DISEÑO DE UN ALGORITMO DE POSICIONAMIENTO PARA INTERIORES BASADO EN MACHINE LEARNING

Darwin Rigoberto Mejía Chamorro
Cristhian Javier Pacichana Matituy

Director: MsC. Claudia Milena Hernández Bonilla

Universidad del Cauca
Facultad de Ingeniería Electrónica y de Telecomunicaciones
Departamento de Telecomunicaciones
Grupo de Radio e Inalámbricas GRIAL
Sistemas de Comunicaciones Móviles e Inalámbricos
Popayán
2022

DISEÑO DE UN ALGORITMO DE POSICIONAMIENTO PARA INTERIORES BASADO EN MACHINE LEARNING

Trabajo de Grado presentado como requisito parcial para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Darwin Rigoberto Mejía Chamorro
Cristhian Javier Pacichana Matituy

Director: MsC. Claudia Milena Hernández Bonilla

Universidad del Cauca
Facultad de Ingeniería Electrónica y de Telecomunicaciones
Departamento de Telecomunicaciones
Grupo de Radio e Inalámbricas GRIAL
Sistemas de Comunicaciones Móviles e Inalámbricos
Popayán
2022

TABLA DE CONTENIDO

1. CONCEPTOS GENERALES 3

1.1. POSICIONAMIENTO EN INTERIORES 3
1.2. PARÁMETROS DE POSICIONAMIENTO PARA INTERIORES 3
1.2.1. Ángulo de llegada 4
1.2.2. Tiempo de llegada 5
1.2.3. Diferencia en tiempo de llegada 5
1.2.4. Indicador de potencia de señal recibida 5
1.2.5. Estado de información del canal 6
1.3. TÉCNICAS DE POSICIONAMIENTO PARA INTERIORES 7
1.3.1. Trilateración 7
1.3.2. Triangulación 9
1.3.3. *Fingerprinting* 10
1.4. INTELIGENCIA ARTIFICIAL 12
1.4.1. *Machine learning* 13
1.5. ALGORITMOS DE PREDICCIÓN Y MACHINE LEARNING 15
1.5.1. K-vecinos más cercanos 15
1.5.2. Árbol de decisión 17
1.5.3. *Naive Bayes* 19
1.5.4. *Support vector machine* 21
1.5.5. Redes neuronales 23
1.6. BASES DE DATOS 30
1.6.1. *UJIIndoorLoc* 30
1.6.2. *Tampere* 31
1.6.3. Base de datos RSSI para localización en interiores 32
1.6.4. Base de datos CSI MAMIMO ultra denso para interiores 33
1.7. MÉTRICAS DE DESEMPEÑO 34
1.7.1. Tasa de aciertos 35
1.7.2. Exactitud 35
1.7.3. Precisión 35
1.7.4. Complejidad 35
1.8. DESARROLLO METODOLÓGICO 36
1.8.1. Definición de tareas 37

2. FASE DE PLANEACIÓN Y DISEÑO 38

2.1. HISTORIAS DE USUARIO 38
2.2. REQUERIMIENTOS 39
2.2.1. Requerimientos funcionales 39
2.2.2. Requerimientos no funcionales 39
2.3. DISEÑO 39

2.3.1. Análisis y selección de parámetros y técnicas 40
2.3.2. Diseño de fingerprinting 42
2.3.3. Análisis y selección de base de datos 43
2.3.4. Procesamiento de base de datos 44
2.3.5. Análisis y selección de algoritmos de machine learning 45
2.3.6. Sistema de posicionamiento con fingerprinting y RSSI 46

3. FASE DE IMPLEMENTACIÓN Y PRUEBAS 50

3.1. IMPLEMENTACIÓN DE ALGORITMOS 50
3.1.1. Implementación algoritmo K vecinos más cercanos 50
3.1.2. Implementación algoritmo redes neuronales 51
3.2. ENTRENAMIENTO DE ALGORITMOS 53
3.2.1. Entrenamiento algoritmo K vecinos más cercanos 53
3.2.2. Entrenamiento algoritmo redes neuronales 54
3.3. VALIDACIÓN DE ALGORITMOS 57
3.3.1. Pruebas y resultados 57
3.3.2. Predicción de edificio UJI 58
3.3.3. Predicción de piso UJI 61
3.3.4. Predicción de coordenadas con UJI 66
3.3.5. Predicción de coordenadas con BDRSSI 72
3.4. MODELO DE PREDICCIÓN FINAL 77

4. CONCLUSIONES Y TRABAJOS FUTUROS 80

4.1. CONCLUSIONES 80
4.2. TRABAJOS FUTUROS 82

5. REFERENCIAS 83

ANEXO A: RESULTADOS DE LOS ALGORITMOS DE POSICIONAMIENTO... 89

LISTA DE FIGURAS

Figura 1-1: Sistema basado en AOA. 4
Figura 1-2: Nivel de señal para RSSI. 6
Figura 1-3: Método de trilateración bidimensional. 8
Figura 1-4: Técnica de triangulación. 9
Figura 1-5: Fase de calibración. 11
Figura 1-6: Fase de posicionamiento fingerprinting. 12
Figura 1-7: Inteligencia artificial y Machine Learning. 14
Figura 1-8: Algoritmo K-NN, parámetro k=3 y k=5. 17
Figura 1-9: Árbol de decisión. 18
Figura 1-10: Tipos de ganancia de información. 19
Figura 1-11: Trazo hiperplano para el algoritmo SVM. 22
Figura 1-12: Transformación de entrada con Kernel. 22
Figura 1-13: Estructura de una neurona artificial. 23
Figura 1-14: Red Neuronal Secuencial Densa. 24
Figura 1-15: Redes neuronales convolucionales. 26
Figura 1-16: Función sigmoide y RELU. 28
Figura 1-17: Base de datos UJIIndoorLoc en 3D. 31
Figura 1-18: Base de datos Tampere en 3D. 32
Figura 1-19: Escenarios de entrenamiento. 33
Figura 1-20: Recolección de muestras CSI. 34
Figura 1-21: Metodología de desarrollo XP. 37
Figura 1-22: Ciclo de desarrollo XP. 37
Figura 2-1: Diagrama de bloques de un IPS. 40
Figura 2-2: Diagrama de bloques técnica fingerprinting. 42
Figura 2-3: a) BD entrenamiento edificio UJI, b) BD Validación edificio UJI. 44
Figura 2-4: a) BD entrenamiento piso UJI, b) BD Validación piso UJI. 44
Figura 2-5: a) BD entrenamiento coordenadas UJI, b) BD Validación coordenadas UJI. 44
Figura 2-6: a) BD entrenamiento escenario 3, b) BD Validación escenario 3. 45
Figura 2-7: Sistema IPS con fingerprinting y ML. 47
Figura 2-8: Diagrama de flujo algoritmo KNN. 48
Figura 2-9: Diagrama de flujo algoritmo NN. 49
Figura 3-1: Pseudocódigo KNN. 51
Figura 3-2: Pseudocódigo DSNN. 52
Figura 3-3: Pseudocódigo CNN. 53
Figura 3-4: Matriz de confusión predicción edificio KNN. 60
Figura 3-5: Matriz de confusión predicción edificio DSNN. 61
Figura 3-6: Matriz de confusión predicción piso KNN. 65
Figura 3-7: Matriz de confusión predicción piso DSNN. 65
Figura 3-8: Ajuste de distribuciones KNN UJI. 71
Figura 3-9: Ajuste de distribuciones DSNN UJI. 71
Figura 3-10: Ajuste de distribuciones CNN UJI. 72
Figura 3-11: Ajuste de distribuciones KNN BDRSSI. 76
Figura 3-12: Ajuste de distribuciones DSNN BDRSSI. 76

Figura 3-13: Ajuste de distribuciones CNN BDRSSI. 77
Figura 3-14: Modelo de predicción UJI. 78
Figura 3-15: Modelo de predicción BDRSSI. 79

LISTA DE TABLAS

Tabla 1-1: Ejemplo de base de datos para algoritmos ML. 15
Tabla 2-1: Historia de usuario 1. 38
Tabla 2-2: Historia de usuario 2. 38
Tabla 2-3: Historia de usuario 3. 39
Tabla 2-4: Parámetros IPS. 40
Tabla 2-5: Técnicas IPS. 41
Tabla 2-6: Bases de datos para IPS. 43
Tabla 2-7: Algoritmos de ML para IPS. 46
Tabla 3-1: Hiperparámetros algoritmo KNN. 54
Tabla 3-2: Hiperparámetros fijos algoritmo NN. 55
Tabla 3-3: Hiperparámetros algoritmo NN. 56
Tabla 3-4: Hiperparámetros algoritmo CNN. 56
Tabla 3-5: Resultados predicción edificio KNN. 59
Tabla 3-6: Resultados predicción de edificio DSNN. 59
Tabla 3-7: Hiperparámetros finales para predicción de edificio. 59
Tabla 3-8: Resultados predicción de edificio. 60
Tabla 3-9: Resultados predicción de piso KNN. 62
Tabla 3-10: Resultados arquitectura predicción de piso DSNN. 62
Tabla 3-11: Resultados optimizador y LR predicción de piso DSNN. 63
Tabla 3-12: Resultados épocas predicción de piso DSNN. 63
Tabla 3-13: Hiperparámetros finales para predicción de piso. 64
Tabla 3-14: Resultados predicción de piso. 64
Tabla 3-15: Resultados predicción de coordenadas KNN UJI. 67
Tabla 3-16: Resultados arquitectura predicción de coordenadas DSNN UJI. 68
Tabla 3-17: Resultados optimizador y LR predicción de coordenadas DSNN UJI. 68
Tabla 3-18: Resultados épocas predicción de coordenadas DSNN UJI. 69
Tabla 3-19: Hiperparámetros finales para predicción de coordenadas UJI. 69
Tabla 3-20: Resultados predicción de coordenadas UJI. 70
Tabla 3-21: Resultados predicción de coordenadas KNN BDRSSI. 73
Tabla 3-22: Resultados arquitectura de predicción de coordenadas DSNN BDRSSI. 73
Tabla 3-23: Resultados optimizador y LR predicción de coordenadas DSNN BDRSSI. 74
Tabla 3-24: Resultados épocas predicción de coordenadas DSNN BDRSSI. 74
Tabla 3-25: Hiperparámetros finales para predicción de coordenadas BDRSSI. 75
Tabla 3-26: Resultados predicción de coordenadas BDRSSI. 75

LISTA DE ACRÓNIMOS

AI	*Artificial Intelligence*, Inteligencia Artificial.
AIC	*Akaike Information Criterion,* Criterio de Información de Akaike.
AOA	*Angle Of Arrival*, Ángulo de Llegada.
ASM	*Attribute Selection Measures,* Medidas de Selección de Atributos.
BD	Base de datos.
BDCSI	Base de Datos CSI MAMIMO Ultra Denso para Interiores.
BDRSSI	Base de Datos RSSI para Localización Interior.
CNN	*Convolutional Neural Network,* Red Neuronal Convolucional.
CSI	*Channel State Information*, Información del Estado del Canal.
GLONASS	*Global Navigation Satellite System*, Sistema Global de Navegación por Satélite.
GPS	*Global Positioning System,* Sistema de Posicionamiento Global.
ID3	*Iterative Dichotomiser 3,* Dicotomizador Iterativo 3.
IPS	*Indoor Positioning System*, Sistema de Posicionamiento en Interiores.
KNN	*K-Nearest Neighbor,* K-Vecinos Más Cercanos.
LORAN	*Long Range Navigation*, Navegación de Largo Alcance.
LOS	*Line of Sight,* Línea de Vista.
LR	*Learning Rate,* Tasa de Aprendizaje.
MIMO	*Multiple-Input Multiple-Output,* Múltiple Entrada Múltiple Salida.
ML	*Machine Learning*, Aprendizaje Automático.
MLE	*Maximum Likelihood Estimation,* Estimación de Máxima Verosimilitud.
NLP	*Natural Language Processing,* Procesamiento del Lenguaje Natural.
NN	*Neural Networks,* Redes Neuronales.
OFDM	*Orthogonal Frequency-Division Multiplexing,* Multiplexación por División de Frecuencia Ortogonal.
RM	*Radio Map,* Mapa Radio.
RSSI	*Received Signal Strength Indicator,* Indicador de Potencia de Señal Recibida.
SVM	*Support Vector Machine,* Maquina de Vectores de Soporte.
TAM	*Tampere.*
TDOA	*Time Differences Of Arrival,* Diferencia de Tiempo de Llegada.
TOA	*Time Of Arrival*, Tiempo de Llegada.
UJI	*UJIIndoorLoc.*
UTM	*Universal Transverse Mercator*, Universal Transversal de Mercator.
WAP	*Wireless Access Point*, Puntos de Acceso Inalámbrico.
WIFI	*Wireless Fidelity,* Fidelidad Inalámbrica.
XP	*Extreme Programming,* Programación Extrema.

