Neiva, 31 de septiembre de 2022

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Neiva

El (Los) suscrito(s):

Juan Sebastián Pastrana Ardila, con C.C. No 1075303018,

Luis Daniel Valencia García, con C.C. No. 1075307839,

Autor(es) de la tesis y/o trabajo de grado titulado ESTUDIO DE VIABILIDAD DE UN SISTEMA DE LOCALIZACIÓN EN ZONA RURAL DEL DEPARTAMENTO DEL HUILA BASADO EN LA INTENSIDAD DE LA SEÑAL RECIBIDA EN UNA RED LPWAN, presentado y aprobado en el año 2022 como requisito para optar al título de INGENIERO ELECTRÓNICO;

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores”, los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE:

Juan Sebastián Pastrana Ardila

Firma: _____

EL AUTOR/ESTUDIANTE:

Luis Daniel Valencia García

Firma: _____

TÍTULO COMPLETO DEL TRABAJO: ESTUDIO DE VIABILIDAD DE UN SISTEMA DE LOCALIZACIÓN EN ZONA RURAL DEL DEPARTAMENTO DEL HUILA BASADO EN LA INTENSIDAD DE LA SEÑAL RECIBIDA EN UNA RED LPWAN

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
PASTRANA ARDILA	JUAN SEBASTIÁN
VALENCIA GARCÍA	LUIS DANIEL

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
BRAVO OBANDO	MARTIN DIOMEDES

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre

PARA OPTAR AL TÍTULO DE: Ingeniero Electrónico

FACULTAD: Ingeniería

PROGRAMA O POSGRADO: Ingeniería Electrónica

CIUDAD: Neiva **AÑO DE PRESENTACIÓN:** 2022 **NÚMERO DE PÁGINAS:** 90

TIPO DE ILUSTRACIONES (Marcar con una **X**)**:**

Diagramas X Fotografías X Grabaciones en discos___ Ilustraciones en general X Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___ Tablas o Cuadros X

SOFTWARE requerido y/o especializado para la lectura del documento:

MATERIAL ANEXO:

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

Español	Inglés	Español	Inglés
1. Intensidad	Strength	**6.** Algoritmos	Algorithms
2. Señal recibida	Received signal	**7.** Predicciones	Predictions
3. Sistemas de comunicación	Communication systems		
4. Aprendizaje de máquina	Machine learning		
5. modelamiento	Modeling		

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

En este trabajo se implementó una red LPWAN basada en tecnología LoRa en un conjunto residencial del corregimiento La Ulloa en el municipio de Rivera, este fue dividido en zonas para realizar envío de mensajes y adquirir niveles de intensidad de señal en cada zona para ser utilizados posteriormente con el fin de estimar la ubicación de un nodo LoRa dentro de la zona cubierta por la red mediante el uso de un algoritmo de “Machine Learning” entrenado con estos niveles.

Una finalidad importante en este trabajo es la estimación de localización a partir del nivel de intensidad de señal recibida (RSSI) en las puertas de enlace LoRa, con el objetivo de discriminar la zona dentro de la red a la cual corresponde los niveles de intensidad obtenidos mediante el uso de algoritmos de clasificación, estos basan la estimación a partir de análisis probabilístico, se implementaron los algoritmos de Machine Learning para multiclasificación de máquina de soporte vectorial, los k-vecinos más cercanos, arboles de decisión red neuronal, se comparan los resultados de las predicciones obtenidas, determinando la red neuronal como el algoritmo más eficiente para esta localización basada en RSSI. Finalmente se evalúa el rendimiento del modelo de red neuronal con más detalle mediante la matriz de confusión, analizando las posibles fuentes de error,

finalmente se proponen mejoras a este tipo de implementaciones para trabajos futuros.

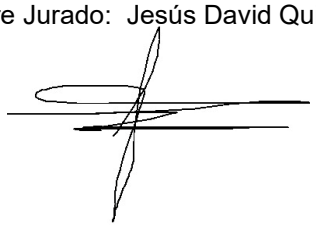
ABSTRACT: (Máximo 250 palabras)

In this work, an LPWAN network based on LoRa technology was implemented in a residential complex in La Ulloa district in the municipality of Rivera, this was divided into zones to send messages and acquire signal intensity levels in each zone to be used later. in order to estimate the location of a LoRa node within the area covered by the network by using a “Machine Learning” algorithm trained with these levels.

An important purpose in this work is the estimation of location from the received signal intensity level (RSSI) in the LoRa gateways, with the aim of discriminating the area within the network to which the intensity levels obtained correspond. through the use of classification algorithms, these base the estimation from probabilistic analysis, replicating the RSSI fingerprint estimation technique, algorithms of Machine Learning for support vector machine multiclassification, k-nearest neighbors, neural network decision trees, the results of the predictions obtained are compared, determining the neural network as the most efficient algorithm for this location based on RSSI. Finally, the performance of the neural network model is evaluated in more detail through the confusion matrix, analyzing the possible sources of error, finally improvements to this type of implementations are proposed for future work.

APROBACION DE LA TESIS

Nombre Jurado: Jesús David Quintero Polanco

Firma: 

Nombre Jurado: Johan Julián Molina Mosquera

Firma: 

UNIVERSIDAD
SURCOLOMBIANA

ESTUDIO DE VIABILIDAD DE UN SISTEMA DE LOCALIZACIÓN EN ZONA RURAL DEL DEPARTAMENTO DEL HUILA BASADO EN LA INTENSIDAD DE LA SEÑAL RECIBIDA EN UNA RED LPWAN

JUAN SEBASTIÁN PASTRANA ARDILA
LUIS DANIEL VALENCIA GARCÍA

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
GRUPO DE INVESTIGACIÓN UNITCOM
NEIVA-HUILA
2022

ESTUDIO DE VIABILIDAD DE UN SISTEMA DE LOCALIZACIÓN EN ZONA RURAL DEL DEPARTAMENTO DEL HUILA BASADO EN LA INTENSIDAD DE SEÑAL RECIBIDA EN UNA RED LPWAN

JUAN SEBASTIÁN PASTRANA ARDILA
LUIS DANIEL VALENCIA GARCÍA

Trabajo de grado presentado como requisito para optar al titulo
Ingenieros Electrónicos

Director:
MARTIN DIOMEDES BRAVO OBANDO
MSc. Ingeniería de Telecomunicaciones.

Codirector:
CARLOS ALBERTO PEREZ CAMACHO
MSc. Ingeniería de Telecomunicaciones.

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
GRUPO DE INVESTIGACIÓN UNITCOM
NEIVA-HUILA
2022

Nota de aceptación

_____

_____

_____

_____

_____

_____

Firma del jurado

_____

Firma del jurado

Neiva, Huila agosto 02 de 2022.

A mi familia por apoyarme este camino, en especial a mi madre Julia Elena, quien me apoyó desde el primer día, me acompañó en los días difíciles, fue mi guía para tomar decisiones con su sabiduría, me educó y es a quien le debo quien soy hoy como persona.

A quienes me acompañaron en mis primeros pasos como investigador, con quienes compartimos la experiencia de trabajar como equipo, plantearnos retos adicionales a nuestra carrera y descubrir la esencia de un investigador, mis compañeros Camila, Jaime, Juan y Paula.

Juan Sebastián.

Agradezco a Dios por permitirme culminar esta etapa de mi vida y poder darles tal satisfacción a mis padres los cuales fueron también indispensables en este proceso y gran parte de este logro es gracias a ellos. Agradezco también a otros familiares y amigos que en determinados momentos me brindaron su apoyo moral y económico

Finalmente, a mis compañeros de carrera e ingenieros instructores que además de instruirme académicamente durante toda la carrera también me enseñaron la importancia de los buenos valores y las buenas relaciones.

Luis Daniel

AGRADECIMIENTOS

Los autores de este proyecto de investigación expresan sinceros agradecimientos a:

Los ingenieros Martín Diomedes Bravo Obando y Carlos Pérez Camacho, por brindarnos su tiempo y apoyo para la realización de este proyecto, en especial al ingeniero Martín quien nos guió sobre sistemas de comunicación, a tomar decisiones en bien y mejoría de este proyecto y al ingeniero Carlos por plantear la idea base del proyecto de investigación, inducirnos a la tecnología LoRa, darnos su apoyo en múltiples ámbitos y ser un apoyo constante antes las dificultades que surgieron en la realización de este proyecto.

A nuestros compañeros Leyder Badillo y Albert Plaza, fueron un apoyo para entender las bases de las redes LPWAN y LoRa, tecnologías que fueron una novedad en su momento y totalmente desconocida para nosotros, y aunque sus objetivos tomaron caminos distintos a los nuestros, consideramos que fueron parte de este camino.

Finalmente, a nuestros compañeros, que nos apoyaron todo este tiempo, nos motivaron a avanzar y no rendirnos, en especial a Camilo Arce, David Reina, David Arango, Heidy Rincón, Katerine Perdomo, Lina Puentes, Luisa Aristizabal, Samuel Brand.

RESUMEN

TÍTULO

ESTUDIO DE VIABILIDAD DE UN SISTEMA DE LOCALIZACIÓN EN ZONA RURAL DEL DEPARTAMENTO DEL HUILA BASADO EN LA INTENSIDAD DE LA SEÑAL RECIBIDA EN UNA RED LPWAN.

AUTORES:

JUAN SEBASTIÁN PASTRANA ARDILA
LUIS DANIEL VALENCIA GARCÍA

PALABRAS CLAVES:

Intensidad de señal recibida, LoRa, LPWAN, sistemas de comunicación, aprendizaje de máquina, modelamiento, algoritmos, predicciones.

DESCRIPCIÓN:

En este trabajo se implementó una red LPWAN basada en tecnología LoRa en un conjunto residencial del corregimiento La Ulloa en el municipio de Rivera, este fue dividido en zonas para realizar envío de mensajes y adquirir niveles de intensidad de señal en cada zona para ser utilizados posteriormente con el fin de estimar la ubicación de un nodo LoRa dentro de la zona cubierta por la red mediante el uso de un algoritmo de “Machine Learning” entrenado con estos niveles.

Una finalidad importante en este trabajo es la estimación de localización a partir del nivel de intensidad de señal recibida (RSSI) en las puertas de enlace LoRa, con el objetivo de discriminar la zona dentro de la red a la cual corresponde los niveles de intensidad obtenidos mediante el uso de algoritmos de clasificación, estos basan la estimación a partir de análisis probabilístico, a diferencia de trabajos previos donde se implementan algoritmos de regresión basados en análisis estadístico, replicando la técnica de estimación de huella digital del RSSI, se implementaron los algoritmos de Machine Learning para multiclasificación de máquina de soporte vectorial, los k-vecinos más cercanos, arboles de decisión red neuronal, se comparan los resultados de las predicciones obtenidas, determinando la red neuronal como el algoritmo más eficiente para esta localización basada en RSSI. Finalmente se evalúa el rendimiento del modelo de red neuronal con más detalle mediante la matriz de confusión, analizando las posibles fuentes de error, finalmente se proponen mejoras a este tipo de implementaciones para trabajos futuros.

ABSTRACT

TITLE

VIABILITY STUDY OF A LOCATION SYSTEM IN RURAL ZONE OF THE DEPARTMENT OF HUILA BASED ON THE INTENSITY OF THE RECEIVED SIGNAL IN A LPWAN NETWORK.

AUTHORS:

JUAN SEBASTIÁN PASTRANA ARDILA
LUIS DANIEL VALENCIA GARCÍA

KEYWORDS:

Received signal strength, LoRa, LPWAN, communication systems, machine learning, modeling, algorithms, predictions.

DESCRIPTION:

In this work, an LPWAN network based on LoRa technology was implemented in a residential complex in La Ulloa district in the municipality of Rivera, this was divided into zones to send messages and acquire signal intensity levels in each zone to be used later. in order to estimate the location of a LoRa node within the area covered by the network by using a “Machine Learning” algorithm trained with these levels.

An important purpose in this work is the estimation of location from the received signal intensity level (RSSI) in the LoRa gateways, with the aim of discriminating the area within the network to which the intensity levels obtained correspond. through the use of classification algorithms, these base the estimation from probabilistic analysis, unlike previous works where regression algorithms based on statistical analysis are implemented, replicating the RSSI fingerprint estimation technique, algorithms of Machine Learning for support vector machine multiclassification, k-nearest neighbors, neural network decision trees, the results of the predictions obtained are compared, determining the neural network as the most efficient algorithm for this location based on RSSI. Finally, the performance of the neural network model is evaluated in more detail through the confusion matrix, analyzing the possible sources of error, finally improvements to this type of implementations are proposed for future work.

CONTENIDO

LISTA DE FIGURAS.

LISTA DE TABLAS

GLOSARIO

RSSI: (Received Signal Strength Indicator) Escala de referencia para medir el nivel de potencia de las señales recibidas en una red inalámbrica.

IoT: (Internet de las cosas) Red de dispositivos físicos capaz de interactuar con datos mediante conexión a internet.

LoS: (Line of Sight) Línea de vista es la referencia al camino sin interrupciones entre antenas de emisores y receptores

GPS: (Global Positioning System) Sistema para posicionar elementos sobre la Tierra con alta precisión

LPWAN (Low Power Wide Area Network) Tecnología de comunicación inalámbrica para conectar dispositivos a larga distancia con poco consumo energético

LoRa (Long Range) Protocolo de comunicación inalámbrica que permite conectar sensores a larga distancia con poco costo energético para desarrollar aplicaciones IoT

CSS: (Chirp Spread Spectrum) Técnica de espectro ensanchado que usa módulos de frecuencia lineal en una banda ancha

FTP (File Transfer Protocol) Protocolo para transferir archivos a través de una red

API (Application Program Interface) Conjunto de definiciones y protocolos para diseñar software que ofrece alguna biblioteca.

SMOTE (Synthetic Minority Over-sampling Technique) Algoritmo de sobremuestreo para tartar conjuntos de datos con clases desequilibradas

KNN (K-Nearest Neighbor) Técnica de aprendizaje de máquina basada en distancias euclidianas para realizar predicciones

SVM (Supported Vector Machine) Técnica de aprendizaje de máquina basado en muestrear los datos en un nuevo espacio vectorial mediante vectores de soporte

ANN (Artificial Neural Network) Modelo computacional para realizar aprendizaje de máquina basado en conjuntos de unidades matemáticas conocidas como neuronas que en conjunto aprenden y evolucionan a partir de los datos.

INTRODUCCIÓN.

En la actualidad la conexión de dispositivos a internet ha tomado un enorme auge en distintos campos, recolectando y almacenando datos en internet, esta es la definición del término usado desde 1999 como "Internet de las cosas" (IoT), la posibilidad de conectar distintos dispositivos a internet ha llevado a desarrollar nuevas aplicaciones basadas en este paradigma que permitan realizar actividades cotidianas y laborales que se llevan a cabo día a día en la sociedad con mayor facilidad, lo que ha motivado el desarrollo de tecnologías que facilitan la implementación de ecosistemas IoT y puedan explotar al máximo su potencial, tal es el caso de las redes de área amplia y baja potencia LPWAN que plantean implementar ecosistemas IoT minimizando costos satisfaciendo el tipo de requerimientos de comunicación.

La localización de objetos es un tema de investigación frecuente debido al incremento de dispositivos que requieren dar a conocer su ubicación[1], actualmente se pueden encontrar diversos sistemas de localización que utilizan diferentes técnicas para encontrar la ubicación global o local de alguna persona u objeto, la mayoría de estas técnicas utilizan fenómenos físicos y relaciones matemáticas sumadas a una conjunta operación de dispositivos específicos para determinar la ubicación; estas técnicas tienen algo en común, y es el hecho de que todas necesitan un parámetro en el cual basarse para estimar la posición, este parámetro puede ser de una señal inalámbrica, una dirección IP o incluso una imagen, estas tecnologías que permiten dar una solución rápida tienen en contra su costo de implementación; a largo plazo no suele ser rentable debido a las limitantes de consumo energético o área de cobertura de estas, en este sentido las redes LPWAN han demostrado ser eficientes para reducir las limitaciones ya mencionadas[2].

Aunque ya existen investigaciones sobre sistemas de localización en distintos tipos de redes utilizando características de un sistema de comunicación, tal es el caso de la intensidad de la señal recibida en un receptor, la cual, apoyada en modelos matemáticos basados en triangulación, permite estimar la localización del transmisor en un área determinada, independiente de la información transmitida, sin embargo esta técnica resulta no ser muy eficiente en términos de precisión debido a la linealidad de estos modelos y el comportamiento no lineal de la intensidad de la señal en un sistema de comunicación; del mismo modo existe un

[1] AZMI, Nur A; SAMSUL, Shafiqa; YAMADA, Yoshihide; YAKUB, Mohd F M; ISMAIL, Mohd I M; DZIYAUDDIN, Rudzidatul A. A Survey of Localization using RSSI and TDoA Techniques in Wireless Sensor Network: System Architecture. En: *2018 2nd International Conference on Telematics and Future Generation Networks (TAFGEN)* [en linea]. Kuching, Malaysia, diciembre 2018. DOI https://doi.org/10.1109/TAFGEN.2018.8580464 E-ISBN 978-1-5386-1275-0

[2] MAHNOOR Anjum; MUHAMMAD, Abdullah K; SYED, Ali. H; AAMIR Mahmood y MIKAEL Gidlund. Analysis of RSSI Fingerprinting in LoRa Networks. En: *15th International Wireless Communications Mobile Computing Conference (IWCMC)* [en línea], Tangier, Morocco, julio 2019. p 1178-1183. https://doi.org/10.1109/IWCMC.2019.8766468 E-ISBN 978-1-5386-7747-6

auge en el uso de algoritmos de aprendizaje de máquina para resolver problemas de modelamiento no lineal a partir del uso de datos, por ende utilizar estos algoritmos con características no lineales como lo es la intensidad de la señal resulta ser una solución tentativa, método el cual tiene pocos estudios realizados hasta ahora demostrando su eficiencia[3].

En vista de lo anterior y buscando aportar a esta área de investigación en la región se realiza un estudio para determinar la viabilidad de un sistema de localización basado en el nivel de intensidad de señal recibida en una red LPWAN mediado el uso de algoritmos de aprendizaje de máquina basados en modelos probabilísticos, revisando la eficiencia de implementar este tipo de sistemas y sus respectivas limitantes.

[3] DARAMOUSKAS, Ioannis; KAPOULAS, Vaggelis y PARASKEVAS, Michael. Using Neural Networks for RSSI Location Estimation in LoRa Networks. En: *10th International Conference on Information, Intelligence, Systems and Applications (IISA)* [en línea], Patras, Greece, noviembre 2019, p. 1-7. DOI https://doi.org/10.1109/IISA.2019.8900742 E-ISBN 978-1-7281-4959-2

OBJETIVOS.

OBJETIVO GENERAL.

Realizar un estudio de viabilidad de un sistema de localización basado en la fuerza de la señal recibida en una red LPWAN.

OBJETIVOS ESPECÍFICOS.

- Examinar dispositivos de la tecnología LPWAN LoRa que se adecue a un sistema de comunicación, comprendiendo su funcionamiento, ventajas y sus limitantes para transmisión.
- Implementar un sistema de comunicación LPWAN en una zona rural delimitada por los gateways LoRa para obtener valores del indicador de intensidad de señal recibida en distintos puntos de un área en el corregimiento la Ulloa del municipio de Rivera.
- Registrar los parámetros parciales de mediciones del RSSI de acuerdo con la ubicación del nodo respecto a los gateways.
- Determinar un modelo que establezca una relación entre el RSSI recibida en los gateways y la ubicación del nodo LoRa.
- Validar el modelo de red neuronal propuesto con datos y pruebas en un entorno real, definiendo límites, margen de error y precisión de este

ANTECEDENTES.

Números trabajos se han realizado para la implementación de sistemas de localización basados en redes LPWAN como LoRa o Sigfox, algunos de estos tienen como base el uso del indicador RSSI para estimar localización:

"Las técnicas de localización y seguimiento enfrentan en el actual escenario de Internet de las Cosas"[4]. Con el fin de dar soluciones de localización en agricultura y ganadería se implementa un sistema de rastreo utilizando un dispositivo GPS y una estación base con protocolos de red LPWAN, validando el sistema con datos reales y dejando abierta la propuesta a la implementación de una API para monitorear varios geteways a la vez.

[5]Se implementa un sistema de localización entre un terminal móvil y una estación base con protocolos LPWAN, basado en la propagación de la señal dado que un dispositivo GPS consume mucha energía, finalmente el sistema implementado con una precisión similar a la de un GPS (90%) y que se debe seguir explorando para tomar decisiones más razonables respecto a la localización.

[6]LoraLoc machine learning based fingerprinting for outdoor Geolocation using Lora. Un desafío en la actualidad implementar sistemas de geolocalización usando dispositivos LoRa cuando no existe línea de vista, por lo tanto, se presenta una solución basada en el uso de diferencia de hora de llegada (TDOA) y algoritmos de Machine Learning cuyos datos verdaderos son tomados mediante el uso de GPS, verificados en simulación con técnicas de redes neuronales, bosques aleatorios y arquitecturas de redes neuronales recurrentes.

[7] A Comparison of Signal Strength Localization Methods with Sigfox. Se presenta una comparación de tres métodos de proximidad, un método de huellas digitales y tres métodos de alcance con un error de 586 metros en entornos sin línea de vista.

---

[4] DA SILVA, Wesley R; OLIVEIRA Luiz; KUMAR, Neerai; RABELO, Ricardo A.L; MARINS, Carlos N.M y RODRIGUEZ Joel J. P. C. An Internet of Things Tracking System Approach Based on LoRa Protocol. En: *IEEE Global Communications Conference (GLOBECOM)* [en línea], Abu Dhabi, United Arab Emirates, febrero 2019, p. 1-7. DOI https://doi.org/ 10.1109/GLOCOM.2018.8647984 E-ISBN 978-1-5386-4727-1

[5] LEMIC, Filip; BEHBOODI, Arash; FAMAEY, Jeroen y MATHAR, Rudolf. Location-Based Discovery and Vertical Handover in Heterogeneous Low-Power Wide-Area Networks (2019). En: *IEEE Internet of Things Journa* [en línea]*I*, marzo 2018, vol. 6, nro. 6, p 10150-10165. DOI https://doi.org/10.1109/JIOT.2019.2935804

[6] FRANCESCO, Carrino; ALES, Janka; OMAR, Abou K y ELENA, Mugellini. LoRaLoc: Machine Learning-Based Fingerprinting for Outdoor Geolocation using LoRa. En*: 6th Swiss Conference on Data Science (SDS)* [en línea], Bern, Switzerland, agosto 2019, p. 82-86. https://doi.org/10.1109/SDS.2019.000-2 E-ISBN 978-1-7281-3105-4

[7] BELLENSKENS, Ben; AERNOUTS, Michiel; BERKENS, Rafael y WEYN, Maarten A Comparison of Signal Strength Localization Methods with Sigfox. En: 2018 15th Workshop on Positioning, Navigation and Communications (WPNC) [en línea]. Breme, Germany diciembre 2018. DOI https://doi.org/10.1109/WPNC.2018.8555743 E-ISBN 978-1-5386-6436-0

[2]Analysis of RSSI Fingerprinting in LoRa Networks. Investigación de la idoneidad de la tecnología LoRa para implementar un sistema de posicionamiento utilizando la intensidad de la señal recibida huella digital del indicador de intensidad de señal recibida (RSSI) con línea de vista y sin línea de vista y huellas digitales como el uso de diferencia horaria de llegada (TDOA) mostrando que estos pueden ser utilizados para hacer sistemas de localización robustos.

