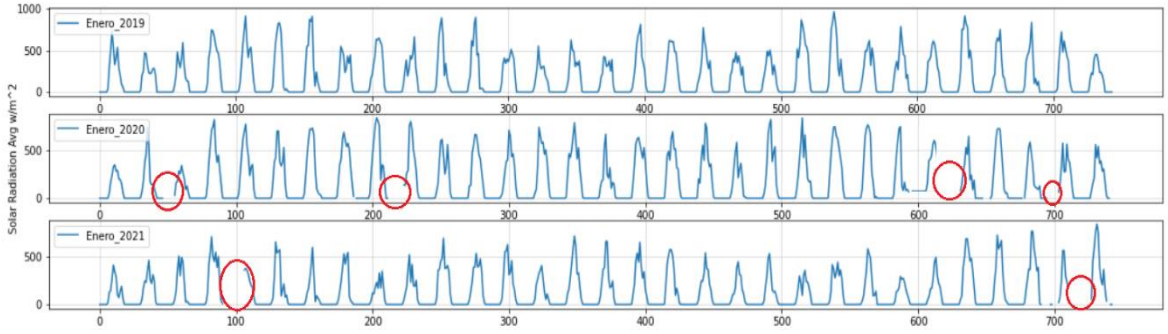


Fig. 31. Comparación de la radiación solar mes enero.

Los datos concatenados en un solo dataframe se imprime la información y se conoce los valores nulos que se encuentran registrados, el dataframe contiene 41 variables, se presenta en la siguiente figura.

Resumen del dataframe

0	Temp Avg	21587 non-null	float64
1	Temp Low	21346 non-null	float64
2	Temp High	21346 non-null	float64
3	Heat Index	21587 non-null	float64
4	Wind Chill	21587 non-null	float64
5	Temp (Day) Low	21336 non-null	float64
6	Temp (Day) High	21336 non-null	float64
7	Hum Avg	21587 non-null	float64
8	Hum Low	21346 non-null	float64
9	Hum High	21346 non-null	float64
10	Dew Point	21587 non-null	float64
11	Hum (Day) Low	21415 non-null	float64
12	Hum (Day) High	21415 non-null	float64
13	Baro Avg	21587 non-null	float64
14	Baro Low	21346 non-null	float64
15	Baro High	21346 non-null	float64
16	Baro (Day) Low	21415 non-null	float64
17	Baro (Day) High	21415 non-null	float64
18	Windspeed	21587 non-null	float64
19	Wind Direction	21587 non-null	float64
20	Gust	21587 non-null	float64
21	Gust Direction	21346 non-null	float64
22	Gust (Day) Max	21415 non-null	float64
23	Gust (Day) Direction	21346 non-null	float64
24	Leaf Wetness (Minutes)	0 non-null	float64
25	Leaf Wetness (Day)	0 non-null	float64
26	Solar Radiation Avg	21587 non-null	float64
27	Solar Radiation Sum (Interval)	21587 non-null	float64
28	Solar Radiation Sum (Day)	21587 non-null	float64
29	Interval Precip	21587 non-null	float64
30	Day Precip	21587 non-null	float64
31	Inside Temp Avg	21587 non-null	float64
32	Inside Temp Low	21346 non-null	float64
33	Inside Temp High	21346 non-null	float64
34	Inside Temp (Day) Low	21415 non-null	float64
35	Inside Temp (Day) High	21415 non-null	float64
36	Station Voltage	21755 non-null	float64
37	UV Index	0 non-null	float64
38	Solar Radiation 2 Avg	21587 non-null	float64
39	Solar Radiation 2 Sum (Interval)	21587 non-null	float64
40	Solar Radiation 2 Sum (Day)	13821 non-null	float64

Resumen valores nulos

Temp Avg	2509
Temp Low	2750
Temp High	2750
Heat Index	2509
Wind Chill	2509
Temp (Day) Low	2760
Temp (Day) High	2760
Hum Avg	2509
Hum Low	2750
Hum High	2750
Dew Point	2509
Hum (Day) Low	2681
Hum (Day) High	2681
Baro Avg	2509
Baro Low	2750
Baro High	2750
Baro (Day) Low	2681
Baro (Day) High	2681
Windspeed	2509
Wind Direction	2509
Gust	2509
Gust Direction	2750
Gust (Day) Max	2681
Gust (Day) Direction	2750
Leaf Wetness (Minutes)	24096
Leaf Wetness (Day)	24096
Solar Radiation Avg	2509
Solar Radiation Sum (Interval)	2509
Solar Radiation Sum (Day)	2509
Interval Precip	2509
Day Precip	2509
Inside Temp Avg	2509
Inside Temp Low	2750
Inside Temp High	2750
Inside Temp (Day) Low	2681
Inside Temp (Day) High	2681
Station Voltage	2341
UV Index	24096
Solar Radiation 2 Avg	2509
Solar Radiation 2 Sum (Interval)	2509
Solar Radiation 2 Sum (Day)	10275

Fig. 32. Resumen de datos válidos y datos nulos.

Para el llenado de datos faltantes y datos Null, se aplicó la media aritmética considerando que en las horas de la noche los valores de radiación solar son cero, se realizó la revisión de los datos exportando al formato .CSV por facilidad de visualización y completar los datos faltantes de todas las mediciones, luego de ser depurados los registros medidos se guardan en formato .CSV para el diseño del algoritmo, se muestra la gráfica del mes de enero con los datos completos y depurados.

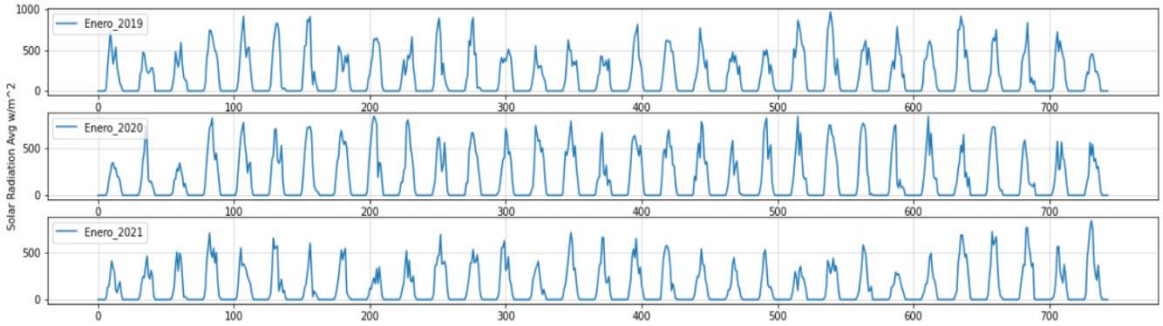


Fig. 33. Datos completos de Radiación solar mes enero.

El preprocesamiento de datos se realizó al dataframe, se imprime el resumen de los resultados para le verificación, se presenta en la siguiente figura.

Resumen del dataframe

0	Temp Avg	24096 non-null	float64
1	Temp Low	24096 non-null	float64
2	Temp High	24096 non-null	float64
3	Heat Index	24096 non-null	float64
4	Wind Chill	24096 non-null	float64
5	Temp (Day) Low	24096 non-null	float64
6	Temp (Day) High	24096 non-null	float64
7	Hum Avg	24096 non-null	float64
8	Hum Low	24096 non-null	float64
9	Hum High	24096 non-null	float64
10	Dew Point	24096 non-null	float64
11	Hum (Day) Low	24096 non-null	float64
12	Hum (Day) High	24096 non-null	float64
13	Baro Avg	24096 non-null	float64
14	Baro Low	24096 non-null	float64
15	Baro High	24096 non-null	float64
16	Baro (Day) Low	24096 non-null	float64
17	Baro (Day) High	24096 non-null	float64
18	Windspeed	24096 non-null	float64
19	Wind Direction	24096 non-null	float64
20	Gust	24096 non-null	float64
21	Gust Direction	24096 non-null	float64
22	Gust (Day) Max	24096 non-null	float64
23	Gust (Day) Direction	24096 non-null	float64
24	Solar Radiation Avg	24096 non-null	float64
25	Solar Radiation Sum (Interval)	24096 non-null	float64
26	Solar Radiation Sum (Day)	24096 non-null	float64
27	Interval Precip	24096 non-null	float64
28	Day Precip	24096 non-null	float64
29	Inside Temp Avg	24096 non-null	float64
30	Inside Temp Low	24096 non-null	float64
31	Inside Temp High	24096 non-null	float64
32	Inside Temp (Day) Low	24096 non-null	float64
33	Inside Temp (Day) High	24096 non-null	float64
34	Station Voltage	24096 non-null	float64
35	UV Index	24096 non-null	float64
36	Solar Radiation 2 Avg	24096 non-null	float64
37	Solar Radiation 2 Sum (Interval)	24096 non-null	float64
38	Solar Radiation 2 Sum (Day)	24096 non-null	float64
39	Unnamed: 42	24096 non-null	int64
40	Unnamed: 43	24096 non-null	float64

Resumen valores nulos

Temp Avg	0
Temp Low	0
Temp High	0
Heat Index	0
Wind Chill	0
Temp (Day) Low	0
Temp (Day) High	0
Hum Avg	0
Hum Low	0
Hum High	0
Dew Point	0
Hum (Day) Low	0
Hum (Day) High	0
Baro Avg	0
Baro Low	0
Baro High	0
Baro (Day) Low	0
Baro (Day) High	0
Windspeed	0
Wind Direction	0
Gust	0
Gust Direction	0
Gust (Day) Max	0
Gust (Day) Direction	0
Solar Radiation Avg	0
Solar Radiation Sum (Interval)	0
Solar Radiation Sum (Day)	0
Interval Precip	0
Day Precip	0
Inside Temp Avg	0
Inside Temp Low	0
Inside Temp High	0
Inside Temp (Day) Low	0
Inside Temp (Day) High	0
Station Voltage	0
UV Index	0
Solar Radiation 2 Avg	0
Solar Radiation 2 Sum (Interval)	0
Solar Radiation 2 Sum (Day)	0
Unnamed: 42	0
Unnamed: 43	0

Fig. 34. Resumen de datos depurados.

3.2.3 Carga y lectura de los datos depurados

Los datos depurados se guardaron en el drive personal, se procede al accedo al drive y cargar en dataframe separados por cada año 2019, 2020 y 2021, se conserva los nombres de cada columna para identificar la variable medida y el tiempo es de suma importancia las fechas de registro.

```
from google.colab import drive   # Acceso a los archivos del drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).


# Dataframe con registros depurados del año 2019
df2019_depurado=pd.read_csv('/content/drive/My Drive/Colab Notebooks/MEDICIONES_MES/AÑ0_2019_depurado.csv',parse_dates=['Time'],index_col=['Time'])
# Dataframe con registros depurados del año 2020
df2020_depurado=pd.read_csv('/content/drive/My Drive/Colab Notebooks/MEDICIONES_MES/AÑ0_2020_depurado.csv',parse_dates=['Time'],index_col=['Time'])
# Dataframe con registros depurados del año 2021
df2021_depurado=pd.read_csv('/content/drive/My Drive/Colab Notebooks/MEDICIONES_MES/AÑ0_2021_depurado.csv',parse_dates=['Time'],index_col=['Time'])
```

Fig. 35. Carga de datos depurados 2019, 2020 y 2021.

Los datos registrados son de 33 meses, para identificar el comportamiento de la radiación en el tiempo de registro, se procede a la visualización por cada hora, para conocer los valores de los registros no se visualiza del año completo es decir los 12 meses porque no se puede diferenciar en la gráfica, por tal razón se conoce el comportamiento comparando los 10 primeros días del mes de enero del 2019, 2020, 2021, se identifica que la radiación solar es cíclica y mantiene un estado estacionario todos los días, solo varia la intensidad de radiación en el día, por lo que es considerado una serie temporal por tener secuencia cíclica.