NOTACIÓN MATEMÁTICA

Los vectores son considerados estructuras de datos, que contienen diferentes medidas de un determinado parámetro y se representan en mayúsculas con una flecha encima, con la fuente Cambria Math y en negrilla, i.e., $\vec{\boldsymbol{X}}$.

Los escalares que se presenten en unidades logarítmicas se representan con letras mayúsculas y con la fuente Cambria Math, i.e., X.

Los escalares que se presenten en unidades lineales se representan en letras minúsculas con la fuente Times New Roman, i.e., x.

Las variables aleatorias se representan en letras mayúsculas con la fuente Times New Roman, i.e., X.

Las realizaciones de las variables aleatorias se representan en minúsculas con la fuente Calibri, i.e., x.

El operador de probabilidad se representa con la letra P y paréntesis usando la fuente Cambria Math, i.e., $P(.)$.

Se utiliza el punto como separador decimal.

INTRODUCCIÓN

El ser humano desde sus orígenes ha tenido la necesidad de explorar nuevos territorios y expandir su hábitat, por lo cual conocer la posición de una persona u objeto ha sido una necesidad fundamental. A través del tiempo el concepto del posicionamiento ha tomado fuerza en diferentes campos como en la navegación marítima, la navegación aérea y la navegación terrestre, llevando a la humanidad a desarrollar diferentes sistemas que permitan obtener una estimación en tiempo real de la posición geográfica de personas u objetos. Los primeros sistemas de posicionamiento utilizaban cuerpos celestes para determinar una ubicación, sin embargo, con el desarrollo tecnológico a través de los años han surgido nuevos sistemas de posicionamiento basados en señales de radiofrecuencia, siendo uno de los más conocidos el de Navegación de Largo Alcance (LORAN, *Long Range Navigation*). Las principales desventajas con este tipo de sistemas es que no cubren toda la superficie terrestre y dependen de las condiciones atmosféricas. Para solucionar este inconveniente se desarrollaron sistemas de posicionamiento global tales como: el Sistema de Posicionamiento Global (GPS, *Global Positioning System*) propiedad de los Estados Unidos; el Sistema Global de Navegación por Satélite (GLONASS, *Global Navigation Satellite System*) controlado por la Federación Rusa; y el sistema GALILEO desarrollado por la Unión Europea [1].

Los sistemas de posicionamiento global usan satélites artificiales distribuidos alrededor de la tierra para estimar una posición, sistemas necesitan Línea de Vista (LOS, *Line of Sight*) entre el dispositivo en tierra y el satélite, es decir, cuando el dispositivo está dentro de un edificio, casa, centro comercial o cualquier escenario interior se dificulta obtener su posición. En estos casos se emplean los Sistemas de Posicionamiento en Interiores (IPS, *Indoor Positioning System*), en los cuales se deben tener en cuenta las condiciones del escenario, tales como: obstáculos, falta de visión directa, interferencias, flujo de personas y atenuaciones de la señal, lo cual afecta el comportamiento de la señal de radiofrecuencia, por lo tanto, los IPS deben estar en constante actualización, buscando implementar nuevas tecnologías, herramientas y conceptos que mitiguen el efecto de las condiciones del escenario en la señal de radiofrecuencia.

En el trabajo de grado se propone utilizar el concepto de Aprendizaje Automático (ML, *Machine Learning*), con el fin de reducir el margen de error y el tiempo de procesamiento al obtener la posición de un dispositivo, enfocándose en los diferentes algoritmos de ML usados en el campo del posicionamiento en interiores tales como K-Vecinos Más Cercanos (KNN, *K-Nearest Neighbor*), Árbol de decisión, *Naive Bayes*, Máquina de Vectores de Soporte (SVM, *Support Vector Machine*) y Redes Neuronales (NN, *Neural Networks*), además, se analiza el desempeño utilizando las métricas de precisión y exactitud [1].

El trabajo de grado se divide en cuatro capítulos: el Capítulo 1 contiene el marco teórico con los conceptos necesarios para para la ejecución del trabajo de grado; el Capítulo 2 plantea la adaptación de los algoritmos de ML seleccionados al posicionamiento en interiores; el Capítulo 3 presenta la implementación, entrenamiento y validación de los algoritmos en diferentes escenarios; y en el Capítulo 4 se presentan las conclusiones obtenidas en el desarrollo de los algoritmos de ML para posicionamiento en interiores y los trabajos futuros relacionados con el trabajo de grado.

1. CONCEPTOS GENERALES

1.1. POSICIONAMIENTO EN INTERIORES

Los sistemas de posicionamiento vía satélite como GPS, GLONAS y GALILEO han mostrado gran avance al determinar la ubicación de un objeto cuando existe LOS entre el dispositivo en tierra y el satélite, con resultados aceptables al obtener coordenadas en términos de latitud y longitud, ofreciendo una gran variedad de servicios y aplicaciones. Por el contrario, los IPS presentan diferentes inconvenientes al momento de obtener la posición de un dispositivo, debido a que elementos como paredes, muebles, ventanas, incluso el flujo de personas generan perturbaciones y cambios en el comportamiento de la señal, lo cual se ve reflejado como un error cuando se desea obtener la posición, por todo esto se considera que el posicionamiento en interiores es un campo activo de investigación [2].

Los IPS han ganado popularidad en la actualidad, su uso se ha vuelto indispensable en diferentes entornos como hospitales, fábricas, empresas de seguridad, edificios gubernamentales, museos, aeropuertos entre otros, en los cuales es de vital importancia conocer la posición de una persona, dispositivo o herramienta en tiempo real, para facilitar la interacción con el entorno y en caso de emergencias disminuir el tiempo de reacción. Los IPS existentes se basan en señales de radio frecuencia usando tecnologías inalámbricas como WIFI (*Wireless Fidelity*), Bluetooth y Zigbee [3], debido a su fácil acceso y bajo costo de implementación, estas tecnologías permiten a los IPS adaptarse a diferentes redes locales; la complejidad de estos sistemas se debe las perturbaciones que sufre la señal debido a los obstáculos en un entorno interior, esto obliga a los IPS a estar en constante actualización, buscando implementar nuevas herramientas, tecnologías y conceptos que le permitan adaptarse a diferentes escenarios [4].

1.2. PARÁMETROS DE POSICIONAMIENTO PARA INTERIORES

Los IPS se encargan de estimar la posición de una persona o dispositivo. La información suministrada por la señal inalámbrica debe ser clasificada por algún tipo de medida, por lo tanto, es necesario seleccionar el parámetro que más se adapte a las condiciones del entorno cerrado, como la red inalámbrica y las tecnologías asociadas, por ejemplo, WIFI, Bluetooth o Zigbee. Existen diferentes parámetros de IPS, basados en señales de radiofrecuencia, entre ellos los que se basan en dispositivos fijos o nodos ancla, de los cuales se conoce su posición, por ejemplo, en el caso de las redes WIFI se utilizan Puntos de Acceso Inalámbrico (WAP,

Wireless Access Point) [5], estos hacen parte de la infraestructura interna del edificio siendo elementos de la vida cotidiana. Los IPS utilizan diferentes medidas [6], tales como:

1.2.1. Ángulo de llegada

El Ángulo de Llegada (AOA, *Angle Of Arrival*) permite estimar la ubicación de un dispositivo, por medio del ángulo de llegada de la señal propagada por los WAP, su principal ventaja es su fácil implementación, en el caso de dos dimensiones se necesitan como mínimo dos WAP capaces de emitir la señal al entorno y un dispositivo móvil encargado de recibir información; para obtener una buena estimación los nodos fijos deben estar en LOS con el dispositivo móvil, esta condición es muy difícil de conseguir en entornos cerrados. La Figura 1-1 presenta un sistema basado en AOA, el cual costa de tres WAP y un dispositivo móvil, mostrando dos posiciones estimadas y su error de posicionamiento.

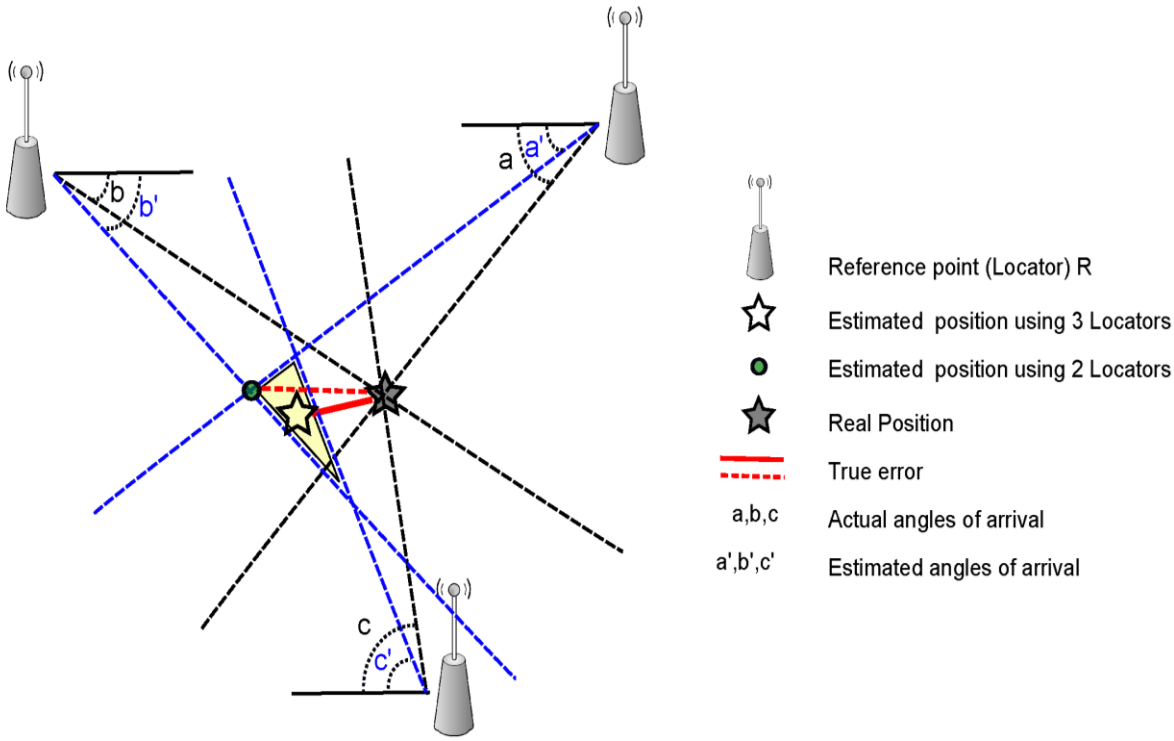

Figura 1-1: Sistema basado en AOA.
Tomada de [7]

La estimación de los ángulos se obtiene con antenas directivas, por lo general, se necesita de hardware adicional, lo cual representa un costo elevado, además, AOA es muy susceptible a obstáculos generando una mala estimación del ángulo, afectando su correcto desempeño [7].

1.2.2. Tiempo de llegada

En el Tiempo de Llegada (TOA, *Time Of Arrival*) la distancia entre los terminales fijos y el dispositivo móvil se calcula a partir del tiempo que tardan en llegar las señales, esto requiere una buena sincronización entre los dispositivos. Para obtener la ubicación se debe tener al menos tres terminales fijos encargados de medir los tiempos, entonces la ubicación del dispositivo móvil se determina midiendo el tiempo de propagación de la señal de ida y la distancia entre el terminal fijo y el dispositivo móvil.