[8]A Survey of Localization using RSSI and TDoA Techniques in Wireless Sensor Network: System Architecture. Se propone un sistema de localización usando redes de largo alcance (LoRa) sin utilizar GPS mediante distintas técnicas basadas en distancia como el uso de la intensidad de señal recibida.

[9]Evaluating indoor and outdoor localization services for LoRaWAN in Smart City applications. En este trabajo, los autores hacen uso conjunto de LPWAN y técnicas de localización aceptadas y sistemas de localización en tiempo real para aplicaciones de Smart Campus. Los resultados experimentales demuestran la viabilidad del enfoque propuesto; en particular, los errores de ubicación están en el orden de pocas decenas de metros para GPS.

[10]Low-Cost Car Park Localization Using RSSI in Supervised LoRa Mesh Networks. Se implementa un sistema de localización basado en redes LoRa y RSSI, diseñado para ubicar autos en grandes concesionarios, se deja claro que estos sistemas deben ser calibrados para distintos ambientes.

[11]LoRaIn: Making a Case for LoRa in Indoor Localization. Se analiza la viabilidad de usar LoRa en localización de interiores con espacios variados en obstáculos como paredes y objetos, considerando la cobertura, estabilidad y regularidad de las señales, precisión de localización, capacidad de respuesta, poder y costo se concluyó que LoRa es una opción factible para solución de localización en interiores.

---

[8] AZMI, Nur A; SAMSUL, Shafiqa; YAMADA, Yoshihide; YAKUB, Mohd F M; ISMAIL, Mohd I M; DZIYAUDDIN, Rudzidatul A. A Survey of Localization using RSSI and TDoA Techniques in Wireless Sensor Network: System Architecture. En: *2018 2nd International Conference on Telematics and Future Generation Networks (TAFGEN)* [en linea]. Kuching, Malaysia, diciembre 2018. DOI https://doi.org/10.1109/TAFGEN.2018.8580464 E-ISBN 978-1-5386-1275-0

[9] BONANAFI, F; FERNANDEZ, Carvalho D; DEPARI, A; FERRARI, P; FLAMMINI, A; PASETTI M; RINALDI, S y SISINNI, E. Evaluating indoor and outdoor localization services for LoRaWAN in Smart City applications. En: *II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 IoT)* [en línea]. Naples, Italy, agosto 2019, p. 300-305. DOI https://doi.org/10.1109/METROI4.2019.8792901 E-ISBN 978-1-7281-0429-4

[10] GOTTHARD Petr y JANKECHK Tomáš. Low-Cost Car Park Localization Using RSSI in Supervised LoRa Mesh Networks. En: *15th Workshop on Positioning, Navigation and Communications (WPNC)* [en línea], Bremen, Germany, diciembre 2018, p. 1-6. DOI https://doi.org/10.1109/WPNC.2018.8555792 E-ISBN 978-1-5386-6436-0

[11] ISLAM, Bashima; ISLAM, Md Tamzeed y KAUR, Jasleen. LoRaIn: Making a Case for LoRa in Indoor Localization. En: *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* [en línea], Kyoto, Japan, p. 423-426. DOI https://doi.org/10.1109/PERCOMW.2019.8730767 E-ISBN 978-1-5386-9151-9

[12]Outdoor Fingerprinting Localization using Sigfox. Se implementa un sistema de localización haciendo uso de redes LPWAN, el indicador de intensidad de señal recibida RSSI y el algoritmo k-vecinos más cercanos kNN, si bien los resultados no son los óptimos, se observa la viabilidad de uso del RSSI como mecanismo para lograr localización de objetos y se deja en claro la necesidad de obtener más datos para alimentar el algoritmo kNN para una mejor precisión.

[13]New RSSI-based LoRa localization Algorithms for Very Noisy Outdoor Environment. Partiendo de estudios que comprueban la utilidad de usar dispositivos LoRa en exteriores y la eficiencia de usar algoritmos basados en la RSSI se proponen dos algoritmos nuevos para ambientes ruidosos.

[14]Towards Location Enhanced IoT: Characterization of LoRa Signal For Wide Area Localization. Este documento proporciona una evaluación sistemática de sistemas de ubicación LoRa haciendo uso de RSSI tanto interior como exterior con porcentajes de error a favor de esta tecnología por sus amplios rangos y costos.

[15]Se realiza un estudio de un sistema de localización utilizando un base de datos en los cuales se utiliza el método de huella digital para estimar localización, comparan los resultados de este estudio con el modelo implementado usando redes neuronales artificiales, probando distintas arquitecturas y métodos de procesamiento de datos logrando un margen de error de 300 metros, el cual se considera aceptable debido a que la tecnología LoRa abarca grandes cantidades de terreno, por lo cual esta incertidumbre puede ser considerablemente útil en ciertos casos de estimación de localización

Al realizar la búsqueda a nivel nacional, no se encuentran trabajos de grado que usen LPWAN para implementar sistemas de localización, de hecho, se encuentran muy pocos trabajos realizados con esta tecnología, sin embargo, se encontró un sistema de localización basado en RSSI con redes Wi-fi

---

[12] JANSSEN, Thomas; AERNOUTS, Michiel; BERKENS, Rafael y WEYN, Maarten. Outdoor Fingerprinting Localization Using Sigfox. En: *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)* [en línea], Nantes, France, noviembre 2018, p 1-6. DOI https://doi.org/10.1109/IPIN.2018.8533826 E-ISBN 978-1-5386-5635-8

[13] LAM, Ka-Ho; CHEUNG, Chi-Chung y LEE, Wah-Ching. New RSSI-Based LoRa Localization Algorithms for Very Noisy Outdoor Environment. En: *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)* [en línea]. Tokyo, Japan, junio 2018, p. 794-799. DOI https://doi.org/10.1109/COMPSAC.2018.10340 E-ISBN 978-1-5386-2667-2

[14] LI, You; HE, Zhe; LI, Yuqi; XU, Hongliang; Pei, Ling y Zhang, Yu. Towards Location Enhanced IoT: Characterization of LoRa Signal For Wide Area Localization. En: *2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS)* [En línea]. Wuhan, China. diciembre 2018, p. 1-7. DOI https://doi.org/10.1109/UPINLBS.2018.8559844 E-ISBN 978-1-5386-3755-5

[15] DARAMOUSKAS, Ioannis; KAPOULAS, Vaggelis y PARASKEVAS, Michael. Using Neural Networks for RSSI Location Estimation in LoRa Networks. En: *10th International Conference on Information, Intelligence, Systems and Applications (IISA)* [en línea], Patras, Greece, noviembre 2019, p. 1-7. DOI https://doi.org/10.1109/IISA.2019.8900742 E-ISBN 978-1-7281-4959-2

[16]Aplicativo móvil para localizar y guiar personas a través del campus universitario de la Pontificia Universidad Javeriana. Se implementó un aplicativo móvil para estimar la ubicación de personas en tiempo real en zonas del edificio del edificio de ingeniería de la Pontificia Universidad Javeriana, mediante el uso de la intensidad de señal Wi-Fi en distintos puntos y el método de predicción k-vecinos más cercano KNN.

A nivel regional en la Revista Ingeniería y Región de la Universidad Surcolombiana se puede encontrar un artículo de revisión sobre los métodos utilizados para detectar localización de objetos ToA, DToA y AoA, donde se concluye que la precisión de estas técnicas es muy afectada por el ruido del ambiente e interferencia causada por elementos que interfieran en la red [17]

[16] CRUZ, Gómez D. E y MENDOZA, Ortiz J. C. Aplicativo móvil para localizar y guiar personas a través del campus universitario de la Pontificia Universidad Javeriana. En: *Pontificia Universidad Javeriana* [en línea]. Bogota: Universidad Javeriana, 2016, Disponible en: http://repository.javeriana.edu.co/handle/10554/21446

[17] GARCIA, A. F; GOMEZ, C; SANCHEZ, T; RENDONDO, A. D; BETANCOUR, L y HINCAPIE, R. C. Algoritmos de Radiolocalización basados en ToA, TDoA y AoA. En: *Ingeniería y Región* [en linea], Neiva: Universidad Surcolombiana, febrero 2016 v.14, p. 9-22. DOI https://doi.org/10.25054/22161325.689

1. CAPÍTULO UNO: FUNDAMENTOS BÁSICOS

1.1 REDES LPWAN.

Las redes LPWAN (low-power wide-area network) representan una alternativa de red inalámbrica y evolución de tecnologías IoT[18], cuenta con características deseables en las que destacan largo alcance de transmisión y bajo consumo energético, al mismo tiempo presenta una solución de bajo costo económico [19], como contrapartida se presenta baja velocidad de transmisión de datos, capacidad de cálculo limitada al hardware de los dispositivos finales, por esta razón las redes LPWAN suelen estar integradas a una arquitectura inalámbrica en la nube, por último presenta baja tasa de transmisión de datos aunque esta última no es fundamental en una red IoT.

Es posible encontrar redes LPWAN con rango de transmisión entre 10-40 km en zonas rurales y 5 km en zonas urbanas, consumo energético reducido, permitiendo una duración de la batería de hasta 10 años, estas características acompañadas al bajo costo de los dispositivos consolidaron las redes LPWAN con un futuro prometedor para implementar aplicaciones IoT que requieran transmitir pocos datos, impulsando estudios experimentales en ambientes internos y externos y el surgimiento de tecnologías emergentes como NB-IoT, Sigfox y LoRa[20], aunque estas utilizan técnicas de modulación diferentes, comparten similitud en la arquitectura de red, conformada por tres capas. La figura 1.1 muestra una representación de una red LPWAN estándar y sus capas.

[18] SONG, Yonghua; LIN, Jin; TANG, Ming y DONG, Shufeng. An Internet of Energy Things Based on Wireless LPWAN. En: *Engineering* [en línea]. Chinese Academy of Engineering, agosto 2017. vol. 3, nro. 4. p. 2. DOI https://doi.org/10.1016/J.ENG.2017.04.011. ISSN 2095-8099

[19] MEKKI, Kais; BAJIC, Eddy; CHAXEL, Frederic y MEYER, Fernand. Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT. En: *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* [en linea]. Athens, Greece, octubre 2018. p. 1. DOI https://doi.org/10.1109/PERCOMW.2018.8480255 E-ISSN 978-1-5386-3227-7

[20] MEKKI, Kais; BAJIC, Eddy; CHAXEL, Frederic y MEYER, Fernand. A comparative study of LPWAN technologies for large-scale IoT deployment. En: *ICT Express* [en linea]. The Korean Institute of Communications Information Sciences, marzo 2019. vol. 5, nro. 1. p. 1. DOI https://doi.org/10.1016/j.icte.2017.12.005 ISSN 2405-9595

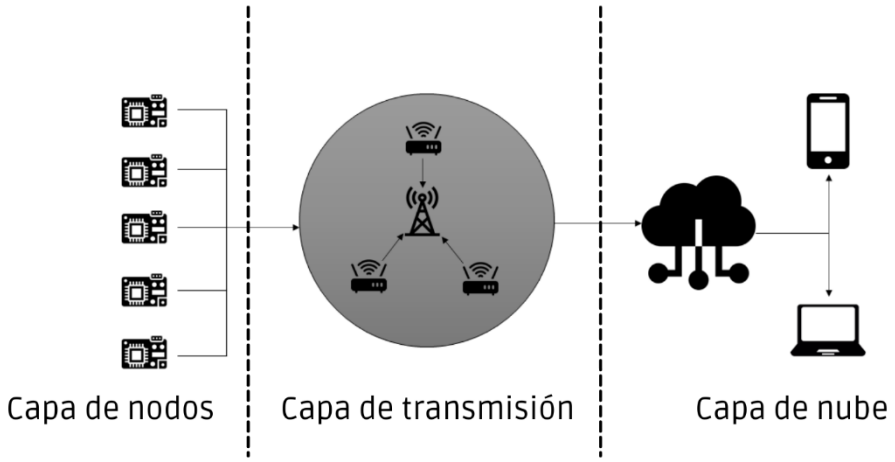


Figura No. 1.1 Arquitectura de una red LPWAN para desarrollo de aplicaciones IoT

En primer lugar se encuentra la capa de nodos, donde es posible encontrar gran cantidad de sensores que envían información por la red hasta la nube y actuadores que reciben ordenes provenientes de esta, en segundo lugar se considera como capa de transmisión todos los dispositivos dispersos geográficamente que ofrecen canales inalámbricos de comunicación, en esta capa es posible encontrar grandes distancias entre dispositivo y dispositivo que en redes tales como WiFi o ZigBee resultan imposibles de implementar, en la figura 1.2 se puede observar una comparativa de distancia. Así mismo se suelen encontrar redes con topología de estrella en donde existe una única estación base a la cual envían información todos los nodos, finalmente se encuentra la capa de nube, la cual contiene aplicaciones e intercambia datos entre el usuario final y la capa de nodos.

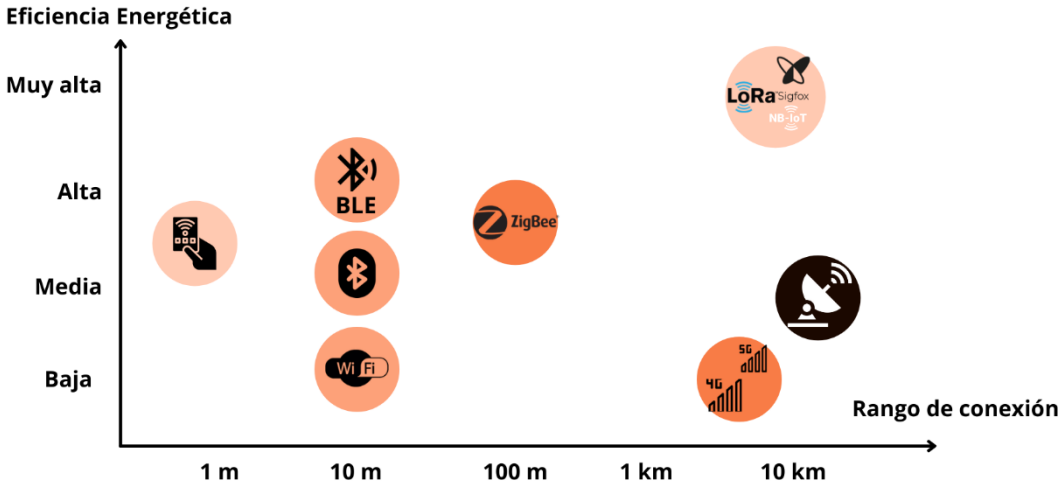


Figura No. 1.2 Comparación distancia entre tecnologías inalámbricas

1.2 LoRa.

LoRa es una tecnología inalámbrica, consiste en la técnica de modulación de radiofrecuencia patentada por Semtech, los dispositivos que hacen uso de esta y las redes de comunicación implementadas para desarrollar aplicaciones IoT desarrolladas a partir de estos [21].

La capa física de la red LoRa utiliza una variante de la modulación de espectro ensanchado o SSM, conocida como chirp spread spectrum o CSS. SSM codifica la señal original con una secuencia de alta frecuencia, aumentando el ancho de banda de la señal, pero reduciendo la potencia presente en esta y mejorando ante interferencias electromagnética, para el caso de CSS, la señal es codificada con una señal sinusoidal de frecuencia modulada en banda ancha, esta tendrá variaciones en el tiempo, aumentando o disminuyendo y es conocida como un pitido o "chirp"[22]. Adicionalmente LoRa permite modificar la cantidad de bits para codificar un símbolo, este valor es conocido como factor de propagación SF, para el caso de LoRa se permite usar seis valores de SF (7 - 12 bits), tres anchos de banda (125KHz, 250 KHz y 500KHz) regulados según el continente donde se implementa LoRa.

LoRa opera en la banda ISM, libre en todo el mundo con fines no comerciales, con variaciones en las frecuencias utilizadas en continentes, 915 MHz para América 868MHz para Europa, reduciendo el costo de implementación al poder operar en una red propia sin pagar licencias para uso de espectro, siendo la primera implementación de CSS de manera comercial y a bajo costo.

En una red LoRa existen dos tipos de dispositivos que serán el eje de toda la arquitectura, los dispositivos finales o nodos, encargados de monitorizar, encargados de interactuar con el ambiente donde están ubicados como receptores de información proveniente de sensores o controlar los actuadores mediante ordenes provenientes de la red LoRa, al mismo tiempo operan los gateways, pues son el puente de la red LoRa entre los nodos y las señales de comunicación estándar para acceso a internet , enviando información proveniente de los nodos a un servidor en la nube o entregando a los nodos información proveniente de esta.
Respecto al protocolo de conexión a internet, existe la posibilidad de usar la especificación LoRaWAN, la cual abarca el conjunto de protocolos de comunicaciones y arquitectura de red [23], siendo esta adoptada por algunas compañías en el mundo para brindar conectividad, sin embargo no siempre es

[21] RAY, Brian. What Is LoRa? A Technical Breakdown [blog]. Blogs *Link Labs*. 26 de junio de 2018. Disponible en https://www.link-labs.com/blog/what-is-lora

[22] PICKERING, Paul. Desarrollar con LoRa para aplicaciones IoT de baja tasa y largo alcance [blog]. Blogs *Digi-Key*. 29 de junio de 2017. Disponible en https://www.digikey.com/es/articles/develop-lora-for-low-rate-long-range-iot-applications

[23] Lora Alliance. A technical overview of LoRa® and LoRaWAN® [documento digital]. *Lora Alliance - Resources*, noviembre 2015. Disponible en https://lora-alliance.org/resource_hub/what-is-lorawan/

posible usar LoRaWAN , motivo por el cual la red puede ser adaptada al medio de implementación mediante el protocolo LoRa-MAC o LoRa RAW, siendo este protocolo el adecuado para zonas donde el acceso a internet es limitado.

1.3 MACHINE LEARNING Y DEEP LEARNING.

El aprendizaje automático o *machine learning* es un área de estudio en el campo de la Inteligencia Artificial que estudia modelos estadísticos y algoritmos a fin de que un sistema de cómputo pueda lograr realizar tareas sin ser estas programadas directamente, esos sistemas logran encontrar patrones en un conjunto de datos y a futuro utilizará realizando nuevas predicciones a partir de nuevos datos, entre mayor cantidad y mejor calidad de datos, el algoritmo encontrado logrará predicciones precisas y fiables, una definición muy aceptada en ingeniería dicta:

"Se dice que un programa de computadora aprende de experiencia E, con respecto a alguna tarea T y alguna medida de desempeño P, mejora con experiencia E, si su desempeño en T, como fue medido con P, mejora con experiencia E." [24]

Con respecto a desarrollar un sistema mediante de machine learning, es importante resaltar que su finalidad es realizar una predicción, esta puede estar en un conjunto de valores continuos (Regresión) o un conjunto de valores discretos (Clasificación), estas predicciones se evaluaran a fin de determinar qué tan acertadas son lo que permitirá al programador ajustar el modelo a partir de una métrica estadística, realizando este proceso cíclicamente hasta obtener un modelo adecuado al conjunto de datos.

1.3.1 MÁQUINAS DE SOPORTE VECTORIAL.

Las máquinas de soporte vectorial (SVM) son un algoritmo de Machine Learning que surgieron como evolución de la clásica regresión lineal, muy utilizado cuando los conjuntos de datos para clasificar son bastantes complejos y tienen una cantidad pequeña o mediana en cantidad de datos para aprendizaje [25], aunque en la regresión lineal se puede lograr desarrollar un hiperplano lineal que logre separar las clases en el conjunto de datos, este no siempre es óptimo, SVM trazará nuevos hiperplanos paralelos al original con ayuda de un vector que determinará la posición hiperplano paralelo más cercano a cada clase, este vector es conocido como vector de soporte y dan nombre al algoritmo, así mismo, la distancia entre los nuevos

[24] MITCHEL, Tom Michael. Machine Learning [en línea]. 1 ed. *McGraw-Hill Education*. 1997, 870-877 p. Disponible en https://www.amazon.com/Machine-Learning-Tom-M-Mitchell/dp/0070428077 ISBN 0070428077

[25] GERON, Aurelien. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems [en linea]. 2 ed. *O'Reilly Media, Inc* 2019. Disponible en https://www.amazon.com/-/es/Aur-C3-A9lien-G-C3-A9ron-dp-1492032646/dp/1492032646/ref=dp_ob_image_bk ISBN: 9781492032649

hiperplanos será conocida como margen máximo, la ecuación principal para una SVM, como puede ser observada a continuación, describe una combinación lineal de las entradas, siendo X el vector de las entradas y β el vector de parámetros para ponderar las entradas, finalmente β_0 corresponderá al sesgo introducido en el modelo.

$$f(x) = X^T\beta + \beta_0 \quad (1)$$

Evidentemente, estos hiperplanos no serán siempre lineales, sin embargo, SVM adopta el uso de funciones matemáticas conocidas como kernel que trasladará el conjunto de datos a un nuevo espacio matemático donde puede lograr una solución lineal, una vez lograda esta se transforma al espacio original [26].

1.3.2 LOS K-VECINOS MÁS CERCANOS.

Los K-vecinos más cercanos consisten en uno de los algoritmos de machine learning más sencillos, consiste en determinar la clase a la cual un dato (*xo*) pertenece mediante el uso de métodos de similitud entre este dato y una cantidad (*k*) de puntos ($x_{(i)}$) cercanas al dato, tal es el caso de la distancia euclidiana, así mismo la determinación de la clase se hace en base a la mayoría de clases más cercanas obtenidas y no requiere entrenar un modelo, motivo por el cual se le conoce este modelo como un clasificador basado en memoria [27].

$$d_{(i)} = \left\| x_{(i)} - x_o \right\| \quad (2)$$

El factor característico de este método y su único hiperparámetro es *k,* sin embargo, no existe un método preciso para determinar el mejor valor para este, por lo cual su búsqueda se convierte en un método heurístico de ensayo y error, no obstante es muy común encontrar situaciones similares donde un valor pequeño de *k* logra predicciones poco precisas, por su parte valores grandes implica menor varianza en la predicción pero un mayor sesgo, además de malas predicciones ante datos desconocidos y costo computacional más elevado.

En conclusión, este algoritmo es sencillo de implementar debido a la presencia de un único hiperparámetro, su matemática lo hace idóneo para clasificación y multiclasificación, al mismo tiempo el calcular cada distancia propuesta en él implica considerar los datos de entrenamiento como parte del modelo, siendo estos

[26] VERSLOOT, Christian. Understanding SVM and SVR for Classification and Regression [blog]. Blogs *MACHINECURVE* 20 de septiembre de 2019. Disponible en https://www.machinecurve.com/index.php/2019/09/20/intuitively-understanding-svm-and-svr/

[27] HASTIE, Trevor; TIBSHIRANI, Robert y FRIEDMAN, Jerome. The Elements of Statistical Learning: Data Mining, Inference, and Prediction [en linea]. 2 ed. *Springer* 2016. Disponible en https://www.amazon.com/-/es/Trevor-Hastie/dp/0387848576 ISBN: 0387848576

utilizados para realizar los cálculos, motivo por el cual deben ser almacenados para ser utilizados en el posterior cálculo de distancias, aumentando los tiempos de procesamiento.

