TONATIUH: Interfaz gráfica para planeación energética basada en algoritmos predictivos de radiación solar

Carla López - A00822301, Alejandro Salinas - A01282503, and Bernardo Barrera - A01194182

(Fecha: 2 de diciembre de 2020)

El proyecto TONATIUH busca planear el óptimo consumo diario de energía para las necesidades de una casa en promedio basándose en la energía que sale de un panel solar con las predicciones de la radiación que llega a éste, a partir de un modelo que utilice parámetros del tiempo como humedad, temperatura, velocidad y dirección del viento, presión y hora respectiva.

Keywords: Energía solar, Machine Learning, Planeación energética, Ánalisis de regresiones

I. RESUMEN EJECUTIVO

TONATIUH es una interfaz gráfica para la planeación energética basada en algoritmos predictivos de radiación solar. Toma su nombre del dios del Sol en la mitología nahua, siendo considerado por los mexicas como el líder del cielo. Está inspirado en el reto postulado por la incubadora de innovación de la NASA para la competencia de SpaceApps de 2017 denominada "You are my sunshine", el cual motiva a sus participantes a desarrollar un medio que le permita a las personas comprender la rendimiento de la energía producida por un panel solar.

Dado que el concurso antes mencionado está enfocado al proyecto HI-SEAS (por sus siglas en inglés Hawaii Space Exploration Analog and Simulation) localizado en Hawaii, los datos corresponden a los recabados para este sitio en específico. A través de la lectura de estos datos, se entrenaron varias modelos computacionales de machine learning para predecir la radiación solar basado en las variables de temperatura, humedad, presión atmosférica, velocidad del viento, dirección del viento y hora del día. Los modelos que se crearon son: Support Vector Machine, Linear Regression, Decision Trees, Random Forest, entre otros. Una vez obtenidos los resultados de la predicción, se implementó un modelo de panel solar, basado en el visto en clase, para predecir la producción fotovoltaica correspondientes a un número límitado de opciones presentes en el mercado actualmente.

Finalemente, diseñamos una interfaz gráfica a manera que los resultados obtenidos puedan ser visualizados por el usuario. El objetivo es que el usuario sea capaz de ingresar los valores númericos de las variables previamente especificadas para recibir una predicción de la radiación recibida en la hora espeificada, pudiendo escoger el modelo de predicción deseado. Acto seguido, será capaz de escoger el modelo del panel solar LG disponible y recibir la producción fotovoltaica diaria. Como parte clave de la utilidad del proyecto, el usuario es capaz de escoger dentro de una lista de opciones un númeor de artículos de uso diario del hogar para que pueda analizar el rendimiento de la energía y su relación con las actividades típicas de un día. Esto permite visualizar de manera orgánica y natural el alcance de la potencia generada por un panel solar para satisfacer las necesidades humanas básicas, o averiguar la distribución óptima del consumo de energía.

II. OBJETIVOS

El objetivo general del proyecto es proporcionar las herramientas necesarias para el analisis del consumo diario de energía de las necesidades de una casa promedio, con base en la energía que sale de un panel solar con las predicciones de la radiación solar que recibiera, determinada a partir de modelos que utilicen parámetros del tiempo como humedad, temperatura, velocidad y dirección del viento, presión y hora respectiva.

Los objetivos específicos personales son:

1. Leer y comprender los algoritmos y modelos hechos por los colaboradores.
2. Integrar estos algortimos en un solo código que tenga como variables de entrada los parámtetros del día, el modelo a usar de predicción y el tipo de panel, y como salida la predicción de la radiación solar y la producción fotovoltaica.
3. Diseñar una interfaz gráfica con cajas de "entry", listas de opciones, y botones de interacción para que el usuario sea capaz de insertar y esoger los parámetros indicados.
4. Implementar una sección con opciones de aparatos electrodomésticos y una barra de progreso que refleje la cantidad de energía disponible y suministrada de acuerdo al nivel máximo de energía del panel y los kWh consumidos por los aparatos escogidos.
5. Comprobar que el funcionamiento sea correcto y que no hayan errores.

III. INTRODUCCIÓN

Las transiciones hacia energías limpias son una inversión en nuestro futuro. No tiene sentido que invirtamos en algo que daña la sustentabilidad de nuestro planeta puesto que pone en riesgo el retorno de nuestra inversión. Por ello, buscamos ayudar en alguna forma a construir un consumo responsable de la energía producida por paneles solares. Porque sabemos que la tendencia va hacia las energías renovables y queremos que, desde el comienzo del uso de estas energías, las

personas puedan hacerlo responsablemente y bajo esta misma metodología en la que se debería gastar el dinero: bajo la mentalidad del ahorro y no el despilfarro irresponsable. El proyecto, como se ha dicho, está basado en el reto llevado a cabo en la competencia de SpaceApps de 2017 denominada "You are my sunshine", para la incubadora de innovación de la NASA. Nuestra propuesta consiste en usar la idea principal del concurso como inspiración para producir un análisis valioso que permita medir la utilidad y el funcionamiento de los arreglos fotovoltaicos en las actividades usuales del hogar.

Entendiendo el uso de los paneles solares según lo aprendido en clase, planteamos diversos escenarios para la generación de energía a partir de un modelo refinado de la intensidad de radiación recibida, considerando factores como humedad, velocidad y dirección del viento, temperatura, hora del día, entre otros. Los datos corresponden a los recabados en Mauna Loa, una región en Hawaii donde se encuentra un volcán, puesto que es el sitio en donde se encuentra la misión HI-SEAS (por sus siglas en inglés Hawaii Space Exploration Analog and Simulation). En esta región nos encontramos con fuertes variaciones en la temperatura, por lo que no se asemeja a lo esperado en una región hawaiiana, llegando incluso a caer nevadas en el sitio.

La materialización del proyecto la encontramos en la interfaz gráfica para que el usuario pueda insertar los datos con los que se realizó el modelo y pueda tener la predicción de la radiación. Aunado a ello, calcula la potencia según el modelo de panel utilizado y optimiza la distribución en los diversos electrodomésticos básicos del hogar. Como ya se mencionó esto lo logramos a partir de la evaluación de distintos modelos, cada uno con un método diferente de predecir la radiación (8 tipos de regresiones, Random Forest y Árboles de decisión) y optamos por el que posea la mayor precisión que en este caso fue el de Random Forest. Nuestro trabajo se divide en la sección que explica el procesamiento de la base de datos, la construcción de los algoritmos y la estadística en las variables, encargada a Alejandro Salinas; el modelo del panel solar, destinada a Bernardo Barrera, y el análisis de rendimiento diario, dirigida por su servidora, Carla López.

IV. MACHINE LEARNING Y ESTADÍSTICA DE LA RADIACIÓN SOLAR

A fin de buscar predecir la radiación solar con base en determinados parámteros y gozando de la vastedad de la base de datos obtenida, se propuso realizar varios modelos de predicción usando Machine Learning para después optar por el de mayor precisión y que sea ese el utilizado en nuestros siguientes cálculos de potencia fotovoltaica. En esta sección se explicarán los pasos desarrollados desde la limpieza de la base de datos, hasta la implementación de los algoritmos y el cálculo de la precisión de cada uno.

Después de importar las múltiples librerías necesarias, se exportaron los datos para tenerlos como DataFrame y que sean de un fácil manejo. Lo primero era entender las unidades en las que estaban los datos, para ello fue útil entrar al sitio del concurso, en donde se encuentra un archivo de texto con las generalidades de cada variable, al igual que la base de datos misma. Nos dimos cuenta que las unidades no correspondían a aquellas que dicta el Sistema Internacional de Unidades, razón por la cual procedimos a convertirlas a ello. En concreto, cambiamos la temperatura de Fahrenheit a Celsius, la presión de unidades de mercurio (Hg) a Pascales y la velocidad de millas por hora a metros por segundo. El resto de las unidades eran ya bien adimensionales, ya bien correspondían a lo buscado como la radiación en Watts por metro cuadrado ($\frac{W}{m^2}$). Si bien esto es prácticamente trivial, es de gran importancia por dos factores: conocer las unidades en las que trabajaremos o ajustarlas a nuestra conveniencia (recordemos la pérdida de una nave espacial debido a que Lockhead Martin usó el sistema inglés de unidades), y el segundo factor que es considerar nuestro mercado meta, ¿qué sistema de unidades es el tradicional en la región de nuestros usuarios? En nuestro caso, optamos por el Sistema Internacional de Unidades. Aunado a lo anteriormente expuesto, la variable de UNIXTIME fue ajustada a manera de índice, convirtiendo la hora de la zona horaria de Honolulu al Tiempo Coordinado Universal (CTU), facultándonos así para la creación de nuevas variables que fueron desahogadas en nuevas columnas. Estas son: mes del año, día del año, semana del año, hora del día, minuto del día, segundo del día. Todas ellas están relacionadas con la medición que se hizo y de la cual están depositados los datos en cada fila. Conviene aclarar que la medición de datos es, en promedio, cada cinco minutos.