TOA es un parámetro bastante complejo, por esta razón posee algunos inconvenientes, el primero es la necesidad de una sincronización precisa entre todos los nodos tanto los terminales fijos como el dispositivo móvil y el segundo es que la señal transmitida necesita de LOS para determinar la posición con precisión [8].

1.2.3. Diferencia en tiempo de llegada

La Diferencia de Tiempo de Llegada (TDOA, *Time Differences Of Arriva*l) consiste en calcular la ubicación de un dispositivo móvil midiendo la diferencia de tiempo de llegada de la señal. Este parámetro no utiliza los tiempos absolutos de las señales propagadas, por lo tanto, no es necesaria la sincronización de los terminales fijos y el dispositivo móvil. Para determinar la posición, el dispositivo móvil se encarga de emitir la señal hacia los terminales fijos y estos registran el tiempo que se tardó la señal en llegar [9].

El dispositivo móvil es el encargado de propagar la señal, generalmente se necesitan dos señales, una señal de radiofrecuencia y otra de ultrasonido, debido a que el tiempo de propagación de las señales en el medio son diferentes, los terminales fijos pueden determinar la distancia calculando la diferencia de tiempos entre las dos señales. En el caso de tener más de dos dispositivos móviles deben estar sincronizados para emitir las señales, lo cual conlleva a la misma desventaja del parámetro TOA [10].

1.2.4. Indicador de potencia de señal recibida

El Indicador de Potencia de Señal Recibida (RSSI, *Received Signal Strength Indicator*) se basa en la medida del nivel de potencia de señal recibida, para obtener una aproximación de la distancia entre los WAP y el dispositivo móvil. El valor de RSSI depende de la atenuación sufrida por la señal al propagarse en el medio, además, puede sufrir múltiples interferencias al interactuar con objetos o personas, esto afecta la relación entre la potencia y la distancia.

La medición de la potencia de la señal de radiofrecuencia se realiza mediante el indicador RSSI, este representa la intensidad de campo recibida por un dispositivo inalámbrico [11]. La Figura 1-2 muestra el nivel de señal para los dispositivos móviles A y B, en la cual el valor de RSSI es diferente debido a sus distancias con el WAP.

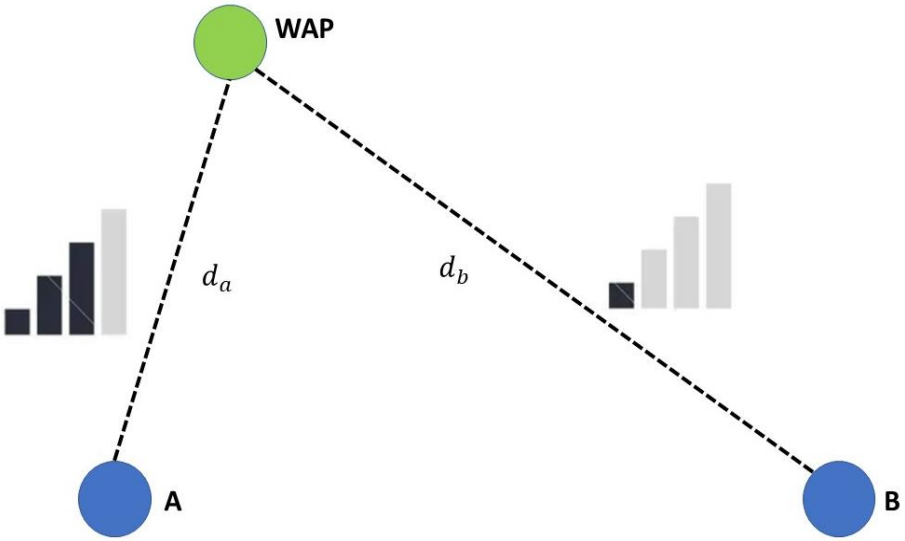

Figura 1-2: Nivel de señal para RSSI.
Por los autores.

RSSI se basa principalmente en dos enfoques: el enfoque de pérdida de ruta consiste en la relación entre la potencia y la distancia, para determinar la posición de un dispositivo móvil; las señales emitidas al medio son bastantes susceptibles al ruido, al multitrayecto ocasionado por los diferentes obstáculos del entorno, los resultados de RSSI pueden ser inexactos, lo cual puede causar errores en el sistema de posicionamiento. Por otro lado, el enfoque *fingerprinting* basado en huellas digitales de RSSI, construye una Base de Datos (BD) de acuerdo a un número determinado de puntos de acceso y dispositivos móviles midiendo los valores de RSSI en una área de interés determinada, estos registros son recopilados en una BD en la fase fuera de línea [12].

1.2.5. Estado de información del canal

La Información del Estado del Canal (CSI, *Channel State Information*) es un parámetro que ha tomado fuerza en los últimos años, debido a su capacidad de describir las características de la señal WIFI, usando el sistema de Multiplexación por División de Frecuencia Ortogonal (OFDM, *Orthogonal Frequency-Division Multiplexing*) y la tecnología inalámbrica de Múltiple Entrada Múltiple Salida (MIMO, *Multiple-Input Multiple-Output*) [13]. En la actualidad se utiliza este parámetro con la

ayuda de una tarjeta de red inalámbrica, como la tarjeta Intel 5300, la cual admite el acceso a la información del Estado del Canal de la capa física [14].

CSI proporciona información de amplitud y fase de un número determinado de subportadoras con la finalidad de describir detalladamente la propagación de la señal en el medio. Para hacer un uso más completo del ancho de banda, se utiliza OFDM el cual permite trasmitir una cantidad determinada de subportadoras ortogonales en paralelo, aportando información de atenuación, dispersión y desvanecimiento que sufre la señal inalámbrica, esta información otorga un mayor número de características que pueden ser usados en un enfoque de *fingerprinting* con el fin de generar BD más robustas que permitan mejorar el desempeño de los IPS [15].

1.3. TÉCNICAS DE POSICIONAMIENTO PARA INTERIORES

Independiente de la tecnología usada para la implementación del IPS, es necesario realizar distintas mediciones de señales de RF para determinar la ubicación de un dispositivo móvil, una vez obtenidas las mediciones se calcula la posición por medio de diversas técnicas. A continuación, se describe las técnicas trilateración, triangulación y *fingerprinting*, encargadas de calcular la posición del dispositivo móvil dentro de ambientes interiores.

1.3.1. Trilateración

La trilateración determina la posición de un dispositivo móvil con varios WAP, midiendo distancias con la ayuda de caculos geométricos, por lo general, se utiliza el parámetro RSSI, porque no necesita de sincronización entre transmisores o entre transmisores y receptores, por lo tanto, la distancia entre el objetivo y el punto de acceso se obtiene mediante la atenuación de la señal, sin embargo, esta técnica puede asociar los parámetros TOA y TDOA [16].

En el caso del plano bidimensional se necesita como mínimo las coordenadas de 3 terminales fijos para lograr un mejor desempeño y obtener una alta precisión. La técnica de trilateración utiliza terminales fijos no colineales para calcular la posición física del dispositivo móvil.

La Figura 1-3 presenta un caso de trilateración circular, usando el parámetro RSSI con tres terminales fijos WAP1, WAP2, WAP3, de los cuales ya se conocen sus respectivas coordenadas, el radio del círculo representa la distancia desde el WAP hasta el dispositivo móvil. La intersección de los radios de los puntos de acceso da como resultado un punto o un área, en el cual se sitúa el dispositivo móvil.

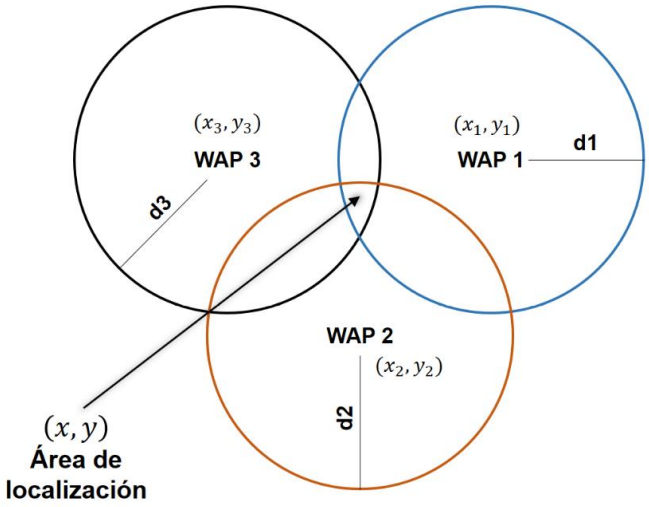

Figura 1-3: Método de trilateración bidimensional. Adaptada de [16]

El cálculo de la distancia d_n entre el dispositivo móvil y el WAP n-ésimo se realiza mediante un modelo de propagación para interiores, en este caso se utiliza el modelo de pérdida de trayecto log-distancia [17]:

$$R_m = -10\rho \log\left(\frac{d_n}{d}\right) + R_o, \quad (1)$$

donde, R_m es el valor de RSSI que llega al receptor en dBm, ρ es el exponente de pérdida de trayectoria, el cual depende del entorno, d_n corresponde a la distancia del dispositivo móvil al WAP n-ésimo dada en metros, d es la distancia de referencia, usualmente esta distancia es de 1 metro y R_o es la medida de RSSI a la distancia de referencia d.

Del modelo de perdida de trayecto de (1) se obtiene la expresión de la distancia d_n, presentada a continuación:

$$d_n = 10^{\left(\frac{R_o - R_m}{10\rho}\right)}, \quad (2)$$

donde, la distancia de referencia d es igual a 1 metro. Teniendo en cuenta las coordenadas de los WAP1, WAP2, WAP3 y las distancias d_1, d_2 y d_3 de la Figura 1-3, se obtiene el siguiente sistema de ecuaciones:

$$(x_1 - x)^2 + (y_1 - y)^2 = d_1^2, \quad (3)$$

$$(x_2 - x)^2 + (y_2 - y)^2 = d_2^2, \quad (4)$$

$$(x_3\text{-}x)^2+(y_3\text{-}y)^2=d_3{}^2. \quad (5)$$

La solución para el sistema de ecuaciones es la coordenada (x,y) del dispositivo móvil. El valor medido de la señal en el punto de interés no es exacto puesto que presenta algunas variaciones generando un margen de error, sin embargo, es una de las técnicas más utilizadas en el posicionamiento en interiores debido a su fácil implementación [18].

1.3.2. Triangulación

Esta técnica calcula la posición de un dispositivo móvil a partir del ángulo de llegada de señales inalámbricas, también conocido como ángulo de llegada AOA. Para determinar el AOA del nodo objetivo se requiere de antenas emisoras direccionales en los diferentes WAP, de los cuales se conocen sus respectivas coordenadas.

La estimación de la posición del dispositivo móvil se calcula en función del ángulo de referencia α, se necesita como mínimo la medición de los ángulos de dos WAP en el plano bidimensional. En la Figura 1-4 se observa la triangulación en dos dimensiones [19].

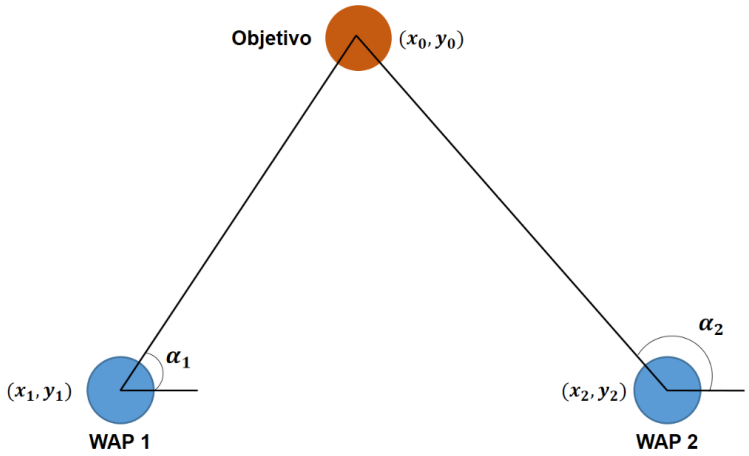

Figura 1-4: Técnica de triangulación.
Por los autores.