1.3.3 ÁRBOL DE DECISIÓN.

El árbol de decisión es un algoritmo de machine learning que analiza las variables predictoras y todos los posibles conjuntos de división sobre estas para generar dos o más conjuntos de datos nuevos homogéneos basados en la característica que más significativa encuentre en estos.

Las variables más significativas son puestas en la cima del árbol y a partir de estas se generan dos nuevos conjuntos de datos o regiones a las cuales el algoritmo establecerá la pertenencia a estas regiones mediante condiciones o desigualdades matemáticas se consiguen calculando el índice de impureza de Gini, descrita en la ecuación 3 donde P_{Ci} corresponde a la probabilidad de seleccionar aleatoriamente una variable de una clase C_i, el índice de impureza de Gini representa el valor esperado de clasificaciones incorrectas sí la clasificación se realiza de manera aletoria.

$$Gini = 1 - \sum_{i=1}^{n} P^2(C_i) \tag{3}$$

La probabilidad de elegir aleatoriamente una muestra de la clase $C_i$es $P(C_i)$, por el contrario, la probabilidad de predecir la clase incorrecta es $1 - P(C_i)$. Realizando la suma $P(C_i)$*$(1 - P(C_i))$ sobre todas las clases obtenemos la fórmula para la impureza de Gini

La idea detrás de un árbol de decisión es lograr dividir los datos en nuevos conjuntos, de manera similar crecer el árbol, decidir las características que serán condiciones para dividir, matemáticamente una técnica común conocida como división binaria recursiva que mediante alguna función de costo generará las condiciones para lograr la división de menor costo.

1.3.4 CLASIFICADOR BAYESIANO INGENUO.

Los clasificadores bayesianos son un algoritmo probabilístico y estadístico utilizado para determinar si una muestra pertenece a una clase, específicamente fundamentado en el teorema de Bayes, mientas que otros algoritmos parten de correlacionar los datos de entrada utilizados en el entrenamiento, estos asumen que

las variables predictoras tienen valores independientes entre ellas, es decir, no consideran que exista una correlación entre variables, esta hipótesis permite simplificar los métodos de computación, por esta razón son conocidos como ingenuos.

Cabe destacar el eje principal de este clasificador, el teorema de Bayes, busca la probabilidad de cada combinación específica de variables predictoras asignadas a una clase, esto implicaría grandes cantidades de datos, sin embargo, recordando que este algoritmo asume que estas son independientes entre sí minimiza el cálculo, este cálculo es utilizado en el teorema de Bayes que en literatura puede ser descrito de la forma:

$$Conocimiento\ Posterior = \frac{Conocimiento\ Previo * Probabilidad\ de\ ocurrir}{Evidencia} \quad (4)$$

En la práctica, representando como y las salidas o clases a predecir y X como las características o entradas, la ecuación 4 puede ser descrita matemáticamente como:

$$p(y_i|X_1, X_2, \dots X_n) = \frac{p(X_1, X_2, \dots X_n|y_i).p(y_i)}{p(X_1, X_2, \dots X_n)} \quad (5)$$

Sobre todo, es importante analizar el numerador, debido a que el denominador es constante puede ser eliminado para reducir costos computacionales, la primera parte del numerador o conocimiento previo muestra la probabilidad de cada combinación de entrada para la clase a predecir, debido a la previa hipótesis de variables predictoras independientes esta puede ser omitida, por otro parte la probabilidad de ocurrir esta simplemente está descrita lingüísticamente como:

El algoritmo almacenará las distribuciones de probabilidad para cada clase independientemente, similar al algoritmo de K-Vecinos más cercanos se evaluará para cada clase y se elegirá la que obtenga mayor probabilidad

1.3.5 REDES NEURONALES ARTIFICIALES.

Durante la historia de la humanidad, el ser humano desarrolló soluciones para facilitar tareas que realizaba a diario, algunas de estas fueron inspiradas en la naturaleza, de la misma forma ocurrió al explorar el cuerpo humano mismo, donde la arquitectura del cerebro sirvió de inspiración para desarrollar el concepto de redes neuronales artificiales (RNA)[25]

Resulta lógico pensar que las RNA en sus inicios fueron bastante primitivas, siendo únicamente útiles en problemas de cómputo binario[28] donde se estableció la característica de una neurona artificial para activar su salida en función de las entradas y conectar esta salida a otra neurona para implementar versiones más avanzadas de cómputo, con el tiempo este problema se enfocaría a resolver problemas mayores donde las entradas y salidas dejarían atrás valores binarios y se enfocarían en valores numéricos continuos siendo esta arquitectura básica pero muy implementada, conocida como el perceptrón, el cual asociaba cada entrada de una neurona a un peso y la salida de esta correspondería a la suma ponderada de estas entradas añadiendo una función no lineal que activará cada neurona, esta ecuación es descrita como:

$$Z = \sigma(W_1X_1 + W_2X_2 + \cdots + W_nX_n + b) = \sigma(X^TW + b) \quad (6)$$

Donde σ representa la función no lineal conocida como función de activación y puede variar dado que existen distintas funciones de activación adecuadas para ciertas ocasiones. Por esto puede definirse una neurona como una transformación no lineal a características lineales de las entradas (X^T) ponderadas en función de un vector de parámetros conocidos como pesos de la neurona (W) y un sesgo añadido (b) ,finalmente una red neuronal como un conjunto de neuronas conectadas entre sí y serán conocidos como capas en principio salidas a entradas desde la primera capa o capa de entrada encargada de recibir las variables predictoras, estas entregan sus salidas a las siguientes capas que se encargaran de añadir más características no lineales al modelo y son conocidas como capas ocultas, por último las capas ocultas entregan a la última capa, encargada de dar la predicción, será conocida como capa de salida.

Cabe destacar que el proceso de diseño de una red neuronal abarca desde seleccionar una arquitectura de conexión entre neuronas, el tipo de función de activación, el método de aprendizaje, tipo de capa final que puede variar para regresión o clasificación, evidentemente es un algoritmo muy completo si se compara con los ya mencionados de machine learning, sin embargo el precio a pagar para implementar este tipo de algoritmo es un elevado aumento de costo computacional en función de la cantidad de neuronas y capas implementadas.

[25] GERON, Aurelien. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems [en linea]. 2 ed. *O'Reilly Media, Inc* 2019. Disponible en https://www.amazon.com/-/es/Aur-C3-A9lien-G-C3-A9ron-dp-1492032646/dp/1492032646/ref=dp_ob_image_bk ISBN: 9781492032649

[28] MCCULLOCH, Warren Sturgis y WALTER, Pitts. A logical calculus of the ideas immanent in nervous activity. En: *Springer - Bulletin of Mathematical* Biophysics. The University of Chicago, Chicago, USA, diciembre 1943. vol. 5, nro. 1. DOI https://doi.org/10.1007/BF02478259.

2. CAPÍTULO DOS: CONFIGURACIÓN DE RED LORA.

En este capítulo se describen los dispositivos (gateway y nodos) utilizados para implementar la red LoRa, como acceder a ellos, configurar los parámetros de red, la instalación de librerías y uso de estas para programar las rutinas para realizar el envío de mensajes y almacenamiento de información de valores de RSSI y zona de envío de mensajes en la memoria ROM de cada gateway.

2.1 DISPOSITIVOS UTILIZADOS.

2.1.1 DRAGINO LORA SHIELD.

Es un módulo de Arduino desarrollado pro Dragino que permite enviar y recibir datos utilizando la tecnología LoRa, en otras palabras, es un transceptor (transmisor-receptor), que permite enviar pequeños paquetes de datos por medio de señales de radiofrecuencia, capaces de alcanzar rangos considerablemente amplios con un bajo consumo energético[29].

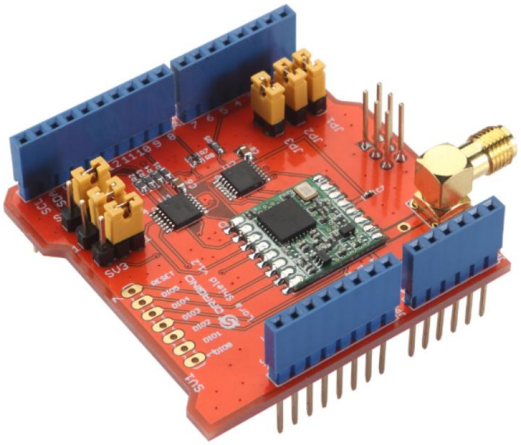


Figura No. 2.1 Módulo Dragino LoRa Shield.

Este módulo utiliza el chip transceptor SX1276/SX1277 desarrollado y fabricado por Semtech, módulo desarrollado para operar en el rango de frecuencias entre 137MHz y 1020MMhz usando la técnica de modulación LoRa patentada por la misma compañía para lograr mayor alcance de comunicación, menor consumo energético y mayor inmunidad a interferencias.

El módulo permite la modulación LoRa con un rango dinámico de 127 dB, de la misma manera otros dispositivos como sensores o actuadores pueden ser integrados para desarrollar una red IoT basada en LoRa accediendo a las entradas

[29] Dragino. Lora Shield [en linea]. *Dragino,* julio 2016. Disponible en https://wiki.dragino.com/index.php?title=Lora_Shield

digitales y analógicas que puede ser observadas en la figura 2.2, facilitando la operación como emisor o receptor, para la red LoRa implementada se utilizó como emisor.

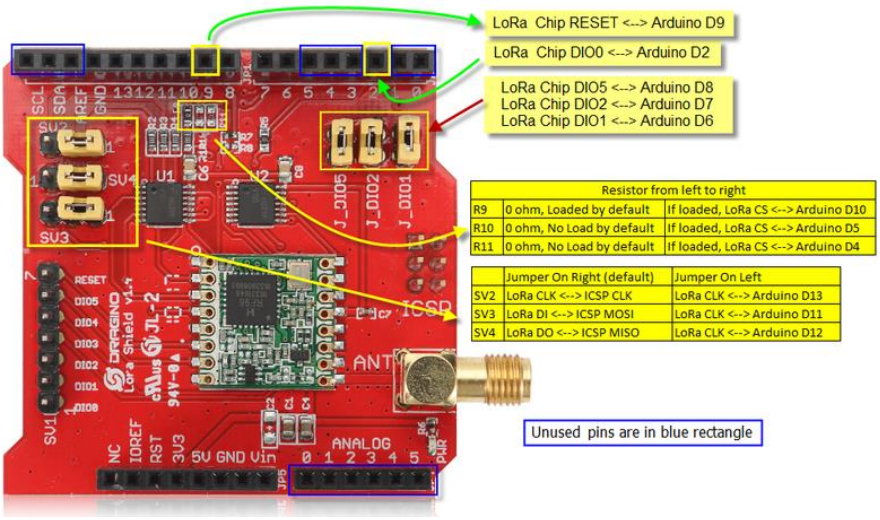


Figura No. 2.2 Diagrama de pines Dragino LoRa Shield

2.1.2 LG01 LORA GATEWAY.

LG01 es un gateway LoRa de la compañía Dragino, código abierto y de un canal, permite conectar la red LoRa a una red IP mediante protocolo WiFi, ethernet o telefonía 3G/4G permitiendo implementar una red inalámbrica destinada a IoT con ventajas típicas de una red LoRa. La arquitectura de este dispositivo observada en la figura 2.4 dada por el fabricante permite visualizar los dos componentes principales del dispositivo

En primer lugar, se encuentra el microcontrolador ATMega 328P conectado por un bus de datos SPI al chip transceptor SX1276/SX1277 encargado de realizar los procesos de la red desde el lado de LoRa, como emisor o receptor de mensajes además de proveer funciones adicionales programables mediante el entorno de desarrollo de Arduino, algunos modelos como el LG01-S permiten integrar sensores directamente a este microcontrolador para expandir las aplicaciones de la red.

En segundo lugar, se encuentra el módulo Dragino HE AR933, esta parte del gateway controla el acceso a la red IP y puede ser configurado mediante el entorno basado en Linux Open Wrt, accediendo por medio del protocolo de acceso remoto SSH, además de contar con memoria flash de 16MB y memoria RAM de 64MB, además de la posibilidad de añadir almacenamiento externo por medio de la interfaz USB

Ambos entornos se encuentran conectados mediante los protocolos de interfaz periférica serial (SPI) y transmisor-receptor asíncrono universal (UART), lo que facilita el intercambio de información entre el entorno Arduino y el entorno Linux, esto permite manipular el gateway desde el entorno de Arduino con mayor facilidad y enviar al módulo Linux para el posterior almacenamiento de información.

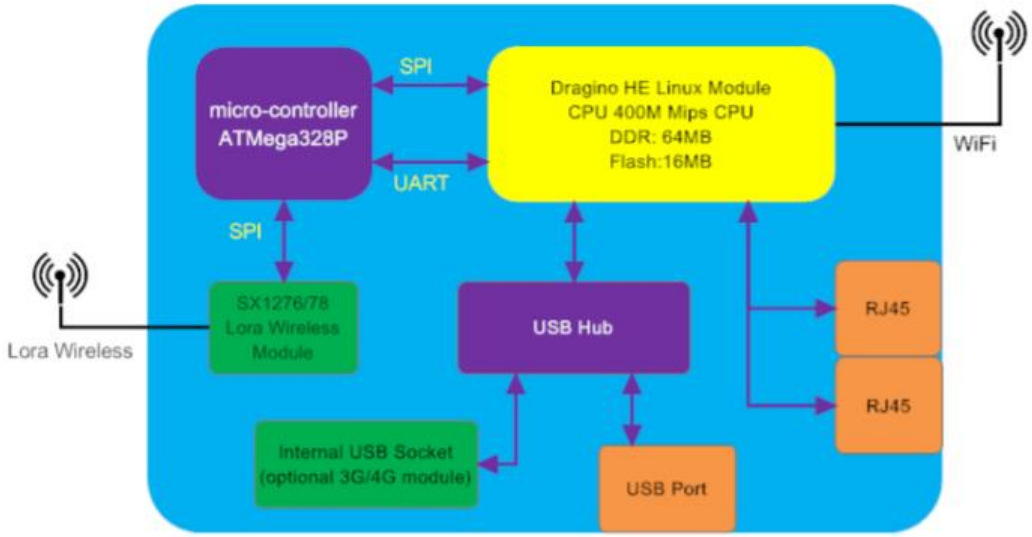


Figura No. 2.3 Arquitectura Dragino LG01

2.2 ENTORNO ARDUINO IDE Y LIBRERÍA RADIOHEAD RF95.

Para empezar, es importante adaptar el IDE Arduino, dentro del gestor de tarjeta de la IDE es posible incluir el soporte de la compañía Dragino añadiendo la URL:

http://www.dragino.com/downloads/downloads/YunShield/package_dragino_yun_test_index_test.json

Esto da acceso al gestor de tarjetas para poder acceder a dispositivos Dragino mediante el IDE Arduino para realizar control sobre el microcontrolador ATMega 328P

Para poder acceder al microcontrolador del dragino LG01, una vez energizado el gateway crea un punto de acceso WiFi con el nombre dragino, esta red por defecto permite ingresar al usuario *root* con la contraseña *dragino,* esta contraseña puede ser modificada ingresando a la interfaz gráfica o conectando de manera remota por medio del protocolo SSH al sistema Open Wrt, la ip por defecto para acceder es 10.130.1.1. Por medio de esta red es posible cargar el sketch desde el entorno Arduino al Dragino LG01.

Cabe destacar que para este punto, es posible acceder al microcontrolador y por medio del protocolo SPI conectarse al chip SX1276/SX1277, sin embargo existen librería adaptadas a este protocolo recomendadas por el fabricante, tal es el caso del paquete público RadioHead Packet Radio Library, una biblioteca desarrollada bajo el paradigma de la programación orientada a objetos para envío y recepción de mensajes en diferentes protocolos de comunicación inalámbrica, específicamente se hace uso del objeto RH_RF95, el soporte oficial para LoRa para el chip transceptor mencionado previamente que incluye el soporte de configuración de parámetros de red, envío y recepción de mensajes como puede ser visto en la 2.4

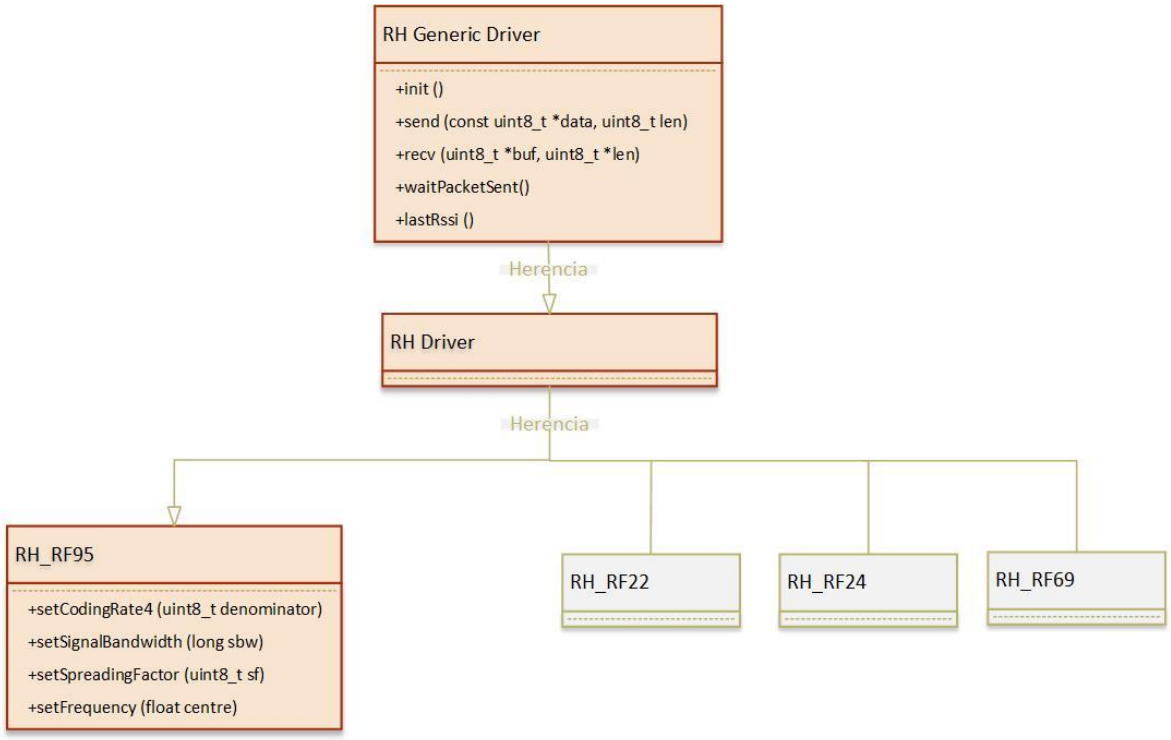


Figura No. 2.4 Diagrama UML RadioHead RF95

Al observar el diagrama UML es posible encontrar los parámetros de configuración estándar de la red LoRa en el objeto RF95 como son el ancho de banda, frecuencia, factor de progragación y tasa de codificación, este objeto hereda los métodos de envío y recepción genéricos de la librería RadioHead para implementar un protocolo de comunicación básico con LoRa Raw sin acceder a la interfaz gráfica o el sistema Open Wrt.

2.3 ALGORITMO DE ENVÍO Y RECEPCIÓN DE LOCALIZACIÓN.

Para implementar el sistema de comunicación se hace uso de la librería RadioHead descrita previamente, el gateway Dragino LG01 será el receptor de mensajes y una tarjeta Arduino con el módulo Dragino LoRa Shield funcionará como nodo o emisor de mensajes, es importante resaltar que el mensaje enviado por el dispositivo Arduino indica la zona desde la cual se emite el este, al ser recibido por el gateway es almacenado en la memoria de este junto al RSSI obtenido.

Configuración del nodo.

El diagrama de flujo de datos en la figura 2.5 representa la rutina programada en el Arduino operando como nodo.

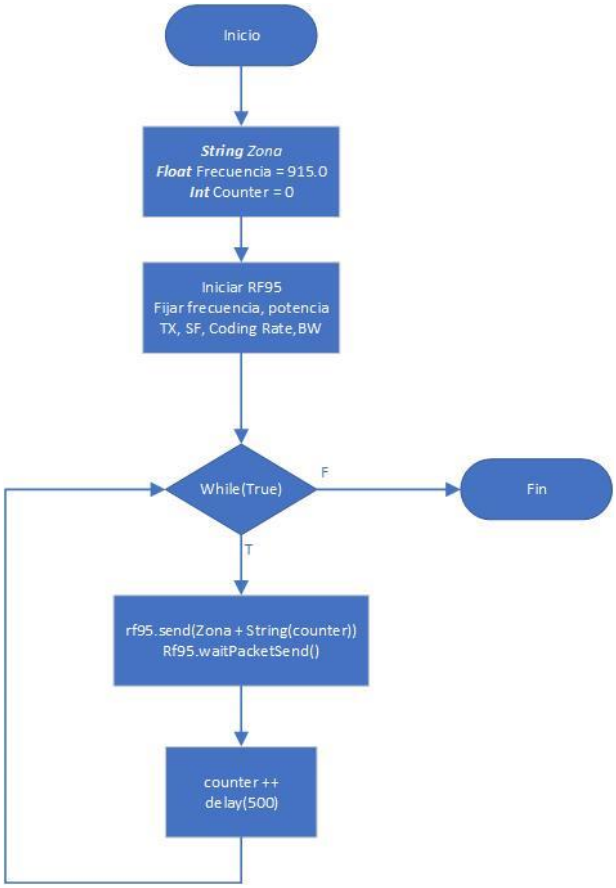


Figura No. 2.5 Diagrama flujo de datos nodo

Se inicializan las variables zona, counter y frecuencia, la variable zona corresponde al dato tipo string que contendrá la zona de envío del mensaje, la variable counter es una variable tipo integer que lleva el registro del número mensajes enviados, finalmente la variable frecuencia es definida tipo flotante con valor 915.0 de acuerdo

con la documentación de la librería RadioHead [30], para operar en la frecuencia permitida en la región. El siguiente paso consiste fijar los parámetros de la red LoRa correspondientes al ancho de banda, factor de propagación, tasa de codificación, finalmente se inicia un ciclo infinito de envío de mensajes mediante el método send de rf95 y se espera a la bandera de envío, en este método se envía en la zona y el contador como guía para sincronizar los datos posteriormente.

Configuración de gateway.

El diagrama de flujo de datos en la figura 2.6 representa la rutina programada en el LG01 operando como gateway o receptor.