Se revisó que la tabla estuviera llena de valores numéricos, es decir sin campos nulos y corroboró que no posee ni uno solo. Dentro del análisis estadístico a los datos de radiación recabados encontramos que hay un total de 32686 mediciones, cuyo promedio es de 207.12 $\frac{W}{m^2}$ y donde el valor mínimo es de 1.11 y el máximo es de 1601.26. Encontramos una dispersión de datos muy grande, con desviación de estándar de 315.92. Podemos apreciar en la Fig. 1 la frecuencia en la distribución de los valores de radiación.

Profundizando aún más en el análisis de los datos, observemos las siguientes gráficas incluidas en la Fig. 2 que relacionan el tiempo de medición a algunas variables como temperatura, presión, humedad y radiación. Vemos que se dividen en promedio por hora y en promedio mensual de los 4 meses que hay datos.

Algunas conclusiones que podemos obtener de estos datos son que la radiación solar suele incrementarse a mediodía, como es de esperarse, atenuándose al punto de desaparecer por las noches y en las mañanas antes de las 6. Asimismo, notamos que la mayor radiación se encuentra en los meses previos al invierno, en lugar del invierno per se, lo cual también es un comportamiento esperado. Podemos desde ya identificar cierta correlación entre, por ejemplo, la radiación, la temperatura y la presión, puesto que las 3 se incrementan durante el mismo intervalo de tiempo del día. Esto se puede confrimar al graficar la

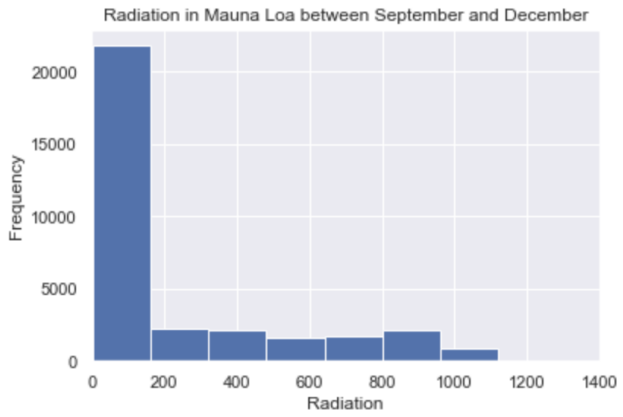


FIG. 1. Radiación en Mauna Loa entre los meses de septiembre y diciembre

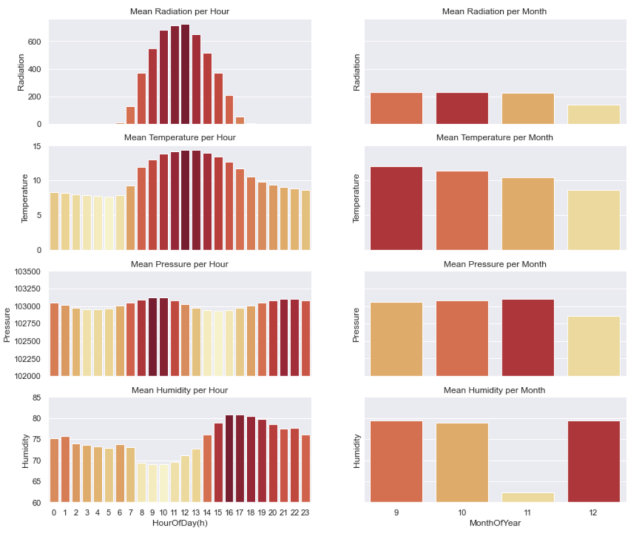


FIG. 2. Resumen y visualización de los datos a usar en promedios mensuales.

radiación frente a estas variables como se muestra en las siguientes figuras. Conviene comentar que se aprovechó para también enfrenta a la radiación con la humedad, la dirección del viento y su velocidad.

Después de un analisis comparativo de la relación entre las variables, es posible anticipar una relación directamente proporcional entre la temperatura y la presión, como ya habíamos predicho. Asimismo, es menester notar la ausencia de radiación cuando el ángulo de la dirección del viento varía entre 180^{o} y 270^{o}. El incremento en la velocidad, por otra parte, parece disminuir la radiación tanto en su densidad, como en la cantidad medida.

Una vez que hemos limpiado y entendido ampliamente los datos con los que trabajaremos, es debido realizar los modelos pertinentes. Para ello, primero dividiremos nuestros datos en 2: variable dependiente, que es la radiación, y variables independientes, que son la temperatura, la presión, humedad, dirección de viento, velocidad y hora del día. Posteriormente, volvemos a partir en 2 cada una de nuestras variables, a fin de tener el 80% de los datos para entrenar los modelos y el resto (20%) para probar y medir la precisión de los mismos.

Comenzamos con el modelo de regresión lineal, para el cual obtenemos los siguientes 'pesos' o coeficientes: temperatura = 69.117, presión = -0.2136, humedad = -0.02788, dirección del viento = -0.0278188 , velocidad = 37.41, hora del día = -6.362. Esto no es tan coherente con lo hallado en las gráficas, por lo que probablemente no sea un modelo tan preciso. Podemos incluso construir una línea de regresión que tome a la temperatura como variable independiente y que se vería de la siguiente manera: y = 69.117 x + 21518.336.

Algo que se notará en algunos puntos del trabajo, es probar casos para dar un vistazo inicial en la precisión del modelo de predicción. Por ello, obtenemos los datos del primer renglón y los insertamos en el modelo para ver qué tan cerca estuvo de la radiación. Nuevamente, conviene aclarar que esto no posee ninguna validez o apoyo a la precisión, es algo azaroso y estético a prima facie. Al insertar los datos en el modelo obtenemos una radiación de 168.33 $\frac{W}{m^2}$, cuando la radiación real era de 2.58 $\frac{W}{m^2}$, por lo que evidentemente no será el mejor modelo.

Probamos ahora con el modelo de clasificación Random Forest, mismo que en teoría es mejor para problemas multivariables. Haciendo el test case igual que con el regresor lineal obtenemos 1.3836 como radiación predicha, acercándose en demasía al target. Haciendo lo mismo para el modelo de Árboles de Decisión con una profundidad de 8 y probando el mismo caso, nos arroja una radiación de 2.6156, muy cercana al 2.58.

Intentamos hacer el algoritmo K nearest neighbors pero éste tiene problemas cuando recibe datos de tipo float, por lo que optamos por usar los siguientes clasificadores y que obtuvieron estos resultados para los mismos datos: Support Vector Machine (2.6135), Stochastic Gradient Regressor (8.23 x 10^{17}), Bayesian Ridge (168.204), Lasso Lars (171.1768), Automatic Relevance Determination Regression (168.2505), Passive Agressive Regressor (-2.017) y Theil-Sen Regressor para una población máxima de 10000 (160.51899). Podemos observar que muchos de ellos no estuvieron tan cerca de la radiación, sin embargo, recordemos esto es muy informal, por lo que es mejor realizar ciertas métricas de precisión para determinar al mejor modelo en general y no para un caso aleatorio y específico. Las métricas son el error absoluto medio, el error cuadrático medio y el coeficiente de determinación.

Los modelos encontrados varían en eficiencia y presición, sin embargo al haber creado un amplio número de opciones, hemos obtenido conocimiento y experiencia en el funcionamiento de machine learning y hemos decidido dejarlos todos como reflejo de nuestro trabajo y grado de libertad para el usuario en la interaz. A continuación, se describirá el modelo matemático de la producción fotovoltaica.

V. MODELO DEL PANEL SOLAR

Para poner en práctica nuestro modelo, decidimos incluir en la interfaz una función que nos permite calcular la producción fotovoltaica diaria para tres distintos modelos de páneles solares de la compañía LG. Los modelos pertenecen al módulo LG NeON 2, y se venden en potencias nominales de 335 Watts (modelo LG335N1C-A5), 330 Watts (modelo LG330N1C-A5), y 325 Watts (modelo LG325N1C-A5). La hoja de especificaciones técnicas para los tres se encuentra anexa al final de este documento. El objetivo de esta sección es encontrar la manera de calcular la energía total producida por cada uno de estos páneles en un día, de acuerdo con los niveles de radiación que hayan sido predichos anteriormente.