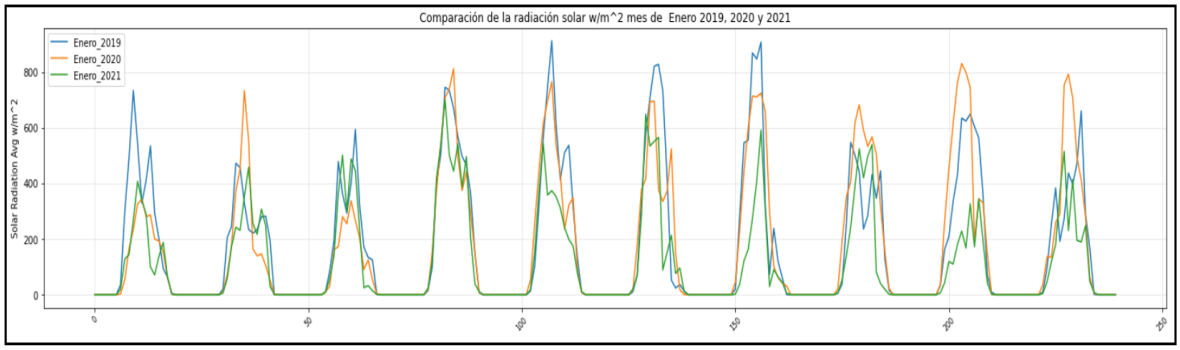


Fig. 36. Mediciones mes de enero 2019, 2020 y 2021

3.2.4 Correlación de variables

Se realiza la correlación a todas las variables del registro con la radiación solar, se conocen las siguientes variables que tienen alta relación:

- *Sola radiation Avg:* Radiación solar promedio en w/m^2 (vatio/metro cuadrado)
- *Hum Avg:* humedad promedion en % (porcentaje)
- *Windspeed:* velocidad del viento en m/s (metro/segundo)
- *Temp Avg:* Temperatura promedio en °C (grados Celsius)
- *Baro Avg:* Presión barométrica promedia en mb (milibares)

La correlación se obtiene con *df2019.corr(), df2020.corr()* y *df2021.corr()* que al comparar los valores solo tienen decimas de variación, se utilizara las variables identificadas como directrices en el entrenamiento del algoritmo y obtener una mejor aproximación de resultados.

	Solar Radiation Avg w/m^2	Baro Avg mb	Hum Avg %	Windspeed m/s	Temp Avg °C
Solar Radiation Avg 2019	1.000000	-0.049694	-0.703033	0.515986	0.707080
Solar Radiation Avg 2020	1.000000	-0.056684	-0.701950	0.499796	0.711919
Solar Radiation Avg 2021	1.000000	-0.011157	-0.695091	0.485567	0.608614

Fig. 37. Correlación de variables climáticas

3.2.5 Selección de la población y muestra.

La población de los registros presentó 44 variables semejantes, y para la elección de la muestra con la ayuda de la correlación entre la variable radiación solar se identificaron solo 5 variables con alto nivel de correlación, se unen los años 2019, 2020 y del 2021 se utiliza hasta el mes de agosto.

```
df2019 = df2019_depurado['2019-01-01':'2019-12-31']  # Dataframe del 2019
df2020 = df2020_depurado['2020-01-01':'2020-12-31']  # Dataframe del 2020
df2021 = df2021_depurado['2021-01-01':'2021-08-31']  # Dataframe del 2021
frames = [df2019,df2020,df2021]    # Selección de los Dataframe para concatenar
df = pd.concat(frames)             # Concatenación de los Dataframe
# Selección de las variables identificadas con mayor relación en correlación
df = df[["Solar Radiation Avg","Baro Avg","Hum Avg","Windspeed","Temp Avg"]]
```

Fig. 38. Concatenación en un solo dataframe "df"

El Dataframe consolidado se denomina "*df*" con 974 días con los valores en frecuencia de las 24 horas genera una dataframe de dimensiones 23.376 x 5 datos por cada variable para procesar, la información se obtuvo con el comando *"df.info()"*

```
DatetimeIndex: 23376 entries, 2019-01-01 00:00:00 to 2021-08-31 23:00:00
Freq: H
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Solar Radiation Avg  23376 non-null  float64
 1   Baro Avg             23376 non-null  float64
 2   Hum Avg              23376 non-null  float64
 3   Windspeed            23376 non-null  float64
 4   Temp Avg             23376 non-null  float64
dtypes: float64(5)
memory usage: 1.1 MB
```

Fig. 39. Información del Dataframe consolidado.

Las horas de radiación solar dependen del amanecer y atardecer, se ha identificado que las horas no varían iniciando desde las 6:00 horas de la mañana hasta las 18:00 horas del atardecer, se debe considerar los valores a procesar dentro del rango establecido por motivo que en la noche no hay valores de radiación, con la eliminación de los valores cero las dimensiones se reducen a 12.662 x 5

```
# Eliminación de valores de radiación menores de 0
df = df.drop(df[df['Solar Radiation Avg']<1].index)
```

Fig. 40. Eliminación de valores de radiación menores a 0.

3.2.6 Procesamiento de datos de entrenamiento.

Los datos consolidado sen un solo dataframe se denominaron *"df"*, para comenzar a procesar los datos se copia la información en un *dataset_train_actual*, se revisa los valores nulos y faltantes, procesándolo para cuantificar la cantidad de datos faltantes, se imprime el resumen de resultados identificando que los datos a trabajar son una matriz de 12.662x6.

```
dataset_train_actual = df.copy()              #Copia del dataframe actual a df
dataset_train_actual.isnull().sum()           #Resumen del dataframe
dataset_train_actual.dropna( inplace=True )   #Eliminación de valores Nan y Nulos
#processamiento de datos Nan
dataset_train_actual = dataset_train_actual.fillna(dataset_train_actual.mean())
#procesamiento de los datos con el index
dataset_train_actual = dataset_train_actual.reset_index()
#visualización para comprobar datos nulos
dataset_train_actual.isnull().sum()
# Imprimir las dimensiones del Dataframe
dataset_train_actual.shape
```

Fig. 41. Procesamiento de datos de entrenamiento.

3.2.7 Procesamiento de fechas

Con la librería datetime se realiza el procesamiento de las fechas, se ha seleccionado la columna time como índice para no alterar la secuencia de los datos, se realiza una copia del dataset actual para el procesamiento, guardando las cinco variables con mayor correlación, se crea la lista de fechas con rango de cada hora de 6:00 a 18:00 y se guarda en *datelis_train* ordenando con un ciclo *for* para ordenar la nueva columna de fechas.

```
# Fijación de la columna de TIME como índice
dataset_train_timeindex = dataset_train_actual.set_index('Time')
# Copia de los valores a dataset_train
dataset_train = dataset_train_actual.copy()
# Lista de las variables seleccionado despues de la columna de fechas
cols = list(dataset_train)[1:6]
# Creación de las fechas de 6:00 a 18:00 en rango de una hora
datelist_train = pd.date_range(start=pd.Timestamp('06:00'), end=pd.Timestamp('18:00'), freq='1H')
# Ordenamiento en de las fechas en una lista
datelist_train = list(dataset_train['Time'])
# Creanción de la columna de fechas
datelist_train = [date for date in datelist_train]
```

Fig. 42. Procesamiento de fechas con datetime.

3.2.8 Ordenamiento de los datos

Los datos del Dataframe se ordenaron utilizando las fechas como índice y *'Solar Radiation Avg', 'Baro Avg', 'Hum Avg', 'Windspeed', 'Temp Avg'* como proditores que aporten en la predicción, para que los datos sean interpretados por Python es necesario convertir las fechas en una lista en formato Timestamp que mantenga la secuencia de fechas y los parámetros medidos convertirlos en un array en una matriz con la ayuda de un ciclo for.

```
# Procesamiento de los datos de entrenamiento
dataset_train = dataset_train[cols].astype(str)
for i in cols:
    for j in range(0, len(dataset_train)):
        dataset_train[i][j] = dataset_train[i][j].replace(',', '')
dataset_train = dataset_train.astype(float)
# Declaración de los datos predictores
training_set = dataset_train.values
```

Fig. 43. Datos de registro en Array

3.2.9 Escalamiento de datos

Los datos del array están en diferentes escalas de medición como la humedad en el primer registro tiene un valor de 93.55 y la temperatura es de 8.74, los valores tienen diferencia, pero es porque no son la misma variable, entonces se utiliza la librería de StandardScaler para normalizar los datos como convertir en valores por unidad, los valores normalizados estarán entre 0 y 1.

```
# Escalamiento de los valores con StandardScaler
from sklearn.preprocessing import StandardScaler
# Desiganción de StandardScaler como sc
sc = StandardScaler()
# Escalamiento de training_set
training_set_scaled = sc.fit_transform(training_set)
# Desiganción de StandardScaler() como sc_predict
sc_predict = StandardScaler()
# Escalamiento de datos predictores
sc_predict.fit_transform(training_set[:, 0:1])
```

Fig. 44. Escalamiento de los registros

3.2.10 Datos de entrenamiento y prueba

Se crean las variables *x_train* y *y_train* para en entrenamiento, considerando que se aplica series de tiempo se estable 91 horas a predecir que corresponde a 7 días de 6:00 a 18:00 horas, los datos del pasado a procesar son de 180 horas, como se están borrando los datos menores a cero los datos ingresados deben ser de un mes, los valores a separar lo realizar el automáticamente la librería, se ha considerado por recomendaciones mantener el 80% de entrenamiento y 20% de prueba.

```
# Separación de datos de entrenamiento y prueba
from sklearn.model_selection import train_test_split
X_train = []     # Variable X de datos de entrenamiento
y_train = []     # Variable Y de datos de entrenamiento
n_future = 91    # Número de horas a predecir en el futuro
n_past = 180     # Número de horas del pasado a revisar para predecir
# Llena los valores escalados en las variables de entremaniento y prueba
for i in range(n_past, len(training_set_scaled) - n_future +1):
    X_train.append(training_set_scaled[i - n_past:i, 0:dataset_train.shape[1]])
    y_train.append(training_set_scaled[i + n_future - 1:i + n_future, 0])
# Estructura de los datos de entremaniento y prueba con 20% de separación
X_train,xtest, y_train, ytest = train_test_split( np.array(X_train), np.array(y_train),test_size=0.2,random_state=0)
```

Fig. 45. Datos de entrenamiento y prueba

3.2.11 Estructura de la red neuronal

Para la creación de la red neuronal se utiliza la librería de keras y tensor Flow, importando Secuancial, Dense, LSTM, Dropout y Adam, se ha considerado una red LSTM por la memoria que guarda las grabaciones de variación de comportamiento de datos y al tratarse de una serie de datos con el Dropout para evitar la memorización de lados.

```
# Importación de librerias para crear la red neuronal
from keras.models import Sequential   # Usar datos secuenciales
from keras.layers import Dense        # Procesamiento de varias capas
from keras.layers import LSTM         # Neurona con memoria
from keras.layers import Dropout      # Evita que memorice los valores
from tensorflow.keras.optimizers import Adam  # Optimizador del algoritmo
```

Fig. 46. Librerías para la red neuronal

La estructura de la red neuronal se realiza con tres capas, la primera capa de entrada con 64 neuronas, la segunda capa oculta con 32 neuronas, se activa el Dropout con el 25% para la

regularización y evitar el overfitting , borra capas de forma aleatoria solo para en el entrenamiento y para la validación todas las capas funcionan y una capa para la salida, la activación se realiza con tangente hiperbólica y el optimizador Adam, las perdidas con erro cuadrático medio, se ha revisado los resultados con diferentes valores como la variación de los pasos del aprendizaje desde 0,1, 0,01, 0,001 y 0,0001, se obtuvo mejores resultados con el valor de 0,00045 y la cantidad de neuronas definidas en cada capa.

```
def build_model():
  model = Sequential()
 # Capa de entrada con 64 neuronas
  model.add(LSTM(units=64, return_sequences=True, input_shape=(n_past, df.shape[1])))
 # Capa oculta con 32 neuronas
  model.add(LSTM(units=32, return_sequences=False))
 # Dropout con 25% para evitar la memorizacion de datos
  model.add(Dropout(0.25))
 # Capa de salida con una sola salida
  model.add(Dense(units=1, activation='linear'))
 # Optimizador del algoritmo y perdidas loss
  model.compile(optimizer = Adam(learning_rate=0.00045), loss='mean_squared_error')
  return model
```

Fig. 47. Estructura red neuronal

3.2.12 Entrenamiento de la red neuronal

En el entrenamiento de la red neuronal se ha considerado utilizar un *Callbacks* para evitar que memorice los datos en el entrenamiento, se ha utilizado un valor de paciencia de 10, se ha activado *shuffle* para mezclar las muestras en cada época, 500 épocas considerando que dependerá de la actuación del *callbacks* para la finalización temprana antes de llegar a 500, la validación el 20%, activado verbose para la impresión de los resultados del entrenamiento, el *bacth_size* de 150 para las limitaciones de las muestras.

```
# Callbacks configurado para que cuando repetidamente empieze a memorizar finalice el aprendizaje
es=keras.callbacks.EarlyStopping(monitor='val_loss',mode='min',patience=10)
# Funcion de la neurona
model=build_model()
# Entrenamiento de la neurona con 500 epocas, 20% de validacion y un batch_size de 150
history = model.fit(X_train, y_train, shuffle=True, epochs=500, validation_split=0.2, verbose=1,batch_size=150,callbacks=[es])
```

Fig. 48. Entrenamiento de la red neuronal

3.2.13 Métricas del entrenamiento y prueba

La variación de los parámetros de la red neuronal influye en los valores resultantes, se ha experimentado variando cada una de los parámetros y se ha determinado los valores de pérdidas de entrenamiento como de prueba, manteniendo las neuronas y el dropout , se ha variado los pasos del optimizador y la cantidad del size, se recuerda que se utiliza el callbacks para evitar la memorización por lo las épocas varían dependiendo la activación automática, los mejores resultados se han obtenido con pasos muy pequeños de 0.0004 y con una pequeña deferencia entre los size de 150 y 300, las gráficas de los errores y de prueba y entrenamiento se encuentra en el Anexo 6, los valores de respuestas son presentados en la tabla 3.