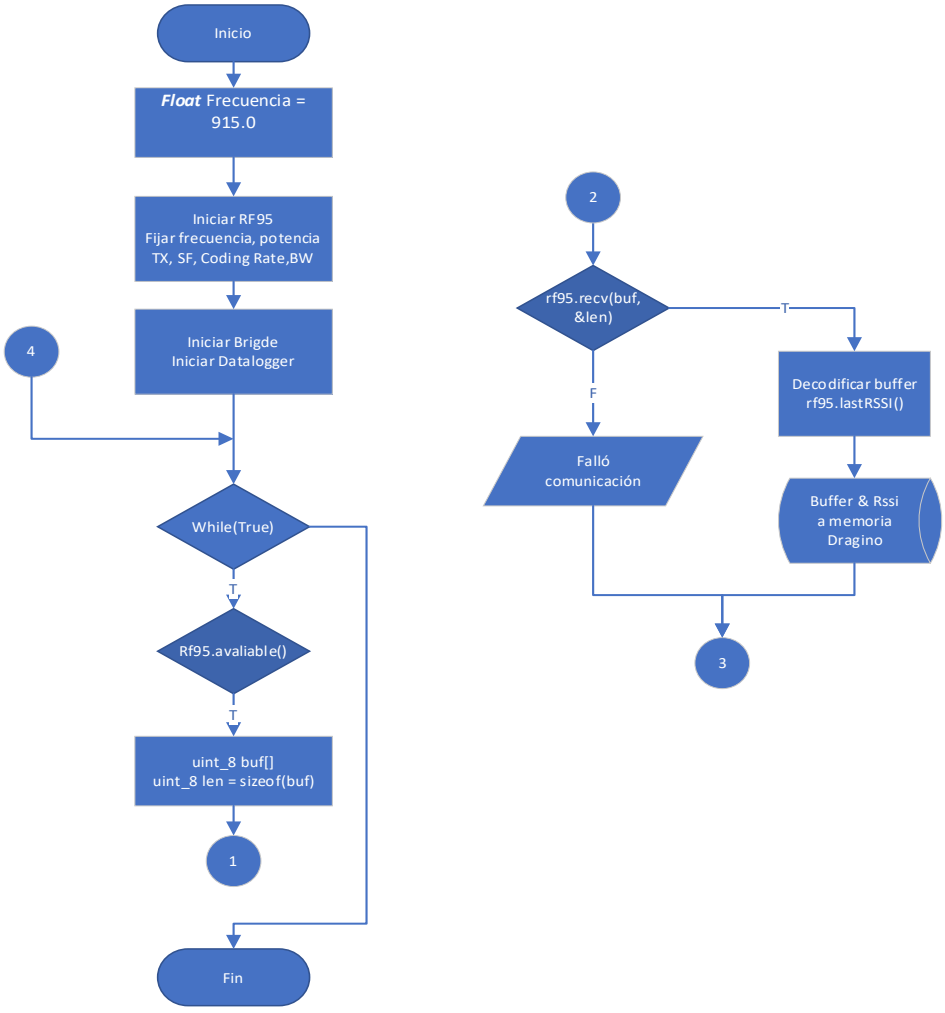


Figura No. 2.6 Diagrama flujo de datos gateway.

[30] Airspayce. RadioHead Packet Radio library for embedded microprocessors [en linea]. *Airspayce*, arbil 2014. Disponible en https://www.airspayce.com/mikem/arduino/RadioHead/

En primer lugar, se inicializan las mismas variables para establecer comunicación inalámbrica por medio de LoRa, evidentemente estas deben tener los mismos valores para realizar la conexión, adicional se inicia el puente entre el microcontrolador y el módulo Linux para poder acceder al sistema de almacenamiento disponible en este lado del gateway y el datalogger de Arduino que permite la creación y edición de archivos en la memoria del dispositivo.

A diferencia del nodo, el gateway se pone en modo de esperar hasta recibir un mensaje proveniente, cuando se recibe un mensaje este lo decodificará y almacenará junto al valor de RSSI con el cual recibió el mensaje en un archivo csv, este proceso es posible por medio del puente que permite enviar desde el ATMega al módulo Linux y almacenar en la memoria flash de este.

2.4 IMPLEMENTACIÓN DE RED Y ADQUISICIÓN NIVELES DE RSSI.

Con los algoritmos listos para transmitir y almacenar, se procede a implementar la red en el terreno de pruebas que puede ser observado a continuación, fundamentalmente el experimento consiste en obtener valores de RSSI en distintas zonas de este terreno para implementar un modelo que logre predecir la ubicación del nodo en alguna zona determinada dentro del terreno, correspondiente al conjunto residencial Colinas de Compostela, ubicado en el corregimiento La Ulloa del municipio Rivera, en un área aproximada de 48000m^2.

La ubicación de los gateways comprende las zonas resaltadas, esta es determinada buscando cobertura de la red LoRa en la mayor parte del área del terreno, aunque en pruebas preliminares se logra un rango de conexión de 2km con línea de vista, es importante considerar la presencia de árboles y casas que logran dificultar la conexión en algunas zonas, ocasionando que no todos los mensajes lleguen a los gateway, una vez instalados los gateway se procede a realizar envío de mensajes del nodo en cada zona, estas últimas corresponden a las divisiones físicas del conjunto residencial, es decir, cada lote presente es usado como guía para establecer un punto de envío de mensajes desde el nodo, cada zona tiene asignado un identificador, estas divisiones pueden ser vistas en las figuras

Figura No. 2.7 Imagen Satelital terreno de pruebas

Figura No. 2.8 Conjunto de zonas sector social.

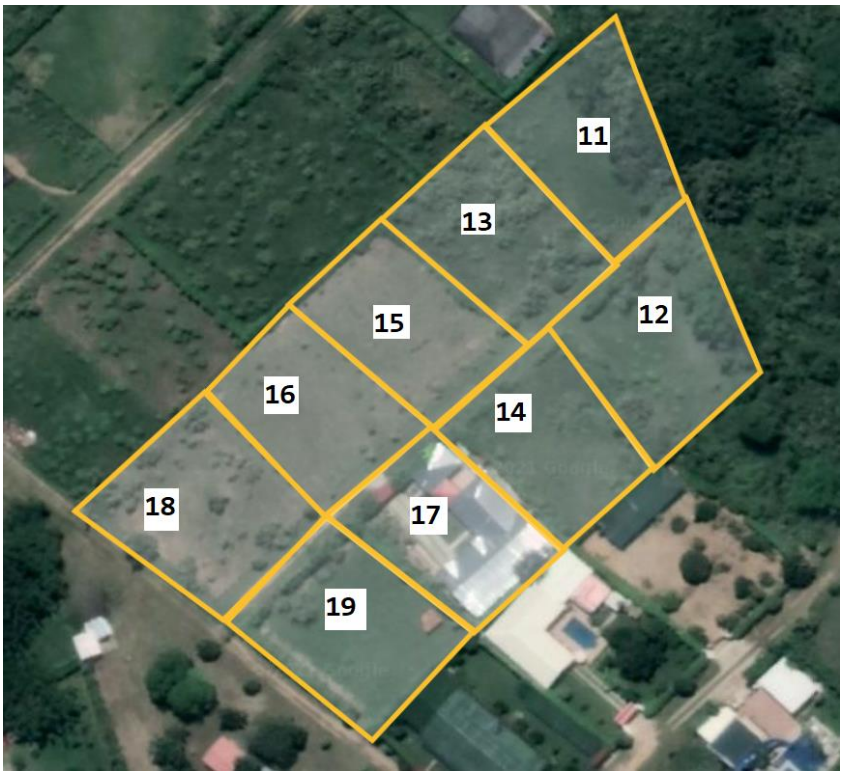


Figura No. 2.9 Conjunto 1 zonas de lotes.

Figura No. 2.10Conjunto 2 zonas de lotes

Figura No. 2.11 Conjunto 3 zonas de lotes

Figura No. 2.12 Conjunto 4 zonas de lotes

2.5 RECOLECCIÓN Y FILTRADO DE DATOS.

Después de haber capturado los distintos niveles de RSSI en las zonas planteadas inicialmente, transmitir esta información por medio de los nodos a los gateways y almacenarla dentro de la memoria interna de este último, es posible acceder a esta memoria por medio de protocolo FTP y extraer el conjunto de datos almacenado en cada gateway.

En ocasiones, los mensajes enviados pueden perderse y no llegar a todos los gateway, lo anterior representa un problema al momento de realizar el análisis y modelamiento de los datos debido a que es necesario exista una uniformidad en los datos para obtener modelos confiables. Por esta razón es necesario realizar una limpieza y filtrado de los datos, eliminando aquellos paquetes que no hayan llegado a los tres gateways instalados, esta tarea podría ser realizada manualmente revisando los datos almacenados en cada uno de los gateways y borrando aquellos que no se encuentren en todos los gateways, sin embargo, debido a la gran cantidad de datos que se registran en cada una de las zonas, realizar esta tarea de manera manual es ineficiente.

Por este motivo, como ayuda del software Excel, se programó un algoritmo que detecta cuando un paquete no se encuentra en todos los gateways y lo elimina, de esta manera, al final del filtrado, únicamente quedaran los niveles de RSSI de los paquetes que llegaron correctamente a todos los gateway, en este punto, el contador almacenado indicando el número de mensajes enviados es la clave para indicar si el mensaje no llegó a todos los gateways y será la variable que permite automatizar el proceso de filtrado, es importante eliminar información donde falten mensajes para lograr un proceso de entrenamiento de modelos de machine learning limpio. El diagrama de flujo del algoritmo está representado en figura 2.13

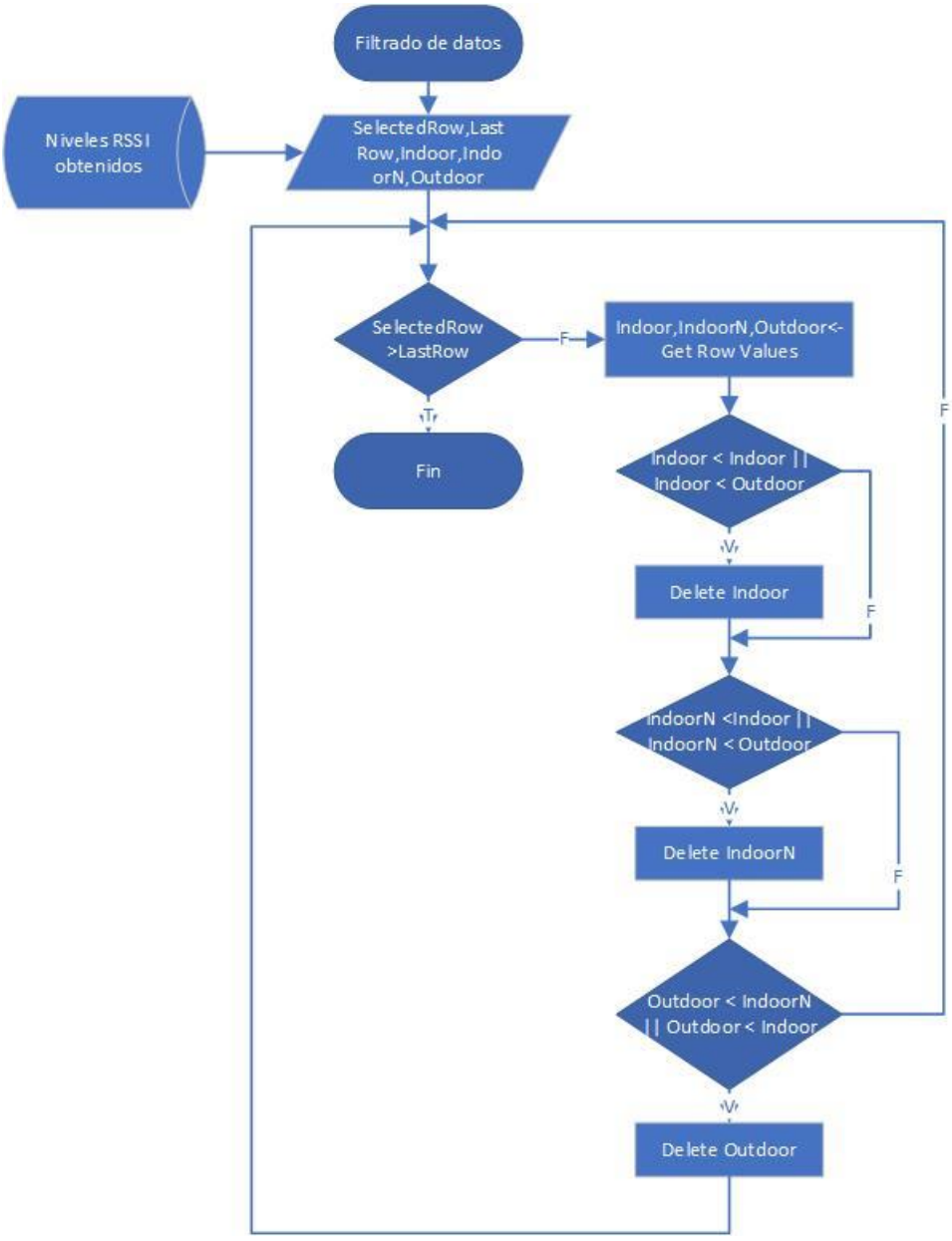


Figura No. 2.13 Diagrama filtrado de datos RSSI

El proceso de filtrado de datos se realiza de la siguiente manera:

SELECCIÓN DE RANGO

En este punto se selecciona el rango de celdas en las cuales se ejecutará el algoritmo, se selecciona un rango en vez de un número de elementos fijos para trabajar con un lote de datos, en este punto se crean las variables de *SelectedRow* que es la fila inicial del rango, *LastRow* que es la fila final del rango y por último se

declaran las variables I*ndoor*, I*ndoorN* y *Outdoor*, que corresponde al contador de la muestra, en cada uno de los tres gateways de la fila.

VERIFICACION DE LA ULTIMA FILA

Una vez se establecen las variables anteriores el programa comienza con el filtrado de los datos, en primer lugar, verifica si la fila actual en la que se va a realizar la comparación de datos es menor a la última fila, si esta condición es verdadera, significa que la fila actual no es la última fila y daría fin al proceso de filtrado.

COMPARACION DE CONTADORES Y BORRADO DE DATOS

El proceso de filtrado se realiza fila por fila de la selección, cuando el algoritmo inicia, este procede a verificar la columna correspondiente a los contadores, en caso de encontrar si un contador tiene un valor menor procederá a borrar las celdas relacionadas a este contador.

A continuación, se presenta en la tabla la cantidad de datos obtenidos en cada zona, además se incluye la información de área para cada zona.

Tabla 1 Síntesis zonas de experimento post-filtrado

CLASE	LUGAR	CANTIDAD MENSAJES ALMACENADOS	ÁREA (m²)
1	Lote salón social	197	875
2	Lote piscina	165	637,32
3	Lote puerta llanera e	165	457
4	Lote puerta llanera 1	197	731
5	Cancha voleibol	146	434
6	Lote esquina Lucho	165	541
7	Lote 1 Lucho	167	416
8	Lote 2 Lucho	174	459
9	Asador	174	1042
10	Planta tratamiento	180	696
11	Lote Panel	189	1131
12	Lote Panel 1	133	1301
13	Lote Panel 2	164	1144
14	Lote Panel 3	206	1097
15	Lote Panel 4-1	286	1118,48
16	Lote Panel 4-2	346	1065,64
17	Lote Panel 5	269	1083,2
18	Lote Panel 6	433	1325,44
19	Lote Panel 7	293	1230,47
20	L-01	361	1317,73

21	L-02	303	1036,35
22	L-03	298	1319,16
23	L-04	120	1136,33
24	L-05	139	1514,53
25	L-06	120	1296,27
26	L-07	174	908,22
27	F-01	323	921,6
28	F-02	189	801,92
29	F-03	286	1593,94
30	F-04	274	1192,78
31	F-05	159	1360,72
32	M	142	1082
33	M-P	58	1075
34	M-1	195	1551
35	M-2	193	1143
36	M-3	363	1175,9
37	V	171	1118
38	V-2	225	1172
39	V-3	234	1077,78
40	V-4	213	997,76
41	V-5	210	1414,16
42	V-6	165	1105
43	V-7	185	1137,89
44	V-8	172	1746,96
45	Casa	85	1106

3. CAPÍTULO TRES: DESARROLLO DE MODELO

En este capítulo se presenta el proceso de entrenamiento de distintos modelos de Machine Learning, abarca todo el procedimiento de procesamiento de datos, implementación de modelo y ajuste de hiperparámetros, finalizando con el desarrollo de una red neuronal.

3.1 PYTHON Y PROCESAMIENTO DE DATOS.

Una vez almacenados los datos y realizado el filtrado de limpieza, es necesario el conjunto de datos dentro de Python, lenguaje de programación interpretado, con licencia de código abierto y librerías implementadas para desarrollo de modelos de machine learning de alto nivel, fundamentalmente para procesar datos con Python se utilizan las librerías NumPy, Pandas, Matplotlib y Scikit-learn, este procedimiento consta de tres partes que pueden ser observadas en la figura 3.1

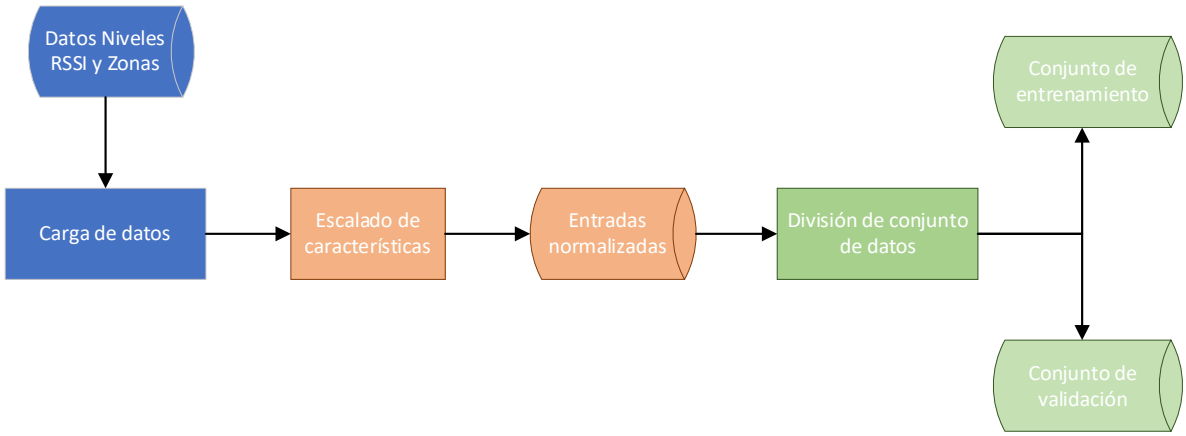


Figura No. 3.1 Procesamiento de datos

3.1.1 CARGA DE DATOS.

Para cargar los datos que fueron recolectados, filtrados y se encuentran en formato de hoja de cálculo *xlsx* se hace uso de Pandas, esta entrega un objeto tipo DataFrame definido como datos tabulares bidimensionales, con tamaño mutable y potencialmente heterogéneos este objeto puede ser visto en la figura 3.2

	Indoor 1	Indoor 2	Outdoor	Zona
0	-93	-56	-93	1
1	-94	-56	-93	1
2	-93	-56	-93	1
3	-95	-55	-92	1
4	-97	-63	-97	1
...	...	...	...	...
9446	-43	98	-98	45
9447	-45	95	-94	45
9448	-44	98	-98	45
9449	-55	96	-99	45
9450	-47	98	-89	45

9451 rows × 4 columns

```
<class 'pandas.core.frame.DataFrame'>
```

Figura No. 3.2 Pandas DataFrame RSSI

Este DataFrame permite realizar operaciones algebraicas en el conjunto de datos, así como separar la columna zona que será posteriormente la variable objetivo durante el entrenamiento, así como Indoor 1, Indoor 2 y Outdoor se mantendrán para hacer parte del conjunto de variables de entrada.

3.1.2 ESCALADO DE CARACTERÍSTICAS: NORMALIZACIÓN ESTÁNDAR.

Para desarrollar un algoritmo de Machine Learning, es ideal poseer características en los datos como media estadística cercana o idealmente con valor cero y varianza similar, esta última debido a casos donde una característica al presentar una varianza mayor implicará que el algoritmo estimador de mayor prioridad al aprendizaje en esta característica y empeorar para otras, para enfrentar estos dos problemas una solución estándar es la normalización estándar o z-score descrita por la ecuación.

$$z = \frac{x - \mu}{\sigma} \tag{7}$$

Donde μ corresponde a la media de los datos y σ la desviación estándar, como resultado el conjunto de datos será escalado a valores entre 0 y 1 y una desviación estándar unitaria, el proceso de normalización estándar puede ser implementado en Python mediante scikit-learn en el conjunto de funciones de preprocesamiento.

3.1.3 DIVISION DE DATOS: ENTRENAMIENTO Y PRUEBA

Para finalizar el procesamiento de datos se realiza la división del conjunto de datos en dos grupos, el primero de estos grupos será utilizado y visto por el algoritmo durante el proceso de entrenamiento, el segundo grupo únicamente será utilizado para evaluar el rendimiento del modelo después de finalizado el entrenamiento.

Debido a que el conjunto de datos no tiene una cantidad elevada de datos se opta por una división de conjunto de datos estándar correspondiente a 80% de datos para entrenamiento y 20% para pruebas, esta división se realiza con el módulo de scikit-learn para selección de modelos, aunque esta selección es aleatoria, se genera un número pseudo-aleatorio para realizar la división con el fin de poder replicar el experimento con los distintos modelos en las mismas condiciones de división del conjunto de datos.

3.2 ENTRENAMIENTO DE MODELOS.

El proceso de entrenamiento de modelos de Machine Learning como puede ser visto en la figura 3.3 consiste en un proceso cíclico para evaluar el rendimiento del modelo, previamente se debe tener el conjunto de datos, definir el objetivo para implementar el modelo y los hiperparámetros de este, de igual manera el conjunto de datos debe ser limpio de ruido, previamente procesado para evitar salidas defectuosas que afectan el rendimiento.

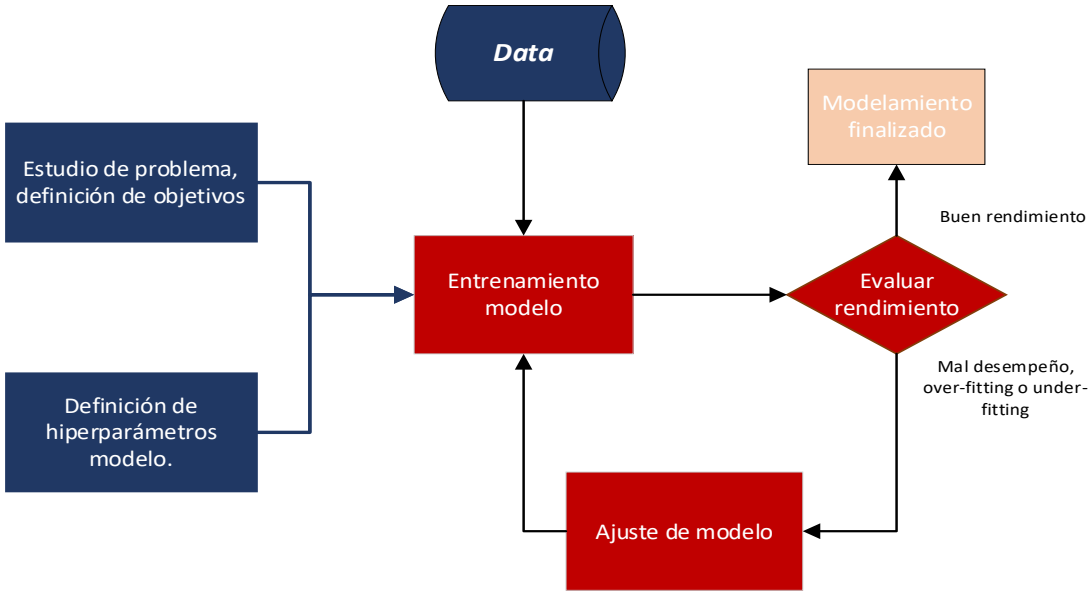


Figura No. 3.3 Procedimiento entrenamiento de algoritmos Machine Learning

3.2.1 OPTIMIZACIÓN HIPERPARÁMETROS: BUSQUEDA EN CUADRILLA

Dentro del desarrollo del modelo interviene la selección de hiperparámetros, en un algoritmo de Machine Learning, hiperparámetro hace referencia a las variables que son fijadas previamente al entrenamiento y no pueden cambiar su valor durante este, aunque no existe una forma óptima de elegir un valor correcto para estos, se puede tomar algún tipo de estrategia para acercarse a un rendimiento deseable.