Una vez se obtienen los ángulos $\propto_1$ y $\propto_2$ y las coordenadas de los WAP, se establece un sistema de ecuaciones para encontrar la coordenada (x_0, y_0), como se muestra continuación:

$$\tan(\propto_1)=\frac{y_0\text{-}y_1}{x_0\text{-}x_1}, \quad (6)$$

$$\tan(\propto_2) = \frac{y_0 - y_2}{x_0 - x_2}. \quad (7)$$

Este método es bastante sencillo y eficiente ya que no requiere de sincronización para su implementación, sin embargo, necesita de hardware adicional como antenas de tipo direccional. En esta técnica tanto el nodo objetivo como los puntos de acceso deben estar en LOS [20].

1.3.3. Fingerprinting

Fingerprinting es una de las técnicas más utilizadas actualmente debido a que no requiere hardware adicional ni en los puntos de acceso ni en el dispositivo móvil, además, no necesita de sincronización. Esta técnica se basa en dividir el área de interés en diferentes segmentos o puntos, en cada uno se realiza un número determinado de mediciones con el dispositivo móvil y diferentes puntos de acceso, lo cual se conoce como Mapa Radio (RM, *Radio Map*). Posteriormente se construye una BD con la información recopilada de cada RM. El posicionamiento de un dispositivo móvil se determina haciendo coincidir las medidas de RSSI en tiempo real con las medidas del RM previamente guardadas en la BD [21].

La técnica de fingerprinting consta de dos fases: la primera fase, es la de calibración o entrenamiento y la segunda es la de posicionamiento. En la primera fase se construyen los RM y se almacena la información en una BD; en la segunda se determina la posición de un dispositivo móvil usando un algoritmo de posicionamiento.

1.3.3.1. Fase de calibración

También conocida como fase offline, se mide el parámetro de la señal en una posición determinada y se construye un RM para cada posición, estas mediciones son almacenadas en una BD. En la Figura 1-5 se presenta la distribución de RM, con un dispositivo móvil en diferentes posiciones, con el fin de obtener la variación de la intensidad de la señal en distintas coordenadas [22].

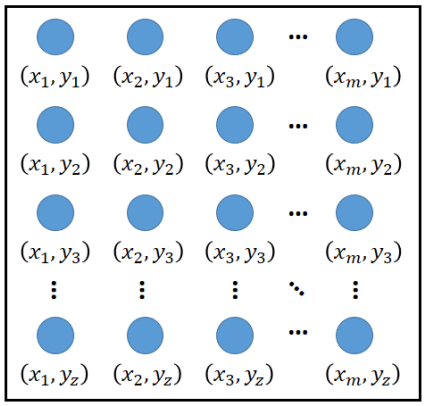

Figura 1-5: Fase de calibración.
Por los autores.

En la Figura 1-5 se utilizan $m \times z$ puntos de referencia en el área de interés, en los cuales se mide la intensidad de la señal proveniente de los n WAP. En cada punto se obtienen un conjunto de medidas de la potencia proveniente de cada WAP, representado de forma vectorial como:

$$\vec{\boldsymbol{R}}_s = (R_{s1}, R_{s2}, \dots, R_{sn}), \qquad (8)$$

donde, $\vec{\boldsymbol{R}}_s$ representa un registro que contiene n medidas RSSI en un punto de referencia; $s \in \mathbb{N}$, representa el número del registro y toma valores desde 0 hasta i; $i \in \mathbb{N}$ y corresponde al número total de registros que se miden en la fase de calibración, el valor s=0 se reserva para el registro del dispositivo móvil; $n \in \mathbb{N}$ y representa el número de WAP con los que se cuenta. Las medidas RSSI R_{sj} están dadas en dBm; $j \in \mathbb{N}$ y toma valores desde 1 hasta n. Este proceso se realiza para todos los puntos de referencia considerados en el escenario, en cada punto de referencia se toman como mínimo 10 registros, cada registro cuenta con n medidas de RSSI y se almacenan en una BD, la cual se usa en la fase de posicionamiento.

1.3.3.2. Fase de posicionamiento

En la fase de posicionamiento o fase online se determina la posición de un dispositivo móvil, el cual se encuentra en la coordenada (x_0, y_0) dentro del área de interés. El dispositivo móvil se encarga de recibir las medidas de RSSI de los n WAP, [23], este proceso se realiza en tiempo real y se representa como:

$$\vec{\boldsymbol{R}}_0 = (R_{01}, R_{02}, \dots, R_{0n}). \qquad (9)$$

En la Figura 1-6 se observa un diagrama de bloques de la fase de posicionamiento de *fingerprinting*. Una vez obtenidas las medidas RSSI en el dispositivo móvil, se

utiliza un algoritmo de posicionamiento para determinar su posición con la ayuda de la información almacenada en la BD.

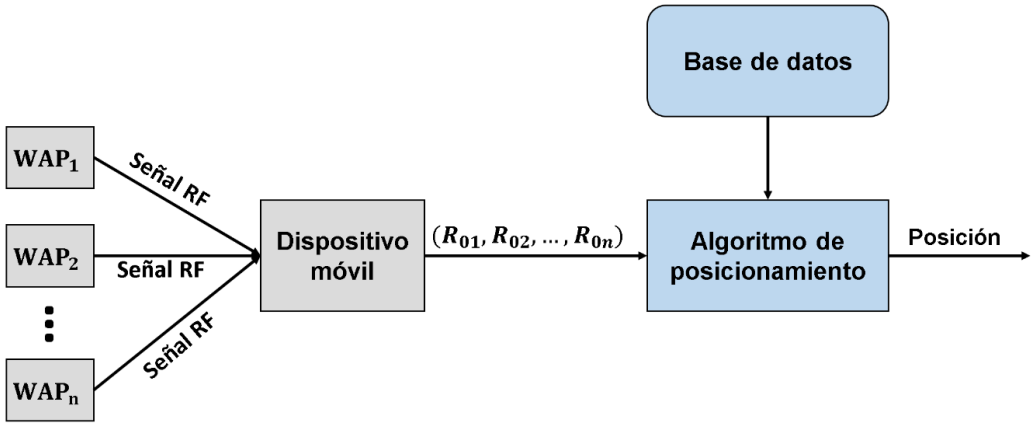

Figura 1-6: Fase de posicionamiento fingerprinting.
Por los autores.

Fingerprinting tiene algunos inconvenientes en la fase de calibración debido a la complejidad de construcción de una BD. Múltiples factores como la sensibilidad al cambio dentro del escenario, el movimiento de objetos o personas generan cambios en la propagación de las señales en el medio, para mitigar los efectos mencionados anteriormente, se realizan mediciones periódicas y se construye nuevos RM para mantener la calidad del IPS.

1.4. INTELIGENCIA ARTIFICIAL

El concepto de máquinas inteligentes se ha convertido en un tema de gran interés en la actualidad, debido a la implementación de máquinas, programas o accesorios capaces de simular comportamientos cognitivos en diferentes entornos; a pesar de parecer algo relativamente nuevo, la creación y diseño de máquinas capaces de replicar acciones inteligentes se viene desarrollando desde hace ya varios años, generando un nuevo campo de investigación asociado a la informática denominado Inteligencia Artificial (AI, *Artificial Intelligence*). Definir el termino AI genera gran controversia entre la comunidad científica ya que depende de la misma definición de inteligencia, la cual al día de hoy tiene múltiples interpretaciones; en 1959 John McCarthy la definió como “la ciencia e ingenio de hacer máquinas inteligentes” [24], esta definición fue muy bien aceptada en su época. Actualmente existen múltiples interpretaciones, siendo más acertada la definición que considera a AI “una subdisciplina del campo de la informática, encargada de la creación de máquinas capaces de imitar comportamientos inteligentes” [25], por lo tanto, se interpreta a AI como la capacidad de las máquinas de usar un conjunto de algoritmos, aprender de

datos y utilizar lo aprendido, en la toma de decisiones, resolviendo un problema en un número finito de pasos.

La capacidad de una máquina para realizar múltiples tareas al tiempo, es un tema bastante estudiado en la actualidad por parte las empresas dedicadas al desarrollo de AI, teniendo en cuenta esto nace una primera clasificación de las AI entre débiles y fuertes. Las AI débiles hacen referencia a las inteligencias capaces de cumplir con un conjunto limitado de tareas, este tipo de AI no se toma como una inteligencia general sino específica, diseñada para ser inteligente en la tarea asignada, por el contrario, las AI fuertes son aplicables a gran variedad de problemas y dominios diferentes, siendo capaz de realizar todas las funciones cognitivas, con un comportamiento similar al de un ser humano; al día de hoy no se cuenta con AI fuertes todas son consideradas como débiles [26].

Dentro del campo de la AI se encuentran diferentes subcategorías correspondientes a diversas situaciones en que ser aplicadas, por ejemplo, el campo de la Robótica encargado de la capacidad de moverse y adaptarse a diferentes entornos, el Procesamiento del Lenguaje Natural (NLP, *Natural Language Processing*) estudia la comprensión del lenguaje, el campo de voz analiza la conversión de voz a texto y de texto a voz, estos son algunos campos de estudios diferentes dentro de AI, no obstante, existe un campo considerado como uno de los más importantes, este es el aprendizaje automático [27].

1.4.1. Machine learning

ML se define como la “rama del campo de la Inteligencia Artificial, que busca dotar a las máquinas de capacidad de aprendizaje”, ML se considera una subdisciplina dentro del campo de la AI, destacándose por ser un componente central en relación con el resto de categorías, esto se representa en la Figura 1-7 [28].

Con las cantidades cada vez mayores de información en formato electrónico, la necesidad de métodos automatizados que permitan analizar datos sigue creciendo, el objetivo de ML es desarrollar métodos capaces de detectar automáticamente patrones en los datos, con el fin de predecir resultados futuros u otros resultados de interés. ML permite a un sistema aprender de un conjunto de datos en lugar de utilizar programación clásica, este no es una tarea sencilla; a medida que el algoritmo recibe registros de entrenamiento se generan modelos de ML de mayor complejidad, después de este proceso, se genera un modelo capaz de predecir un resultado a partir de unos registros de entrada.

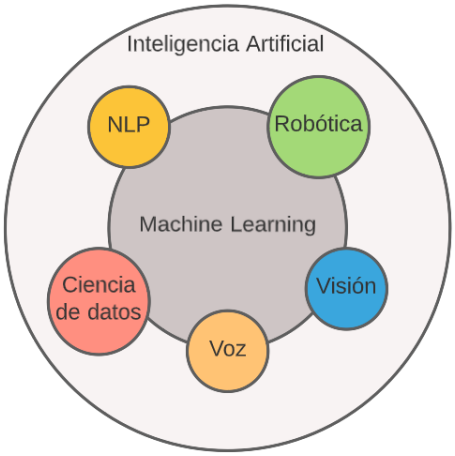

Figura 1-7: Inteligencia artificial y Machine Learning.
Por los autores.

Dependiendo de la naturaleza del problema se pueden usar diferentes enfoques de ML basados en el tipo y cantidad de datos. El aprendizaje de los algoritmos de ML basados en reconocimiento de patrones se clasifica en supervisado y no supervisado.

1.4.1.1. Aprendizaje supervisado

El aprendizaje supervisado se basa en encontrar la relación existente entre unas variables de entrada y salida. Para realizar este proceso es necesario utilizar un conjunto de datos de entrenamiento previamente ordenado en clases con sus respectivas etiquetas, con estos registros generalmente consignados dentro de una BD, se analiza cuales atributos y patrones hacen que ciertos datos pertenezcan a una determinada clase y según el tipo de información se genera un modelo capaz de clasificar nuevos datos en las distintas clases.

Este tipo de aprendizaje es uno de los enfoques de ML con más aplicaciones, debido al buen desempeño de sus modelos predictivos, su desventaja es la necesidad de clasificar previamente todos los registros de entrenamiento, esto parece una tarea simple si se tiene una pequeña cantidad de registros, pero en la actualidad la cantidad de datos que manejan los sistemas es muy grande y el proceso de clasificar cada uno se vuelve una tarea compleja [29].