TABLA 3.
COMPARACIÓN DE MÉTRICAS DE ENTRENAMIENTO Y PRUEBA

1RA CAPA	2DA CAPA	DROPOUT	ADAM	EPOCHS	SIZE	LOSS	VAL_LOSS
64	32	0.25	0.00045	162	150	0.1459	0.2286
64	32	0.25	0.0004	180	150	0.2341	0.2935
64	32	0.25	0.004	103	150	0.8214	0.7363
64	32	0.25	0.04	25	150	0.8304	0.7769
64	32	0.25	0.4	24	150	1.11	1.16
64	32	0.25	0.00045	192	300	01827	0.2542
64	32	0.25	0.0004	258	300	0.205	0.2764
64	32	0.25	0.004	74	300	0.8341	0.7833
64	32	0.25	0.04	39	300	0.8199	0.7877

3.2.14 Predicciones de la radiación

Los valores ingresados para el entrenamiento y prueba son hasta el mes de septiembre 2021, el algoritmo se ha entrenado con los valores del 2019, 2020 y 2021, para la predicción se ha separado el mes de agosto 2021 para predecir los 7 días posteriores, se realiza el tratamiento a las fechas a predecir con Datetime para generar la secuencia de los días predecidos y completar las 24 horas porque el entrenamiento se realizó de 6:00 a 18:00.

```
# Generación de las fechas a partir del último registro
datelist_future = pd.date_range(datelist_train[-1], periods=182, freq='1h').tolist()
rangopredicion = range(182)         # Designación a rangopredicion un rago de 182 valores
list(rangopredicion)                # Convertir a rango predicción en lista
fake = pd.DataFrame(rangopredicion) # Creación de las dimensiones
# Fijación de datelist como indice
temporal = pd.DataFrame(fake, columns=['Solar Radiation']).set_index(pd.Series(datelist_future))
# Rango de horas de 6:00 a 18:00
temporal = temporal.between_time('06:00:00', '18:00:00')
# Reseteo del indice y la primera es la de fechas
temporal = temporal.rename_axis('index').reset_index()
# Convertir en formato timestamp
temporal1 = temporal['index'].tolist()
del temporal1[0:3]
# Ordenar en en formato año, mes, dia
datelist_future_ = []
for this_timestamp in temporal1:
    datelist_future_.append(this_timestamp.date())
# Función para la conversion de fechas
def datetime_to_timestamp(x):
    '''
        x : a given datetime value (datetime.date)
    '''
    return datetime.strptime(x.strftime('%Y%m%d'), '%Y%m%d')
```

Fig. 49. Procesamiento de fechas de la predicción

Se selecciona el mes de datos a ingresar al algoritmo para la predecir los días consecutivos, los datos de entrenamiento y prueba son hasta agosto 2021, para la predicción se ingresa el mes de septiembre 2021 que son los registro que no ha visto el algoritmo, se realiza el procesamiento de escalar los valores, el mismo proceso se realizó con los datos de entrenamiento, los registros al encontrarse en un solo dataframe se selecciona escribiendo el rango de fechas que va desde el 27 de agosto al 23 de septiembre 2021.

```
# Selección del rango de fechas que no ha visto el algoritmo
df2021 = df2021_depurado['2021-08-27':'2021-09-23']
```

Fig. 50. Selección de datos para realizar la predicción.

Para la predicción se realiza con *model.prdit*, los datos procesado están en la variable *X_*, en función de los datos se predice los siete días consecutivos, como los valores son escalados la predicción es en valores menores a uno, se convierten en valores reales invirtiendo el escalamiento, y se organiza los valores con las fechas para identificar la hora y el valor de la radiación solar.

```
#Predicción de los valores posteriores
predicccion_futura = model.predict(X_[-d_pred:])
# Reescalamiento de los valores predecidos
y_pred_future = sc_predict.inverse_transform(predicccion_futura)
# Seleccion del indice las nuevas fechas
PREDICCION_2 = pd.DataFrame(y_pred_future, columns=['Solar Radiation Avg']).set_index(pd.Series(temporal1))
# Seleccion de los valores predecidos de la radación solar
week_predicted = pd.DataFrame(y_pred_future, columns=['Solar Radiation Avg'])
```

Fig. 51. Predicción días posteriores.

Para verificar si la predicción realizada con el algoritmo se acerca a valores reales se procede a cargar el registro realizado por la estación que corresponde del 24 de al 30 de septiembre para contrastar con los valores predecidos que son las mismas fechas, se realiza el procesamiento de datos de selecciona la coluna de radiación solar, borrar los datos de cero y valores NAN.

```
# Selección de registro medido por la estación para la comparación con lo predecido
MES_PRUEBA = df2021_depurado['2021-09-24' : '2021-09-30']
#  Selección de la radiación solar de todos los registros
MES_PRUEBA = MES_PRUEBA[["Solar Radiation Avg"]]
# Representación por cada hora de medición
MES_PRUEBA = MES_PRUEBA.resample('H').mean()
# Eliminación de los alores menores a cero
MES_PRUEBA = MES_PRUEBA.drop(MES_PRUEBA[MES_PRUEBA['Solar Radiation Avg']<1].index)
# Tratamiento de valores NAN
MES_PRUEBA = MES_PRUEBA[MES_PRUEBA['Solar Radiation Avg'].notna()]
# Grafica de validación
plt.scatter(range(len(MES_PRUEBA)),MES_PRUEBA,c='g')
plt.scatter(range(len(PREDICCION_2)),PREDICCION_2,c='r')
plt.title('validación ')
plt.show()
```

Fig. 52. Selección de datos medidos por la estación para comparar.

3.2.15 Comparación de la predicción con valores reales

Para comprobar la efectividad del algoritmo, aparte de los valores de error en el entrenamiento se compara los valores de la predicción generado por el algoritmo de los días consecutivos, los días predecidos corresponden del 24 de octubre al 1 de noviembre, de la misma manera se cargan los registros correspondientes a los mismos días que fueron gravados por la estación meteorológica, las mediciones se presentan en un solo gráfico.

```
# Valores de radiación predecidos
plt.plot(PREDICCION_2.index, PREDICCION_2['Solar Radiation Avg'], color='r', label='Prediccion radiación solar Avg w/m^2')
# Valores de radiación medidos estación
plt.plot(MES_PRUEBA.index,MES_PRUEBA['Solar Radiation Avg'], color='orange', label='Octubre radiación solar Avg w/m^2')
# Linea de inicio
plt.axvline(x = min(PREDICCION_2.index), color='green', linewidth=2, linestyle='--')
# Regilla
plt.grid(which='major', color='#cccccc', alpha=0.5)
# Leyenda
plt.legend(shadow=True)
# Titulo de grafica
plt.title('Predicciones y valores medidos de radiación solar Avg values w/m^2', family='Arial', fontsize=12)
# Estilo eje x
plt.xlabel('Timeline', family='Arial', fontsize=10)
# Estilo eje y
plt.ylabel('radiación solar Avg Value w/m^2', family='Arial', fontsize=10)
# Rotación letras eje x
plt.xticks(rotation=45, fontsize=8)
plt.show()
```

Fig. 53. Código de comparación valor predecido con valor medido de la estación

La curva de color amarillo representa los registros de la estación meteorológica, los valores en horas picos se observa con altos y bajos, la curva de color rojo corresponde a los valores predecidos del algoritmo, se observa que se ajusta a los dos valores por cada hora.

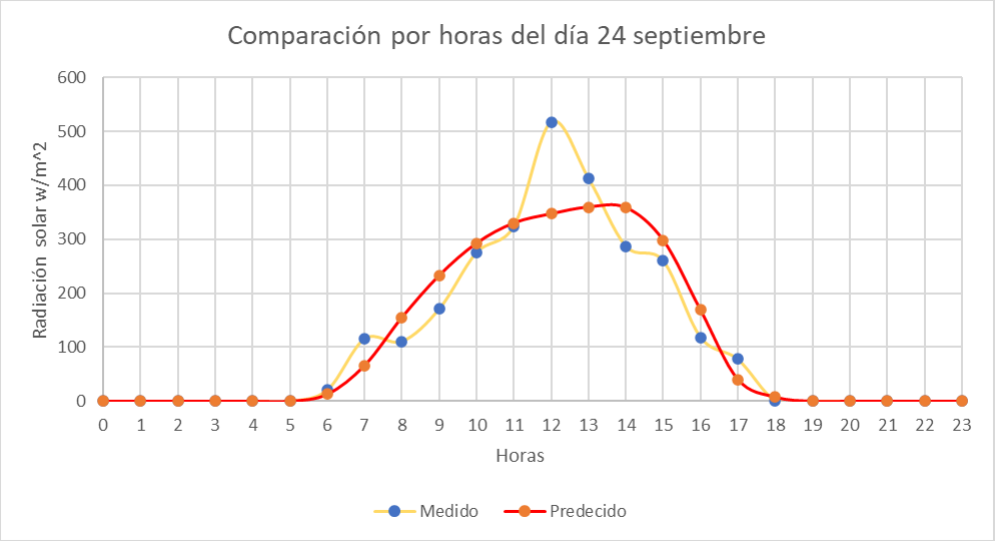


Fig. 54. Gráfica de comparación valor predecido con valor medido de la estación

TABLA 4.
COMPARACIÓN DE LA RADIACIÓN SOLAR w/m^2, DE VALORES MEDIDOS CON LA ESTACIÓN Y LOS VALORES PREDECIDOS POR EL ALGORITMO.