Se optó por implementar la búsqueda de cuadrilla, un método fácil de desarrollar, aunque no muy recomendado para algoritmos de mayor complejidad, la búsqueda de cuadrilla consiste en definir un conjunto o vector del hiperparámetro a ajustar, una vez entrenado el modelo se realiza una nueva búsqueda en un nuevo conjunto de valores limitados por los dos que mayor rendimiento presentaron, este método puede ser automatizado limitado por el tiempo y costo computacional, este proceso se repite en el entrenamiento de todos los modelos de machine learning implementados para encontrar el mejor desempeño de estos.

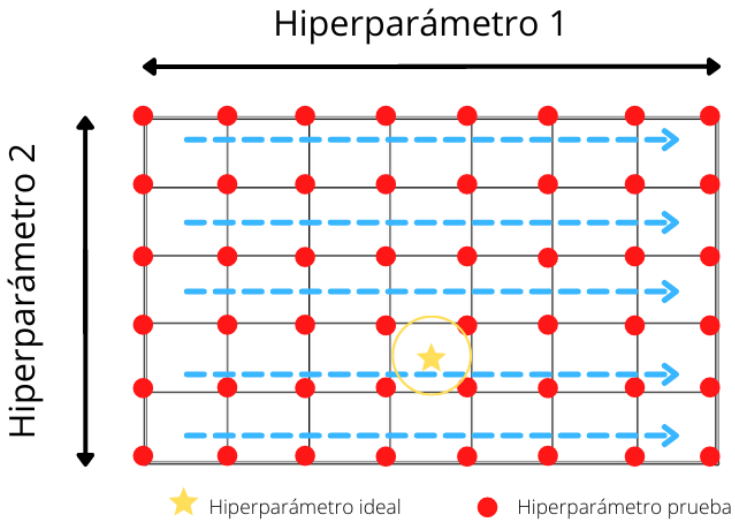


Figura No. 3.4 Método de la búsqueda de cuadrilla.

3.2.2 MÉTRICAS DE RENDIMIENTO.

Para evaluar el rendimiento del modelo se evalúan haciendo uso del subconjunto de datos de validación, cada algoritmo predice a partir de las variables de entrada y se comparan las salidas obtenidas con las salidas almacenadas, las métricas de evaluación implementadas corresponden a exactitud del modelo y puntaje R^2.

Naturalmente en clasificación el desempeño del modelo es evaluado en función de las veces que predijo una clase correctamente, para clasificación binaria se

consideran cuatro estados para una predicción, verdadero positivo (VP), verdadero negativo (VN), falso positivo (FP) y falso negativo (FN)

EXACTITUD.

En clasificación, exactitud corresponde a la fracción de datos predichos del total de predicciones, puede ser representada como el porcentaje de predicciones correctas realizada por algoritmo y está descrita por la ecuación 8

$$Exactitud = \frac{Número\ de\ predicciones\ correctas}{Número\ total\ de\ predicciones} = \frac{VP + VN}{VP + VN + FP + FN} \qquad (8)$$

En modelos de multiclasificación, resulta lógico considerar para cada subconjunto de datos que exactitud fue lograda para cada clase, como puede ser observado en la ecuación 9, donde $\hat{y}_i$ corresponde a la i-enésima predicción, evidentemente la generalización de exactitud se logra al ponderar las exactitudes logradas por clase.

$$Exactitud(y, \hat{y}) = \frac{1}{n_{muestras}} = \sum_{i=0}^{n_{muestras}^{-1}} 1(\hat{y}_i = y_i) \qquad (9)$$

PUNTAJE R^2.

El puntaje R^2 representa el coeficiente de determinación obtenido, en otras palabras, representa la proporción de varianza para la variable predicha, proporciona un indicador de que tan probable es que el modelo prediga bien ante muestras o datos no vistos mediante la varianza, esta métrica está determinada por la ecuación 10.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y_i})^2}\ , donde\ \overline{y_i} = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad (10)$$

3.2.3 MODELOS MACHINE LEARNING.

Para realizar el entrenamiento de modelos se hace uso de los módulos de modelamiento de scikit-learn, como puede ser visto en la figura 3.5 estos modelos tienen los métodos train en los cuales se insertarán las variables para aprendizaje y predict, este solo puede ser implementado si el modelo fue entrenado y retornará las predicciones para un arreglo de datos, este método será utilizado para validar posteriormente el rendimiento de cada algoritmo.

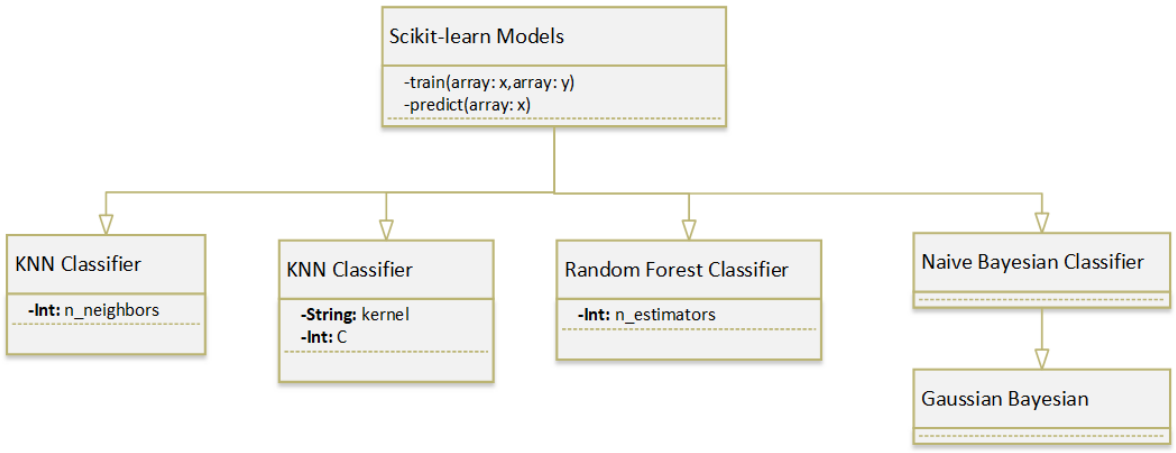


Figura No. 3.5 Diagrama UML Scikit-learn modelamiento.

Todos los modelos de Machine Learning heredan los métodos de aprendizaje y predicción y en su constructor añaden las variables correspondientes a sus hiperparámetros, se implementa el método de búsqueda en cuadrilla para cada modelo y se almacena el rendimiento de estos para posterior comparación y selección del mejor rendimiento, una vez finalizado el entrenamiento de cada modelo para cada grupo de hiperparámetros se escogen como modelos finales aquellos con el mejor rendimiento y se almacenan para comparar los resultados de rendimiento para cada modelo implementado.

3.3 REDES NEURONALES PROFUNDAS.

La implementación de redes neuronales se lleva a cabo con la API Keras, una extensión de la librería de grafos computacionales de Python conocida como TensorFlow, implementar una red neuronal el Keras permite probar configuraciones de redes neuronales sin modificar cada grafo computacional, facilitando el diseño de redes, experimentar la prueba de arquitecturas mediante los objetos que dispone para las capas neuronales y hacer uso de distintos modelos matemáticos para optimización y cálculo de pérdidas en un alto nivel de programación.

3.2.1 ESCALADO DE CARACTERÍSTICAS: FUNCIÓN MINMAX

MinMax es una técnica de escalado de características, de manera similar a la normalización estándar, tiene como objetivo acercar la media del conjunto de datos a cero y tener varianza unitaria, a diferencia de la normalización estándar, en MinMax se busca que las características de los datos sean movidas a un conjunto de datos en un rango de valores deseados, ideal para redes neuronales donde se buscan valores de entradas en el rango de valores de cero y uno para evitar que valores muy grandes sean escalados sin control en las funciones de activación de

las neuronas, al momento de optimizar la velocidad de cómputo de gradientes presente en la red, empeorando el aprendizaje de la red neuronal, la función MinMax está descrita por la ecuación 11, donde *max* y *min* corresponden a los valores del rango al cual se desea ajustar el conjunto de datos, estos son omitidos para el caso donde el rango a ajustar está entre cero y uno como se desea para el caso de las redes neuronales

$$Ex = \frac{x - x_{min}}{x_{max} - x_{min}} (max - \min) + min \tag{11}$$

3.3.2 ARQUITECTURA.

La arquitectura de red implementada es un perceptrón multicapa conocida en ingles también como Fully Connected Network, consiste en una serie de capas que conectan todas sus salidas a todas las entradas de la siguiente capa, matemáticamente cada nodo o grafo computacional tomará un conjunto de datos entrantes y retornada una salida al siguiente conjunto de grafos, este proceso se conoce como propagación y es realizado por la neurona en entrenamiento y operando como modelo, la topología de la red implementada es ilustrada en la figura 3.6

Entre las características de este tipo de red, cada grafo computacional tendrá un conjunto de pesos que serán utilizados para ponderar las entradas de este y una función de transferencia no lineal para recrear características no lineales de la ponderación realizada por el grafo, teóricamente la cantidad de neuronas en cada grafo permite la generar hiperplanos de decisión mientras la cantidad de capas extiende la capacidad de generar hiperplanos para decisiones no lineales, sin embargo introducir muchas capas ocultas implica mayor costo computacional para el procesamiento de la red neuronal resultante, debido a la cantidad de clases a segmentar y la cercanía de niveles de RSSI en cada zona se optó por implementar varias capas ocultas en la red.

Por último, se implementa la capa de salida, en esta capa se definen las características que permiten que la red opere como algoritmo de clasificación, específicamente como multiclasificación, al haber elegido 45 zonas y por tal motivo tener 45 clases a predecir, la capa de salida tiene obligatoriamente 46 salidas, siendo esta salida adicional para casos donde no es posible clasificar en alguna de las clases con las cuales fuese entrenado.

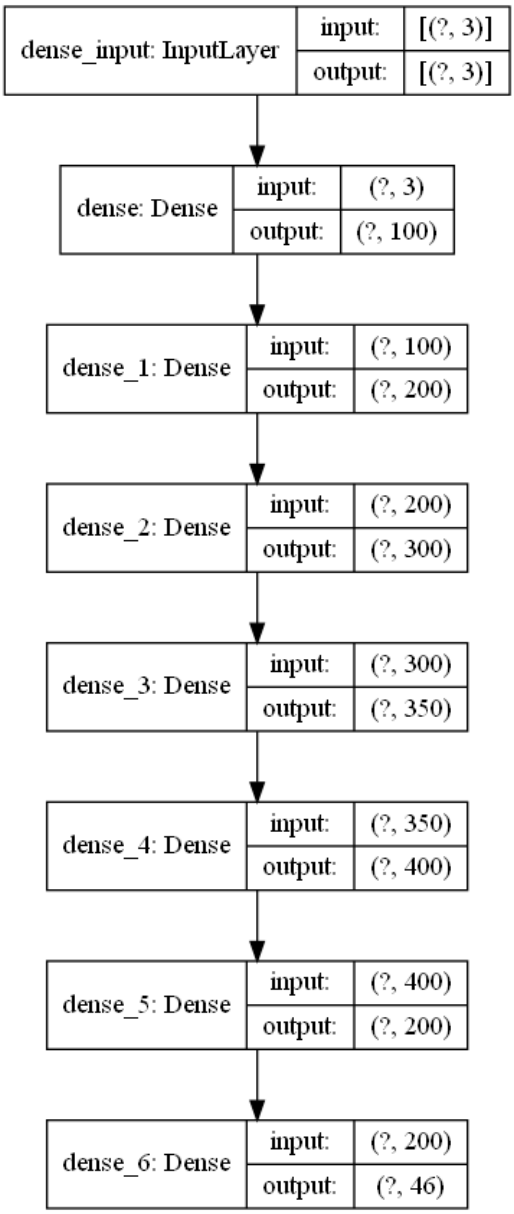


Figura No. 3.6 Arquitectura red implementada.

3.3.3 SMOTE.

En clasificación un problema muy recurrente es la presencia de conjuntos de datos desequilibrado, es decir, datos donde una clase puede tener una cantidad reducida de datos para entrenar si es comparada con alguna otra clase, pues la red neuronal al pasar pocas veces esta clase durante su entrenamiento no considerará lo suficiente esta clase para generar hiperplanos de decisión, una forma de solventar este problema consiste en duplicar la información de las clases minoritarias, sin embargo no se añadiría información adicional para mejorar el entrenamiento.

Una forma propuesta para abordar este problema consiste aumentar las muestras de datos en las clases minoritarias, generando muestras de datos sintéticas, de ahí el nombre en inglés de esta técnica conocida como Synthetic Minority Oversampling Tecnique, la cual busca sobre ejemplos aleatorios de la clase minoritaria cercanos en características y sus vecinos más cercanos, creando datos nuevos entre el ejemplo escogido aleatoriamente y uno de sus vecinos al azar,el nuevo dato sintético corresponde a una combinación convexa de los dos datos, este proceso de generar un dato sintético se realiza por medio de la ecuación 12, donde $X_i$corresponde al dato elegido aleatoriamente, λ un número aleatorio entre 0 y 1 y X_{zi} un vecino cercano al dato aleatorio

$$X_{nuevo} = X_i + \lambda(X_{zi} + X_i) \tag{12}$$

3.3.4 FUNCIÓN DE ACTIVACIÓN.

La función de activación elegida para las capas de entrada y capas ocultas es el rectificador lineal unitario o ReLU, debido a que demanda bajo costo computacional, converge rápidamente y genera salidas inestables que otras funciones pueden llegar a presentar, adicionalmente tiene características deseables como volver cero los valores negativos, algo deseable para este modelo debido a que al ser clasificación los valores negativos no son deseable en este modelo, está descrito por la ecuación donde *z* representa la salida de cada neuronal.

$$(z) = \max(0, z) \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases} \tag{13}$$

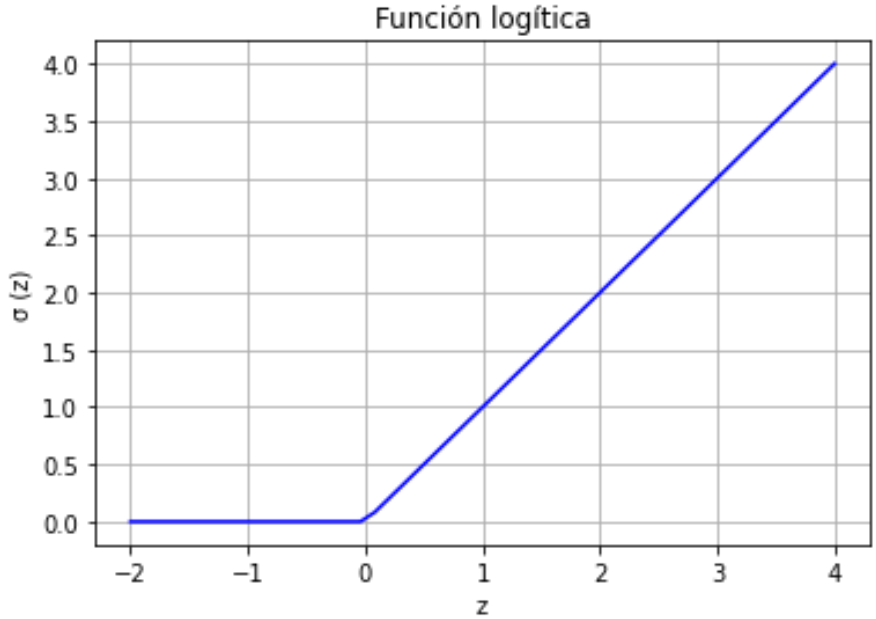


Figura No. 3.7 Función de activación ReLU

Para la capa de salida esta función no es conveniente, al ser un sistema de clasificación la salida será un valor estimando la probabilidad para que un conjunto de datos sea una clase, esta salida tiene como estándar la función logística ilustrada en la ecuación 13

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{14}$$

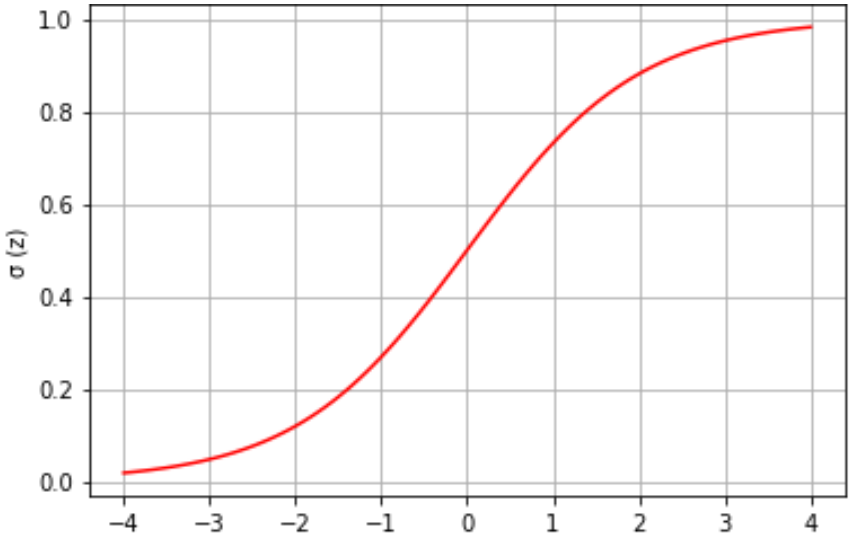


Figura No. 3 8 Función logística.

En esta función se toma la salida ponderada de la neurona y la transforma a un valor entre 1 y 0, sin embargo, sigue sin ser eficiente para la salida del tipo de modelo a recrear con la red neuronal, pues esta función solo permite una salida binaria, en otras palabras, solo es eficiente si el objetivo es determinar una clase, para solucionar este conflicto, se hace uso de una solución que consiste en la generalización de esta función para salidas multiclases conocida como la función softmax, esta toma como entrada el vector z de dimensión C correspondiente a las salidas producidas por la capa de salida de la red neuronal y entrega un vector de salida y de dimensión C , matemáticamente esto es representado de acuerdo a la ecuación 14

$$y_c(z) = \frac{e^{z_c}}{\sum_{d=1}^{C} e^{z_d}}, para\ c = 1, \dots C \tag{15}$$

Para poder entender la ecuación 15 como una función de probabilidad para la capa de salida de una red neuronal, es posible expresar esta ecuación de la siguiente forma.

$$\begin{bmatrix} P(t=1|C) \\ . \\ . \\ . \\ P(t=C\ |z) \end{bmatrix} = \begin{bmatrix} y_c(z)_1 \\ . \\ . \\ . \\ y_c(z)_C \end{bmatrix} = \frac{1}{\sum_{d=1}^{C} e^{Z_d}} \begin{bmatrix} e^{Z_1} \\ . \\ . \\ . \\ e^{Z_c} \end{bmatrix} \quad (16)$$

Donde $P(t = C\ |z)$ representa la probabilidad de que a partir de una entrada Z corresponda a una clase C, en otras palabras, la función softmax aplicada a una red neuronal generará un vector de probabilidades para cada clase, siendo la clase que estimará como salida aquella que tenga mayor probabilidad en este vector.

3.3.5 ENTRENAMIENTO DE RED NEURONAL.

El proceso de aprendizaje de la red neuronal consiste en dos partes, la propagación que incluiría el realizar el cálculo de los grafos computacionales los cuales contienen los vectores de parámetros de la red conocidos como pesos seguido de la propagación hacia atrás o propagación regresiva, donde se evalúa el desempeño de la red y se ajustan los vectores de parámetros, esto es un procedimiento cíclico que permite a la red mejorar su desempeño en cada ciclo o época, gracias a este procedimiento las redes neuronales tienen mejor desempeño para realizar tareas de predicción respecto a los algoritmos de predicción tradicionales, para este modelo el entrenamiento es llevado a cabo con mil épocas.

Función de pérdidas.

La función de pérdidas es la métrica que permite a la red neuronal evaluar su desempeño durante el aprendizaje, comparará sus predicciones con la información del conjunto de base de datos de entrenamiento y ejecutará un cálculo de error, debido al objetivo de la red neuronal a implementar se hace uso de la función Entropía Cruzada Categórica, en ingles Categorical Cross-Entropy loss, esta computa cálculo de pérdida en función de un vector one-hot, idealmente se espera que solo la clase a predecir tenga como valor uno, lo equivale a que es la clase positiva para la predicción, en los otros casos debería obtenerse idealmente cero, castigando y aumentando el cálculo de error por todas las demás clases donde la red tienda a generar una probabilidad distinta a cero.

$$CE = \mathcal{L} = -\sum_{i}^{C} t_i \log f(Z_i)\ , donde\ f(Z_i) = \frac{e^{Z_i}}{\sum_{j}^{C} e^{S_j}} \quad (17)$$

Optimizador.

El optimizador es el método utilizado para ajustar los parámetros de la red neuronal, este busca minimizar el error encontrado en la función de pérdidas, lo que implicaría que la red neuronal puede tener mejores predicciones, se utiliza el algoritmo ADAM, en inglés Adaptative Moment Estimator, este es una combinación de los algoritmos RMSProp y Gradiente Descendiente Estocástico , ajustando en cada iteracción el factor de aprendizaje, mediante el cálculo de los dos primeros momentos del gradiente para ajustar el factor de aprendizaje encargado de ajustar el peso de las neuronas, este cálculo está descrito en la ecuación 17

$$W_t = W_{t-1} - \eta \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \epsilon} \tag{18}$$

Los momentos $\widehat{m}_t$ y $\widehat{v}_t$ son calculados a partir de las ecuaciones descritas acontinuacion, donde β es el factor de aprendizaje, hiperparámetro que debe ser definido previo al entrenamiento de la red neuronal.

$$m = \beta_1 m_{t-1} + (1 - \beta_1) * g \tag{19}$$

$$\widehat{m}_t = \frac{m}{1 - \beta_1^t} \tag{20}$$

$$v = \beta_2 v_{t-1} + (1 - \beta_1) * g^2 \tag{21}$$

$$\hat{v}_t = \frac{v}{1 - \beta_2^t} \tag{22}$$

Donde g corresponde al gradiente de la función de error respecto al vector de parámetros, en otras palabras, g está definido por la ecuación, siendo $\mathcal{L}$ el vector de pérdidas obtenido y θ el vector de parámetros de la red neuronal.

$$g = \frac{\partial \mathcal{L}}{\partial \theta} \tag{23}$$

Gráficas de entrenamiento.