1.4.1.2. Aprendizaje no supervisado

El aprendizaje no supervisado busca establecer una relación entre distintos registros de entrada sin necesidad de definir los datos de salida, por lo tanto, este tipo de aprendizaje no puede ser aplicado directamente a problemas de regresión y clasificación, para llevar a cabo este proceso, se requiere una gran cantidad de datos sin una clase definida previamente. El aprendizaje no supervisado se encarga

de descubrir patrones desconocidos entre los datos de entrada agrupando los datos con atributos o características similares entre ellos, a esto se lo conoce como métodos de *Clustering* y asociación; las aplicaciones de este tipo de aprendizaje se relacionan con el tema de *Big Data*, y se reconoce como el futuro de ML [29].

1.5. ALGORITMOS DE PREDICCIÓN Y *MACHINE LEARNING*

El objetivo del aprendizaje automático es entrenar un algoritmo determinado a partir de un conjunto de datos previamente establecido, para construir un modelo de clasificación o regresión, capaz de predecir una clase o un valor, al cual pertenece un nuevo registro de entrada, en IPS existen algoritmos relevantes debido a su alto desempeño, estos se caracterizan por ser algoritmos de aprendizaje supervisado [30].

Para entender el funcionamiento de algunos de estos algoritmos en los IPS, es necesario tener en cuenta los registros de la Tabla 1-1, los cuales se interpretan como una BD de entrenamiento, donde la primera columna representa los registros de entrada como vectores, los cuales contienen las medidas RSSI en dBm de los n WAP; $\mathrm{i} \in \mathbb{N}$ y corresponde al número de registros que contiene la base de datos, el valor 0 se reserva para el registro con la información RSSI del dispositivo móvil; Cada $\vec{\boldsymbol{R}}_{\mathrm{i}}$ pertenece a una clase C_{l} donde , $\mathrm{l} \in \mathbb{N}$ toma valores entre 1 y h; h representa el número de puntos de referencia o clases que se tuvieron en cuenta en la creación de la base de datos.

Registro	$RSSI_1$	$\cdots$	$RSSI_{\mathrm{n}}$	Clase
$\vec{\boldsymbol{R}}_1$	R_{11}	$\cdots$	$R_{1\mathrm{n}}$	C_{l}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\vec{\boldsymbol{R}}_{\mathrm{i}}$	$R_{\mathrm{i}1}$	$\cdots$	R_{in}	C_{h}

Tabla 1-1: Ejemplo de base de datos para algoritmos ML. Por los autores.

Los algoritmos de ML pueden inferir la posición del objetivo móvil basado en señales percibidas desde los diferentes puntos de acceso. Los algoritmos más relevantes y los más usados en investigaciones en el campo de posicionamiento en interiores son K-Vecinos Más Cercanos, Árbol de decisión, *Naive Bayes*, Máquina de Vectores de Soporte y Redes Neuronales.

1.5.1. K-vecinos más cercanos

Los algoritmos KNN son conocidos por ser clasificadores basados en instancia, además son algoritmos de aprendizaje supervisado lo cual hace necesario contar

con un conjunto de datos de entrenamiento, con unas clases definidas o etiquetadas. KNN se basa en clasificar un nuevo caso de entrada en la clase más frecuente a la que pertenecen sus k vecinos más cercanos, determina la cercanía por medio de la distancia, la cual en este caso representa un valor de error , el estar fundamentado en una idea tan intuitiva hace a los algoritmos KNN más fáciles de implementar, aumentando su uso a nivel científico en el campo de ML [31].

Para determinar el valor de error entre el registro $\vec{\boldsymbol{R}}_0 = (R_{01}, R_{02}, \cdots, R_{0n})$ y un registro $\vec{\boldsymbol{R}}_s = (R_{s1}, R_{s2}, \cdots, R_{sn})$ del conjunto entrenamiento de la Tabla 1-1, se usa la distancia euclidiana y se expresa de la siguiente forma:

$$d(\vec{\boldsymbol{R}}_0, \vec{\boldsymbol{R}}_s) = \sqrt{(R_{01} - R_{s1})^2 + (R_{02} - R_{s2})^2 + \cdots + (R_{0n} - R_{sn})^2}, \qquad (10)$$

donde, $s \in \mathbb{N}$ y toma valores desde 0 hasta i. La expresión de la distancia euclidiana es una generalización del teorema de Pitágoras, y sirve para determinar la distancia o el valor de error entre dos puntos en un plano n-dimensional, de manera general el cálculo de la distancia euclidiana se puede expresar como:

$$d(\vec{\boldsymbol{R}}_0, \vec{\boldsymbol{R}}_s) = \sqrt{\sum_{j=1}^{n} (R_{0j} - R_{sj})^2}, \qquad (11)$$

donde, $j \in \mathbb{N}$ y toma valores desde 1 hasta n. El cálculo del error, debe realizarse con todos los registros del conjunto de entrenamiento, posteriormente se ordenan los valores y se escoge a los k registros en los cuales el error es menor. El parámetro k debe ser decidido a priori y en función de la decisión tomada se obtiene los resultados.

Después de encontrar los vecinos más cercanos se realiza la predicción, para esto se lleva a cabo un método conocido como criterio de decisión, existen dos formas por votación o distancia media, depende del tipo de problema. En la Figura 1-8 se observa, cómo influye el parámetro k en el resultado del algoritmo usando el método de votación, en el caso de elegir un valor de k=3, el punto verde se clasifica como un triángulo rojo puesto que dos de sus 3 vecinos pertenecen a dicha clase; no obstante, si k=5, el circulo verde se clasifica como cuadrado azul, debido a que tres de sus cinco vecinos más cercanos pertenecen a esa clase, generalmente se recomienda usar un valor de k impar.

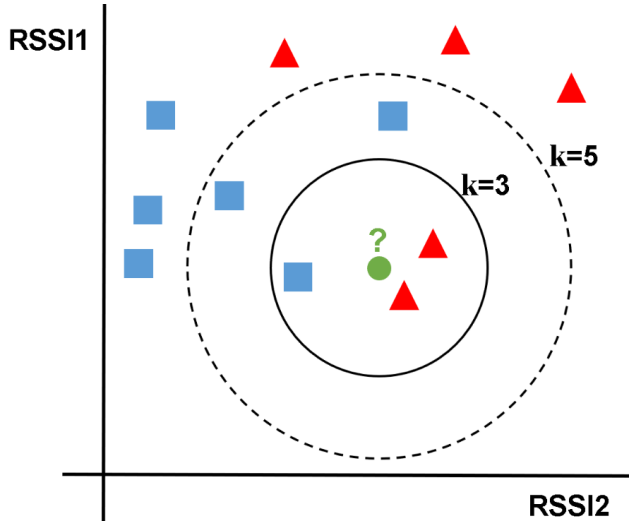

Figura 1-8: Algoritmo K-NN, parámetro k=3 y k=5. Adaptada de [32].

En el área de posicionamiento en interiores se aplica el algoritmo KNN una vez obtenida la BD en la fase de entrenamiento, en el caso de *fingerprinting*; en la BD se almacena cada clase con sus respectivos atributos. El tiempo que tarda en buscar los vecinos más cercanos es proporcional al número de clases que componen la BD. Para el diseño de KNN es necesario definir el número k y el criterio de decisión [33].

- **Parámetro k**

Para escoger el parámetro k no existen reglas estandarizadas, en muchos casos se define un número que tenga relación con la cantidad de clases de predicción o se escoge un número bajo y se aumenta hasta alcanzar un valor de k con resultados aceptables, por medio de un proceso de prueba y error.

- **Criterio de decisión**

El criterio de decisión del algoritmo KNN depende del problema bajo análisis, en casos de clasificación se escoge comúnmente el criterio por votación, y casos de regresión se elige el criterio de distancia media.

1.5.2. Árbol de decisión

Es un algoritmo de uso común en el aprendizaje automático, este aprende en función de registros de los cuales se conoce la clase a la que pertenecen, un árbol de decisión es una estructura jerárquica consta de nodos de decisión, ramas y nodos hoja, estos representan atributos, condiciones y clases respectivamente. Cada nodo de decisión contiene una condición para determinar la rama a seguir,

cuando el algoritmo llega a un nodo hoja, la etiqueta almacenada en la hoja regresa como una clase, lo mencionado anteriormente se representa en la Figura 1-9.

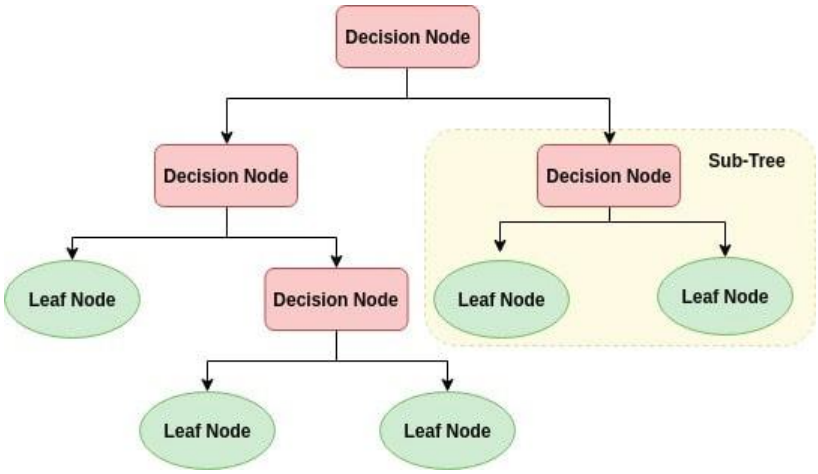

Figura 1-9: Árbol de decisión.
Tomado de [34].

Un árbol de decisión se considera como una estructura semejante a un diagrama de flujo, en el cual un nodo de decisión representa un atributo, la rama una regla de decisión y cada nodo hoja representa el resultado, esta semejanza con un diagrama de flujo ayuda a facilitar su interpretación [35].

El nodo superior en un árbol de decisión se conoce como nodo raíz. El algoritmo aprende a clasificar en función del valor del atributo, dividiendo el árbol mediante un método de partición recursiva, seleccionando el mejor atributo utilizando Medidas de Selección de Atributos (ASM, *Attribute Selection Measures*); este atributo se convertirá en un nodo decisión dividiendo el conjunto de datos en subconjuntos más pequeños, a los cuales se aplica nuevamente las ASM, este proceso se repite hasta llevar los ejemplos a un nodo hoja, encargado de proporcionar la clase a la cual pertenece el nuevo ejemplo [36].

La medida de selección de atributos es una heurística que ayuda a seleccionar el criterio para realizar la división de los datos, proporcionando una puntuación para cada atributo del conjunto de datos, al final el atributo con mayor puntuación se selecciona como el encargado de dividir los datos. La ASM más popular es la Ganancia de información, esta es una propiedad estadística, su función es medir qué tan bien un atributo separa los ejemplos de entrenamiento de acuerdo con su clasificación; en la Figura 1-10 (a) se muestra como un atributo con baja ganancia de información divide los datos de manera uniforme, lo cual no ayuda a su clasificación, mientras que en la Figura 1-10 (b) un atributo con alta ganancia de información divide los datos de forma desigual, esto permite diferenciarlos entre sí.