Hora	día 24 (w/m^2)		día 25 (w/m^2)		día 26 (w/m^2)		día 27 (w/m^2)		día 28 (w/m^2)		día 29 (w/m^2)		día 30 (w/m^2)	
	medido	predecido	medido	predecido	medido	predecido	medido	predecido	medido	predecido	medido	predecido	medido	predecido
6:00	21.0	13.0	28.5	14.2	25.9	30.6	22.7	38.2	25.9	10.7	19.4	0.7	16.1	7.5
7:00	115.8	66.5	150.4	122.4	82.1	98.1	144.4	135.3	82.1	77.0	112.2	63.3	74.7	79.7
8:00	110.7	155.4	314.2	296.8	144.0	198.3	338.7	263.1	144.0	170.3	242.0	173.9	164.3	195.5
9:00	171.0	233.2	280.9	462.1	155.3	301.4	394.6	404.5	155.3	245.4	332.5	303.1	323.9	321.0
10:00	275.5	292.7	335.8	572.1	367.2	375.5	480.3	517.7	167.2	285.0	556.2	424.5	375.0	424.1
11:00	324.2	330.0	480.8	617.3	410.8	412.0	302.3	561.6	207.3	300.7	337.6	507.3	527.5	479.3
12:00	517.2	347.0	518.6	591.3	302.9	425.1	383.6	538.8	212.9	300.0	289.8	540.1	304.2	476.8
13:00	412.5	359.5	365.3	494.3	405.8	412.5	451.8	452.7	224.7	278.8	305.2	536.9	251.9	445.1
14:00	287.0	358.4	364.2	355.4	187.1	383.2	335.3	321.0	187.1	226.5	340.1	488.8	298.7	397.6
15:00	259.7	298.2	329.0	216.5	163.0	319.0	299.7	196.1	163.0	141.2	149.4	372.1	136.0	328.5
16:00	118.4	168.9	215.6	118.1	84.3	208.9	208.1	116.9	84.3	60.5	152.5	207.5	24.9	208.6
17:00	77.7	40.5	74.2	43.2	32.1	80.3	133.8	49.6	32.1	14.6	42.6	74.8	5.6	74.8
18:00	1.5	-7.7	3.0	9.6	1.5	23.3	1.5	1.5	1.5	-9.3	1.5	7.8	1.5	11.9
AVG	207.1	204.3	266.2	301.0	181.7	251.4	269.0	276.7	129.8	161.7	221.6	284.7	192.6	265.4
% diferencia	1.4%		13.1%		8.4%		2.9%		24.5%		28.5%		37.8%	

Los valores predecidos de los 7 días, se exportaron y se presentan en la tabla 2 para analizar los valores medidos con los predecido y comprobar la efectividad del algoritmo, la horas comprenden de 6:00 a 18:00 horas, iniciando desde el día 24 hasta el 30 de octubre, se ha realizado un valor promedio del día para la comparación y se ha calculado el porcentaje de diferencia de cada día, se observa que los valores en la gráfica y en los valores en la tabla se acercan a la realidad, y en el valor promedio en ocasiones es muy bajo el error, como es el caso del día 24 el error es de 1.4% y el día 26 es del 38.4% , aunque las curvas en la imagen sean similares, la variación se produce en los valores de cada hora que en ocasiones se mantiene altos, pero al calcular el error promedio de los siete días se obtiene 20,9% comprando al valor de error en la prueba es del 22.8%, los valores están en el rango de entrenamiento.

3.3 Conclusiones del capítulo III.

La estación meteorológica es un instrumento que permite recopilar información de las variables climáticas, guardando en una nube del instrumento, y para ser analizado se exporta en formato .CSV que es procesado en lenguaje Python en Google Colab, la mayor parte del tiempo se dedica al análisis y verificación de las mediciones, corrigiendo valores Null NAN y vacíos, por la facilidad la revisión se realizó minuciosamente por cada mes a través de Excel, el procesamiento realizado con datos sin procesar el resultado son erróneos, afecta en la secuencia de datos para el entrenamiento y la predicción arroja horas con saltos de tiempo e incompletos y los errores en el entrenamiento y la validación son altos, después se realizó el entrenamiento con datos depurados completos y el resultado se refleja en los valores predecidos, ajustándose a los valores reales y los errores en el entrenamiento y prueba se redujeron, también el análisis está enfocado para el uso de la radiación solar en cada hora, en los dimensionamiento de sistemas solares es necesario conocer la curva diaria solar, para la estructura de la red neuronal se aplicó la LSTM por la capacidad de memorizar los eventos, en estudios realizados sobre series de tiempo se ha encontrado mejor aproximación con este tipo de red neuronal, los datos ingresados corresponden a dos años, es necesario tener más tiempo registrado para mejorar el entrenamiento del algoritmo.

Conclusiones

Las referencias bibliográficas destinadas a la predicción de la radiación solar se encuentran en publicaciones de trabajos similares, pero se utiliza métodos tradicionales de Machine Learning como regresión lineal, arboles de dediciones, Boosting, también la información se ha encontrado en libros, Blog y páginas de personas que han dedicado al análisis de datos, se ha encontrado varios curso de capacitación pero se exponen los técnicas convencionales y la información sobre series temporales es escasa, lo poco que se encuentra está relacionado a la parte de economía a la predicción de bolsa de valores que se aplica por valores medias de cada día por, tal motivo no hay información para predicción sobre cada hora como es lo requerido para diseños de energía solar.

El registro de parámetros climáticos mediante la estación meteorológica se dificulta al momento de instalar la estación en un sitio que no tiene red eléctrica porque no se puede energizar el servidor para la recepción de datos, el alcance de la estación es de un kilómetro hacia el modem de recepción y no se tiene el acceso a internet en las zonas alejadas para el monitoreo, también los registros que realiza la estación es en una nube que se puede ingresar mediante el internet para descargar los registro en valores separados por coma .CSV que es un formato estándar para la lectura y análisis en programas de análisis de datos.

El diseño del algoritmo para la predicción de la radicación solar se ha aplicado series de tiempo con el tensor Flow para le conexión de los datos de entrada y keras en el entrenamiento del algoritmo, se ha creado una red secuencial con la LSTM por la facilita que grava el comportamiento de los datos, se ha estructurado con 64 neuronas de entrada una capa oculta y una salida, para el entrenamiento es necesario que los datos sean escalados para que no altere las diferentes valores

El algoritmo en el entrenamiento muestra las métricas de error, pero los datos ingresados son vistos por el algoritmo, y para verificar la efectividad del algoritmo se ha decidido ingresar los valores registrados por la estación de un nuevo mes que no haya sido visto por

el algoritmo, se ha ingresado el mes de septiembre para predecir los días consecutivos de octubre, el resultado óptimo se obtuvo ingresando los datos depurados, completando valores faltantes, nulos o vacíos, se realizó una comparación de los días predecidos con los medidos con el valor medio verificando que el valor del error del testeo es de 22.8%, y el valor promedio de los datos de los siete días predecidos es de 20,9% verificando que los valores son cercanos al entrenamiento de la red.

Recomendaciones

La búsqueda de información sobre la aplicación de técnicas de Machine Learning es recomendable usar varias fuentes como publicaciones y estudios relacionados variables climáticas para conocer el rendimiento del método aplicado y revisar los libros de los autores que presentan códigos similares, es recomendable guiarse con los códigos aplicados a finanzas porque son los que más se encuentran información con la descripción del funcionamiento para modificar las estructura y obtener el resultado esperado.

Es necesario identificar el sitio donde reposará la estación para el registro del clima, considerando que la selección del modelo de estación debe cubrir las áreas alejas y poseer un panel fotovoltaico como fuente de alimentación autónoma en la estación, el alcance para envío de los datos debe estar dentro del rango de cobertura que no se pierda los datos en el envío por interrupciones de la señal de envió, también el registro realizado para el análisis de al programa es necesario que se exporte en formato .CSV por ser estándar en los lenguajes de programación la lectura.

El análisis de datos con el tiempo como variable es recomendable aplicar series de tiempo porque analiza los datos de tiempo atrás y en función de lo datos revisados predice el comportamiento de la variable, con la utilización de librerías de Python como Keras y tensor Flow ayudan a que la creación de las redes neuronales sea fácil en la creación como el ingreso de parámetros porque la estructura ya integra la codificación y solo con la

creación de tensores de uniones entre capas facilita el procesamiento de los datos, también se debe ir cambiando las configuraciones como neuronas , pasos, cantidad de datos para obtener diferentes resultado y elegir el óptimo.

Es necesario identificar los valores a comparar corresponden a valores promedios de un día o a valores por cada hora, porque al ser analizados en cada no se pueden modificar el orden altera el resultado final como en el caso de radiación se eliminó Las horas de la noche para evitar predicciones que se malinterpreten como un valor de radiación en la noche cuando ya no hay la presencia del Sol.

Referencias bibliográficas.

[1] J. P. Castanedo-Cázares, B. Torres-Álvarez, B. Portales-González, K. Martínez-Rosales, y D. Hernández-Blanco, «Análisis de la radiación solar ultravioleta acumulada en México», *Rev Med Inst Mex Seguro Soc*, p. 7.

[2] I. Z. Ducun, D. J. M. Álvarez, y D. M. G. Solano, «Departamento de Ingeniería Eléctrica y Electrónica», p. 100.

[3] A. M. Vélez-Pereira, E. L. Vergara-Vásquez, W. D. Barraza-Coronell, y D. C. Agudelo-Yepes, «Evaluación de un modelo estadístico para estimar la radiación solar en Magdalena, Colombia», *TecnoLógicas*, vol. 18, n.º 35, p. 35, ago. 2015, doi: 10.22430/22565337.196.

[4] C. Voyant *et al.*, «Machine learning methods for solar radiation forecasting: A review», *Renew. Energy*, vol. 105, pp. 569-582, may 2017, doi: 10.1016/j.renene.2016.12.095.

[5] I. Samson, R. Echarri, S. Vera, A. Sartarelli, y E. Cyrulies, «Medición de la radiación solar en Santo Domingo», *Cienc. Soc.*, vol. 35, n.º 4, pp. 555-565, dic. 2010, doi: 10.22206/cys.2010.v35i4.pp555-565.

[6] J. Wright, «MEDICIÓN Y PREDICCIÓN DE LA RADIACIÓN SOLAR GLO- BAL ULTRAVIOLETA (0.295-0.385 μm) EN CONDICIONES DE CIELOS CLAROS Y SIN NUBES», p. 11.

[7] J. W. Gilmore, «MEDICIÓN Y PREDICCIÓN DE LA RADIACIÓN SOLAR GLOBAL UV-B BAJO CIELOS CLAROS Y SIN NUBES», p. 11, 2010.

[8] M. Aguiar, «Modelo predictivo de radiación solar mediante técnicas de machine learning: aplicación a la isla de Gran Canaria», p. 230, 2015.

[9] A. G.-T. Chávez, Y. C. Huerta, L. M. M. Manilla, D. Fernández, y A. F. G.-T. Rojas, «PREDICCIÓN ESPACIAL DE CARBONO ORGÁNICO EDÁFICO SUPERFICIAL EN ZONAS FORESTALES MEDIANTE ANÁLISIS DIGITAL DE TERRENO Y SIG: USO DE LA RADIACIÓN SOLAR POTENCIAL», p. 8.

[10] F. V. G. Corea, «Predicción espacio-temporal de la irradiancia solar global a corto plazo en España mediante geoestadística y redes neuronales artificiales», p. 169.

[11] S. Sayago, M. Bocco, G. Ovando, y E. Willington, «RADIACIÓN SOLAR HORARIA: MODELOS DE ESTIMACIÓN A PARTIR DE VARIABLES METEOROLÓGICAS BÁSICAS», p. 8.

[12] E. D. Obando, S. X. Carvajal, y J. Pineda Agudelo, «Solar Radiation Prediction Using Machine Learning Techniques: A Review», *IEEE Lat. Am. Trans.*, vol. 17, n.º 04, pp. 684-697, abr. 2019, doi: 10.1109/TLA.2019.8891934.

[13] L. C. Ruiz Cárdenas, D. Amaya Hurtado, y R. Jiménez Moreno, «Predicción de radiación solar mediante deep belief network», *Rev. Tecnura*, vol. 20, n.º 47, p. 39, feb. 2016, doi: 10.14483/udistrital.jour.tecnura.2016.1.a03.

[14] S. Fan, C. Mao, J. Zhang, y L. Chen, «Forecasting Electricity Demand by Hybrid Machine Learning Model», en *Neural Information Processing*, vol. 4233, I. King, J. Wang, L.-W. Chan, y D. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 952-963. doi: 10.1007/11893257_105.

[15] G. Rubio Flores, «Modelos avanzados de inteligencia computacional para aproximación funcional y predicción de series temporales en arquitecturas paralelas», Editorial de la Universidad de Granada, Granada, 2010.

[16] N. Mora, «Predicción a corto plazo de radiación solar Short term solar radiation prediction». 2019.

[17] J. R. Viloria, *Estudios de viabilidad de instalaciones solares. Determinación del potencial solar*. Editorial Paraninfo, 2012.

[18] J. J. García-Badell, *Cálculo de la energía solar*. IGME, 1983.

[19] «Definición de Machine Learning», *Aprende IA*, mar. 22, 2018. https://aprendeia.com/definicion-de-machine-learning/ (accedido sep. 12, 2021).