Ahora bien, para realizar el cálculo de la potencia en función de la radiación solar, se utilizaron las consideraciones cubiertas en clase para llevar a cabo el cálculo de la potencia. Se asumió un panel solar en el plano horizontal. Para ello, primero se expresó la potencia de la forma,

$$P = I_{sc}V_{oc}FF, \tag{1}$$

donde su asumió un factor de relleno constante e igual a,

$$FF = \frac{I_{mpp}V_{mpp}}{I_{sc}V_{sc}} \tag{2}$$

con I_{mpp}, V_{mpp} la corriente y el voltaje en el punto de máxima potencia. Se asumió una relación lineal entre la radiación solar y la corriente de corto circuito, y se calculó la constante de proporción utilizando los valores de la corriente y radiación I_{stc}, G_{stc} en condiciones estándares de prueba,

$$I_{sc} = \kappa G \qquad \kappa = \frac{I_{stc}}{G_{stc}}. \tag{3}$$

Además, se asumió una dependencia lineal entre la diferencia entre las temperaturas de la placa y ambiente y la radiación solar, y se utilizó la temperatura normal de operación NOCT para el cálculo de la constante de proporción.

$$T_{placa} - T_{amb} = \alpha G \qquad \alpha = \frac{T_{noct} - T_{stc}}{G_{noct}}. \tag{4}$$

Finalmente, la hoja técnica especifica la constante de proporcionalidad entre la temperatura y el cambio de voltaje de circuito abierto,

$$\beta = \frac{\Delta V_{oc}}{\Delta T} = -0.27\frac{\mathrm{V}}{^{\circ}\mathrm{C}}. \tag{5}$$

Por lo tanto, el voltaje de circuito abierto se puede escribir como,

$$V_{oc} = V_{stc} + \beta\Delta T \tag{6}$$
$$= V_{stc} + \beta\alpha G. \tag{7}$$

Con estas consideraciones en mente, la ecuación (1) se puede reescribir como,

$$P = \kappa G(V_{stc} + \beta\alpha G)FF, \tag{8}$$

y solamente queda reemplazar los valores proyectados por nuestro modelo para G para obtener el perfil horario para la producción fotovoltaica en función de la temperatura, humedad, y las otras variables del día. Hemos entonces alcanzado el objetivo, y se usarán estos resultados como herramienta clave en el analisis de consumo y rendimiento de energía que será descrito a a continuación.

VI. ANALISIS DE RENDIMIENTO DIARIO E INTERFAZ GRÁFICA

En las secciones pasadas, se detalló el proceso seguido para la obtención de un modelo de predicción de la radiación dado un número de variables, así como la producción fotovoltaica en un día en kWh de una celda solar. Este último dato es especialmente útil para analizar la cantidad de dispositivos en el hogar que podrían ser suministrados completamente con la celda. Para poder darle al proyecto una utilidad concreta, se diseñó una interfaz gráfica usando Python, más específicamente la biblioteca tkinter. Está compuesta de tres subsecciones: la obtención de datos insertada por el usuario, el reporte de la predicción de la radiación y la producción fotovoltaica producida, y el analisis de rendimiento de energía en función de una serie de artículos disponibles.

En primer lugar, se le pide al usuario insertar los datos del día que desea analizar, cumpliendo las variables de temperatura, presión, humedad, dirección del viento, velocidad, y la hora del día. Los datos introducidos nos pueden dar una idea de las características del día. En la Tabla I, se muestran distintos días escogidos como representaciones de escenarios metorológicos, basado en historiales meteorológicos de la región disponibles en World Weather Online[1]. La primera fila muestra el nombre de las variables, donde la dirección del viento es W.D (por sus siglas en inglés wind directon), la velocidad de viento es W.V. (wind velocity), la radiación observada reportada en la base de datos es R.R, y la radiación predicha por el modelo Random Forest es R.P. Los días escogidos para los escenarios son los siguientes: 13 de septiembre para el día lluvioso, 11 de octubre para el soleado, 15 de noviembre para el nublado, y 1 de diciembre para el día nevado.

TABLE I. Ejemplos representativos de escenarios meteorológicos, a las 10 hrs del día.

Escenario	Temp °C	Pres kPa	Hum %	W.D. °	W.S m/s	R.R W/m^2	R.P. W/m^2
Lluvioso	15.84	103.183	57	357.48	0	516.36	609.21
Soleado	14.20	103.149	87	47.8	3.37	831.77	886.65
Nublado	11.47	103.284	94	93.36	13.5	442.75	728.19
Nevado	7.65	103.103	100	141.8	9	527.28	289.36

La Fig. 3 muestra un recorte de la interfaz gráfica, que inlcuye los cajones de recolección de información, el menú desplegable de las opciones de modelos, y los botones de interacción. Los datos insertados pueden ser decimales y deben ser en las unidades especificadas correspondientes

a la variable indicada a la izquierda de cada cajón. Una vez insertados los datos, el usuario deberá seleccionar "Guardar" para que el algoritmo calcule los datos.

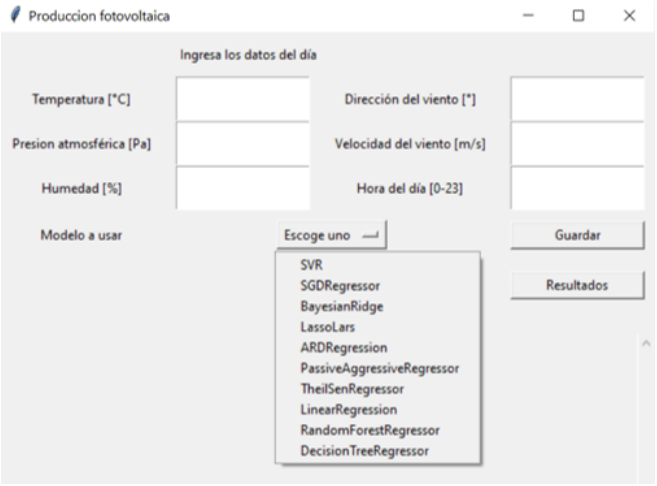


FIG. 3. Recorte de la interfaz gráfica, ingresar datos y elegir un modelo.

Al seleccionar el botón "Resultados", se desplegará el reporte de la radiación calculada de acuerdo al modelo seleccionado, así como la energía que generaría un panel solar en un día con las características determinadas. Las opciones de paneles solares a usar son LG335N1C-A5, LG33ON1C-A5 y LG325N1C-A5, producidos por la compañía L.G. La Fig. 4 muestra los resultados de radiación y producción fotovoltaica para un día con 15°C de temperatura, 103.013 kPa de presión, 47% de humedad, con un viento con 0.4° de dirección y 3.5 m/s de velocidad.

FIG. 4. Recorte de la interfaz gráfica, despliege del analisis realizado.

Finalmente, el usuario será capaz de escoger un número de artículos del hogar y visualizar cuántos y qué combinación de ellos el panel sería capaz de abastecer. Se incluyeron treinta y nueve opciones de aparatos distintos o funcionalidades, cada uno asociado con el consumo de enrgía promedio en un día en kWh de un ciudadano americano[2]. Las opciones se encuentran clasificadas en las siguientes categorías,

- Aire acondicionado (7 artículos)
- Ventiladores (3 artículos)
- Refrigeración de alimentos (6 artículos)
- Aparatos de cocina (14 artículos)
- Luz (3 artículos)
- Entretenimiento (3 artículos)
- Lavandería (3 artículos)

Cada categoría tiene su propio menú despegable del cual se puede seleccionar una de las opciones disponibles a la vez y agregar al análisis usando el botón "Añadir", como se muestra en la Fig. 5.

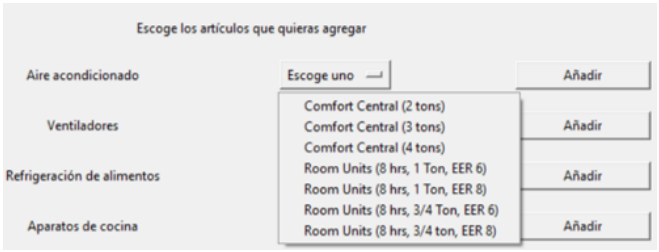


FIG. 5. Recorte de la interfaz gráfica, selección de artículos.

Al añadir un artículo, este se despliega en una lista a la derecha, para que el usuario tenga una manera de verificar que el sistema está registrando correctamente su selección. En teoría, no hay límite para el numero de artículos que se pueden escoger, pero visualmente hay un límite práctico de doce artículos. Seleccionar la opción default "Escoge uno" crea un error, pero se puede borrar usando el botón "Borrar artículos", el cual elimina la lista de artículos seleccionados y reinicia cualquier dato que haya sido calculado con respecto a la potencia total consumida hasta ese momento. Al seleccionar el botón "Evaluar", se guarda en una variable la potencia total requerida y se compara contra la potencia total generada generada por el panel en un día. La fracción del uso total de la energía producida por el panel se muestra en la barra de progreso a la derecha. La interpretación es que sila barra está al cien por ciento de su capacidad (totalmente verde), quiere decir que se está usando toda la energía generada por el panel. Por otra parte, si la barra está vacía significa que toda la energía está disponible para ser usada. Usando los mismos datos que en la Fig. 5, si el usuario escoge la opción de luz, para 4 a 5 cuartos y la radio, el panel estará suministranto 1.9 kWh de sus 2.06 kWh totales, como de muestra en la Fig. 6.