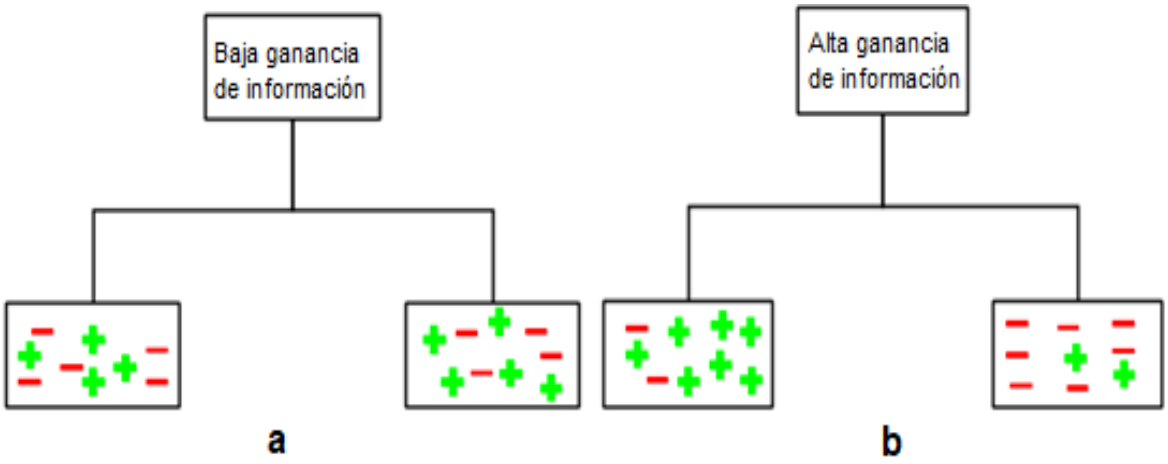

Figura 1-10: Tipos de ganancia de información.
Adaptada de [37].

El algoritmo Dicotomizador Iterativo 3 (ID3, *Iterative Dichotomiser 3*) de Quinlan y su sucesor el algoritmo C4.5, son los más populares entre los algoritmos de árbol de decisión, se usan en clasificación por esta razón, se denominan clasificadores estadísticos; estos algoritmos generan un árbol de decisión mediante la estrategia de recorrido en profundidad, encargada de recorrer todo el árbol de decisión de forma ordenada pasando por todos los nodos de decisión, explorando las rutas sin comenzar una nueva hasta completar todo el recorrido, este proceso se realiza hasta llegar a todos los nodos hoja.

El árbol de decisión J48 en WEKA[1] usa entropía e implementa el algoritmo C4.5 de Quinlan, para generar un modelo de predicción. Primero crea un árbol de decisión para clasificar un nuevo elemento utilizando los atributos de los datos de entrenamiento, luego elige un atributo encargado de discriminar las diversas instancias, hasta encontrar el atributo capaz de proporcionar la mayor ganancia de información. Continúa el proceso hasta encontrar instancias de subconjuntos que pertenecen a la misma clase, creando el nodo hoja y se detiene cuando utiliza todos los atributos [39].

1.5.3. Naive Bayes

Los modelos *Naive Bayes* son una clase especial de algoritmos de clasificación de ML, estos se basan en el Teorema de *Bayes*, también conocido como teorema de probabilidad condicionada. Su expresión es la siguiente:

[1] WEKA es un software de código abierto, que contiene una colección de algoritmos de ML, con herramientas para tareas de clasificación, agrupación, asociación, regresión y visualización [38].

$$P(\mathrm{C_l}|\vec{\mathbf{R}}_\mathrm{s}) = \frac{P(\mathrm{C_l})P(\vec{\mathbf{R}}_\mathrm{s}|\mathrm{C_l})}{P(\vec{\mathbf{R}}_\mathrm{s})}, \quad (12)$$

donde, $\mathrm{C_l}$ es la variable aleatoria que representa una determinada clase y $\vec{\mathbf{R}}_\mathrm{s}$ representa un vector de variables aleatorias que denota los valores RSSI provenientes de los n WAP; $\mathrm{s} \in \mathbb{N}$ y toma valores desde 0 hasta el número i de registros que contiene la base de datos. Este clasificador asume a las medidas de RSSI de los n WAP como independientes entre sí, es decir, la presencia de una cierta característica en un conjunto de datos no está en absoluto relacionada con la presencia de cualquier otra característica [40].

Para entender el funcionamiento de los algoritmos *Naive Bayes* se necesita de (12), esta al ser aplicada a un registro de entrada $\vec{\mathbf{R}}_0 = (\mathrm{R_{01}},\mathrm{R_{02}},\cdots,\mathrm{R_{0n}})$ da como resultado:

$$P(\mathrm{C_l}|\vec{\mathbf{R}}_0) = \frac{P(\mathrm{C_l})P(\mathrm{R_{01}},\mathrm{R_{02}},\cdots,\mathrm{R_{0n}}|\mathrm{C_l})}{P(\mathrm{R_{01}},\mathrm{R_{02}},\cdots,\mathrm{R_{0n}})}, \quad (13)$$

donde, el numerador representa un modelo de probabilidad conjunta y el denominador por su parte es un valor que solo depende de los valores $(R_{01}, R_{02}, \cdots, R_{0\mathrm{n}})$, es decir, no afecta de forma directa al resultado por convertirse en una constante. Al numerador de (13) se le aplica la regla de la cadena para la definición de probabilidad condicional obteniendo:

$$P(\mathrm{C_l}, \mathrm{R_{01}},\cdots,\mathrm{R_{0n}}) = P(\mathrm{C_l})P(\mathrm{R_{01}},\mathrm{R_{02}},\cdots,\mathrm{R_{0n}}|\mathrm{C_l}). \quad (14)$$

Si se continúa aplicando la regla de la cadena se tiene que:

$$P(\mathrm{C_l})P(\mathrm{R_{01}},\cdots,\mathrm{R_{0n}}|\mathrm{C_l}) = P(\mathrm{C_l})P(\mathrm{R_{01}}|\mathrm{C_l})P(\mathrm{R_{02}}|\mathrm{C_l}, \mathrm{R_{01}})P(\mathrm{R_{03}},\cdots,\mathrm{R_{0n}}|\mathrm{C_l}, \mathrm{R_{01}}, \mathrm{R_{02}}). \quad (15)$$

Y así sucesivamente, hasta el valor $\mathrm{R_{0n}}$; luego se usa el supuesto de independencia entre las n medidas RSSI de un registro de entrada, es decir, el valor $\mathrm{R_{01}}$ no depende de cualquier otro valor $\mathrm{R_{0n}}$, teniendo en cuenta esta independencia se obtiene la expresión:

$$P(\mathrm{C_l})P(\mathrm{R_{01}},\mathrm{R_{02}},\cdots,\mathrm{R_{0n}}|\mathrm{C_l}) = P(\mathrm{C_l})P(\mathrm{R_{01}}|\mathrm{C_l})P(\mathrm{R_{02}}|\mathrm{C_l}) \cdots P(\mathrm{R_{0n}}|\mathrm{C_l}), \quad (16)$$

la cual se remplaza en (13), obteniendo la expresión:

$$P(\mathrm{C_l}|\vec{\mathbf{R}}_0) = \frac{P(\mathrm{C_l})P(\mathrm{R_{01}}|\mathrm{C_l})P(\mathrm{R_{02}}|\mathrm{C_l})\cdots P(\mathrm{R_{0n}}|\mathrm{C_l})}{P(\mathrm{R_{01}},\mathrm{R_{02}},\cdots,\mathrm{R_{0n}})}, \quad (17)$$

donde, el numerador presenta la multiplicación de la probabilidad de la clase $\mathrm{C_h}$ por la probabilidad de cada $R_{0\mathrm{n}}$ dado $\mathrm{C_l}$; la multiplicación de la probabilidad condicional se puede expresar mediante una multiplicadora, además, el denominador no afecta directamente el resultado por convertirse en una constante, dando como resultado:

$$P(\mathrm{C_l}|\vec{\mathbf{R}}_0) = P(\mathrm{C_l}) \prod_{j=1}^{\mathrm{n}} P(\mathrm{R_{0j}}|\mathrm{C_l}), \quad (18)$$

donde, $j \in \mathbb{N}$ y toma valores desde 1 hasta n. Para construir el algoritmo *Naive Bayes* se usa el modelo probabilístico de (18) con una regla de decisión conocida como máximo a posteriori, en la cual se clasifica un caso de entrada $\vec{\mathbf{R}}_{0\mathrm{n}}$, en la clase h que tenga mayor probabilidad. Su expresión se presenta a continuación:

$$Solución = \arg \max_{\mathrm{l}=1}^{\mathrm{h}} P(\mathrm{C_l}) \prod_{j=1}^{\mathrm{n}} P(\mathrm{R_{0j}}|\mathrm{C_l}). \quad (19)$$

donde, $\mathrm{l} \in \mathbb{N}$ y toma valores desde 1 hasta h. El algoritmo Naive Bayes presenta precisión y velocidad media cuando se aplica a grandes bases de datos, construye modelos de clasificación más complicados, por lo tanto, es ampliamente utilizado en tareas de clasificación [41].

1.5.4. *Support vector machine*

SVM es un algoritmo de aprendizaje supervisado utilizado para clasificación y regresión, ofrece una alta exactitud en comparación con otros algoritmos clasificadores como los árboles de decisión y KNN. SVM necesita de separadores lineales encargados de dividir los datos de entrada en subconjuntos o clases generando un espacio que sea capaz de minimizar errores [42].

Para establecer un hiperplano capaz de dividir el conjunto de datos, es necesario conocer los puntos de cada clase más cercanos entre sí, a estos se les denomina vectores de soporte, una vez definidos estos puntos se traza el hiperplano, si el margen existente entre las 4 clases es máximo, la clasificación del algoritmo es mucho más acertada.

En la Figura 1-11 se expone la división de dos clases en color azul y morado, en este caso las clases son linealmente separables, por lo tanto, el hiperplano se traza

como una línea continua. Los dos puntos azules y el punto morado son los vectores de soporte.

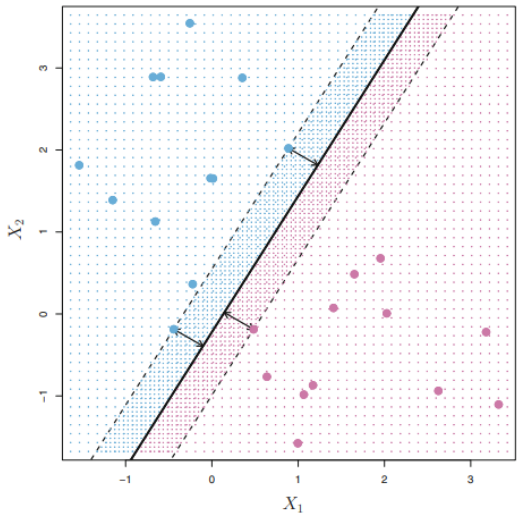

Figura 1-11: Trazo hiperplano para el algoritmo SVM.
Tomada de [43].

En algunos casos los registros de entrada no son linealmente separables en el espacio original, por lo tanto, es necesario mapear los datos de entrada a un espacio de mayor dimensión, debido a las limitaciones computacionales actuales esto no es posible, para solucionar este problema se hace uso de la función de Kernel, la cual se encarga de transformar los registros de entrada en la forma requerida [44]. En la Figura 1-12 se muestra el mapeo de datos hacia un nuevo espacio, en el cual es posible realizar la clasificación.

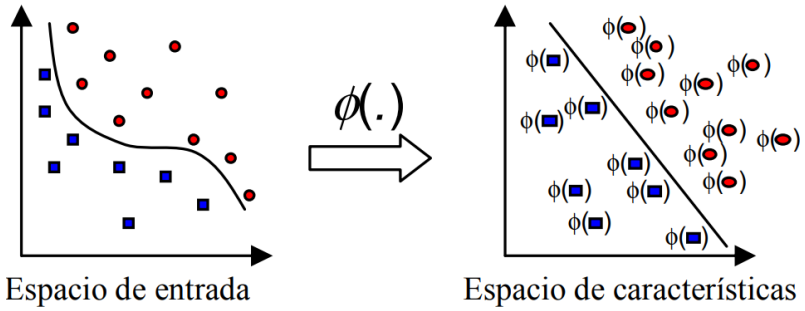

Figura 1-12: Transformación de entrada con Kernel.
Adaptada de [43].

En la Figura 1-12 se muestra un mapeo de un espacio de entrada de dos dimensiones a un espacio de características de dos dimensiones, por medio de la función ∅(.), obteniendo la siguiente expresión:

$$\emptyset\left(\vec{\boldsymbol{R}}_{\mathrm{s}}\right) = \left(\emptyset(R_{\mathrm{s1}}) + \emptyset(R_{\mathrm{s2}}) + \cdots + \emptyset(R_{\mathrm{sn}})\right). \quad (20)$$

La forma más común en que las máquinas de aprendizaje lineal pueden aprender una función objetivo es cambiando su representación, esto quiere decir, mapear los datos de entrada a un nuevo espacio de características de mayor dimensión.