[20] V. V. J. Antonio, G. A. Julio, P. R. Francisco, y B. P. Mauricio, *Métodos de Data Science aplicados a la Economía y a la Dirección y Administración de Empresas*. Editorial UNED, 2019.

[21] A. Pulido, *Guía para usuarios de predicciones económicas*. ECOBOOK, 2006.

[22] M. R. Arahal, M. B. Soria, y F. R. Díaz, *Técnicas de predicción con aplicaciones en Ingeniería*. Universidad de Sevilla, 2006.

[23] S. T. Porras y E. M. Moreno, *Modelos Neuronales Aplicados en Economía: Casos Prácticos mediante Mathematica / Neural Networks*. Addlink Software Científico, 2013.

[24] R. F. López y J. M. F. Fernández, *Las Redes Neuronales Artificiales*. Netbiblo, 2008.

[25] A. Sabharwal y B. Selman, «Artifical_Intelligence», *Artif. Intell.*, vol. 175, n.º 5-6, pp. 935-937, abr. 2011, doi: 10.1016/j.artint.2011.01.005.

[26] C. S. Prat, T. M. Madhyastha, M. J. Mottarella, y C.-H. Kuo, «Relating Natural Language Aptitude to Individual Differences in Learning Programming Languages», *Sci. Rep.*, vol. 10, n.º 1, p. 3817, dic. 2020, doi: 10.1038/s41598-020-60661-8.

[27] F. and A. O. of the U. Nations, *Evapotranspiracion Del Cultivo: Guias Para Determinacion Los Requerimientos De Agua De Los Cultivos*. Food & Agriculture Org., 2006.

[28] RainWise, «Rainwise_mk-iii_spec». 2014. [En línea]. Disponible en: https://www.fondriest.com/pdf/rainwise_mk-iii_spec.pdf

[29] A. PETITJEAN y R. LEMESLE, *Windows PowerShell: Los fundamentos del lenguaje*. Ediciones ENI, 2015.

[30] S. C. Antonio, *Gestión de la información web usando Python*. Editorial UOC, 2017.

[31] W. Vance, *CIENCIA DE DATOS: Métodos y estrategias avanzados para aprender ciencia de datos para empresas*. joiningthedotstv.

[32] J. C. Bautista, *micro:bit y Python (Edici—n en Blanco y Negro)*. Lulu.com, 2018.

[33] G. D. Bocanegra, *Didáctica y aplicación de la administración de operaciones contaduría y administración*. IMCP, 2016.

[34] L. J. Gitman, *Principios de administración financiera*. Pearson Educación, 2003.

[35] «Python | Pandas dataframe.corr()», *GeeksforGeeks*, nov. 20, 2018. https://www.geeksforgeeks.org/python-pandas-dataframe-corr/ (accedido jul. 02, 2021).

[36] «Amanecer y atardecer en Ecuador», *DatosMundial.com*. https://www.datosmundial.com/america/ecuador/puesta-del-sol.php (accedido jul. 08, 2021).

[37] R.- ASALE y RAE, «amanecer | Diccionario de la lengua española», *«Diccionario de la lengua española» - Edición del Tricentenario*. https://dle.rae.es/amanecer (accedido jul. 08, 2021).

[38] «Salida y puesta del sol en Latacunga, Cotopaxi, Ecuador». https://www.citipedia.info/city/sunriseandsunset/Ecuador__Cotopaxi_Latacunga_id_3654870_lang_es (accedido jul. 08, 2021).

[39] R.- ASALE y RAE, «anochecer | Diccionario de la lengua española», *«Diccionario de la lengua española» - Edición del Tricentenario*. https://dle.rae.es/anochecer (accedido jul. 08, 2021).

[40] C. S. JULIÁN, *Configuración de instalaciones solares fotovoltaicas*. Ediciones Paraninfo, S.A., 2016.

[41] «Python | Pandas dataframe.resample()», *GeeksforGeeks*, nov. 20, 2018. https://www.geeksforgeeks.org/python-pandas-dataframe-resample/ (accedido jul. 08, 2021).

[42] J. Brownlee, «How to Use StandardScaler and MinMaxScaler Transforms in Python», *Machine Learning Mastery*, jun. 09, 2020. https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/ (accedido jul. 09, 2021).

[43] J. Brownlee, *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python*. Machine Learning Mastery, 2020.

[44] «About Feature Scaling and Normalization», *Dr. Sebastian Raschka*, jul. 11, 2014. https://sebastianraschka.com/Articles/2014_about_feature_scaling.html (accedido jul. 09, 2021).

[45] «Train and Test Set in Python Machine Learning - How to Split», *DataFlair*, ago. 06, 2018. https://data-flair.training/blogs/train-test-set-in-python-ml/ (accedido jul. 09, 2021).

[46] J. I. S. Martínez, «Estimación del esfuerzo de proyectos de software con algoritmos de aprendizaje de máquinas», *2019*, p. 23.

[47] F. G. Hernando, «Análisis de líneas de costa con redes neuronales LSTM», p. 57.

[48] codificandobits, *Redes Neuronales Recurrentes: EXPLICACIÓN DETALLADA*. Accedido: jul. 09, 2021. [En línea Video]. Disponible en: https://www.youtube.com/watch?v=hB4XYst_t-I

[49] C. Antona, «HERRAMIENTAS MODERNAS EN REDES NEURONALES: LA LIBRERIA KERAS». 2017.

[50] kexugit, «Serie de pruebas: introducción a las celdas de LSTM mediante C#». https://docs.microsoft.com/es-es/archive/msdn-magazine/2018/april/test-run-understanding-lstm-cells-using-csharp (accedido oct. 04, 2021).

[51] «¿Qué son las Redes LSTM?», jul. 20, 2019. https://www.codificandobits.com/blog/redes-lstm/ (accedido jul. 10, 2021).

[52] «Sequential - Keras Documentation». https://faroit.com/keras-docs/1.1.0/models/sequential/ (accedido jul. 12, 2021).

[53] R. F. López y J. M. F. Fernández, *Las Redes Neuronales Artificiales*. Netbiblo, 2008.

[54] M. J. Evans y J. S. Rosenthal, *Probabilidad y estadística*. Reverte, 2005.

[55] J. Heizer y B. Render, *Principios de administración de operaciones*. Pearson Educación, 2004.

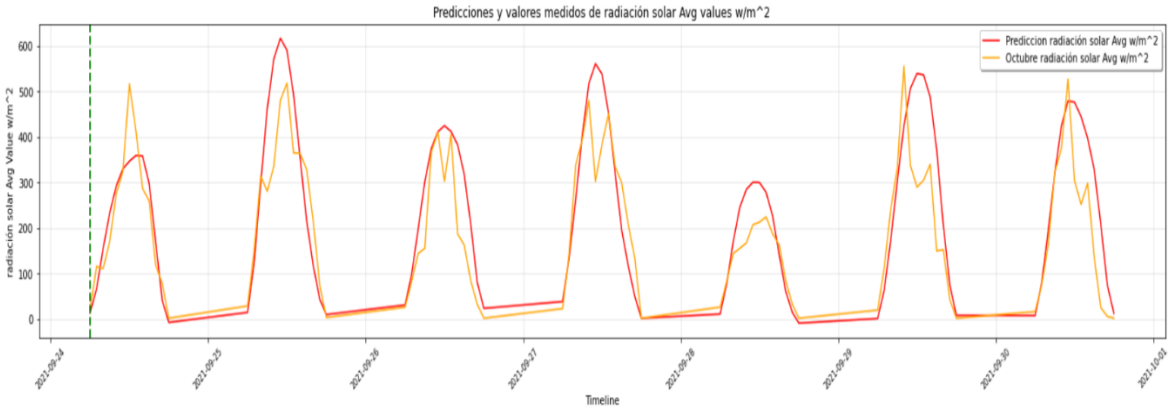
ANEXO 1. Data original utilizada

Time	Temp Avg	Temp Low	Temp High	Heat Index	Wind Chill	Temp (Day) Low	Temp (Day) High	Hum Avg	Hum Low	Hum High	Dew Point	Hum (Day) Low	Hum (Day) High	Baro Avg	Baro Low	Baro High	Baro (Day) Low	Baro (Day) High	Windspeed	Wind Direction	Gust	Gust Direction	Gust (Day) Max	Di
2019-01-01 00:00:00	7.5	7.5	7.5	7.5	7.5	7.3	16.7	91.0	92.0	92.0	6.1	47.0	96.0	720.0	719.9	719.9	717.1	720.8	1.9	67.0	3.5	270.0	48.4	
2019-01-01 00:01:00	7.5	7.5	7.5	7.5	7.5	7.5	7.5	92.0	92.0	92.0	6.3	92.0	92.0	719.9	719.9	719.9	719.9	719.9	5.3	67.0	7.9	67.0	7.9	
2019-01-01 00:02:00	7.5	7.5	7.5	7.5	7.5	7.5	7.5	91.0	92.0	92.0	6.1	92.0	92.0	719.9	719.9	719.9	719.9	719.9	6.6	67.0	7.9	67.0	7.9	
2019-01-01 00:03:00	7.5	7.5	7.5	7.5	7.5	7.5	7.5	92.0	92.0	92.0	6.3	92.0	92.0	719.8	719.9	719.7	719.7	719.9	4.0	67.0	5.6	67.0	7.9	
2019-01-01 00:04:00	7.5	7.5	7.5	7.5	7.5	7.5	7.5	91.0	92.0	92.0	6.1	92.0	92.0	719.8	719.9	719.7	719.7	719.9	3.2	67.0	5.1	67.0	7.9	
2019-01-01 00:05:00	7.4	7.4	7.5	7.4	7.4	7.4	7.5	92.0	92.0	92.0	6.2	92.0	92.0	719.9	719.9	719.9	719.7	719.9	3.5	54.0	5.1	67.0	7.9	
2019-01-01 00:06:00	7.4	7.4	7.4	7.4	7.4	7.4	7.5	92.0	92.0	92.0	6.2	92.0	92.0	719.8	719.9	719.7	719.7	719.9	3.7	45.0	4.8	67.0	7.9	
2019-01-01 00:07:00	7.4	7.4	7.4	7.4	7.4	7.4	7.5	91.0	92.0	92.0	6.0	92.0	92.0	719.7	719.7	719.7	719.7	719.9	1.4	45.0	3.4	67.0	7.9	
2019-01-01 00:08:00	7.4	7.4	7.4	7.4	7.4	7.4	7.5	91.0	92.0	92.0	6.0	92.0	92.0	719.7	719.9	719.7	719.7	719.9	2.6	45.0	4.2	67.0	7.9	
2019-01-01 00:09:00	7.4	7.4	7.4	7.4	7.4	7.4	7.5	92.0	92.0	92.0	6.2	92.0	92.0	719.8	719.9	719.7	719.7	719.9	1.3	45.0	2.9	67.0	7.9	
2019-01-01 00:10:00	7.4	7.3	7.4	7.4	7.4	7.3	7.5	91.0	92.0	92.0	6.0	92.0	92.0	719.8	719.9	719.7	719.7	719.9	0.0	45.0	2.1	67.0	7.9	
2019-01-01 00:11:00	7.3	7.3	7.3	7.3	7.3	7.3	7.5	92.0	92.0	92.0	6.1	92.0	92.0	719.8	719.9	719.7	719.7	719.9	0.0	45.0	0.0	67.0	7.9	
2019-01-01 00:12:00	7.3	7.3	7.3	7.3	7.3	7.3	7.5	91.0	92.0	92.0	5.9	92.0	92.0	719.7	719.7	719.7	719.7	719.9	0.0	45.0	0.0	67.0	7.9	
2019-01-01	7.3	7.3	7.3	7.3	7.3	7.3	7.5	92.0	92.0	92.0	6.1	92.0	92.0	719.7	719.7	719.7	719.7	719.9	0.0	45.0	0.0	67.0	7.9	