Una vez se implementó la red neuronal con los hiperparámetros descritos previamente se procede a entrenar la red neuronal, durante el proceso de aprendizaje se genera un tercer conjunto de datos utilizados por el algoritmo para validar el desempeño de la red ante datos que aún no ve durante su entrenamiento.

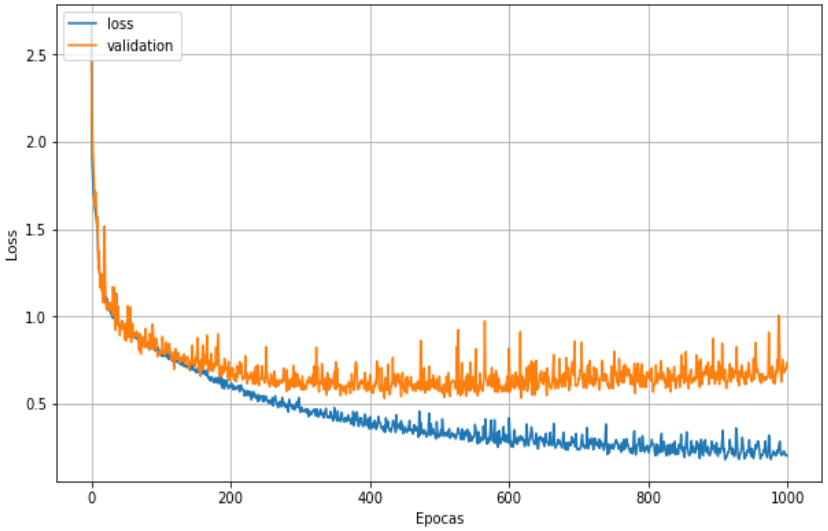


Figura No. 3.9 Gráfica de pérdidas durante entrenamiento red neuronal

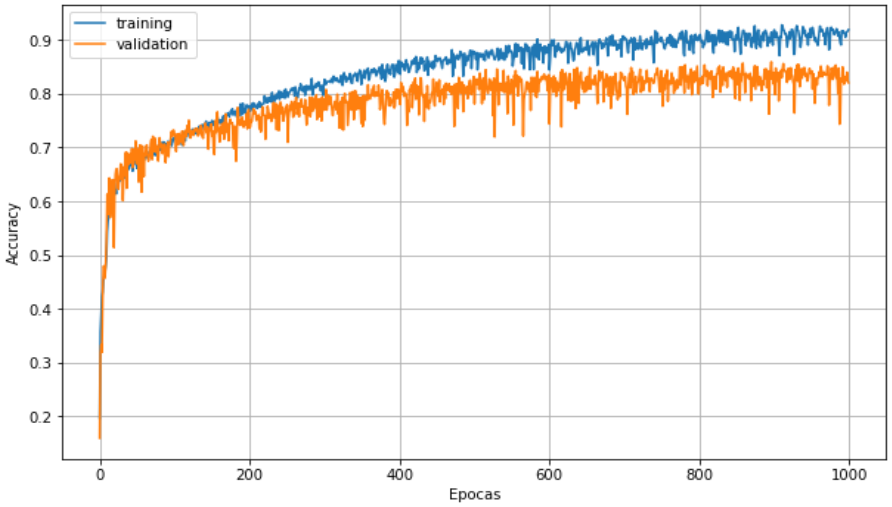


Figura No. 3.10 Gráfica de exactitud durante entrenamiento red neuronal

Como puede ser visto en las figuras 3.9 y 3.10, la red va disminuyendo el error y aumentando la exactitud de sus predicciones, aunque en el conjunto de validación es posible observar como la pendiente de aprendizaje mantiene su valor, lo que quiere expresar es que con esta arquitectura definida la red no es posible obtener un mejor desempeño.

4. RESULTADOS Y DISCUSIONES.

En este capítulo se presentan los resultados de este trabajo de investigación, se presenta el desempeño de los algoritmos entrenados antes datos que no intervinieron durante el proceso de aprendizaje, las gráficas comparativas entre modelos que fueron fundamental para la elección, adicional se realiza la evaluación del desempeño por zonas de los mejores algoritmos mediante un reporte de clasificación y matriz de confusión.

4.1 SELECCIÓN DE MEJORES MODELOS.

El proceso de elección de los mejores modelos es llevado a partir de las métricas de exactitud y puntaje R^2, métricas discutidas previamente, en otras palabras, se eligió directamente tener un gran número de predicciones acertadas en el modelo

A continuación, se presenta el resultado para el algoritmo KNN.

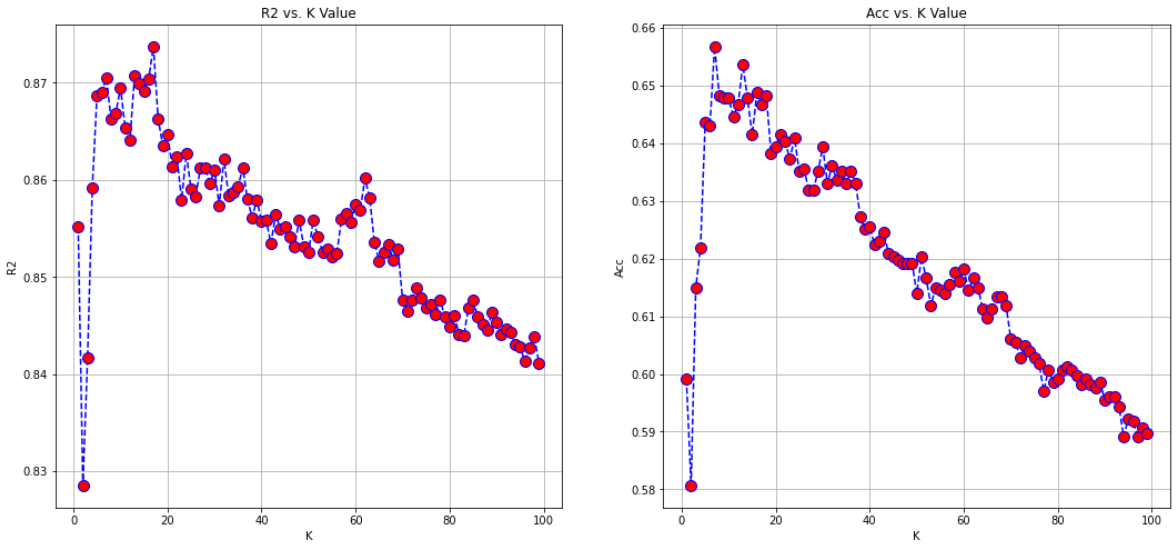


Figura No. 4.1 Resultados ajuste KNN

De la figura 4.1, destaca el desempeño del algoritmo cuando el valor de K es 16, aunque presenta un puntaje R^2, la exactitud muestra que este algoritmo logra 65% de sus predicciones acertadamente.

Con las máquinas de soporte vectorial se evalúa y compara el rendimiento con los cuatro tipos de kernel, después de realizar la comparativa como puede ser visto en la figura 4.2 el kernel que implementa características a los datos mediante la función de base radial o RBF es el que logra un mejor desempeño, este desempeño resulta

curioso pues la condición matemática RBF está familiarizada con la distribución gaussiana y similitud entre puntos, esta segunda es compartida con el algoritmo de KNN, aunque tiene menor exactitud que este último, motivo por el cual se compara al ajustar el parámetro C de las SVM, este resultado puede ser visualizado en la figura 4.3

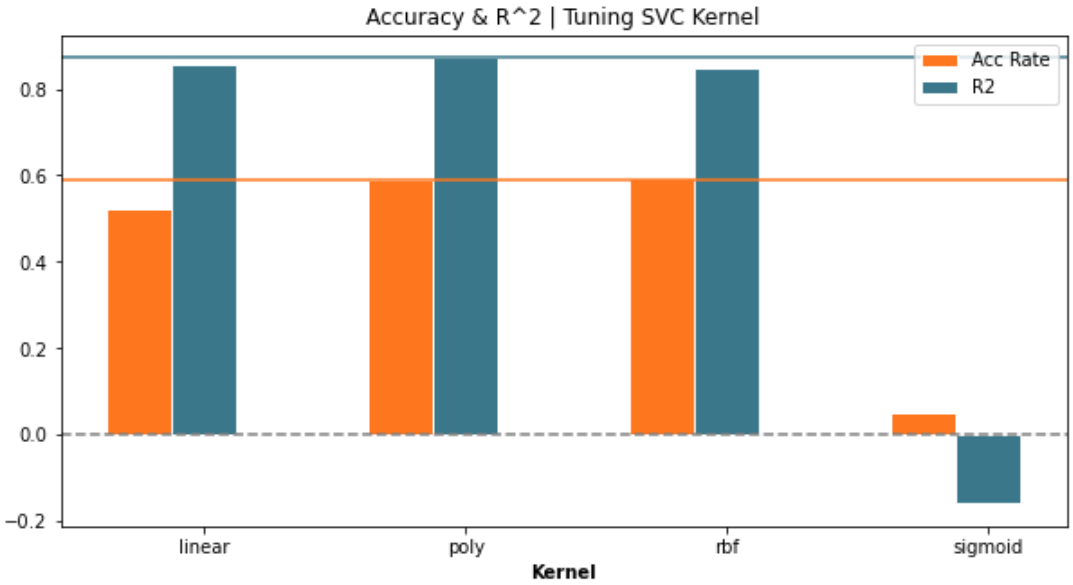


Figura No. 4.2 Ajuste de Kernel en SVM

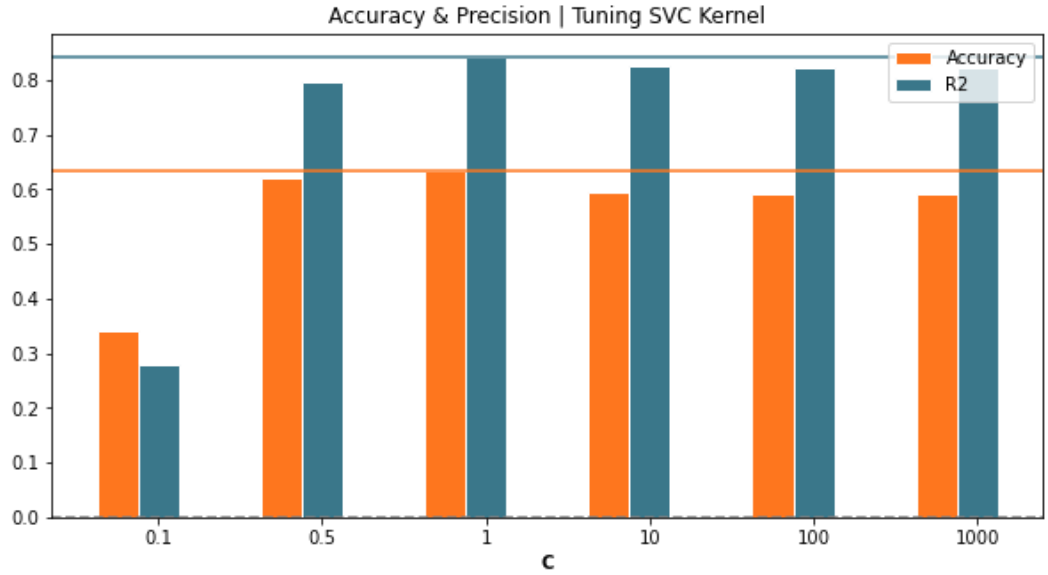


Figura No. 4.3 Ajuste penalidad C en SVM

Se observa un desempeño muy similar al algoritmo KNN en exactitud con valor de C igual a 1, en conclusión, no se presentan mejoras relevantes.

Por otro lado, los algoritmos basados en métodos de árbol y ensamblaje de árboles presentan un desempeño similar, aunque inferior a los algoritmos presentados hasta ahora como puede ser visto en la figura 4.4

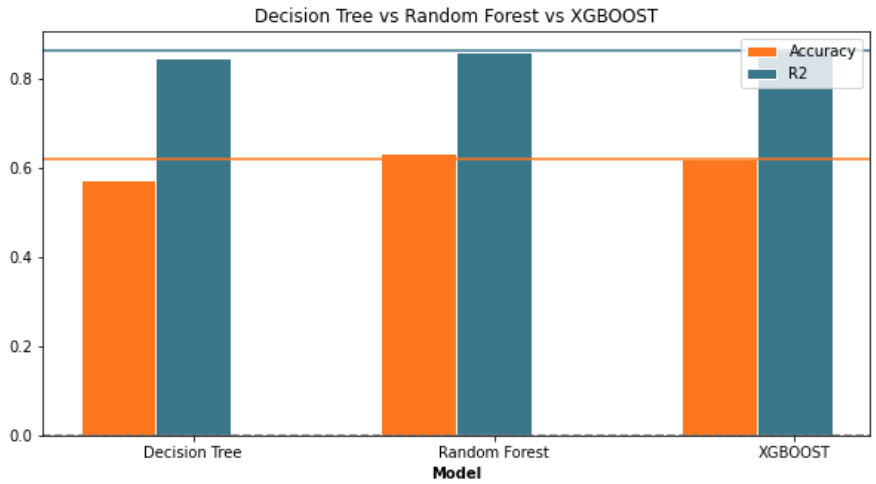


Figura No. 4.4 Comparativa modelos basados en árboles de decisión

Para finalizar la discusión de comparativa en los algoritmos tradicionales de machine learning, se comparan los mejores modelos obtenidos en cada algoritmo junto al algoritmo bayesiano, el cual tuvo el peor rendimiento, KNN se muestra como el algoritmo con mejor desempeño, sin embargo los resultados de estos algoritmo no indican un desempeño ideal para implementar un sistema de localización, pues solo el 65% de las veces logrará predicciones correctas, dejando mucho margen de error para considerar viable este método de localización.

Con respecto a las redes neuronales, parte del desempeño fue presentado durante las gráficas de entrenamiento, sin embargo, se realiza la evaluación con los datos de prueba, los cuales nunca intervinieron en el proceso de entrenamiento, obteniendo notables diferencias respecto a los demás algoritmos como puede ser observado en la figura 4.5 red neuronal logra obtener 84% de predicciones correctas, motivo por el cual es posible considerar las redes neuronales como el mejor algoritmo para esta aplicación.

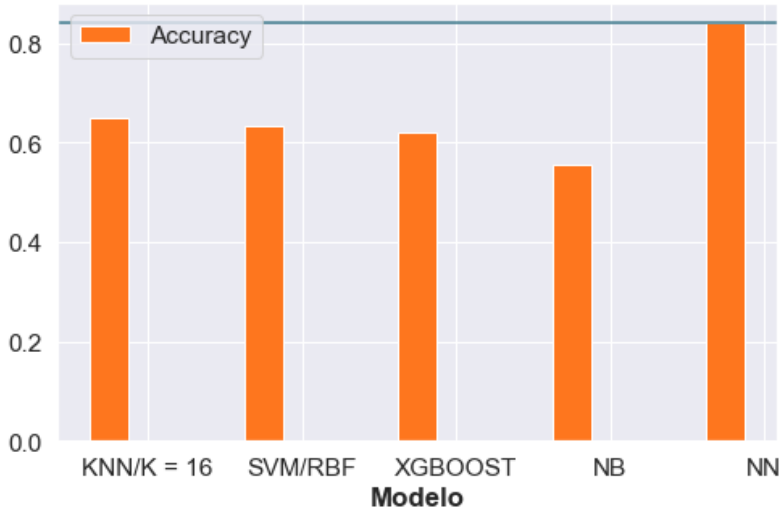


Figura No. 4.5 Comparativa exactitud de modelos entrenados

4.2 REPORTES DE CLASIFICACIÓN.

Para observar mejor el comportamiento en el sistema, se realiza la implementación del reporte de clasificación de los algoritmos, esto permite obtener información específica de como responden estos algoritmos en cada clase o zona a estimar localización, permitiendo observar en cuales es posible estimar con seguridad la ubicación del nodo y en cuales existe un peor desempeño.

Para este reporte se implementan tres nuevas métricas para medir rendimiento de los algoritmos, para clasificación es posible considerar una predicción como verdadera o falsa en función de las clase a predecir, siendo verdadera positiva y verdadera negativa (VP y VN) las salidas deseadas en este tipo de sistemas, como contrapartida falsa positiva o falsa negativa (FP y FN), son salidas que resultan en predicciones empeorando el rendimiento de este, estas salidas son evaluadas con las métricas precisión y exhaustividad (precision y recall en inglés) y la relación entre estas conocida como valor-F(F1 Score en inglés)

Precisión es la métrica que indica en la exactitud del modelo cuantas predicciones fueron positivas realmente, representando la habilidad del clasificador de no clasificar como negativo una salida positiva, una baja precisión disminuye el rendimiento para que el algoritmo pueda identificar correctamente una clase, en esta aplicación, precisión indica que tan bueno es el algoritmo para identificar cada zona, la generalización de esta métrica está dada por la ecuación 23

$$Precision = \frac{VP}{VP + FP} \tag{24}$$

Ahora bien, exhaustividad indica la habilidad del algoritmo para encontrar todas las muestras positivas, otra forma de medir la capacidad del algoritmo para encontrar correctamente la clase y evaluar cuantas veces este fallo en reconocerla, en otras palabras, dio valores de falso negativos, esta relación es vista en la ecuación 24

$$Exhaustividad = \frac{VP}{VP + FN} \tag{25}$$

Finalmente, el valor-F presenta la relación entre precisión y exhaustividad, esta relación puede ser vista en la ecuación 25.

$$F1\ Score = \frac{Precision * Exhaustividad}{Precision + Exhaustividad} \tag{26}$$

En síntesis, valor-F evalúa con igual importancia la exhaustividad y la precisión, a diferencia de la exactitud en estos casos se evalúan realmente cuando se va a clasificar exactamente la zona en función de la RSSI, no la capacidad de este de considerar las otras zonas como negativos de clasificación, a diferencia de la exactitud donde este caso si es considerado, al tener 45 zonas el algoritmo evalúa cada caso, por lo cual la cantidad de veces tendrá que dar resultados verdadero negativo será elevada y esta información es poco relevante, en otras palabras estas métricas presentan la calidad del modelo para identificar correctamente las zonas.

La información de estas métricas se puede encontrar en la tabla 2 (Anexo A), en esta se presenta un informe de resultados de los algoritmos en función de la precisión, para el caso de la tabla 3 (Anexo A) se visualiza la exhaustividad de estos algoritmos, por último, la tabla 4 presenta el valor-F obtenido, adicionalmente en las tablas es posible encontrar la cantidad de datos con por cada zona con los que fue realizado el reporte.

La tabla 5 (Anexo B) ilustra el desempeño de los algoritmos entrenados, específicamente en la red neuronal implementada es posible encontrar un bajo valor-F en la zona 23 (menor a 0.6) indicando que el algoritmo no es capaz de clasificar correctamente esta zona, se encuentra un grupo de rendimiento donde se tiene un valor aceptable pero no lo suficientemente bueno para las zonas 3, 15 y 25 (valor entre 0.6 y 0.69), adicional se mantienen las clases 7, 16, 31 y 41 con un valor muy cercano en el límite de un valor-F considerado en las tres zonas anteriores, las otras 37 zonas restantes indican un valor-F donde el algoritmo será capaz de identificar estas zonas.

De la misma forma se realiza el reporte de clasificación, específicamente el valor-F en la red neuronal con los datos de entrenamiento y se comparan con los de prueba, como puede ser visto en la tabla 5, la razón principal para realizar esta comparativa

se basa en la cantidad de datos totales, donde solo una muestra de 20% fue utilizada para prueba, motivo por el cual se opta por realizar esta comparación.

En la información presentada en la tabla 5 es posible confirmar que el algoritmo no logra clasificar la zona 23 durante el entrenamiento, pero presenta mejores resultados en las demás zonas consideradas anteriormente, llevando las zonas 15, 16 y 41 al grupo donde son clasificadas correctamente y subiendo al grupo de límite las demás zonas puestas en consideración, respecto a las 37 zonas restantes como es de esperarse debido a las gráficas de entrenamiento presentadas anteriormente, con valor-F cercano a 1, indicando que el algoritmo clasifica correctamente estas zonas.

4.3 MATRIZ DE CONFUSIÓN.

En un análisis de clasificación binario, la matriz de confusión únicamente consistiría en mostrar gráficamente los resultados de clasificación en los estados VP, VN, FP, FN, sin embargo, en multiclasificación la matriz de confusión resulta un método gráfico interesante para evaluar el rendimiento de la red neuronal.

En este caso, el algoritmo se evaluó individualmente las predicciones de cada zona hechas con el algoritmo y se comparó con la etiqueta de prueba, si el eje horizontal representa la etiqueta de predicción y el eje vertical la etiqueta esperada, cada predicción hecha correctamente se ubica en la diagonal de la matriz, lo que es lo mismo a que cada predicción mal realizada por la red neuronal se ubicará fuera de la diagonal, esto resulta interesante pues aunque ya es posible saber que el algoritmo presenta errores en zonas, como es el caso de la zona 23, introduce una información adicional a evaluar interesante, mediante la matriz de confusión es posible confirmar el rendimiento de la red neuronal pero adicional permite observar en las clases donde se presentan fallos que está prediciendo la red neuronal, en otras palabras, como se equivoca el algoritmo, esta matriz es presentada en un mapa de calor en la figura 4.6

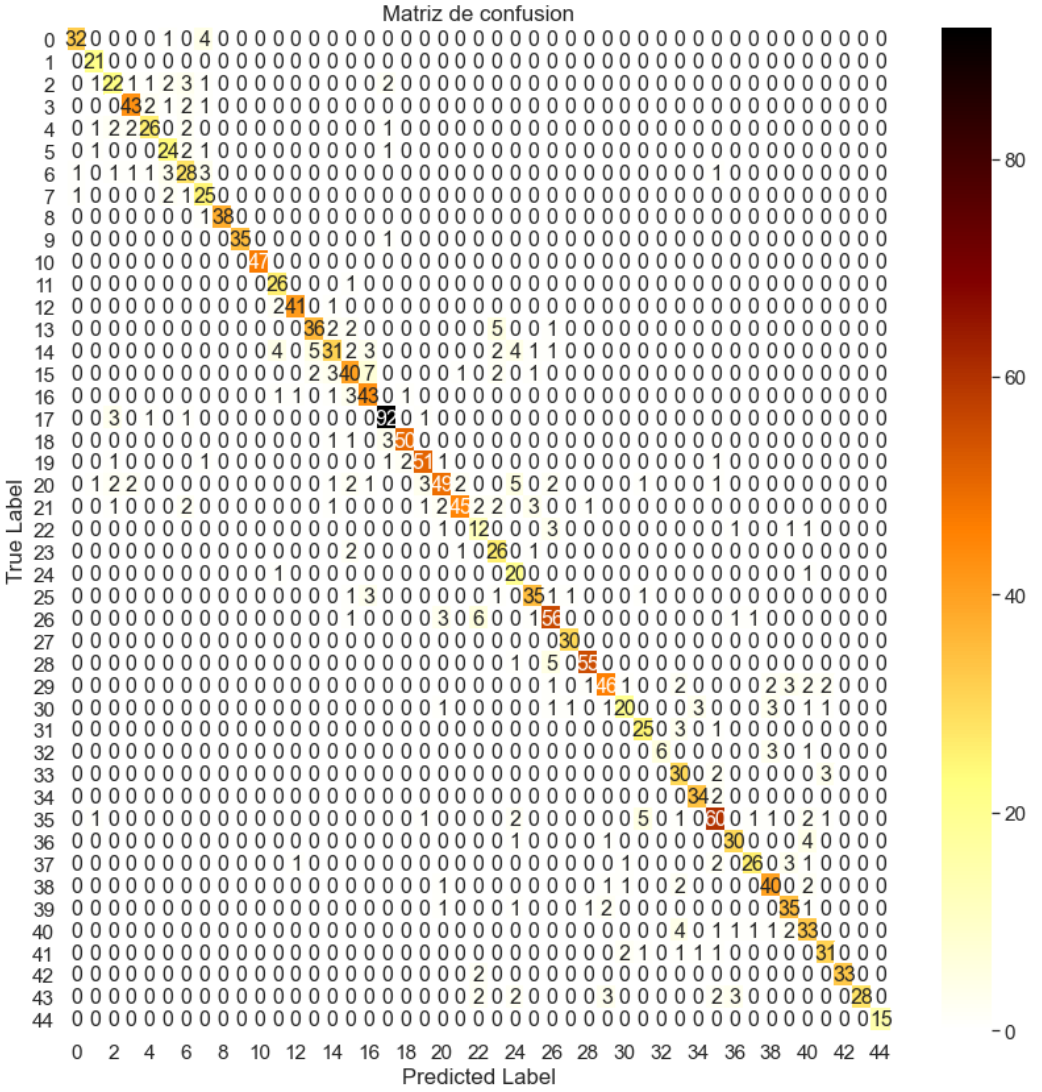


Figura No. 4.6 Matriz de confusión red neuronal.