La implementación de esta interfaz gráfica permite al usuario visualizar el rendimiento de un panel solar de manera efectiva y accesible. Al comparar la energía con los artículos gráficamente más que numéricamente, un ciudadano promdio es capaz de dimensionar la capacidad de producción de energía de un sistema fotovoltaico, o en cualquier caso, la energía requerida para el funcionamiento de aparatos electrodomésticos de uso diario. El código utilizado para correr la simulación se encuentra adjunto en la sección de anexos.

VII. CONCLUSIÓN

Mi trabajo consistió en integrar el trabajo de mis colaboradores, y diseñar la interfaz gráfica para el usuario.

Para poder realizar esto, tuve que leer y comprender los algoritmos de Machine Learning, que eran completamente nuevos para mí. Se me hace sorprendente

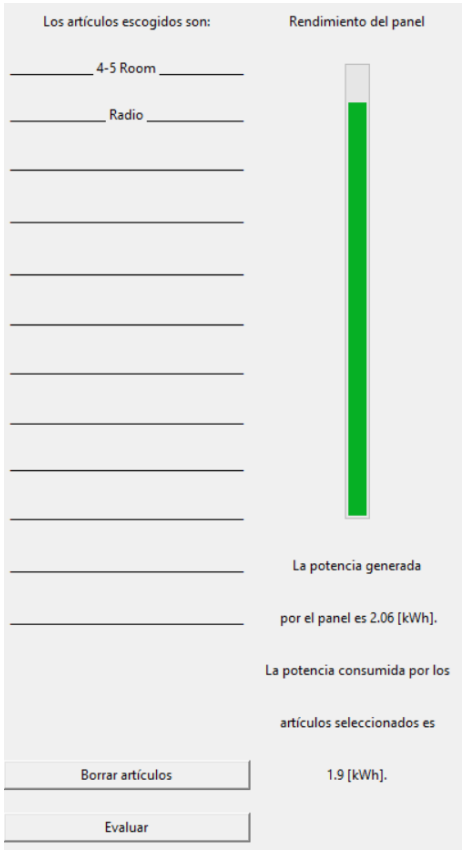


FIG. 6. Recorte de la interfaz gráfica, demostración visual de la administración de la energía suministrada por el panel.

cómo, gracias al desarrollo tecnológico actual, algoritmos tan poderosos como Random Forest Generator o Decision Tree están al alcance de cualquier persona con un conocimiento básico de programación. A través de la lectura de una base de datos extensas, las funciones de Machine Learning buscar dar un pronóstico de cierta variable dependiente con base en las condiciones inciales determinadas. Las funciones de regresión lineal son capaces de asignar un peso a cada variable independiente, mientras que las más avanzadas pueden averiguar cómo se relacionan entre ellas, aunque sean extremadamente complejas y caóticas.

Por otra parte, intregrar la parte de la producción fotovoltaica no resultó tan demandante ya que contaba de antemano con la teoría y los ejercicios realizados en clase, Ya estaba familiarizada con la interpretación de los parámetros y requerimientos de las celdas fotovoltaicas y no resultó nada complejo añadir esta parte del código al general.

Mi reto más grande se basó en usar por primera vez el lenguage de programación Python. La curva de aprendizaje inicial fue un poco alta ya que no estaba familiarizada con la interfaz que debía usar para ponder progrmar, pero recibiendo ayuda de mis comparñeros logré superar este obstáculo. La siguiente complicación surgió debido a las clases de objetos de programación que se deben declarar para poder usarlas como interactivas en la biblioteca tkintre. Usando códigos de ejemplo en internet, fui capaz de ir descubriendo el funcionamiento de cada "widget" y la manera de agregarlos a la interfaz de forma funcional.

Aunque no todos los modelos realizados con Machine Learning pueden ser considerados estables o funcionales, podemos recuperar algunos que si lo son, y si usamos una base de datos más extensa, podríamos lograr un algoritmo realmente destacado. No alcanzamos a incluir diferentes posiciones geográficas como parte de las variables independientes a analiza, ya que subestimamos inicialmente los requerimientos computacionales necesarios para poder llevar a cabo esta tarea. Sin embargo, podemos ahondar más en las características de los paneles solares e incluir más especificaciones del usuario dentro de la interfaz gráfica y así poder simular un número más amplio de celdas. Asimismo, con más tiempo para trabajar sobre este proyecto, se podría mejorar el diseño gráfico de la interfaz, e incluso explorar la posibilidad de migrarlo a otros sistemas operativos.

Considero que hemos logrado realizar un proyecto bastante inclusivo que abarca varios de los temas vistos en clase y tiene como producto final una interfaz funcional con mucho valor. Hemos desarrollado una manera de ilustrar el funcionamiento de los paneles fotovoltaicos en un escenario realista.

VIII. REFERENCIAS

[1] *Mauna Loa Historial Weather*, World Weather Online (2020).
[2] *Typical Electric Usage of Various Appliances*, Keys Energy Services (2020).
[3] J. Hermosillo, Instituto Tecnológico y de Estudios Superiores de Occidente (1995).
[4] *Solar Radiation Prediction*, NASA Hackathon (2020).

Código

Carla Judith López Zurita A00822301

Diciembre 2020

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn import linear_model
from sklearn import svm
from pytz import timezone
import pytz
sns.set()



train_data = pd.read_csv('SolarPrediction.csv')
train_data = train_data.sort_values(['UNIXTime'], ascending = [True])
train_data.head()



TZhawaii= timezone('Pacific/Honolulu')
train_data.index =  pd.to_datetime(train_data['UNIXTime'], unit='s')
train_data.index = train_data.index.tz_localize(pytz.utc).tz_convert(TZhawaii)

train_data['MonthOfYear'] = train_data.index.strftime('%m').astype(int)
train_data['DayOfYear'] = train_data.index.strftime('%j').astype(int)
train_data['WeekOfYear'] = train_data.index.strftime('%U').astype(int)
```

```
train_data['HourOfDay(h)'] = train_data.index.hour
train_data['MinuteOfDay(m)'] = train_data.index.
    minute
train_data['SecondOfDay(s)'] = train_data.index.
    second

train_data.drop(['Data','Time','TimeSunRise','
    TimeSunSet'], inplace=True, axis=1)
train_data


train_data_temp = (train_data['Temperature'] - 32)
    /1.8
train_data['Temperature'] = train_data_temp

train_data_pres = train_data['Pressure'] * 3386.39
train_data['Pressure'] = train_data_pres

train_data_vel = train_data['Speed'] * 0.44704
train_data['Speed'] = train_data_vel
train_data


train_data['Radiation'].describe()


group_month=train_data.groupby('MonthOfYear').mean
    ().reset_index()
group_week=train_data.groupby('WeekOfYear').mean().
    reset_index()
group_day=train_data.groupby('DayOfYear').mean().
    reset_index()
group_hour=train_data.groupby('HourOfDay(h)').mean
    ().reset_index()


X = train_data.iloc[:,np.r_[2:7,10]] # Independent
    variables: Temperature, Pressure, Humidity, Wind
     direction, speed and HourOfDay
y = train_data.iloc[:,1] # Dependent variable:
    Radiation


X.head(-5)
y.head()
```

```
                                                                    64
X_train, X_test, y_train, y_test = train_test_split                 65
    (X, y, test_size = 0.2, random_state = 25)
                                                                    66
                                                                    67
regresor= LinearRegression()                                        68
reg = regresor.fit(X_train, y_train)                                69
regresor_pred = regresor.predict(X_test)                            70
                                                                    71
slope_T = reg.coef_[0]                                              72
intercept = reg.intercept_                                          73
                                                                    74
train_data.iloc[0]                                                  75
                                                                    76
                                                                    77
                                                                    78
rf_reg = RandomForestRegressor()                                    79
rf_reg.fit(X_train, y_train)                                        80
randomforest_pred= rf_reg.predict(X_test)                           81
                                                                    82
                                                                    83
                                                                    84
dtree = DecisionTreeRegressor(max_depth=8,                          85
    min_samples_leaf=0.13, random_state=3)
dtree.fit(X_train, y_train)                                         86
pred_train_tree= dtree.predict(X_test)                              87
                                                                    88
                                                                    89
                                                                    90
train_data.iloc[100]                                                91
                                                                    92
modelos = ["SVR","SGDRegressor","BayesianRidge", "                  93
    LassoLars", "ARDRegression",
                    "PassiveAggressiveRegressor","                  94
                        TheilSenRegressor","
                        LinearRegression",
                    "RandomForestRegressor","                       95
                        DecisionTreeRegressor"]
classifiers = [                                                     96
    svm.SVR(),                                                      97
    linear_model.SGDRegressor(),                                    98
    linear_model.BayesianRidge(),                                   99
    linear_model.LassoLars(),                                       100
    linear_model.ARDRegression(),                                   101
    linear_model.PassiveAggressiveRegressor(),                      102
    linear_model.TheilSenRegressor(),                               103
```