ANEXO 2. Datos depurados

Time	Solar Radiation Avg	Baro Avg	Hum Avg	Windspeed	Temp Avg
2019-01-01 00:00:00	0.000000	719.498333	92.500000	7.410000	7.130000
2019-01-01 01:00:00	0.000000	718.926667	94.600000	12.395000	7.191667
2019-01-01 02:00:00	0.000000	718.576667	93.033333	10.530000	6.936667
2019-01-01 03:00:00	0.000000	718.570000	94.850000	8.930000	5.788333
2019-01-01 04:00:00	0.000000	718.993333	95.216667	6.545000	5.445000
2019-01-01 05:00:00	0.000000	719.531667	93.883333	6.203333	5.026667
2019-01-01 06:00:00	36.216667	720.191667	94.666667	4.716667	3.851667
2019-01-01 07:00:00	300.266667	720.963333	93.116667	4.493333	5.841667
2019-01-01 08:00:00	494.116667	721.010000	74.550000	17.261667	10.513333
2019-01-01 09:00:00	735.433333	720.671667	61.833333	23.195000	13.391667
2019-01-01 10:00:00	543.750000	720.131667	56.833333	22.600000	14.596667
2019-01-01 11:00:00	330.250000	719.533333	55.416667	22.483333	15.038333
2019-01-01 12:00:00	411.566667	718.780000	51.383333	22.663333	15.856667
2019-01-01 13:00:00	535.866667	717.941667	43.300000	22.551667	17.200000
2019-01-01 14:00:00	292.550000	717.378333	43.883333	21.831667	16.785000
2019-01-01 15:00:00	201.966667	717.245000	43.300000	21.170000	16.343333
2019-01-01 16:00:00	94.750000	717.520000	48.766667	19.196667	15.445000
2019-01-01 17:00:00	63.100000	718.148333	52.350000	18.273333	14.438333
2019-01-01 18:00:00	5.200000	718.790000	57.366667	13.576667	13.066667
2019-01-01 19:00:00	0.000000	719.420000	65.716667	12.688333	11.610000
2019-01-01 20:00:00	0.000000	720.145000	78.183333	15.058333	10.681667
2019-01-01 21:00:00	0.000000	720.696667	82.483333	12.073333	10.306667
2019-01-01 22:00:00	0.000000	720.825000	86.600000	7.980000	9.755000
2019-01-01 23:00:00	0.000000	720.713333	89.900000	7.206667	9.426667
2019-01-02 00:00:00	0.000000	720.071667	91.233333	8.875000	9.146667
2019-01-02 01:00:00	0.000000	719.288333	93.733333	10.910000	8.693333
2019-01-02 02:00:00	0.000000	718.591667	93.833333	14.251667	8.196667
2019-01-02 03:00:00	0.000000	718.526667	95.233333	14.073333	7.805000
2019-01-02 04:00:00	0.000000	718.760000	95.650000	15.821667	8.071667
2019-01-02 05:00:00	0.000000	719.321667	95.000000	15.453333	8.416667
2019-01-02 06:00:00	20.483333	719.823333	93.950000	15.801667	8.591667
2019-01-02 07:00:00	206.350000	720.598333	88.266667	14.680000	9.926667
2019-01-02 08:00:00	246.600000	720.773333	80.316667	15.640000	11.488333
2019-01-02 09:00:00	473.250000	720.785000	73.900000	19.035000	13.186667
2019-01-02 10:00:00	457.583333	720.340000	65.183333	22.160000	14.570000

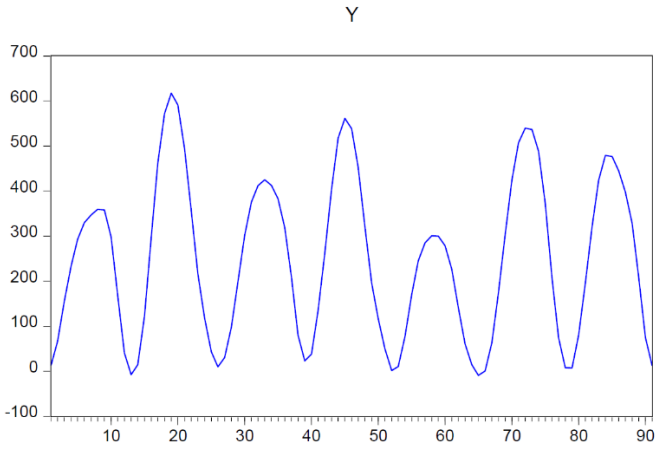
ANEXO 3. Predicción de los 7 días consecutivos



ANEXO 4. Variables de respuestas

1RA CAPA	2DA CAPA	DROPOUT	ADAM	EPOCHS	SIZE	LOSS	VAL_LOSS
64	32	0.25	0.00045	162	150	0.1459	0.2286
64	32	0.25	0.0004	180	150	0.2341	0.2935
64	32	0.25	0.004	103	150	0.8214	0.7363
64	32	0.25	0.04	25	150	0.8304	0.7769
64	32	0.25	0.4	24	150	1.11	1.16
64	32	0.25	0.00045	192	300	01827	0.2542
64	32	0.25	0.0004	258	300	0.205	0.2764
64	32	0.25	0.004	74	300	0.8341	0.7833
64	32	0.25	0.04	39	300	0.8199	0.7877

ANEXO 5. Horizonte de predicción y de control.



Funciones utilizadas:

La función de producción estimada: modelo matemático

==========

Y = 211.78240121 + [AR(1)=0.894549440152,MA(12)=0.665648510997,UNCOND,ESTSMPL="1 91"]

La función de producción estimada:

Coeficientes:

==========

Y = -22.2211564235 + 0.729581963776*Y(-1) + 0.371512865707*Y(-12)

Horizonte de Predicción: f (t+n) ∀ n= 13 frecuencias horarias Δμ (t)

Horizonte de control: MPC = 4 (ciclos + tendencia + estacionalidad + irregularidad)

Factor de ponderación para seguimiento de la trayectoria de la variación solar:

$\Delta^0 Yt = \alpha + \varphi * \mathrm{AR}(1) + \Theta * \mathrm{MA}(12) + \varepsilon t$

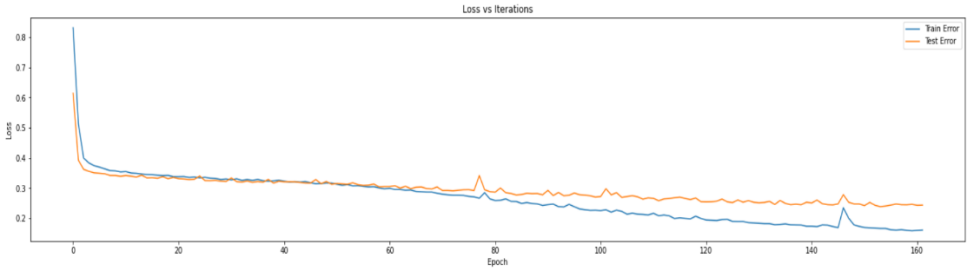
$\Delta^0 Yt = \alpha + 0.89 * \mathrm{AR}(1) + 0.66 * \mathrm{MA}(12) + \varepsilon t$

AR (1) = Proceso autorregresivo de primer orden

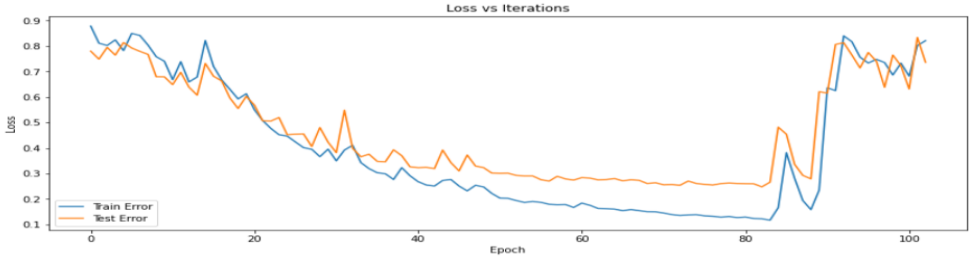
MA(12) = Proceso de promedio móvil de orden doce estacional