En primer lugar, es importante interpretar la forma de presentar las clases en la matriz de confusión, las clases van etiquetadas desde 0 hasta 44, sin embargo, representan las zonas 1 a 45 respectivamente, adicionalmente la matriz es presentada mediante una gráfica tipo mapa de calor o heatmap en inglés, este tipo de gráfico nos permite observar la cantidad de datos acumulados en cada espacio de la matriz, evidentemente gracias al rendimiento del algoritmo descrito previamente, la mayor densidad de datos se encuentra en la diagonal de la matriz debido a la gran cantidad de predicciones acertadas por la red neuronal.

Al observar las etiquetas reales se observa como la red neuronal cuando realiza una predicción errónea, predice una zona muy cercana a la esperada, este es un comportamiento esperado, debido a que en zonas cercanas el nivel de RSSI tiende a ser similar, igualmente resulta interesante observar las zonas 1, 11 y 45 no tienen

errores de predicción, validando la información de las tablas de reporte de clasificación, y sus zonas cercanas tienden a tener pocos errores de predicción como es de esperarse de acuerdo al reporte dado, sin embargo, la matriz permite observar la concentración de errores puede ser observada en la matriz, este conjunto grande de errores que afectan negativamente el rendimiento del algoritmo evitando lograr la totalidad de predicciones correctas se encuentran en las clase de la mitad de la matriz, específicamente la zona 23, la cual fue previamente vista como la zona que peor rendimiento tuvo el algoritmo seguida de las zonas 15 y 16, es posible observar como las predicciones realizadas por la red neuronal son igualmente zonas muy cercanas a estas zonas mencionadas.

Esta información sacada de la matriz de confusión resulta valiosa al revisar la figura 4.6, sobre todo para confirmar la ubicación de estas zonas y las predicciones erróneas ubicadas zonas geográficamente cercanas, además cabe destacar la ubicación geográfica de estas zonas en todo el terreno de experimentación, aproximadamente ubicadas en el centro del terreno, una distancia intermedia a la ubicación entre las tres puertas de enlace, esto permite generar algunas hipótesis sobre el rendimiento del algoritmo, entre las cuales cabe destacar la falta de generar hiperplanos por parte de la red neuronal para segmentar correctamente las clases o zonas ubicadas en la mitad del terreno, al ser puntos donde la distancia entre el nodo y las puertas de enlace es común puede generarse dificultad en la red neuronal para identificar la zona de donde se obtienen los niveles de RSSI.

Esta hipótesis planteada deja abierta la opción de buscar formas para mejorar la red neuronal, una posibilidad interesante por explotar de acuerdo a lo descrito en el párrafo anterior consistiría en añadir una cuarta puerta de enlace ubicada en este punto central donde se encuentran los errores de clasificación para añadir una nueva entrada de datos a la red neuronal que facilite la identificación de estas zonas. Resulta lógico pensar en un incremento de costos de implementación el utilizar una cuarta puerta de enlace, evidentemente, al revisar los casos aislados de zonas alejadas del centro donde también resultan fallos de predicción aunque en una cantidad muchísimo menor en teoría no deberían verse afectados por la hipótesis planteada, de todos modos estas zonas presentan elevados porcentajes de predicción correctos y en una implementación en un contexto aplicado de este método debería evaluarse si este porcentaje de error es aceptable.

Finalmente, si se compara con trabajos previos que involucran redes LPWAN y el uso del RSSI para estimar ubicación, los métodos clásicos de regresión para predicción de latitud y longitud en promedio tienen un margen de error en un radio entre 300 m y 500 m (lo que en áreas circulares equivalen a 250000 m^2), siendo un terreno considerablemente mayor a las áreas de en promedio 800 m^2 sobre las cuales se desarrollaron las pruebas para este proyecto, además del uso de menor cantidad de puertas de enlace LoRa, minimizando el costo de implementación.

5. CONCLUSIONES Y RECOMENDACIONES.

5.1 CONCLUSIONES.

Se logró implementar una red LPWAN basada en LoRa mediante el uso de dispositivos Dragino. La red tuvo un nodo y tres puertas de enlace logrando transmitir sin línea de vista a una distancia máxima de 350m en un área de 48000m^2 ubicada en el corregimiento La Ulloa del municipio de Rivera, para todos los puntos dentro de esta área se logró envío y recepción de mensajes.

La zona de experimentación donde fue implementada la red consistió en un conjunto residencial, las divisiones existentes de las residencias, lotes y áreas comunes se utilizaron como puntos de referencia para el envío de mensajes desde el nodo LoRa, el envío de mensajes en cada zona permitió adquirir niveles de RSSI etiquetados con la información de la zona desde la cual fue enviado el mensaje, este registro es guardado para su posterior uso en la generación de modelos basados en algoritmos de aprendizaje de máquina para estimar la localización del nodo mediante estos modelos.

Se desarrollan cinco modelos de aprendizaje de máquina, cada modelo fue ajustado para obtener el mejor desempeño de estos modelos realizando el ajuste de hiperparámetros, estos modelos probabilísticos o de clasificación buscan predecir a partir del nivel de RSSI obtenido en las puertas de enlace el punto de origen de envío de mensaje utilizando el registro de datos obtenido para determinar la huella digital del RSSI en cada zona, debido a la relación no lineal entre RSSI y localización los modelos de aprendizaje tradicionales logran un buen desempeño con los datos utilizados durante el aprendizaje, sin embargo, con datos nuevos solo logran estimar correctamente el 60% de los casos en promedio, siendo esto un indicativo de sobre-entrenamiento en los modelos, motivo por el cual son descartados para estimar localización basado en intensidad de señal.

Las dificultades para estimar ciertas zonas se deben a que el nivel de RSSI no presentaba cambios significativos, debido a la naturaleza logarítmica de esta escala de referencia, obteniendo niveles de RSSI similares en cada gateway, imposibilitándole al algoritmo estimar correctamente la ubicación del nodo

Mediante el uso de Python y una API de alto nivel de modelamiento se desarrolla una red neuronal basada en modelos probabilísticos, esta red con topología totalmente conectada y salida de multiclasificación genera transformaciones no lineales de los niveles de RSSI para determinar la relación no lineal entre estos y la localización donde se obtuvieron estos niveles, la red tiene una salida de multiclasificación que estima la probabilidad para cada zona , siendo la salida con mayor probabilidad la determinada por el algoritmo como predicción de la red

neuronal, la generación de características no lineales dentro de la red permitió tener en este modelo mejor desempeño sobre los demás algoritmos, logrando predicciones el 94% del conjunto de datos de RSSI con los cuales fue entrenado el algoritmo y 84% sobre un conjunto de datos no vistos durante el entrenamiento.

Aunque el método utilizado para estimar la ubicación de los nodos no es el más preciso, se comprobó que con equipo de bajo costo se pudieron conseguir probabilidades de más del 80% para las zonas establecidas sin demandar un alto costo computacional, siendo esta una alternativa a los métodos vistos en trabajos previos en donde la estimación se basa en una regresión, en la que se relaciona la posición con los valores de latitud y longitud de un punto, obteniendo rangos de estimación con fiabilidad de 800 m^2.

5.2 RECOMENDACIONES.

Una limitante del gateway Dragino LG01 es el uso del chip SX1276, este al ser monocanal solo permite la comunicación por un nodo, se recomienda el uso de gateway LoRa que tengan integrados chip multicanal como el SX1301 o SX1308 para implementar una estructura de red LoRaWAN más sólida.

Diseñar e implementar un sistema de alimentación fotovoltaico para suplir la demanda energética de los dispositivos en vista que este sistema está pensado principalmente para entornos rurales donde se requiera fuentes de energías distintas a la red eléctrica convencional

Aumentar el número de gateways de la red para obtener un mayor número de variables de referencia para facilitar al algoritmo la estimación de la ubicación al aumentar el número de entradas que recibe.

Implementar el sistema elevando los gateways en zonas donde pueda lograr línea de vista dentro de toda la zona para estimar localización, evitando perdidas en los mensajes transmitidos y disminuyendo el problema de sincronización de datos.

Implementar pruebas y estudios con distintas configuraciones de parámetros de la red tales como el SF, ancho de banda y potencia de transmisión y contrastar los distintos resultados posibles.

Automatizar la etapa de recolección de datos para entrenar el algoritmo, implementando una infraestructura que permita guardar los niveles de RSSI en una

base de datos y poder realizar posteriormente con mayor facilidad el proceso de limpieza de estos.

Investigar e implementar modelos de aprendizaje de máquina de mayor complejidad y comparar resultados a fin de lograr mejores oportunidades de estimación de ubicación

Realizar comparativas y estudios de costos con otros sistemas de ubicación.

Implementar el modelo entrenado en un sistema productivo y realizar una investigación respecto a la eficiencia del modelo con el paso del tiempo, evaluando el impacto causado por la degeneración del modelo de aprendizaje de máquina

REFERENCIAS.

AZMI, Nur A; SAMSUL, Shafiqa; YAMADA, Yoshihide; YAKUB, Mohd F M; ISMAIL, Mohd I M; DZIYAUDDIN, Rudzidatul A. A Survey of Localization using RSSI and TDoA Techniques in Wireless Sensor Network: System Architecture. En: *2018 2nd International Conference on Telematics and Future Generation Networks (TAFGEN)* [en linea]. Kuching, Malaysia, diciembre 2018. DOI https://doi.org/10.1109/TAFGEN.2018.8580464 E-ISBN 978-1-5386-1275-0

Airspayce. RadioHead Packet Radio library for embedded microprocessors [en linea]. *Airspayce*, arbil 2014. Disponible en https://www.airspayce.com/mikem/arduino/RadioHead/

BELLENSKENS, Ben; AERNOUTS, Michiel; BERKENS, Rafael y WEYN, Maarten A Comparison of Signal Strength Localization Methods with Sigfox. En: 2018 15th Workshop on Positioning, Navigation and Communications (WPNC) [en línea]. Breme, Germany diciembre 2018. DOI https://doi.org/10.1109/WPNC.2018.8555743 E-ISBN 978-1-5386-6436-0

BONANAFI, F; FERNANDEZ, Carvalho D; DEPARI, A; FERRARI, P; FLAMMINI, A; PASETTI M; RINALDI, S y SISINNI, E. Evaluating indoor and outdoor localization services for LoRaWAN in Smart City applications. En: *II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 IoT)* [en línea]. Naples, Italy, agosto 2019, p. 300-305. DOI https://doi.org/10.1109/METROI4.2019.8792901 E-ISBN 978-1-7281-0429-4

CRUZ, Gómez D. E y MENDOZA, Ortiz J. C. Aplicativo móvil para localizar y guiar personas a través del campus universitario de la Pontificia Universidad Javeriana. En: *Pontificia Universidad Javeriana* [en línea]. Bogota: Universidad Javeriana, 2016, Disponible en: http://repository.javeriana.edu.co/handle/10554/21446

DARAMOUSKAS, Ioannis; KAPOULAS, Vaggelis y PARASKEVAS, Michael. Using Neural Networks for RSSI Location Estimation in LoRa Networks. En: *10th International Conference on Information, Intelligence, Systems and Applications (IISA)* [en línea], Patras, Greece, noviembre 2019, p. 1-7. DOI https://doi.org/10.1109/IISA.2019.8900742 E-ISBN 978-1-7281-4959-

DA SILVA, Wesley R; OLIVEIRA Luiz; KUMAR, Neerai; RABELO, Ricardo A.L; MARINS, Carlos N.M y RODRIGUEZ Joel J. P. C. An Internet of Things Tracking System Approach Based on LoRa Protocol. En: *IEEE Global Communications Conference (GLOBECOM)* [en línea], Abu Dhabi, United Arab Emirates, febrero 2019, p. 1-7. DOI https://doi.org/ 10.1109/GLOCOM.2018.8647984 E-ISBN 978-1-5386-4727-1

Dragino. Lora Shield [en linea]. *Dragino,* julio 2016. Disponible en https://wiki.dragino.com/index.php?title=Lora_Shield

FRANCESCO, Carrino; ALES, Janka; OMAR, Abou K y ELENA, Mugellini. LoRaLoc: Machine Learning-Based Fingerprinting for Outdoor Geolocation using LoRa. En*: 6th Swiss Conference on Data Science (SDS)* [en línea], Bern, Switzerland, agosto 2019, p. 82-86. https://doi.org/10.1109/SDS.2019.000-2 E-ISBN 978-1-7281-3105-4

GARCIA, A. F; GOMEZ, C; SANCHEZ, T; RENDONDO, A. D; BETANCOUR, L y HINCAPIE, R. C. Algoritmos de Radiolocalización basados en ToA, TDoA y AoA. En: *Ingeniería y Región* [en linea], Neiva: Universidad Surcolombiana, febrero 2016 v.14, p. 9-22. DOI https://doi.org/10.25054/22161325.689

GERON, Aurelien. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems [en linea]. 2 ed. *O'Reilly Media, Inc* 2019. Disponible en https://www.amazon.com/-/es/Aur-C3-A9lien-G-C3-A9ron-dp-1492032646/dp/1492032646/ref=dp_ob_image_bk ISBN: 9781492032649

GOTTHARD Petr y JANKECHK Tomáš. Low-Cost Car Park Localization Using RSSI in Supervised LoRa Mesh Networks. En: *15th Workshop on Positioning, Navigation and Communications (WPNC)* [en línea], Bremen, Germany, diciembre 2018, p. 1-6. DOI https://doi.org/10.1109/WPNC.2018.8555792 E-ISBN 978-1-5386-6436-0

HASTIE, Trevor; TIBSHIRANI, Robert y FRIEDMAN, Jerome. The Elements of Statistical Learning: Data Mining, Inference, and Prediction [en linea]. 2 ed. *Springer* 2016. Disponible en https://www.amazon.com/-/es/Trevor-Hastie/dp/0387848576 ISBN: 0387848576

ISLAM, Bashima; ISLAM, Md Tamzeed y KAUR, Jasleen. LoRaIn: Making a Case for LoRa in Indoor Localization. En: *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* [en línea], Kyoto, Japan, p. 423-426. DOI https://doi.org/10.1109/PERCOMW.2019.8730767 E-ISBN 978-1-5386-9151-9

JANSSEN, Thomas; AERNOUTS, Michiel; BERKENS, Rafael y WEYN, Maarten. Outdoor Fingerprinting Localization Using Sigfox. En: *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)* [en línea], Nantes, France, noviembre 2018, p 1-6. DOI https://doi.org/10.1109/IPIN.2018.8533826 E-ISBN 978-1-5386-5635-8

LAM, Ka-Ho; CHEUNG, Chi-Chung y LEE, Wah-Ching. New RSSI-Based LoRa Localization Algorithms for Very Noisy Outdoor Environment. En: *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)* [en línea]. Tokyo, Japan, junio 2018, p. 794-799. DOI https://doi.org/10.1109/COMPSAC.2018.10340 E-ISBN 978-1-5386-2667-2

LEMIC, Filip; BEHBOODI, Arash; FAMAEY, Jeroen y MATHAR, Rudolf. Location-Based Discovery and Vertical Handover in Heterogeneous Low-Power Wide-Area Networks (2019). En: *IEEE Internet of Things Journa* [en línea]*l*, marzo 2018, vol. 6, nro. 6, p 10150-10165. DOI https://doi.org/10.1109/JIOT.2019.2935804

LI, You; HE, Zhe; LI, Yuqi; XU, Hongliang; Pei, Ling y Zhang, Yu. Towards Location Enhanced IoT: Characterization of LoRa Signal For Wide Area Localization. En: *2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS)* [En línea]. Wuhan, China. diciembre 2018, p. 1-7. DOI https://doi.org/10.1109/UPINLBS.2018.8559844 E-ISBN 978-1-5386-3755-5

Lora Alliance. A technical overview of LoRa® and LoRaWAN® [documento digital]. *Lora Alliance - Resources*, noviembre 2015. Disponible en https://lora-alliance.org/resource_hub/what-is-lorawan/

MAHNOOR Anjum; MUHAMMAD, Abdullah K; SYED, Ali. H; AAMIR Mahmood y MIKAEL Gidlund. Analysis of RSSI Fingerprinting in LoRa Networks. En: *15th International Wireless Communications Mobile Computing Conference (IWCMC)* [en línea], Tangier, Morocco, julio 2019. p 1178-1183. https://doi.org/10.1109/IWCMC.2019.8766468 E-ISBN 978-1-5386-7747-6

MCCULLOCH, Warren Sturgis y WALTER, Pitts. A logical calculus of the ideas immanent in nervous activity. En: *Springer - Bulletin of Mathematical* Biophysics. The University of Chicago, Chicago, USA, diciembre 1943. vol. 5, nro. 1. DOI https://doi.org/10.1007/BF02478259.

MEKKI, Kais; BAJIC, Eddy; CHAXEL, Frederic y MEYER, Fernand. Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT. En: *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* [en linea]. Athens, Greece, octubre 2018. p. 1. DOI https://doi.org/10.1109/PERCOMW.2018.8480255 E-ISSN 978-1-5386-3227-7

MEKKI, Kais; BAJIC, Eddy; CHAXEL, Frederic y MEYER, Fernand. A comparative study of LPWAN technologies for large-scale IoT deployment. En: *ICT Express* [en linea]. The Korean Institute of Communications Information Sciences, marzo 2019. vol. 5, nro. 1. p. 1. DOI https://doi.org/10.1016/j.icte.2017.12.005 ISSN 2405-9595

PICKERING, Paul. Desarrollar con LoRa para aplicaciones IoT de baja tasa y largo alcance [blog]. Blogs *Digi-Key*. 29 de junio de 2017. Disponible en https://www.digikey.com/es/articles/develop-lora-for-low-rate-long-range-iot-applications

RAY, Brian. What Is LoRa? A Technical Breakdown [blog]. Blogs *Link Labs*. 26 de junio de 2018. Disponible en https://www.link-labs.com/blog/what-is-lora

SONG, Yonghua; LIN, Jin; TANG, Ming y DONG, Shufeng. An Internet of Energy Things Based on Wireless LPWAN. En: *Engineering* [en línea]. Chinese Academy of Engineering, agosto 2017. vol. 3, nro. 4. p. 2. DOI https://doi.org/10.1016/J.ENG.2017.04.011. ISSN 2095-8099

VERSLOOT, Christian. Understanding SVM and SVR for Classification and Regression [blog]. Blogs *MACHINECURVE* 20 de septiembre de 2019. Disponible en https://www.machinecurve.com/index.php/2019/09/20/intuitively-understanding-svm-and-svr/

ANEXOS.

ANEXO A. TABLAS RENDIMIENTO DE MODELOS DE MACHINE LEARNING

Tabla 2 Reporte de precisión de algoritmos

PRECISION						
ZONA	CANTIDAD DATOS	KNN	SVM	ÁRBOLES	BAYESIANO	RED NEURONAL
1	37	0,74	0,86	0,80	0,57	0,94
2	21	0,50	0,67	0,50	0,31	0,81
3	33	0,36	0,43	0,30	0,12	0,69
4	49	0,67	0,80	0,77	0,44	0,88
5	34	0,55	0,66	0,60	0,63	0,84
6	29	0,51	0,43	0,44	0,25	0,73
7	39	0,36	0,27	0,26	0,23	0,68
8	29	0,40	0,38	0,38	0,30	0,68
9	39	0,93	0,97	0,95	0,95	1,00
10	36	0,94	0,96	0,85	0,85	1,00
11	47	1,00	1,00	1,00	1,00	1,00
12	27	0,67	0,77	0,86	0,65	0,76
13	44	0,92	0,98	0,93	0,90	0,95
14	46	0,53	0,71	0,63	0,34	0,84
15	53	0,44	0,42	0,37	0,52	0,76
16	56	0,38	0,22	0,31	0,39	0,73
17	50	0,64	0,64	0,61	0,54	0,75
18	98	0,83	0,81	0,84	0,72	0,91
19	55	0,84	0,81	0,73	0,72	0,94
20	58	0,58	0,62	0,66	0,41	0,89
21	72	0,44	0,37	0,41	0,24	0,83
22	60	0,57	0,56	0,56	0,05	0,92
23	19	0,31	0,29	0,13	0,17	0,50
24	30	0,44	0,45	0,47	0,47	0,68
25	22	0,37	0,30	0,29	0,15	0,56
26	43	0,72	0,72	0,52	0,54	0,83
27	69	0,58	0,54	0,53	0,48	0,79
28	30	0,86	0,88	0,88	0,86	0,94
29	61	0,87	0,95	0,85	0,78	0,95
30	60	0,49	0,42	0,42	0,40	0,85
31	32	0,33	0,23	0,35	0,83	0,80

32	29	0,70	0,81	0,64	0,62	0,76
33	10	0,56	0,25	0,44	0,42	1,00
34	35	0,52	0,59	0,57	0,51	0,70
35	36	0,79	0,89	0,86	0,83	0,89
36	75	0,66	0,60	0,59	0,62	0,81
37	36	0,80	0,83	0,62	0,60	0,83
38	34	0,65	0,71	0,45	0,33	0,90
39	47	0,61	0,54	0,48	0,40	0,80
40	41	0,42	0,60	0,47	0,45	0,80
41	43	0,53	0,50	0,45	0,45	0,67
42	37	0,81	0,83	0,70	0,55	0,82
43	35	0,83	0,94	0,92	0,90	1,00
44	40	0,78	0,79	0,86	0,81	1,00
45	15	1,00	1,00	0,94	1,00	1,00

Tabla 3 Reporte de exhaustividad de algoritmos.