```
    linear_model.LinearRegression(),
    RandomForestRegressor(),
    DecisionTreeRegressor(max_depth=8,
       min_samples_leaf=0.13, random_state=3)]


"""
Created on Fri Nov 27 16:36:22 2020
@author: Carla L pez
Now that we have our models up  and running, we
   will create a GUI
that allows the user to import their own data
"""

import tkinter as tk
from tkinter import ttk

#SE DEFNIE EL ESPACIO
class ScrolledFrame(tk.Frame):
    def __init__(self, parent, *args, **kw):
        tk.Frame.__init__(self, parent, *args, **kw
           )

        vscrollbar = tk.Scrollbar(self, orient=tk.
           VERTICAL)
        vscrollbar.pack(fill= tk.Y, side=tk.RIGHT,
           expand=False)


        self.canvas = tk.Canvas(self, bd=0,
           highlightthickness=0,
                                yscrollcommand=
                                   vscrollbar.set)
        self.canvas.pack(side=tk.LEFT, fill=tk.BOTH
           , expand=True)
        vscrollbar.config(command=self.canvas.yview
           )
        self.canvas.xview_moveto(0)
        self.canvas.yview_moveto(0)

        self.contenedor = tk.Frame(self.canvas)
        self.contenedor_id = self.canvas.
           create_window(0, 0,

window
=
self
```

```
                                                                    .
                                                                    contenedor
                                                                    ,
                                                                138 anchor
                                                                    =
                                                                    tk
                                                                    .
                                                                    NW
                                                                    )

        self.contenedor.bind('<Configure>', self.                139
            _configure_contenedor)
        self.canvas.bind('<Configure>', self.                    140
            _configure_canvas)
                                                                 141
                                                                 142
    def _configure_contenedor(self, event):                      143
        size = (self.contenedor.winfo_reqwidth(),                144
            self.contenedor.winfo_reqheight())
        self.canvas.config(scrollregion="0␣0␣{}␣{}"              145
            .format(*size))
        if self.contenedor.winfo_reqwidth() != self              146
            .canvas.winfo_width():
            self.canvas.config(width=self.                       147
                contenedor.winfo_reqwidth())
                                                                 148
                                                                 149
    def _configure_canvas(self, event):                          150
        if self.contenedor.winfo_reqwidth() != self              151
            .canvas.winfo_width():
            self.canvas.itemconfigure(self.                      152
                contenedor_id, width=self.canvas.
                winfo_width())
                                                                 153
#SE CREA LA APLICACION                                           154
class Aplicacion:                                                155
    def __init__(self, root):                                    156
        self.master = root                                       157
                                                                 158
        self.frame_conf = tk.Frame(root)                         159
        self.frame_conf.pack()                                   160
                                                                 161
        label  = tk.Label(self.frame_conf,                       162
                          text = 'Ingresa␣los␣datos              163
                              ␣del␣d  a')
```

```
label.grid(row = 0, column = 0, columnspan = 3, sticky ='nsew', padx=10, pady=10)


tk.Label(self.frame_conf,
                  text = 'Temperatura [ C ]').grid(row = 1,
                     column = 0, columnspan = 1, sticky ='nsew',
                     padx=10, pady=10)
self.temp  = tk.Entry(self.frame_conf)
self.temp.grid(row = 1, column = 1, columnspan = 1, sticky =tk.W+tk.E+tk.N+tk.S, padx=10)

tk.Label(self.frame_conf,
                  text = 'Presion atmosf rica [Pa]').grid(row = 2, column = 0, columnspan = 1,
                     sticky ='nsew', padx=10, pady=10)
self.pressure  = tk.Entry(self.frame_conf)
self.pressure.grid(row = 2, column = 1, columnspan = 1, sticky =tk.W+tk.E+tk.N+tk.S, padx=10)

tk.Label(self.frame_conf,
                  text = 'Humedad [%]').grid(row = 3, column = 0, columnspan = 1,
                     sticky ='nsew', padx=10, pady=10)
self.humidity  = tk.Entry(self.frame_conf)
self.humidity.grid(row = 3, column = 1, columnspan = 1, sticky =tk.W+tk.E+tk.N+tk.S, padx=10)

tk.Label(self.frame_conf,
                  text = 'Direcci n del viento [  ]').grid(row = 1, column = 2,
                     columnspan = 1, sticky ='nsew', padx=10,
                     pady=10)
self.winddir  = tk.Entry(self.frame_conf)
```

```
self.winddir.grid(row = 1, column = 3,
    columnspan = 1, sticky =tk.W+tk.E+tk.N+
    tk.S, padx=10)

tk.Label(self.frame_conf,
                  text = 'Velocidad del
                      viento [m/s]').grid(
                      row = 2, column = 2,
                      columnspan = 1, sticky
                       ='nsew', padx=10,
                      pady=10)
self.windvel  = tk.Entry(self.frame_conf)
self.windvel.grid(row = 2, column = 3,
    columnspan = 1, sticky =tk.W+tk.E+tk.N+
    tk.S, padx=10)

tk.Label(self.frame_conf,
                  text = 'Hora del d a
                      [0-23]').grid(row = 3,
                       column = 2,
                      columnspan = 1, sticky
                       ='nsew', padx=10,
                      pady=10)
self.hour  = tk.Entry(self.frame_conf)
self.hour.grid(row = 3, column = 3,
    columnspan = 1, sticky =tk.W+tk.E+tk.N+
    tk.S, padx=10)

#ESCOGEMOS EL MODELO A UTILIZAR
self.options = tk.StringVar()
self.options.set("Escoge uno") # default
    value

tk.Label(self.frame_conf,  text='Modelo a
    usar', width=15 ).grid(row=4,column=0)

self.om1 =tk.OptionMenu(self.frame_conf,
    self.options, *modelos)
self.om1.grid(row=4,column=1,columnspan =
    2)

boton = tk.Button(self.frame_conf, text = '
    Guardar', command = self.save_data)
boton.grid(row = 4, column = 3, columnspan
    = 1, sticky ='nsew', padx=10, pady=10)
```

```
        boton = tk.Button(self.frame_conf, text = '
            Resultados', command = self.next_win)
        boton.grid(row = 5, column = 3, columnspan
            = 1, sticky ='nsew', padx=10, pady=10)
                                                    
                                                    
        self.lbl_info  = tk.Label(root)             
        self.lbl_info.pack(padx=10, anchor = 'w')   
                                                    
        self.frame_botones = ScrolledFrame(root)    
        self.frame_botones.pack(fill = 'x')         
                                                    
        #Guardamos los datos de las entries en
            variables globales.
    def save_data(self):                            
        global data                                 
        global modelo                               
        global radiation                            
                                                    
        data = [float(self.temp.get()),float(self.
            pressure.get()), float(self.humidity.get
            ()),
                float(self.winddir.get()), float(
                    self.windvel.get()), float(self.
                    hour.get())]
                                                    
        modelo=modelos.index(self.options.get())    
                                                    
        # MANDAMOS A LLAMAR LA FUNCI N              
        global clf                                  
        clf = classifiers[modelo]                   
        clf.fit(X_train, y_train)                   
        radiation = clf.predict([data])             
                                                    
    def next_win(self):                             
                                                    
        label  = tk.Label(self.frame_conf,          
                          text = 'La radiaci n
                              predicha por el modelo
                              '+ self.options.get()
                              +
                          ', a las '+str(self.hour.
                              get())+' es '+str(
                              round(radiation[0], 2)
                              )+ ' [W/m^2].')
        label.grid(row = 5, column = 0, columnspan
```

```
            = 3, sticky ='nsew', padx=10, pady=0)
                                                                    242
        tk.Label(self.frame_conf,  text='Panel␣a␣                   243
            usar', width=15 ).grid(row=6,column=0)
                                                                    244
        global paneles                                              245
        paneles = ['LG335N1C-A5', 'LG33ON1C-A5', '                  246
            LG325N1C-A5']
                                                                    247
        self.panels = tk.StringVar()                                248
        self.panels.set("Escoge␣uno")                               249
                                                                    250
        self.om2 =tk.OptionMenu(self.frame_conf,                    251
            self.panels, *paneles)
        self.om2.grid(row=6,column=1,columnspan =                   252
            1)
                                                                    253
        boton = tk.Button(self.frame_conf, text = '                 254
            Producci n␣fotovoltaica', command =
            self.energy_win)
        boton.grid(row = 6, column = 2, columnspan                  255
            = 1, sticky ='nsew', padx=10, pady=10)
                                                                    256
    def energy_win(self):                                           257
                                                                    258
        i=paneles.index(self.panels.get())                          259
        panels                                                      260
            =[[9.83,34.1,10.49,41.0,1000,-0.27,45,25,800],[9.80,33.7,10