1.5.5. Redes neuronales

Las redes neuronales artificiales son modelos computacionales, se encargan de imitar la estructura y funcionamiento del cerebro humano en particular del sistema nervioso, el cual está compuesto por redes de neuronas biológicas que individualmente tienen poca capacidad de procesamiento, sin embargo, toda su capacidad cognitiva se sustenta en la conectividad de estas.

Las redes neuronales se basan en el conocimiento obtenido a través de la experiencia, así, es posible ejecutar algunas tareas mediante el entrenamiento previo. El entrenamiento realizado permite a las redes neuronales crear su propia representación del problema internamente y posteriormente pueden responder antes nuevos datos de entrada. Aunque las neuronas tengan poca capacidad de procesamiento, al estar conectadas en paralelo aumentan significativamente su desempeño [45] [46]. El funcionamiento de NN se basa en la interacción y trabajo en conjunto de múltiples neuronas, una aproximación de su estructura se evidencia en la Figura 1-13.

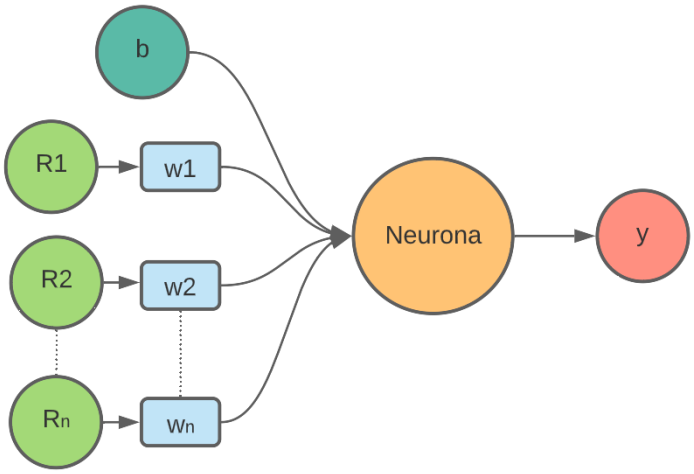

Figura 1-13: Estructura de una neurona artificial.
Por los autores.

En la Figura 1-13 se presenta el procesamiento de una neurona artificial, la información o medidas RSSI de entrada están definidas de R_1 hasta R_n, donde n es el número de WAP. A cada entrada se asigna un peso representado por $w_j \epsilon \mathbb{Q}$, además, se cuenta con el termino independiente b denominado sesgo o bias el cual $\epsilon \mathbb{Q}$. El valor de los pesos y del sesgo se obtienen en la fase de entrenamiento. El

comportamiento interno de una neurona se representa como un modelo de regresión lineal:

$$y = f\left(b + \sum_{j=1}^{n} w_j R_j\right), \quad (21)$$

donde, f es una función de activación encargada de agregar deformaciones no lineales al hiperplano definido por los atributos de entrada R_j, este hiperplano puede cambiar su inclinación y elevación de acuerdo a los parámetros w_j y b; $j \in \mathbb{N}$ y toma valores entre 0 y n.

Después de conocer el funcionamiento de una neurona, es posible comprender como se forma una red neuronal, en la Figura 1-14 se representa la estructura de una Red Neuronal Secuencial Densa (DSNN, *Dense Secuential Neural Network*) utilizada ampliamente como base en la construcción de algoritmos de NN. Las neuronas se agrupan en columnas llamadas capas, las cuales se ordenan de forma secuencial, donde las neuronas que se encuentran en la capa A reciben la misma información de la capa A-1 y sus resultados pasan a las neuronas de la capa A+1; la primera capa es la capa de entrada y su función es recibir los registros, la última capa se conoce como capa de salida encargada de entregar el resultado de predicción, a las capas intermedias se les denomina capas ocultas.

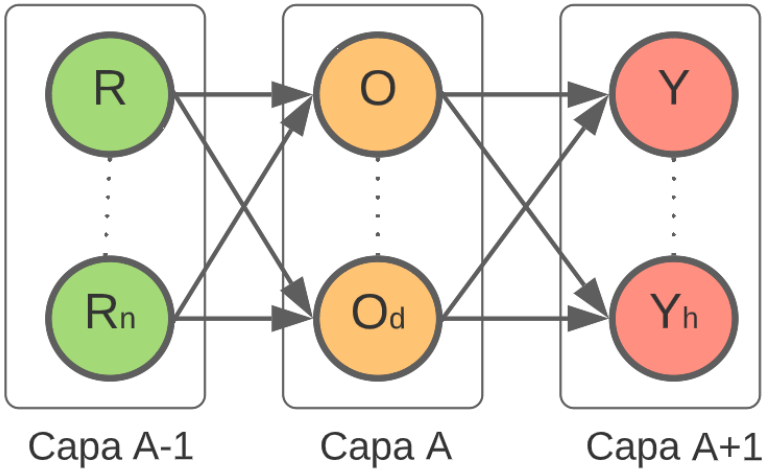

Figura 1-14: Red Neuronal Secuencial Densa.
Por los autores.

El proceso de ordenar las neuronas en capas de manera secuencial ayuda a que la red aprenda de forma jerarquizada, las neuronas de las primeras capas aprenden conceptos básicos, y la información resultante pasa a las siguientes capas donde

las neuronas aprenden conceptos más elaborados y complejos, es decir entre más capas se agreguen más complejo puede ser el conocimiento aprendido por la red.

La función de activación f, se aplica a la estructura de cada neurona agregando deformaciones no lineales, esto permite encadenar de forma efectiva la computación de varias neuronas para modelar información más compleja; sin el uso de las funciones de activación toda la estructura de una red colapsa hasta ser equivalente a una sola neurona. La función de activación se define como un hiperparámetro y se puede elegir diferentes funciones según el tipo de entorno.

Los parámetros internos w_j y b, se ajustan automáticamente en la etapa de entrenamiento, con el fin de crear un modelo capaz de solucionar un problema de clasificación o regresión. En la etapa de entrenamiento es necesario definir una función de costo, con el fin de conocer como varia el error del modelo cuando se presenta un cambio en los parámetros w_j y b, posteriormente se realiza el cálculo de las derivadas parciales de error con respecto a cada uno de los parámetros, este resultado es el vector gradiente, el cual indica hacia qué dirección se debe modificar el valor de w_j y b para minimizar el error, además, se define la cantidad en la que varían los parámetros internos mediante la Tasa de Aprendizaje (LR*, Learning Rate*).

El cálculo del vector gradiente es una tarea de gran dificultad, debido a esto se utiliza el método de *backpropagation*, con el cual se calcula el vector gradiente desde la capa de salida hasta la capa de entrada, realizando una retropropagación de errores, ya que el error de una neurona depende de las neuronas previas a las que está conectada. Al obtener el vector gradiente es posible modificar los parámetros internos de cada una de las neuronas, usando un nuevo hiperparámetro conocido como optimizador, este usa la información del vector gradiente y el LR para definir la variación de los parámetros [47] [48].

- **Redes neuronales convolucionales**

Un concepto dentro de las NN de bastante interés en la actualidad, es el uso de capas convoluciones para mejorar el desempeño de las DSNN, este enfoque se define como Redes Neuronales Convolucionales (*CNN, Convolutional Neural Network*). CNN se usa en el análisis de imágenes, consiste en múltiples capas de filtros convolucionales, con una función de mapeo no lineal al final de cada capa y unas capas de neuronas simples en la salida de la red, esto se muestra en la Figura 1-15. Las capas de convolución se encargan de extraer características de las

imágenes y las capas de neuronas analizan estas características para obtener una predicción.

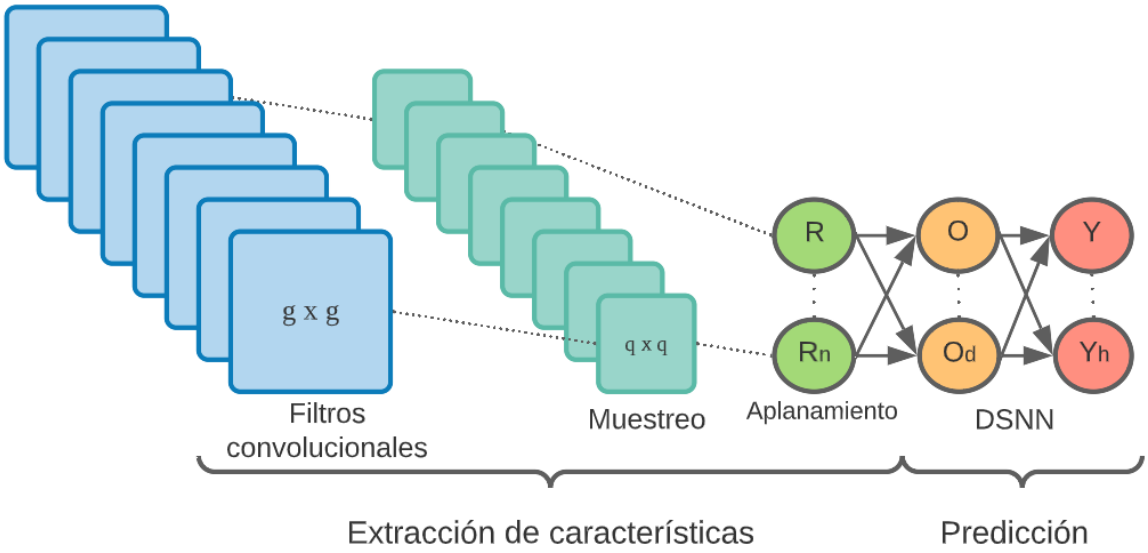

Figura 1-15: Redes neuronales convolucionales.
Por los autores.

Las capas convolucionales de CNN se encargan de tomar un grupo de pixeles cercanos de una imagen y realizar el producto escalar con una matriz de tamaño $g \times g$ denominada filtro, esta operación se realiza de izquierda a derecha y de arriba hacia abajo desplazándose por toda la imagen generando una nueva matriz, la cual se convierte en la entrada de la siguiente capa. Cada capa tiene un número de filtros, estos representan las matrices obtenidas en la salida, a estas se les aplica una función de activación para resaltar las características de la imagen y un muestreo para reducir el tiempo de procesamiento.

Al final de la extracción de características, se realiza el proceso de aplanamiento, permitiendo que los datos de las imágenes pasen a ser unidimensionales y se usen como registros de entrada en las DSNN, encargadas de realizar la predicción. El entrenamiento de las CNN es similar al algoritmo de DSNN, con la diferencia de que los parámetros internos a ajustar en las capas de convolución son los valores de las matrices de filtros y muestreo [49] [50].

Para DSNN y CNN es necesario definir unos hiperparámetros según, el tipo de problema y la naturaleza de los datos.

- **Arquitectura de red**

La arquitectura de red en DSNN define el número de capas ocultas y su número de neuronas, en CNN se establece el número de capas convolucionales, filtros y la red

neuronal al final de modelo, para encontrar estos valores no se ha planteado un procedimiento estándar.

En DSNN no existe razón para usar múltiples capas ocultas, ya que la mayoría de problemas prácticos se resuelven con una sola, tener muchas capas ocultas implica un entrenamiento lento y un vector gradiente inestable. En el caso de implementar más de una capa oculta, existen algunas reglas empíricas para establecer un punto de partida en la construcción de la topología, entre ellas la más utilizada es la regla de la pirámide geométrica [51], esta regla calcula el número de neuronas basada en la suposición de que la cantidad de neuronas en la capa anterior es superior a la cantidad de neuronas de la capa siguiente y se expresa como:

$$e=\sqrt{h\,n}. \qquad (22)$$

Donde, e representa el número de neuronas de la capa y su valor se aproxima al entero mayor; n es el número de WAP y h es el número de clases de la BD, de igual forma esta regla permite agregar una segunda capa oculta, con el uso de:

$$\begin{aligned} q &= \sqrt[3]{n/h}, \\ e_1 &= n\,q^2, \\ e_2 &= n\,q, \end{aligned} \qquad (23)$$

donde, q corresponde a la relación entre el número de entradas y salidas de la red neuronal; e_1 es el número de neuronas en la primera capa oculta y e_2 el número de neuronas de la segunda capa oculta. Al definir el número de neuronas de cada capa oculta los resultados de e_1 y e_2 se aproximan al entero mayor.