ANEXO 6. Función de pérdida utilizada

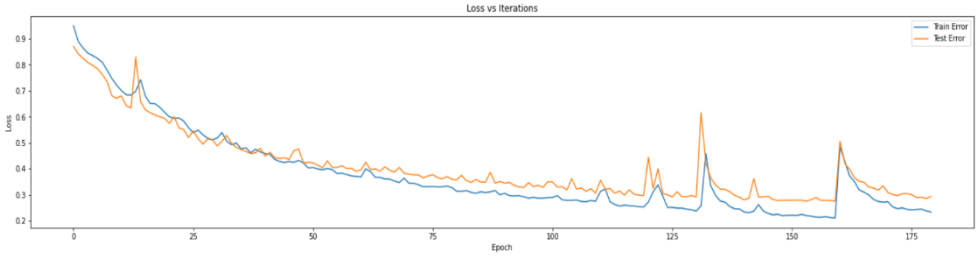
Optimizador Adams con pasos de: 0.00045



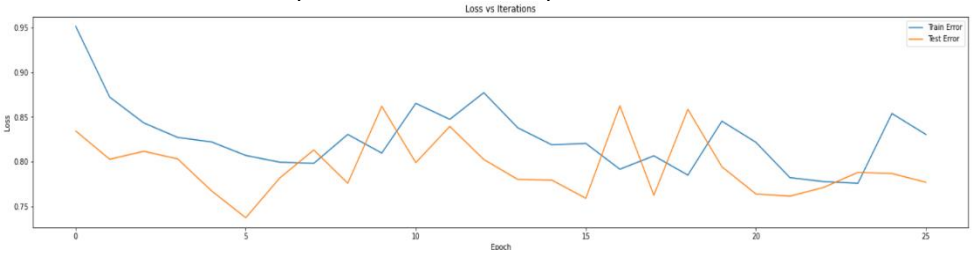
Optimizador Adams con pasos de: 0.004



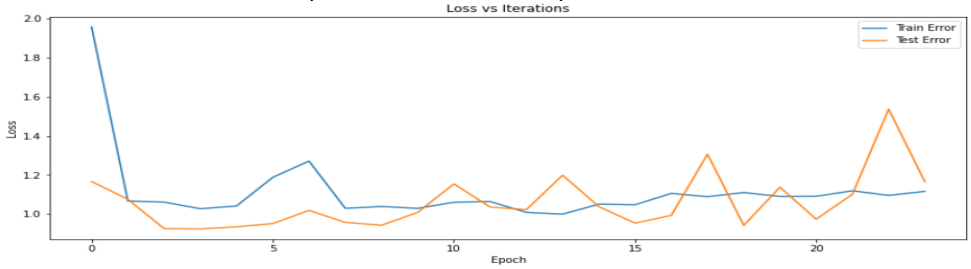
Optimizador Adams con pasos de: 0.0004



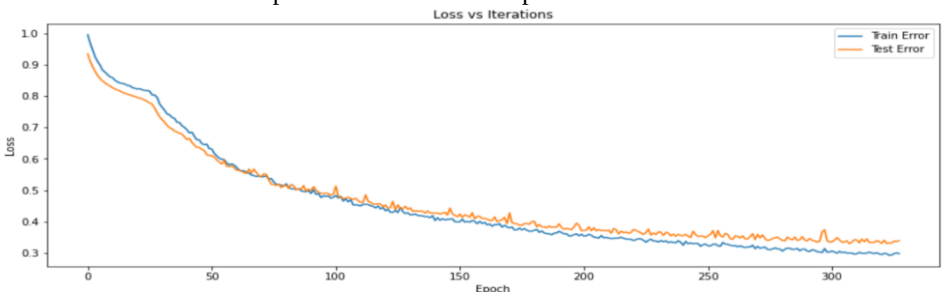
Optimizador Adams con pasos de: 0.04



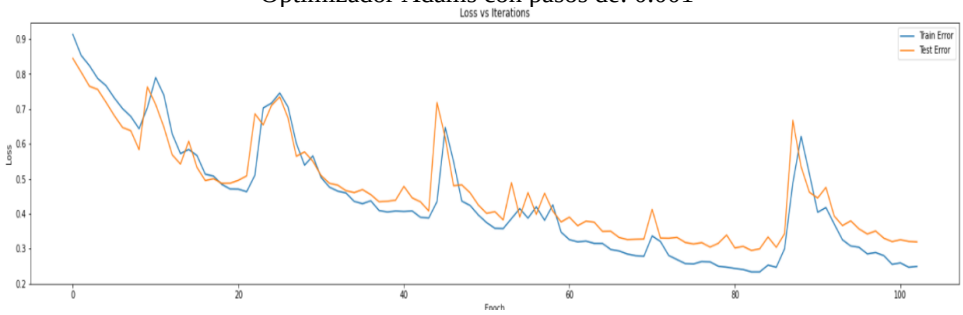
Optimizador Adams con pasos de: 0.4



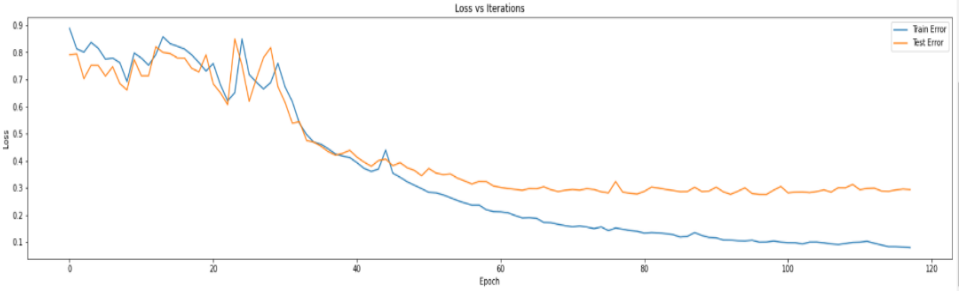
Optimizador Adams con pasos de: 0.0001



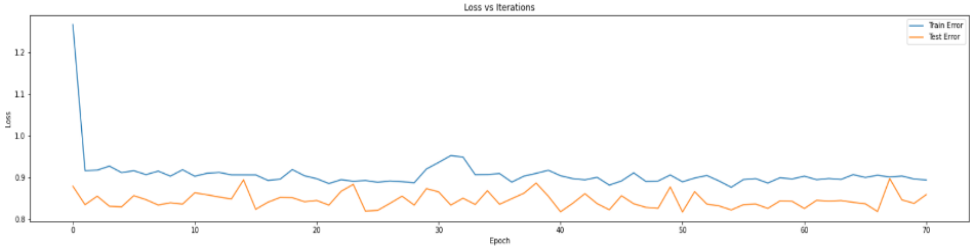
Optimizador Adams con pasos de: 0.001



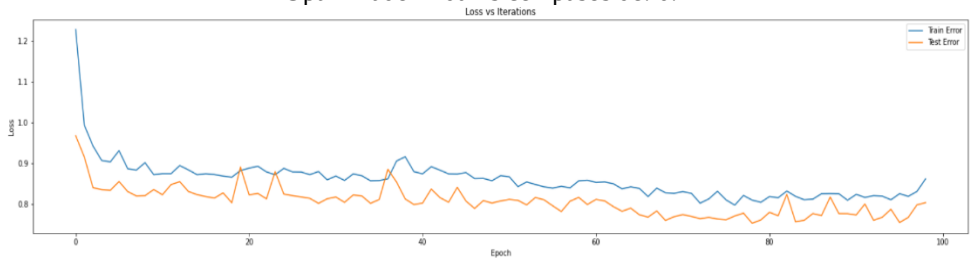
Optimizador Adams con pasos de: 0.01



Optimizador Adams con pasos de: 0.11



Optimizador Adams con pasos de: 0.1



ANEXO 7. Algoritmo desarrollado

```
#Importación de las librerías básicas y sus complementos
import pandas as pd              # Lectura y procesamiento del Dataframe
import numpy as np               # Realizar ecuaciones matemáticas
import matplotlib.pyplot as plt  # Visualización de gráficas
import tensorflow as tf          # Aplicacion de tensores junto con keras
import datetime as dt            # Trabajar con fechas
from datetime import datetime    # Trabajar con fechas
import tensorflow.keras as keras # Aplicación de tensores junto con keras
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint, TensorBoard
%matplotlib inline
from pandas import DataFrame     # Procesamiento del Dataframe
from sklearn.preprocessing import MinMaxScaler # Escalar los valores
```

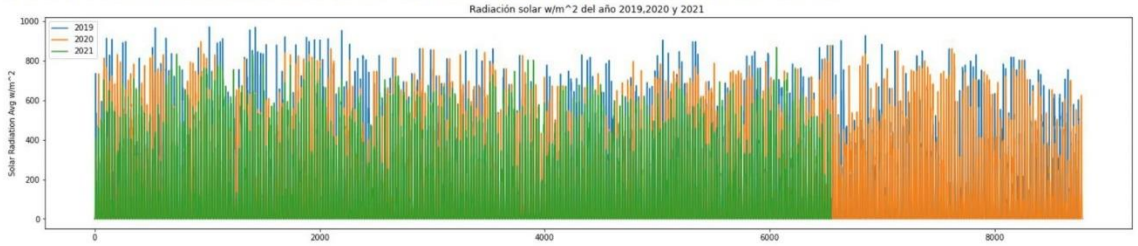
```
from google.colab import drive   # Acceso a los archivos del drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

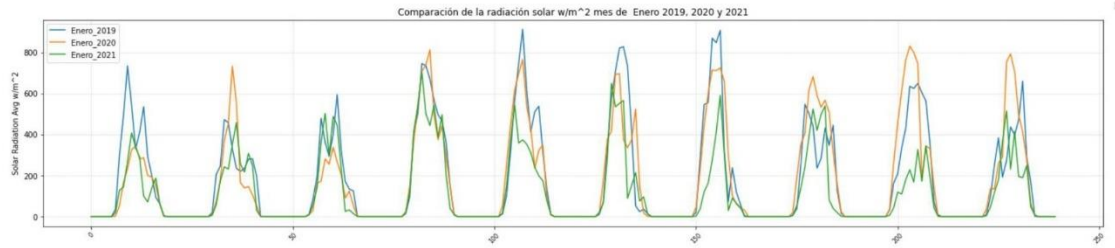
```
# Dataframe con registros depurados del año 2019
df2019_depurado=pd.read_csv('/content/drive/My Drive/Colab Notebooks/MEDICIONES_MES/AÑ0_2019_depurado.csv',parse_dates=['Time'],index_col=['Time'])
# Dataframe con registros depurados del año 2020
df2020_depurado=pd.read_csv('/content/drive/My Drive/Colab Notebooks/MEDICIONES_MES/AÑ0_2020_depurado.csv',parse_dates=['Time'],index_col=['Time'])
# Dataframe con registros depurados del año 2021
df2021_depurado=pd.read_csv('/content/drive/My Drive/Colab Notebooks/MEDICIONES_MES/AÑ0_2021_depurado.csv',parse_dates=['Time'],index_col=['Time'])
```

```
df2019=df2019_depurado                          # Designación de variable df2019
df2019_c=df2019_depurado                        # Designación de variable df2019_c
df2020=df2020_depurado                          # Designación de variable df2020
df2020_c=df2020_depurado                        # Designación de variable df2020_c
df2021=df2021_depurado                          # Designación de variable df2021
df2021_c=df2021_depurado                        # Designación de variable df2021_c
df2019 =df2019.resample('h').mean()             # Represntación en horas df2019
df2020 =df2020.resample('h').mean()             # Represntación en horas df2020
df2021 =df2021.resample('h').mean()             # Represntación en horas df2021
df2019 = df2019[["Solar Radiation Avg"]]        # Selección radiación solar del 2019
df2020 = df2020[["Solar Radiation Avg"]]        # Selección radiación solar del 2020
df2021 = df2021[["Solar Radiation Avg"]]        # Selección radiación solar del 2021
plt.rcParams['figure.figsize'] = 25, 5          # Dimensiones d ela imagen
plt.plot(df2019['2019'].values,label="2019")    # Gráfica 2019
plt.plot(df2020['2020'].values,label="2020")    # Gráfica 2020
plt.plot(df2021['2021'].values,label="2021")    # Gráfica 2021
plt.ylabel("Solar Radiation Avg w/m^2")         # Leyenda eje y
plt.legend(loc="upper left")                    # Ubicación de la leyenda
plt.title('Radiación solar w/m^2 del año 2019,2020 y 2021 ', family='Arial', fontsize=12)
```

```
Text(0.5, 1.0, 'Radiación solar w/m^2 del año 2019,2020 y 2021 ')findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
```




```
df2019 = df2019[["Solar Radiation Avg"]]       #  Selección de la radiación solar 2019
df2020 = df2020[["Solar Radiation Avg"]]       #  Selección de la radiación solar 2020
df2021 = df2021[["Solar Radiation Avg"]]       #  Selección de la radiación solar 2021
enero2019 = df2019['2019-01-01':'2019-01-10']  #  Selección mes enero radiación solar 2019
plt.plot(enero2019.values,label="Enero_2019")  #  Gráfica mes enero radiación solar 2019
enero2020= df2020['2020-01-01':'2020-01-10']   #  Selección mes enero radiación solar 2020
plt.plot(enero2020.values,label="Enero_2020")  #  Gráfica mes enero radiación solar 2020
enero2021= df2021['2021-01-01':'2021-01-10']   #  Selección mes enero radiación solar 2021
plt.plot(enero2021.values,label="Enero_2021")  #  Gráfica mes enero radiación solar 2021
plt.ylabel("Solar Radiation Avg w/m^2")        #  Leyenda eje y
plt.legend(loc="upper left")                   #  Ubicacion de leyenda
plt.grid(which='major', color='#cccccc', alpha=0.5) #  Regilla
plt.title('Comparación de la radiación solar w/m^2 mes de  Enero 2019, 2020 y 2021', family='Arial', fontsize=12)
plt.xticks(rotation=45, fontsize=8)            #  rotacion texto eje x
plt.show()
```



```
df2019_c = df2019_c[["Solar Radiation Avg","Baro Avg","Hum Avg","Windspeed","Temp Avg"]] # selección de variables con alta correlación
df2019_c.corr()                                                                         # Correlación del dataframe 2019
df2020_c = df2020_c[["Solar Radiation Avg","Baro Avg","Hum Avg","Windspeed","Temp Avg"]] # selección de variables con alta correlación
df2020_c.corr()                                                                         # Correlación del dataframe 2020
df2021_c = df2021_c[["Solar Radiation Avg","Baro Avg","Hum Avg","Windspeed","Temp Avg"]] # selección de variables con alta correlación
df2021_c.corr()                                                                         # Correlación del dataframe 2021
df2019 = df2019_depurado['2019-01-01':'2019-12-31']  # Dataframe del 2019
df2020 = df2020_depurado['2020-01-01':'2020-12-31']  # Dataframe del 2020
df2021 = df2021_depurado['2021-01-01':'2021-08-31']  # Dataframe del 2021
frames = [df2019,df2020,df2021]    # Selección de los Dataframe para concatenar
df = pd.concat(frames)             # Concatenación de los Dataframe
# Selección de las variables identificadas con mayor relación en correlación
df = df[["Solar Radiation Avg","Baro Avg","Hum Avg","Windspeed","Temp Avg"]]
df.tail()   #visualización de los ultimos datos
df.shape    #visualización de la estructura datos
df.info()   #información de los datos cargados
df.columns  #número de columnas de los datos guardados
# Eliminación de valores de radiación menores de 0
df = df.drop(df[df['Solar Radiation Avg']<1].index)
df.tail(3)  #visualización de los ultimos datos
#Copia de los datos cargados y luego eliminación los valores nan o nulos para mejorar la eficiencia del algoritmo
dataset_train_actual = df.copy()             #Copia del dataframe actual a df
dataset_train_actual.isnull().sum()          #Resumen del dataframe
dataset_train_actual.dropna( inplace=True )  #Eliminación de valores Nan y Nulos
#processamiento de datos Nan
dataset_train_actual = dataset_train_actual.fillna(dataset_train_actual.mean())
#procesamiento de los datos con el index
dataset_train_actual = dataset_train_actual.reset_index()
dataset_train_actual.isnull().sum() #visualización para comprobar datos nulos
dataset_train_actual.shape   # Imprimir las dimensiones del Dataframe
dataset_train_actual.head()  # visualización de los datos despues del preprocesamiento
dataset_train_actual.info()  # Impresión de la información
dataset_train_actual.shape   # Imprimir las dimensiones del Dataframe
#usar el valor time de fechas para colocarlo en el index luego del procesamiento
# Fijación de la columna de TIME como índice
dataset_train_timeindex = dataset_train_actual.set_index('Time')
# Copia de los valores a dataset_train
dataset_train = dataset_train_actual.copy()
# Lista de las variables seleccionado despues de la columna de fechas
cols = list(dataset_train)[1:6]
# Creación de las fechas de 6:00 a 18:00 en rango de una hora
datelist_train = pd.date_range(start=pd.Timestamp('06:00'), end=pd.Timestamp('18:00'), freq='1H')
# Ordenamiento en de las fechas en una lista
datelist_train = list(dataset_train['Time'])
# Creanción de la columna de fechas
datelist_train = [date for date in datelist_train]
# Impresión de las características
#print('Training set shape == {}'.format(dataset_train.shape))
#print('All timestamps == {}'.format(len(datelist_train)))
#print('Featured selected: {}'.format(cols))
```