                [9.77,33.3,10.41,40.8,1000,-0.27,45,25,800]]        261

        Impp=panels[i][0]                                           262
        Vmpp=panels[i][1]                                           263
        Isc=panels[i][2]                                            264
        Voc=panels[i][3]                                            265
        Gstc=panels[i][4]                                           266
        beta=panels[i][5]                                           267
        Tnoct=panels[i][6]                                          268
        Tstc=panels[i][7]                                           269
        Gnoct=panels[i][8]                                          270
        FF=Impp*Vmpp/(Isc*Voc)                                      271
                                                                    272
        global total                                                273
        total=0                                                     274
        datas = data[:-1]                                           275
                                                                    276
```

```
        for t in range(12,38):                                  277
            t=t/2                                               278
            d=np.concatenate((datas,[t]))                       279
            G = clf.predict([d])                                280
            I=Isc/Gstc*G                                        281
            V=Voc+beta*(Tnoct-Tstc)/Gnoct*G                     282
            E=FF*I*V/1000*0.5 #Energia producida en             283
                media hora en kWh
            total=total+E                                       284
                                                                285
        label  = tk.Label(self.frame_conf,                      286
                          text = 'La producci n                 287
                             fotovolataica para el
                             panel '+ self.panels.
                             get()+
                          ' en todo el d a es '+                288
                             str(round(total[0],2))
                             + ' [kWh].')
        label.grid(row = 7, column = 0, columnspan              289
           = 4, sticky ='nsew', padx=10, pady=0)
                                                                290
        boton = tk.Button(self.frame_conf, text = '             291
           An lisis energ tico', command = self.
           panels_win)
        boton.grid(row = 8, column = 1, columnspan              292
           = 1, sticky ='nsew', padx=10, pady=10)
                                                                293
                                                                294
    def panels_win(self):                                       295
                                                                296
        label  = tk.Label(self.frame_conf,                      297
                          text = 'Escoge los                    298
                             art culos que quieras
                             agregar')
        label.grid(row = 9, column = 0, columnspan              299
           = 3, sticky ='nsew', padx=10, pady=0)
                                                                300
        self.v = tk.StringVar()                                 301
        values = {"Aire acondicionado" : "1",                   302
                  "Ventiladores" : "2", "                       303
                     Refrigeraci n de alimentos" :
                      "3",
                  "Aparatos de cocina" : "4", "Luz"             304
                      : "5",
                  "Entretenimiento" : "6", "                    305
                     Lavander a" : "7"}
```

```
                                                                    306
opcionesac = ["Comfort␣Central␣(2␣tons)",                          307
              "Comfort␣Central␣(3␣tons)",                           308
              "Comfort␣Central␣(4␣tons)",                           309
              "Room␣Units␣(8␣hrs,␣1␣Ton,␣EER␣                      310
                 6)",
              "Room␣Units␣(8␣hrs,␣1␣Ton,␣EER␣                      311
                 8)",
              "Room␣Units␣(8␣hrs,␣3/4␣Ton,␣                        312
                 EER␣6)",
              "Room␣Units␣(8␣hrs,␣3/4␣ton,␣                        313
                 EER␣8)"]
                                                                    314
opcionesvent = ["Whole␣House","Circulating"                        315
   ,"Ceiling"]
                                                                    316
opcionesrefri = ["Refrigerator(␣Manual␣12␣                          317
   cu.␣ft.)","Ref-Freezer␣(Manual␣12-14␣cu.
   ␣ft.)",
                 "Ref-Freezer␣(Frost-free␣                          318
                    14-17␣cu.␣ft.)","Ref-
                    Freezer␣(Frost␣free␣
                    17-20␣cu.␣ft.)",
                 "Freezer␣(Manual␣14.5-17.5                         319
                    ␣cu.␣ft.)","Freezer␣(
                    Frost-␣Free␣␣14.5-␣17.5
                    ␣cu.␣ft.)"]
                                                                    320
opcionescocina = ["Baby␣Food/Bottle␣Warmer"                        321
   ,"Broiler/Rotisserie","Coffee␣Maker",
                  "Dishwasher","Egg␣Cooker"                         322
                     ,"Frying␣Pan",
                  "Microwave␣Oven", "Range␣                         323
                     with␣Oven", "Roaster",
                  "Sandwich␣Grill", "Slow␣                          324
                     Cooker", "Toaster", "
                     Trash␣Compactor", "
                     Waffle␣Iron"]
opcionesluz =["4-5␣Room","6-8␣Room","                               325
   Outdoors,␣1␣Spotlight,␣All␣Night"]
                                                                    326
opcionesentr = ["Radio","Radio/Record␣                              327
   Player","Television␣(color␣solid␣state)"
   ]
                                                                    328
opcioneslav = ["Dryer","Iron","Washing␣                             329
```

```
    Machine"]
                                                    330
                                                    331
self.options1 = tk.StringVar()                      332
self.options1.set("Escoge␣uno")                     333
self.options2 = tk.StringVar()                      334
self.options2.set("Escoge␣uno")                     335
self.options3 = tk.StringVar()                      336
self.options3.set("Escoge␣uno")                     337
self.options4 = tk.StringVar()                      338
self.options4.set("Escoge␣uno")                     339
self.options5 = tk.StringVar()                      340
self.options5.set("Escoge␣uno")                     341
self.options6 = tk.StringVar()                      342
self.options6.set("Escoge␣uno")                     343
self.options7 = tk.StringVar()                      344
self.options7.set("Escoge␣uno")                     345
                                                    346
for (text, value) in values.items():                347
    tk.Label(self.frame_conf, text = text).         348
       grid(row=9+int(value),column=0, padx
       =0, pady=0)
                                                    349
                                                    350
tk.OptionMenu(self.frame_conf, self.                351
    options1,
                  *opcionesac).grid(row             352
                     =10,column=1,
                     columnspan = 2)
boton = tk.Button(self.frame_conf, text = '         353
    A adir', command = self.but1)
boton.grid(row = 10, column = 3, columnspan         354
    = 1, sticky ='nsew', padx=10, pady=10)
                                                    355
tk.OptionMenu(self.frame_conf, self.                356
    options2,
                  *opcionesvent).grid(row           357
                     =11,column=1,
                     columnspan = 2)
boton = tk.Button(self.frame_conf, text = '         358
    A adir', command = self.but2)
boton.grid(row = 11, column = 3, columnspan         359
    = 1, sticky ='nsew', padx=10, pady=10)
                                                    360
tk.OptionMenu(self.frame_conf, self.                361
    options3,
```

```
                    *opcionesrefri).grid(row
                        =12,column=1,
                        columnspan = 2)
boton = tk.Button(self.frame_conf, text = '
    A adir', command = self.but3)
boton.grid(row = 12, column = 3, columnspan
     = 1, sticky ='nsew', padx=10, pady=10)

tk.OptionMenu(self.frame_conf, self.
    options4,
                    *opcionescocina).grid(
                        row=13,column=1,
                        columnspan = 2)
boton = tk.Button(self.frame_conf, text = '
    A adir', command = self.but4)
boton.grid(row = 13, column = 3, columnspan
     = 1, sticky ='nsew', padx=10, pady=10)

tk.OptionMenu(self.frame_conf, self.
    options5,
                    *opcionesluz).grid(row
                        =14,column=1,
                        columnspan = 2)
boton = tk.Button(self.frame_conf, text = '
    A adir', command = self.but5)
boton.grid(row = 14, column = 3, columnspan
     = 1, sticky ='nsew', padx=10, pady=10)

tk.OptionMenu(self.frame_conf, self.
    options6,
                    *opcionesentr).grid(row
                        =15,column=1,
                        columnspan = 2)
boton = tk.Button(self.frame_conf, text = '
    A adir', command = self.but6)
boton.grid(row = 15, column = 3, columnspan
     = 1, sticky ='nsew', padx=10, pady=10)

tk.OptionMenu(self.frame_conf, self.
    options7,
                    *opcioneslav).grid(row
                        =16,column=1,
                        columnspan = 2)
boton = tk.Button(self.frame_conf, text = '
    A adir', command = self.but7)
boton.grid(row = 16, column = 3, columnspan
```