RECALL						
ZONA	**CANTIDAD DATOS**	**KNN**	**SVM**	**ÁRBOLES**	**BAYESIANO**	**RED NEURONAL**
1	37	0,86	0,81	0,89	0,81	0,86
2	21	0,86	0,76	0,67	0,71	1,00
3	33	0,30	0,36	0,27	0,06	0,67
4	49	0,63	0,67	0,76	0,49	0,88
5	34	0,50	0,56	0,53	0,29	0,76
6	29	0,62	0,52	0,48	0,31	0,83
7	39	0,26	0,18	0,23	0,15	0,72
8	29	0,21	0,38	0,41	0,21	0,86
9	39	0,95	0,90	0,92	0,92	0,97
10	36	0,83	0,72	0,81	0,81	0,97
11	47	0,96	0,87	1,00	0,94	1,00
12	27	0,96	0,89	0,93	0,74	0,96
13	44	1,00	1,00	0,98	1,00	0,93
14	46	0,65	0,65	0,63	0,37	0,78
15	53	0,30	0,38	0,32	0,21	0,58
16	56	0,38	0,46	0,34	0,45	0,71
17	50	0,64	0,58	0,60	0,64	0,86
18	98	0,94	0,90	0,90	0,93	0,94
19	55	0,78	0,78	0,75	0,85	0,91
20	58	0,79	0,72	0,71	0,16	0,88
21	72	0,32	0,33	0,29	0,14	0,68

22	60	0,60	0,65	0,60	0,02	0,75
23	19	0,21	0,11	0,11	0,05	0,63
24	30	0,57	0,50	0,57	0,67	0,87
25	22	0,32	0,32	0,32	0,45	0,91
26	43	0,30	0,30	0,33	0,44	0,81
27	69	0,70	0,72	0,61	0,72	0,81
28	30	1,00	1,00	1,00	1,00	1,00
29	61	0,89	0,89	0,92	0,87	0,90
30	60	0,40	0,47	0,37	0,35	0,77
31	32	0,22	0,19	0,38	0,16	0,63
32	29	0,66	0,59	0,62	0,79	0,86
33	10	0,50	0,10	0,40	0,80	0,60
34	35	0,91	0,86	0,71	0,66	0,86
35	36	0,94	0,92	0,89	0,94	0,94
36	75	0,63	0,65	0,59	0,53	0,80
37	36	0,67	0,67	0,64	0,69	0,83
38	34	0,71	0,71	0,59	0,38	0,76
39	47	0,57	0,53	0,55	0,40	0,85
40	41	0,37	0,44	0,41	0,46	0,85
41	43	0,44	0,47	0,30	0,12	0,77
42	37	0,68	0,65	0,62	0,76	0,84
43	35	1,00	0,97	0,97	1,00	0,94
44	40	0,98	0,83	0,75	0,95	0,70
45	15	1,00	0,67	1,00	1,00	1,00

Tabla 4 Reporte de Valor-F de algoritmos.

F1						
ZONA	**CANTIDAD DATOS**	**KNN**	**SVM**	**ÁRBOLES**	**BAYESIANO**	**RED NEURONAL**
1	37	0,80	0,83	0,85	0,67	0,90
2	21	0,63	0,71	0,57	0,43	0,89
3	33	0,33	0,39	0,29	0,08	0,68
4	49	0,65	0,73	0,76	0,47	0,88
5	34	0,52	0,60	0,56	0,40	0,80
6	29	0,56	0,47	0,46	0,28	0,77
7	39	0,30	0,22	0,25	0,18	0,70
8	29	0,27	0,38	0,39	0,24	0,76
9	39	0,94	0,93	0,94	0,94	0,99
10	36	0,88	0,83	0,83	0,83	0,99
11	47	0,98	0,93	1,00	0,97	1,00

12	27	0,79	0,83	0,89	0,69	0,85
13	44	0,96	0,99	0,96	0,95	0,94
14	46	0,58	0,68	0,63	0,35	0,81
15	53	0,36	0,40	0,34	0,30	0,66
16	56	0,38	0,30	0,32	0,42	0,72
17	50	0,64	0,61	0,61	0,59	0,80
18	98	0,88	0,85	0,87	0,81	0,92
19	55	0,81	0,80	0,74	0,78	0,93
20	58	0,67	0,67	0,68	0,23	0,89
21	72	0,37	0,35	0,34	0,18	0,75
22	60	0,59	0,60	0,58	0,03	0,83
23	19	0,25	0,15	0,11	0,08	0,56
24	30	0,49	0,48	0,52	0,55	0,76
25	22	0,34	0,31	0,30	0,23	0,69
26	43	0,43	0,43	0,40	0,49	0,82
27	69	0,63	0,62	0,57	0,57	0,80
28	30	0,92	0,94	0,94	0,92	0,97
29	61	0,88	0,92	0,88	0,82	0,92
30	60	0,44	0,44	0,39	0,38	0,81
31	32	0,26	0,21	0,36	0,26	0,70
32	29	0,68	0,68	0,63	0,70	0,81
33	10	0,53	0,14	0,42	0,55	0,75
34	35	0,67	0,70	0,63	0,58	0,77
35	36	0,86	0,90	0,88	0,88	0,92
36	75	0,64	0,63	0,59	0,57	0,81
37	36	0,73	0,74	0,63	0,64	0,83
38	34	0,68	0,71	0,51	0,36	0,83
39	47	0,59	0,54	0,51	0,40	0,82
40	41	0,39	0,51	0,44	0,46	0,82
41	43	0,48	0,48	0,36	0,19	0,72
42	37	0,74	0,73	0,66	0,64	0,83
43	35	0,91	0,96	0,94	0,95	0,97
44	40	0,87	0,80	0,80	0,87	0,82
45	15	1,00	0,80	0,97	1,00	1,00

ANEXO B. TABLA RENDIMIENTO DE MODELO DE RED NEURONAL

Tabla 5 Comparativa valor-F entrenamiento y prueba de red neuronal

REPORTE RED NEURONAL

ZONA	TRAIN		TEST	
	CANTIDAD	F1-SCORE	CANTIDAD	F1-SCORE
1	161	0,91	37	0,90
2	145	0,86	21	0,89
3	133	0,71	33	0,68
4	149	0,83	49	0,88
5	113	0,74	34	0,80
6	137	0,74	29	0,77
7	129	0,73	39	0,70
8	146	0,81	29	0,76
9	136	0,98	39	0,99
10	145	0,95	36	0,99
11	143	0,98	47	1,00
12	107	0,88	27	0,85
13	121	0,95	44	0,94
14	161	0,88	46	0,81
15	234	0,75	53	0,66
16	291	0,78	56	0,72
17	220	0,84	50	0,80
18	336	0,92	98	0,92
19	239	0,88	55	0,93
20	304	0,86	58	0,89
21	232	0,72	72	0,75
22	239	0,83	60	0,83
23	102	0,61	19	0,56
24	110	0,80	30	0,76
25	99	0,69	22	0,69
26	132	0,74	43	0,82
27	255	0,80	69	0,80
28	160	0,99	30	0,97
29	226	0,92	61	0,92
30	215	0,72	60	0,81
31	128	0,73	32	0,70
32	114	0,84	29	0,81
33	49	0,83	10	0,75
34	161	0,84	35	0,77
35	158	0,94	36	0,92

36	289	0,85	75	0,81
37	136	0,88	36	0,83
38	192	0,79	34	0,83
39	188	0,86	47	0,82
40	173	0,79	41	0,82
41	168	0,78	43	0,72
42	129	0,79	37	0,83
43	151	0,96	35	0,97
44	133	0,85	40	0,82
45	71	1,00	15	1,00

ANEXO C. CÓDIGO DE TRANSMISIÓN NODO LORA

```
#include <SPI.h>
#include <RH_RF95.h>
RH_RF95 rf95;
float frequency = 915.0;
int counter = 0;

char message = "lotecod"; /* Reemplazar cod por identificador zona*/

void setup()
{
  Serial.begin(9600);
  Serial.println("Start LoRa Client");
  if (!rf95.init())
    Serial.println("init failed");
  rf95.setFrequency(frequency);
  rf95.setTxPower(13);
  rf95.setSpreadingFactor(7);
  rf95.setSignalBandwidth(125000);
  rf95.setCodingRate4(5);
}
void loop()
{
  Serial.println("Sending to LoRa Server");
  String str = "Prueba en lotecas: "+String(counter)+"/";
  counter++;
  rf95.send(str.c_str(),str.length());
  rf95.waitPacketSent();
  delay(500);
}
```

ANEXO D. CÓDIGO RECEPCIÓN GATEWAY LORA.

```
#define BAUDRATE 115200
#include <Console.h>
#include <SPI.h>
#include <RH_RF95.h>
#include <FileIO.h>

RH_RF95 rf95;
float frequency = 915.0;

void setup()
{
  Bridge.begin(BAUDRATE);
  Console.begin();
  FileSystem.begin();
  while (!Console) ;
  Console.println("Iniciando comunicación");
  if (!rf95.init())
    Console.println("Inicio fallido");

  // Configuración banda de frecuencia ISM
  rf95.setFrequency(frequency);

  // Configuración potencia señal
  rf95.setTxPower(13);

  // Configuración SF
  rf95.setSpreadingFactor(7);

  // Asignar Ancho de Banda
  rf95.setSignalBandwidth(125000);

  // Configurar CR :5(4/5),6(4/6),7(4/7),8(4/8)
  rf95.setCodingRate4(5);

  Console.print("Escuchando en la frecuencia: ");
  Console.println(frequency);
}

void loop()
{
  if (rf95.available())
  {
```

```
    // Recepción de mensaje
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if (rf95.recv(buf, &len))
    {
      RH_RF95::printBuffer("request: ", buf, len);
      Console.print("got request: ");
      Console.println((char*)buf);
      Console.print("RSSI: ");
      Console.println(rf95.lastRssi(), DEC);


      // Almacenar RSSI en memoria

      // Crear cadena string mensaje y rssi
      String dataString = "";
      dataString = String((char*)buf) + "," + String(rf95.lastRssi(), DEC);

      File dataFile = FileSystem.open("/usr/datos.csv", FILE_APPEND);
        if (dataFile) {
        dataFile.println(dataString);
        dataFile.close();
        // print to the serial port too:
        Console.println(dataString);
      }
        else {
          Console.println("Error abriendo datos.csv");
        }
    }
    else
    {
      Console.println("recv failed");
    }
  }
}
```

ANEXO E. CÓDIGO ENTRENAMIENTO DE MODELOS MACHINE LEARNING Y RED NEURONAL

```
#Librerias Base:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split # Utilidad dividir datos
```

```
#Metricas
from sklearn.metrics import r2_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
 np.random.seed(1234)
datos = pd.read_excel("datosTesis.xlsx",index_col=None)
datos = datos.abs()

#Preprocesamiento, escalado de características.

from sklearn.preprocessing import StandardScaler
Normalizador = StandardScaler()
datosN = Normalizador.fit_transform(X)


#Creación de conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)


#Listas auxiliares para validar modelos:
best_acc = []
best_model = []
best_precision = []
```

Entrenamiento KNN

```
from sklearn.neighbors import KNeighborsClassifier

K_max = 100
error_rate = []
r2_vector = []
acc_vector = []
# Will take some time
for i in range(1,K_max):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    y_pred = knn.predict(X_test)
    error_rate.append(np.mean(y_pred != y_test))
    r2_vector.append(r2_score(y_test, y_pred))
    acc_vector.append(accuracy_score(y_test, y_pred))

#Visualizar resultados:
```

```
plt.figure(figsize=(18,8))

plt.subplot(121)
plt.plot(range(1,K_max),r2_vector,color='blue', linestyle='dashed', marker='o',
        markerfacecolor='red', markersize=10)
plt.title('R2 vs. K Value')
plt.xlabel('K')
plt.ylabel('R2')
plt.grid()


plt.subplot(122)
plt.plot(range(1,K_max),acc_vector,color='blue', linestyle='dashed', marker='o',
        markerfacecolor='red', markersize=10)
plt.title('Acc vs. K Value')
plt.xlabel('K')
plt.ylabel('Acc')
plt.grid()

#Revisión mejor valor de K obtenido (K = 16)

knn = KNeighborsClassifier(n_neighbors=16)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy:",accuracy_score(y_test, y_pred))
print("R^2:",r2_score(y_test, y_pred))
print(classification_report(y_test,y_pred))
best_acc.append(1-np.mean(y_pred != y_test))
best_model.append('KNN/K=16')

report = classification_report(y_test,y_pred, output_dict=True )
best_precision.append(report['macro avg']['precision'])

# Guardar reporte como tabla de excel.
df_reporte_clasificacion = pd.DataFrame(report).transpose()
df_reporte_clasificacion.to_excel("KNN_Class.xlsx")
```

Entrenamiento SVC.

```
from sklearn.svm import SVC

#Ajuste de Kernel.
```

```
Kernels = ['linear', 'poly', 'rbf', 'sigmoid']
acc_rate = []
r2_vector = []
# Will take some time
for i in range(len(Kernels)):
    svc = SVC(kernel = Kernels[i])
    svc.fit(X_train,y_train)
    y_pred = svc.predict(X_test)
    acc_rate.append(1-np.mean(y_pred != y_test))
    r2_vector.append(r2_score(y_test, y_pred))


# Visualizar resultados

barWidth = 0.25
plt.figure(figsize=(10,5))
# set height of bar
bars1 = acc_rate
bars2 = r2_vector
r1 = np.arange(len(bars1))
r2 = [x + barWidth for x in r1]

plt.bar(r1, bars1, color='#FE761E', width=barWidth, edgecolor='white', label='Acc
Rate')
plt.bar(r2, bars2, color='#3A778A', width=barWidth, edgecolor='white', label='R2')

plt.xlabel('group', fontweight='bold')
plt.xticks([r + barWidth for r in range(len(bars1))], Kernels)

plt.legend(loc='upper right')
plt.xlabel('Modelo')
plt.axhline(y=0.0, color='gray', linestyle='--')
plt.axhline(y=acc_rate[1], color='#FE761E', linestyle='-')
plt.axhline(y=r2_vector[1], color='#3A778A', linestyle='-')
plt.title('Accuracy & R^2 | Tuning SVC Kernel')


#Revisión mejor SVC obtenido: (C = 1 , kernel tipo rbf)

svc = SVC(kernel='rbf',gamma='auto', C = 1)
svc.fit(X_train, y_train)

y_pred = svc.predict(X_test)
print("Accuracy:",accuracy_score(y_test, y_pred))
```

```
print("R^2:",r2_score(y_test, y_pred))
print(classification_report(y_test,y_pred))
best_acc.append(1-np.mean(y_pred != y_test))
best_model.append('SVM/RBF')

report = classification_report(y_test,y_pred, output_dict=True )
best_precision.append(report['macro avg']['precision'])

# Guardar reporte como tabla de excel.
df_reporte_clasificacion = pd.DataFrame(report).transpose()
df_reporte_clasificacion.to_excel("SVC_RBF_1C_Class.xlsx")
```

Entrenamiento Árboles de decisión.

```
error_rate = []
r2_vector = []
type_model = []
acc_rate = []


#Árbol de decisión básico
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
print("R^2 Score en test: ", r2_score(y_test,y_pred))
print("Accuracy:",accuracy_score(y_test, y_pred))
#print(classification_report(y_test,y_pred))

#Añadir a vectores para visualizar resultados.
error_rate.append(np.mean(y_pred != y_test))
r2_vector.append(r2_score(y_test, y_pred))
type_model.append('Decision Tree')
acc_rate.append(1-np.mean(y_pred != y_test))


#Bosque aleatorio

from sklearn.ensemble import RandomForestClassifier
n_estimators = [1,10,100,200,1000]
acc_rate_rf = []
r2_vector_rf = []
# Will take some time
```

```
for estimator in n_estimators:
    model = RandomForestClassifier(n_estimators=estimator,
                                   bootstrap = True,
                                   max_features = 'sqrt')
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    acc_rate_rf.append(1-np.mean(y_pred != y_test))
    r2_vector_rf.append(r2_score(y_test, y_pred))


#Método avanzado de ensamblaje, XGBOOST

from xgboost import XGBClassifier
model = XGBClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:",accuracy_score(y_test, y_pred))
print("R^2:",r2_score(y_test, y_pred))
#print(classification_report(y_test,y_pred))

#Añadir a vectores para visualizar resultados.
error_rate.append(np.mean(y_pred != y_test))
r2_vector.append(r2_score(y_test, y_pred))
type_model.append('XGBOOST')
acc_rate.append(1-np.mean(y_pred != y_test))

best_acc.append(1-np.mean(y_pred != y_test))
best_model.append('XGBOOST')

report = classification_report(y_test,y_pred, output_dict=True )
best_precision.append(report['macro avg']['precision'])



#Almacenar en Excel información XGBOOST
df_reporte_clasificacion = pd.DataFrame(report).transpose()
df_reporte_clasificacion.to_excel("Arboles_XGBOOST.xlsx")
```

Entrenamiento NBC

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

```
y_pred = gnb.predict(X_test)
print("Accuracy:",accuracy_score(y_test, y_pred))
print("R^2:",r2_score(y_test, y_pred))
print(classification_report(y_test,y_pred))

best_acc.append(1-np.mean(y_pred != y_test))
best_model.append('Naive Bayesian')

report = classification_report(y_test,y_pred, output_dict=True )
best_precision.append(report['macro avg']['precision'])

#Reporte a Excel
df_reporte_clasificacion = pd.DataFrame(report).transpose()
df_reporte_clasificacion.to_excel("NBC.xlsx")
```

Comparación resultados modelos machine learning.

```
plt.figure(figsize=(10,5))
# set height of bar
bars1 = best_acc
bars2 = best_precision

r1 = np.arange(len(bars1))
r2 = [x + barWidth for x in r1]

plt.bar(r1, bars1, color='#FE761E', width=barWidth, edgecolor='white',
label='Accuracy')
plt.bar(r2, bars2, color='#3A778A', width=barWidth, edgecolor='white',
label='Precision')

plt.xlabel('group', fontweight='bold')
plt.xticks([r + barWidth for r in range(len(bars1))], ['KNN/K = 10', 'SVM/RBF',
'XGBOOST', 'NB'])

plt.legend()
plt.xlabel('Modelo')

plt.axhline(y=best_acc[n], color='#FE761E', linestyle='-')
plt.axhline(y=best_precision[n], color='#3A778A', linestyle='-')
```

Generar matriz de confusión KNN(n = 16)

```
#Create KNN Classifier
```

```
knn = KNeighborsClassifier(n_neighbors=16)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
import seaborn as sns

matrizConfusion = confusion_matrix(y_test,y_pred)

df_cm = pd.DataFrame(matrizConfusion)
df_cm.to_excel("CM_KNN.xlsx")

sns.set(font_scale=1.4)
fig_dims = (14,14)
fig, ax = plt.subplots(figsize=fig_dims)
sns_plot = sns.heatmap(df_cm, annot=True, annot_kws={"size": 16},ax = ax,
cmap="afmhot_r") # font size
sns_plot.set_title("Matriz de confusion")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
```

Entrenamiento Red Neuronal.

```
#MinMax Scaler
from sklearn.preprocessing import MinMaxScaler
#Over Sampling
from imblearn.over_sampling import SMOTE
#Keras utilidades:
from keras.utils.np_utils import to_categorical

#Cargar datos
datos = pd.read_excel("datosTesis.xlsx",index_col=None)
datos = datos.abs()
y = datos['Zona']
X = datos.drop('Zona',axis = 1)

smote = SMOTE()


Normalizador = MinMaxScaler()
XN = Normalizador.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(XN, y,
test_size=0.2,random_state=1234) #Revolver
print("Datasets originales:",X_train.shape, X_test.shape, y_train.shape,
y_test.shape)
X_sm, y_sm = smote.fit_sample(X_train,y_train)
print("Despues de Smote:",X_sm.shape,y_sm.shape)
# Convertir a salidas categóricas.
y_sm = to_categorical(y_sm, num_classes=46)

##Implementación red neuronal con Keras.

from keras.models import Sequential
from keras.layers import Activation, Dense
from keras import optimizers

model = Sequential()

#Arquitectura:
model.add(Dense(100, input_shape = (3, ),activation = 'relu')) # Capa de entrada
model.add(Dense(200,activation = 'relu')) #Capas ocultas
model.add(Dense(300,activation = 'relu')) #Capas ocultas
model.add(Dense(350,activation = 'relu')) #Capas ocultas
model.add(Dense(400,activation = 'relu'))
model.add(Dense(200,activation = 'relu'))
model.add(Dense(46,activation = 'softmax')) #Capa de salida

#Compilación modelo

opt = optimizers.Adam(learning_rate=0.001)
model.compile(optimizer = opt, loss = 'categorical_crossentropy', metrics =
['accuracy'])

history = model.fit(X_sm, y_sm, batch_size = 128, validation_split = 0.2, epochs =
1000, verbose = 1)

#Visualizar reporte entrenamiento.

#Exactitud
plt.figure(figsize = (10,6))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'], loc = 'upper left')
plt.xlabel("Epocas");plt.ylabel("Accuracy")
plt.grid(b = True)
```

```
plt.show()

#Pérdida
plt.figure(figsize = (10,6))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss', 'validation'], loc = 'upper left')
plt.xlabel("Epocas");plt.ylabel("Loss")
plt.grid(b = True)
plt.show()

#Resumen modelo (Aquitectura y parámetros)
model.summary()

#Prueba modelo datos test.
y_pred = model.predict(X_test, batch_size=64, verbose=1)
y_pred_bool = np.argmax(y_pred, axis=1)

print(classification_report(y_test, y_pred_bool))


matrizConfusion = confusion_matrix(y_test,y_pred_bool)
df_cm = pd.DataFrame(matrizConfusion)

sns.set(font_scale=1.4)
fig_dims = (14,14)
fig, ax = plt.subplots(figsize=fig_dims)
sns_plot = sns.heatmap(df_cm, annot=True, annot_kws={"size": 16},ax = ax,
cmap="afmhot_r") # font size
sns_plot.set_title("Matriz de confusion")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")

#Almacenar modelo para posterior uso
from keras.models import model_from_json
json_model = model.to_json()
with open('rssi_model84_smote.json', 'w') as json_file:
    json_file.write(json_model)
model.save_weights('rssi_weights84_smote.h5')

#Como cargar modelo.
with open('rssi_model86_smote.json', 'r') as json_file:
    json_savedModel= json_file.read()
model_j = model_from_json(json_savedModel)
```

```
model_j.summary()

model_j.load_weights('rssi_weights86_smote.h5')
opt = optimizers.Adam(learning_rate=0.001)
model_j.compile(loss='categorical_crossentropy',
        optimizer=opt,
        metrics=['accuracy'])

y_pred_j = model.predict(X_test, batch_size=64, verbose=1)
y_pred_bool_j = np.argmax(y_pred, axis=1)

report = classification_report(y_test, y_pred_bool_j, output_dict=True)
print(classification_report(y_test, y_pred_bool_j))

#Utilidad de visualización arquitectura.
from keras.utils.vis_utils import plot_model
plot_model(model, to_file='model_plot.png', show_shapes=True,
show_layer_names=True)

#Almacenar en Excel resultados.
df_reporte_clasificacion = pd.DataFrame(report).transpose()
df_reporte_clasificacion.to_excel("output.xlsx")
df_cm.to_excel("CM_NN.xlsx")
```