```
datelist_train = [date for date in datelist_train]
# Impresión de las características
#print('Training set shape == {}'.format(dataset_train.shape))
#print('All timestamps == {}'.format(len(datelist_train)))
#print('Featured selected: {}'.format(cols))
```

```
# Procesamiento de los datos de entrenamiento
dataset_train = dataset_train[cols].astype(str)
for i in cols:
    for j in range(0, len(dataset_train)):
        dataset_train[i][j] = dataset_train[i][j].replace(',', '')
dataset_train = dataset_train.astype(float)
# Declaración de los datos predictores
training_set = dataset_train.values
# Impresión de las características
#print('Shape of training set == {}.'.format(training_set.shape))
```

```
# Escalamiento de los valores con StandardScaler
from sklearn.preprocessing import StandardScaler
# Desiganción de StandardScaler como sc
sc = StandardScaler()
# Escalamiento de training_set
training_set_scaled = sc.fit_transform(training_set)
# Desiganción de StandardScaler() como sc_predict
sc_predict = StandardScaler()
# Escalamiento de datos predictores
sc_predict.fit_transform(training_set[:, 0:1])
```

```
# Separación de datos de entrenamiento y prueba
from sklearn.model_selection import train_test_split
X_train = []     # Variable X de datos de entrenamiento
y_train = []     # Variable Y de datos de entrenamiento
n_future = 91    # Número de horas a predecir en el futuro
n_past = 180     # Número de horas del pasado a revisar para predecir
# Llena los valores escalados en las variables de entremaniento y prueba
for i in range(n_past, len(training_set_scaled) - n_future +1):
    X_train.append(training_set_scaled[i - n_past:i, 0:dataset_train.shape[1]])
    y_train.append(training_set_scaled[i + n_future - 1:i + n_future, 0])
# Estructura de los datos de entremaniento y prueba con 20% de separación
X_train,xtest, y_train, ytest = train_test_split( np.array(X_train), np.array(y_train),test_size=0.2,random_state=0)
# Impresión de las características
#print('X_train shape == {}.'.format(X_train.shape))
#print('Xtest shape == {}.'.format(xtest.shape))
#print('y_train shape == {}.'.format(y_train.shape))
#print('ytest shape == {}.'.format(ytest.shape))
```

```
# Importación de librerías para crear la red neuronal
from keras.models import Sequential   # Usar datos secuenciales
from keras.layers import Dense        # Procesamiento de varias capas
from keras.layers import LSTM         # Neurona con memoria
from keras.layers import Dropout      # Evita que memorice los valores
from tensorflow.keras.optimizers import Adam  # Optimizador del algoritmo
```

```
def build_model():
  model = Sequential()
 # Capa de entrada con 64 neuronas
  model.add(LSTM(units=64, return_sequences=True, input_shape=(n_past, df.shape[1])))
 # Capa oculta con 32 neuronas
  model.add(LSTM(units=32, return_sequences=False))
 # Dropout con 25% para evitar la memorizacion de datos
  model.add(Dropout(0.25))
 # Capa de salida con una sola salida
  model.add(Dense(units=1, activation='linear'))
 # Optimizador del algoritmo y perdidas loss
  model.compile(optimizer = Adam(learning_rate=0.00045), loss='mean_squared_error')
  return model
```

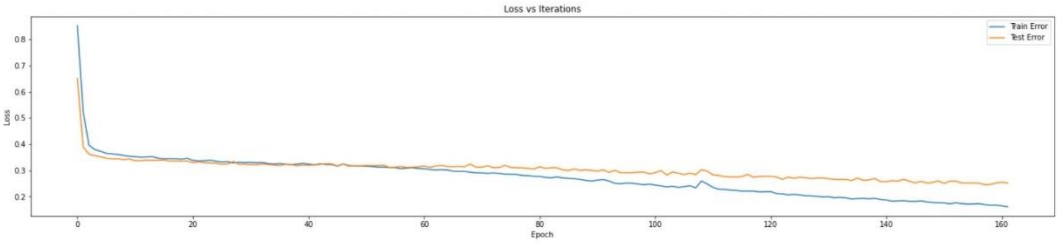
```
model=build_model()   # declaración de función
model.summary()       # Resumen del modelo
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 180, 64)           17920
_________________________________________________________________
lstm_1 (LSTM)                (None, 32)                12416
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 33
=================================================================
Total params: 30,369
Trainable params: 30,369
Non-trainable params: 0
```

```
# Callbacks configurado para que cuando repetidamente empieze a memorizar finalice el aprendizaje
es=keras.callbacks.EarlyStopping(monitor='val_loss',mode='min',patience=15)
# Funcion de la neurona
model=build_model()
# Entrenamiento de la neurona con 500 epocas, 20% de validacion y un batch_size de 150
history = model.fit(X_train, y_train, shuffle=True, epochs=162, validation_split=0.2, verbose=1,batch_size=150,callbacks=[es])
```

```
Epoch 156/162
53/53 [==============================] - 17s 327ms/step - loss: 0.1716 - val_loss: 0.2517
Epoch 157/162
53/53 [==============================] - 18s 331ms/step - loss: 0.1728 - val_loss: 0.2509
Epoch 158/162
53/53 [==============================] - 17s 327ms/step - loss: 0.1693 - val_loss: 0.2453
Epoch 159/162
53/53 [==============================] - 17s 329ms/step - loss: 0.1668 - val_loss: 0.2463
Epoch 160/162
53/53 [==============================] - 17s 328ms/step - loss: 0.1668 - val_loss: 0.2518
Epoch 161/162
53/53 [==============================] - 17s 326ms/step - loss: 0.1643 - val_loss: 0.2544
Epoch 162/162
53/53 [==============================] - 17s 327ms/step - loss: 0.1607 - val_loss: 0.2515
```

```
plot_history(history) # Grafica de entrenamiento
#loss= pérdida de conjunto de entrenamiento
#val_loss= pérdida del conjunto de prueba
```



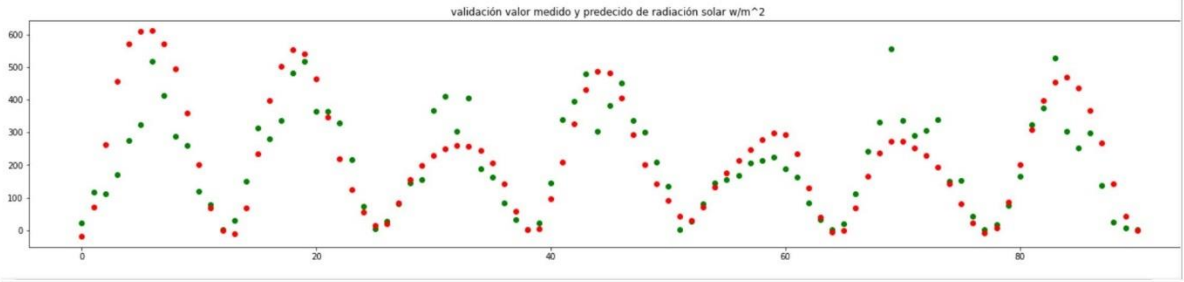

```
# INGRESO DEL MES PARA VERIFICAR LA PREDICCION
# Selección del rango de fechas que no ha visto el algoritmo
df2021 = df2021_depurado['2021-08-27':'2021-09-23']
```

```
# Generación de las fechas a partir del último registro
datelist_future = pd.date_range(datelist_ [-1], periods=182, freq='1h').tolist()
rangopredicion = range(182)          # Designación a rangopredicion un rago de 182 valores
list(rangopredicion)                 # Convertir a rango predicción en lista
fake = pd.DataFrame(rangopredicion) # Creación de las dimensiones
# Fijación de datelist como indice
temporal = pd.DataFrame(fake, columns=['Solar Radiation']).set_index(pd.Series(datelist_future))
# Rango de horas de 6:00 a 18:00
temporal = temporal.between_time('06:00:00', '18:00:00')
# Reseteo del indice y la primera es la de fechas
temporal = temporal.rename_axis('index').reset_index()
# Convertir en formato timestamp
temporal1 = temporal['index'].tolist()
del temporal1[0:3]
# Ordenar en en formato año, mes, dia
datelist_future_ = []
for this_timestamp in temporal1:
    datelist_future_.append(this_timestamp.date())
```

```
# Función para la conversion de fechas
def datetime_to_timestamp(x):
    '''
        x : a given datetime value (datetime.date)
    '''
    return datetime.strptime(x.strftime('%Y%m%d'), '%Y%m%d')
```

```
#Predicción de los valores posteriores
prediccion_futura = model.predict(X_[-d_pred:])
# Reescalamiento de los valores predecidos
y_pred_future = sc_predict.inverse_transform(prediccion_futura)
# Seleccion del indice las nuevas fechas
PREDICCION_2 = pd.DataFrame(y_pred_future, columns=['Solar Radiation Avg']).set_index(pd.Series(temporal1))
# Seleccion de los valores predecidos de la radación solar
week_predicted = pd.DataFrame(y_pred_future, columns=['Solar Radiation Avg'])
```

```
# VALIDACIÓN DE LOS DATOS PREDECIDOS CON LOS VALORES REALES MEDIDO CON LE ESTACIÓN
# Selección de registro medido por la estación para la comparación con lo predecido
MES_PRUEBA = df2021_depurado['2021-09-24' : '2021-09-30']
#  Selección de la radiación solar de todos los registros
MES_PRUEBA = MES_PRUEBA[["Solar Radiation Avg"]]
# Representación por cada hora de medición
MES_PRUEBA = MES_PRUEBA.resample('H').mean()
# Eliminación de los alores menores a cero
MES_PRUEBA = MES_PRUEBA.drop(MES_PRUEBA[MES_PRUEBA['Solar Radiation Avg']<1].index)
# Tratamiento de valores NAN
MES_PRUEBA = MES_PRUEBA[MES_PRUEBA['Solar Radiation Avg'].notna()]
# Grafica de validación
plt.scatter(range(len(MES_PRUEBA)),MES_PRUEBA,c='g',label='Valor medido')
plt.scatter(range(len(PREDICCION_2)),PREDICCION_2,c='r',label='Valor predecido')
plt.title('validación valor medido y predecido de radiación solar w/m^2 ')
plt.show()
```




```
# COMPARACION VALOR PREDECIDO CON VALOR MEDIDO DE LA ESTACION
# Valores de radiación predecidos
plt.plot(PREDICCION_2.index, PREDICCION_2['Solar Radiation Avg'], color='r', label='Prediccion radiación solar Avg w/m^2')
# Valores de radiación medidos estación
plt.plot(MES_PRUEBA.index,MES_PRUEBA['Solar Radiation Avg'], color='orange', label='Octubre radiación solar Avg w/m^2')
# Linea de inicio
plt.axvline(x = min(PREDICCION_2.index), color='green', linewidth=2, linestyle='--')
# Regilla
plt.grid(which='major', color='#cccccc', alpha=0.5)
# Leyenda
plt.legend(shadow=True)
# Titulo de grafica
plt.title('Predicciones y valores medidos de radiación solar Avg values w/m^2', family='Arial', fontsize=12)
# Estilo eje x
plt.xlabel('Timeline', family='Arial', fontsize=10)
# Estilo eje y
plt.ylabel('radiación solar Avg Value w/m^2', family='Arial', fontsize=10)
# Rotación letras eje x
plt.xticks(rotation=45, fontsize=8)
plt.show()
```