```
    = 1, sticky ='nsew', padx=10, pady=10)
                                                                    385
tk.Label(self.frame_conf, text = "Los␣                              386
    art culos␣escogidos␣son:").grid(row=1,
    column=4, padx=0, pady=0)
for n in range(12):                                                 387
    tk.Label(self.frame_conf,                                       388
             text = "␣                                              389
                 ------------------------------------------
                 ␣").grid(row=2+n,column=4,
                 rowspan = 1,padx=10, pady
                 =10)
                                                                    390
                                                                    391
                                                                    392
boton = tk.Button(self.frame_conf, text = '                         393
    Borrar␣art culos', command = self.
    borrar)
boton.grid(row = 15, column = 4, columnspan                         394
    = 1, sticky ='nsew', padx=10, pady=10)
                                                                    395
                                                                    396
boton = tk.Button(self.frame_conf, text = '                         397
    Evaluar', command = self.evaluate)
boton.grid(row = 16, column = 4, columnspan                         398
    = 1, sticky ='nsew', padx=10, pady=10)
                                                                    399
self.count = tk.IntVar()                                            400
self.count = 0                                                      401
                                                                    402
global arti                                                         403
arti = []                                                           404
self.totenergy = tk.DoubleVar()                                     405
self.totenergy = 0                                                  406
                                                                    407
tk.Label(self.frame_conf, text = "                                  408
    Rendimiento␣del␣panel").grid(row=1,
    column=5,padx=0, pady=0)
self.bar = ttk.Progressbar(self.frame_conf,                         409
    length = 400, mode='determinate',orient
    =tk.VERTICAL)
self.bar.grid(row = 2, rowspan=10, column =                         410
    5)
                                                                    411
tk.Label(self.frame_conf, text = 'La␣                               412
    potencia␣generada').grid(row=12,column
```

```
    =5,padx=0, pady=0)
tk.Label(self.frame_conf, text = '␣por␣el␣ panel␣es␣'+str(round(total[0],2))+'␣[kWh].').grid(row=13,column=5,padx=0, pady=0)
tk.Label(self.frame_conf, text = 'La␣ potencia␣consumida␣por␣los').grid(row=14,column=5,padx=0, pady=0)
tk.Label(self.frame_conf, text = ' art culos␣seleccionados␣es␣').grid(row=15,column=5,padx=0, pady=0)
tk.Label(self.frame_conf, text = str(round(self.totenergy,2))+'␣[kWh].').grid(row=16,column=5,padx=0, pady=0)

self.articles = ['Comfort␣Central␣(2␣tons)', 'Comfort␣Central␣(3␣tons)',
          'Comfort␣Central␣(4␣tons)',
          'Room␣Units␣(8␣hrs,␣1␣Ton,␣EER␣6)',
          'Room␣Units␣(8␣hrs,␣1␣Ton,␣EER␣8)',
          'Room␣Units␣(8␣hrs,␣3/4␣Ton,␣EER␣6)',
          'Room␣Units␣(8␣hrs,␣3/4␣ton,␣EER␣8)',
          'Whole␣House','Circulating','Ceiling',
          'Refrigerator(␣Manual␣12␣cu.␣ft.)',
          'Ref-Freezer␣(Manual␣12-14␣cu.␣ft.)',
          'Ref-Freezer␣(Frost-free␣14-17␣cu.␣ft.)',
          'Ref-Freezer␣(Frost␣free␣17-20␣cu.␣ft.)',
          'Freezer␣(Manual␣14.5-17.5␣cu.␣ft.)',
          'Freezer␣(Frost-␣Free␣␣14.5-␣17.5␣cu.␣ft.)',
          'Baby␣Food/Bottle␣Warmer','Broiler/Rotisserie',
          'Coffee␣Maker','Dishwasher','Egg␣Cooker','Frying␣Pan','Microwave␣Oven',
          'Range␣with␣Oven','Roaster',
```

```
                    'Sandwich␣Grill','Slow␣Cooker',
                        'Toaster','Trash␣Compactor'
                       ,'Waffle␣Iron','4-5␣Room',
                    '6-8␣Room','Outdoors,␣Spotlight
                       ,␣All␣Night',
                    'Dryer','Iron','Washing␣Machine
                       ','Radio',
                    'Radio/Record␣Player','
                       Television␣(color␣solid␣
                       state)']

        self.kwh = [48.33333333, 70, 91.66666667,
           4.714666667, 3.536,
                            16.50133333, 12.376, 1,
                               0.133333333, 0.4,
                               2.6,
                            4.166666667,
                               5.666666667,
                               6.833333333, 4.5,
                            6.266666667,
                               0.066666667,
                               0.233333333, 0.3, 1,
                            0.033333333,
                               0.266666667,
                               0.533333333,
                               1.933333333,
                            0.166666667, 0.1, 0.4,
                               0.1, 0.133333333,
                               0.066666667,
                            1.666666667, 2, 1.5,
                               2.5, 0.166666667,
                               0.3, 0.233333333,
                               0.3,0.9]

        self.frame_conf.mainloop()

    def but1(self):
        self.count += 1
        tk.Label(self.frame_conf, text = self.
           options1.get()).grid(row = 1+self.count,
            column = 4)
        arti.append(self.options1.get())
    def but2(self):
        self.count += 1
        tk.Label(self.frame_conf, text = self.
           options2.get()).grid(row = 1+self.count,
```

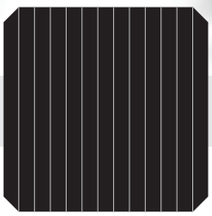
```
            column = 4)
        arti.append(self.options2.get())
    def but3(self):
        self.count += 1
        tk.Label(self.frame_conf, text = self.
           options3.get()).grid(row = 1+self.count,
            column = 4)
        arti.append(self.options3.get())
    def but4(self):
        self.count += 1
        tk.Label(self.frame_conf, text = self.
           options4.get()).grid(row = 1+self.count,
            column = 4)
        arti.append(self.options4.get())
    def but5(self):
        self.count += 1
        tk.Label(self.frame_conf, text = self.
           options5.get()).grid(row = 1+self.count,
            column = 4)
        arti.append(self.options5.get())
    def but6(self):
        self.count += 1
        tk.Label(self.frame_conf, text = self.
           options6.get()).grid(row = 1+self.count,
            column = 4)
        arti.append(self.options6.get())
    def but7(self):
        self.count += 1
        tk.Label(self.frame_conf, text = self.
           options7.get()).grid(row = 1+self.count,
            column = 4)
        arti.append(self.options7.get())


    def borrar(self):
        self.count = 0
        for n in range(12):
            tk.Label(self.frame_conf,
                     text = "␣
                        ----------------------------------------
                        ␣").grid(row=2+n,column=4,
                        rowspan = 1,padx=10, pady
                        =10)
        arti.clear()
        self.totenergy=0
        self.bar['value'] = self.totenergy
```

```
            tk.Label(self.frame_conf, text = '␣␣␣␣␣␣␣␣␣␣
               ␣␣␣␣␣␣␣␣0␣[kWh].␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣').grid(
               row=16,column=5,padx=0, pady=0)
                                                                    488
        def evaluate(self):                                         489
            print(arti)                                             490
            self.totenergy = 0                                      491
            for n in range(len(arti)) :                             492
                energy  = self.kwh[self.articles.index(            493
                   arti[n])]
                self.totenergy = self.totenergy +                   494
                   energy
                                                                    495
            self.bar['value'] = self.totenergy/total                496
               [0]*100
                                                                    497
            tk.Label(self.frame_conf, text = 'La␣                   498
               potencia␣consumida␣por␣los').grid(row
               =14,column=5,padx=0, pady=0)
            tk.Label(self.frame_conf, text = '                      499
               art culos␣seleccionados␣es').grid(row
               =15,column=5,padx=0, pady=0)
            tk.Label(self.frame_conf, text = str(round(             500
               self.totenergy,2))+'␣[kWh].').grid(row
               =16,column=5,padx=0, pady=0)
                                                                    501
                                                                    502
                                                                    503
                                                                    504
    def main():                                                     505
        root = tk.Tk()                                              506
        Aplicacion(root)                                            507
        root.title("Produccion␣fotovoltaica")                       508
        root.mainloop()                                             509
                                                                    510
    if __name__ == '__main__':                                      511
        main()                                                      512
```

LG NeON® 2

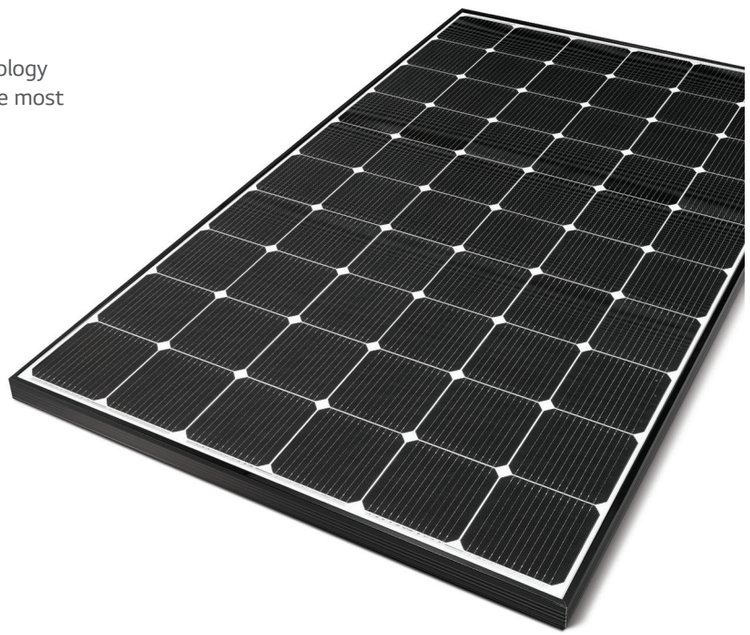
LG335N1C-A5 | LG330N1C-A5 | LG325N1C-A5



60

335W | 330W | 325W

The LG NeON® 2 is LG's best selling solar module. It received the acclaimed 2015 Intersolar AWARD for featuring LG's Cello Technology that increases its power output and reliability making it one of the most powerful and versatile modules on the market.



Feature

Enhanced Performance Warranty

LG NeON® 2 has an enhanced performance warranty. After 25 years, LG NeON® 2 is guaranteed at least 84.8% of initial performance.

High Power Output

Compared with previous models, the LG NeON® 2 has been designed to significantly enhance its output efficiency making it efficient even in limited space.

Aesthetic Roof

LG NeON® 2 has been designed with aesthetics in mind; thinner wires that appear all black at a distance. The product can increase the value of a property with its modern design.

Outstanding Durability

With its newly reinforced frame design, LG has extended the warranty of the NeON® 2 for an additional 2 years. Additionally, LG NeON® 2 can endure a front load up to 6000 Pa, and a rear load up to 5400 Pa.

Better Performance on a Sunny Day

LG NeON® 2 now performs better on a sunny days thanks to its improved temperature coefficient.

Near Zero LID (Light Induced Degradation)

The n-type cells used in LG NeON® 2 have almost no boron, which may cause the initial performance degradation, leading to less LID.

About LG Electronics

LG Electronics is a global big player, committed to expanding its operations with the solar market. The company first embarked on a solar energy source research program in 1985, supported by LG Group's vast experience in the semi-conductor, LCD, chemistry and materials industries. In 2010, LG Solar successfully released its first MonoX® series to the market, which is now available in 32 countries. The NeON® (previous. MonoX® NeON), NeON®2, NeON®2 BiFacial won the "Intersolar AWARD" in 2013, 2015 and 2016, which demonstrates LG Solar's lead, innovation and commitment to the industry.

LG NeON® 2

LG335N1C-A5 | LG330N1C-A5 | LG325N1C-A5

Mechanical Properties

Cells	6 x 10
Cell Vendor	LG
Cell Type	Monocrystalline / N-type
Cell Dimensions	161.7 x 161.7 mm / 6 inches
# of Busbar	12 (Multi Wire Busbar)
Dimensions (L x W x H)	1,686 x 1,016 x 40 mm
	66.38 x 40 x 1.57 in
Front Load	6,000Pa / 125 psf
Rear Load	5,400Pa / 113 psf
Weight	18 kg / 39.68 lb
Connector Type	MC4 (MC)
Junction Box	IP68 with 3 Bypass Diodes
Cables	1,000 mm x 2 ea / 39.37 in x 2 ea
Glass	High Transmission Tempered Glass
Frame	Anodized Aluminium

Certifications and Warranty

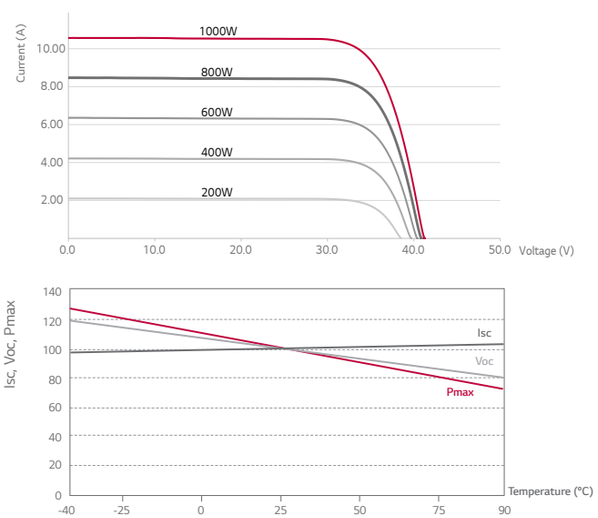
Certifications	IEC 61215, IEC 61730-1/-2
	UL 1703
	IEC 61701 (Salt mist corrosion test)
	IEC 62716 (Ammonia corrosion test)
	ISO 9001
Module Fire Performance	Type 1 (UL 1703)
Fire Rating	Class C (ULC/ORD C 1703, IEC 61730)
Product Warranty	12 Years
Output Warranty of Pmax	Linear Warranty*

* 1) 1st year: 98%, 2) After 1st year: 0.55% annual degradation 3) 84.8% for 25 years

Temperature Characteristics

NOCT	[°C]	45 ± 3
Pmax	[%/°C]	-0.37
Voc	[%/°C]	-0.27
Isc	[%/°C]	0.03

Characteristic Curves



Electrical Properties (STC*)

Model		LG335N1C-A5	LG330N1C-A5	LG325N1C-A5
Maximum Power (Pmax)	[W]	335	330	325
MPP Voltage (Vmpp)	[V]	34.1	33.7	33.3
MPP Current (Impp)	[A]	9.83	9.80	9.77
Open Circuit Voltage (Voc)	[V]	41.0	40.9	40.8
Short Circuit Current (Isc)	[A]	10.49	10.45	10.41
Module Efficiency	[%]	19.6	19.3	19.0
Operating Temperature	[°C]	-40 ~ +90		
Maximum System Voltage	[V]	1000 (UL / IEC)		
Maximum Series Fuse Rating	[A]	20		
Power Tolerance	[%]	0 ~ +3		

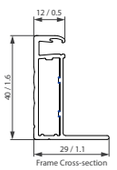
* STC (Standard Test Condition): Irradiance 1000 W/m², cell temperature 25 °C, AM 1.5
The nameplate power output is measured and determined by LG Electronics at its sole and absolute discretion.
The Typical change in module efficiency at 200 W/m² in relation to 1000 W/m² is -2.0%.

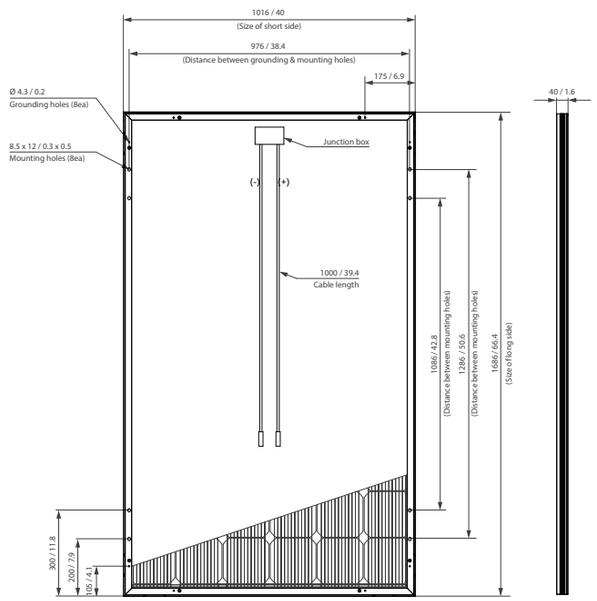
Electrical Properties (NOCT*)

Model		LG335N1C-A5	LG330N1C-A5	LG325N1C-A5
Maximum Power (Pmax)	[W]	247	243	240
MPP Voltage (Vmpp)	[V]	31.5	31.2	30.8
MPP Current (Impp)	[A]	7.83	7.81	7.78
Open Circuit Voltage (Voc)	[V]	38.2	38.1	38.0
Short Circuit Current (Isc)	[A]	8.44	8.41	8.38

* NOCT (Nominal Operating Cell Temperature): Irradiance 800 W/m², ambient temperature 20 °C, wind speed 1 m/s

Dimensions (mm / inch)





* The distance between the center of the mounting/grounding

LG Electronics Inc.
Solar Business Division
LG Twin Towers, 128 Yeoui-daero, Yeongdeungpo-gu, Seoul
07336, Korea

www.lg-solar.com

Product specifications are subject to change without notice.
DS-N5-60-C-G-F-EN-70521