- **Funciones de activación**

La función de activación se define para cada capa, es decir las neuronas de una capa tienen la misma función de activación. A continuación, se describe las funciones de activación más empleadas en el desarrollo de algoritmos de redes neuronales [52].

Función de activación lineal ReLU

Es una de las más empleadas debido a sus buenos resultados y su bajo costo computacional para algoritmos de aprendizaje profundo; esta función activa la neurona si el valor de entrada está por encima de 0 y aumenta de forma lineal con el valor de entrada, en (23) se muestra la función ReLU.

$$\mathrm{ReLU}(x)=\max(0,x)\,. \qquad (24)$$

ReLU es muy utilizada debido a su simplicidad como función, su principal ventaja es el comportamiento de su derivada, puede ser 0 o un valor constante, los valores negativos se transforman en 0 evitando inconvenientes en el momento de encontrar el valor mínimo de la función de costo.

Función de activación sigmoide

La función sigmoide se utiliza para obtener una probabilidad como resultado en la salida de la red neuronal; transforma los valores de entrada x en el rango de $(-\infty, +\infty)$ a valores en el rango de $(0, 1)$. La función es diferenciable, por lo que presenta propiedades adecuadas para el cálculo del gradiente, su representación matemática se presenta a continuación:

$$\text{sigmoide} = \frac{1}{1 + e^{-x}}. \qquad (25)$$

Esta función puede presentar desvanecimiento del gradiente provocando que el modelo se atasque en el proceso de entrenamiento, esto limita la capacidad de aprendizaje. En la Figura 1-16 se muestra el comportamiento de la función sigmoide y ReLU.

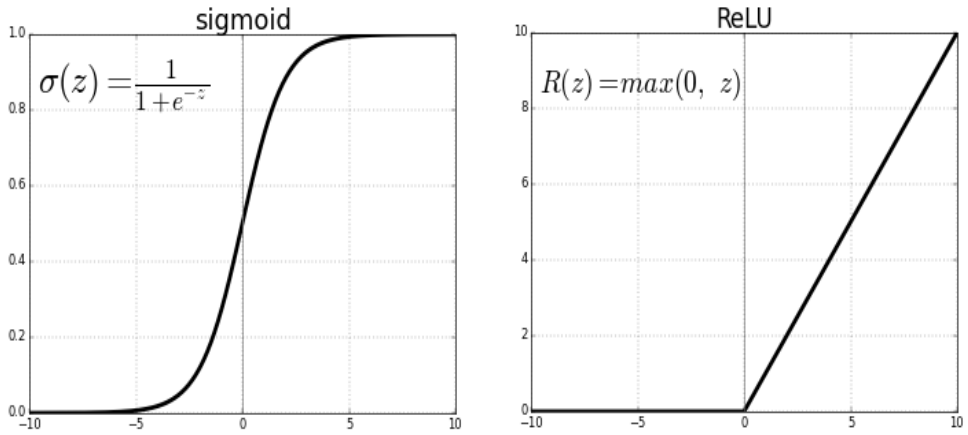

Figura 1-16: Función sigmoide y RELU.
Tomada de [52].

Función de activación softmax

La función Softmax convierte un vector de números en un vector de probabilidades. El uso más común de esta función es en redes de clasificación, principalmente en la capa de salida en la cual se obtiene un vector con las probabilidades de cada clase, la suma de estas probabilidades tiene que ser igual a 1.

- **Algoritmo de optimización y *learning rate***

El principal algoritmo de optimización es el descenso del gradiente, este permite la actualización de los parámetros internos del modelo de redes neuronales para minimizar el error en la función de costo, su función es medir el error de predicción del modelo en el conjunto de datos, encontrando los mínimos locales de la función para mejorar el desempeño de la red. La derivada indica el valor o sentido en el cual se encuentra el mínimo más próximo, este puede ser un mínimo local o un mínimo global, esto presenta inconvenientes puesto que este método no los puede diferenciar. Para evitar este inconveniente, se desarrollaron modificaciones del descenso del gradiente en función de la derivada con la adaptación del LR. La elección de los algoritmos de optimización adecuados puede tener un gran impacto en el modelo de NN, generalmente para datos de gran dimensión se recomienda utilizar el optimizador Adam o RMSProp [53].

Adam

Es uno de los más utilizado debido a su gran rapidez para converger a un mínimo de la función de costo, Adam calcula tasas de aprendizaje capaces de ajustarse a cada parámetro de la red. Este algoritmo se basa en estimaciones de momentos de orden superior y sus actualizaciones son estimadas mediante un promedio entre el primer y segundo momento del gradiente, también puede estimar un promedio exponencial en bajada de gradientes pasados, es adecuado para modelos con grandes conjuntos de datos.

RMSProp

Este algoritmo fue diseñado para acelerar el proceso de optimización en modelos de redes neuronales, establece automáticamente la tasa de aprendizaje para cada parámetro de la red, esto se realiza calculando de forma inicial el tamaño del paso para una función de costo, luego con el valor del tamaño del paso se hace el movimiento en esa dimensión utilizando un promedio decreciente de gradientes parciales. El uso de una media móvil decreciente permite que en lugar de tener un acumulado del gradiente solamente utilice los gradientes parciales más recientes.

El LR se encarga de acelerar el proceso de aprendizaje del algoritmo de NN siendo uno de los hiperparámetros más importantes, su función es controlar el tamaño del paso del descenso del gradiente. El LR es un valor pequeño generalmente varía entre el rango de 0.01 y 0.0001, si este es muy pequeño el modelo de predicción es lento en el proceso de entrenamiento, en cambio si el valor es mayor al rango propuesto esto provoca que el modelo de un salto muy grande en la dirección de la función de error saltando el mínimo.

- **Tamaño de filtro y muestreo**

El tamaño de los filtros y el muestreo se representan mediante matrices $g \times g$ y $q \times q$ respectivamente, se definen para cada capa de convolución, y al igual que para la elección de la arquitectura no existen reglas definidas, los valores g y q se ajustan teniendo en cuenta el tamaño de las imágenes y la capacidad de procesamiento de los equipos.

- **Número de épocas**

Son un hiperparámetro de los algoritmos de NN, su función es establecer el número adecuado de veces en que el conjunto de datos de entrenamiento realiza un paso hacia adelante o *forward* y un paso hacia atrás *backward* sobre la red. No existe un número de épocas estándar que funcione correctamente, este valor puede variar de acuerdo a la BD empleada, la tarea asignada a la red y su profundidad de aprendizaje.

1.6. BASES DE DATOS

Para obtener un modelo de predicción de un algoritmo de ML es esencial la fase de entrenamiento, en la cual es necesario el uso de una base de datos con características que permitan al algoritmo aprender. Diversos trabajos e investigaciones para IPS, plantean un escenario único construyendo una BD propia, esto presenta inconvenientes al momento de comparar los resultados de desempeño de los diferentes algoritmos. Actualmente, existen bases de datos públicas con un conjunto de datos estándar para el posicionamiento en interiores, entre ellas UJIIndoorLoc (UJI), Tampere (TAM), Base de Datos RSSI para Posicionamiento Interior (BDRSSI) y Base de Datos CSI MAMIMO ultra denso para interiores (BDCSI) [53].

1.6.1. UJIIndoorLoc

Se encuentra disponible públicamente en el repositorio de ML de la Universidad de California en Irvine, UJI cubre tres edificios de la Universidad de Jaume I de España, de 4 o 5 pisos dependiendo del edificio, con una superficie de 108.703 m^2; esta BD se usa en problemas de clasificación, en caso de clasificar edificio y piso y de regresión para encontrar las coordenadas geográficas en términos de longitud y latitud, consta de 19.937 datos para entrenamiento y 1.111 datos para validación. Los primeros 520 atributos corresponden a valores de intensidad de la señal RSSI de diferentes WAP, que van desde el rango de -104 a 0 dBm, en el caso de no detectar señal desde algún WAP el valor de RSSI se establece en 100.

La base de datos contiene tres tipos de etiquetas, la primera es el identificador de edificio toma valores de 0 a 2, los cuales representan los tres edificios, la segunda corresponde al identificador de piso con valores de 0 a 4, por último, la etiqueta de coordenadas con valores de longitud y latitud, dadas en el sistema de coordenadas Universal Transversal de Mercator[1] (UTM, *Universal Transverse Mercator*) [54]. En la Figura 1-17 se muestra la distribución de los puntos de referencia de cada uno de los edificios, pisos y su distribución geográfica.

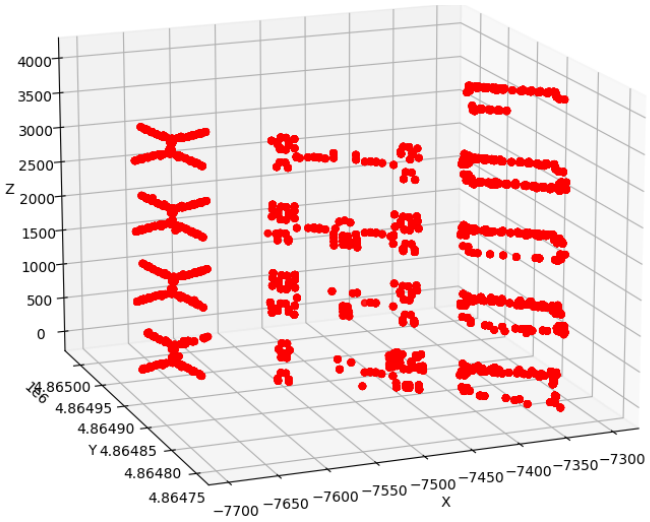

Figura 1-17: Base de datos UJIIndoorLoc en 3D.
Por los autores.

1.6.2. Tampere

La BD cubre un edificio de la Universidad de Tampere en Finlandia, se encuentra disponible en línea, se utiliza el parámetro RSSI de diferentes puntos de acceso y se expresa en valores negativos desde -100 dBm a 0 dBm. Para los WAP que no registren RSSI se toma un valor de 100. La BD cuenta con 4648 registros divididos aleatoriamente y de manera no superpuesta en 697 registros de entrenamiento y 3951 registros de validación. Los registros de RSSI se toman de 992 WAP y cuentan con etiqueta de coordenadas y de altura, sus datos se pueden visualizar en la Figura 1-18 [55].

[1] UTM es un sistema de proyección cartográfico basado en cuadrículas con el cual se pueden referenciar puntos sobre la superficie terrestre.

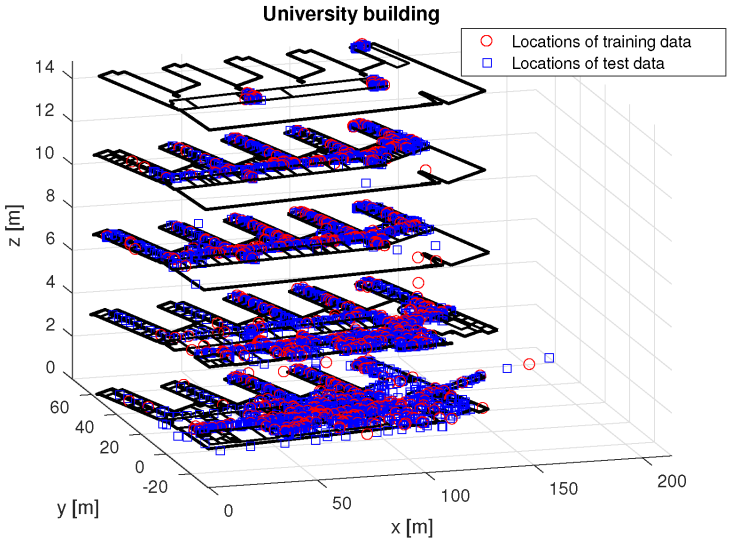

Figura 1-18: Base de datos Tampere en 3D.
Tomada de [55].

1.6.3. Base de datos RSSI para localización en interiores

Esta BD contiene valores del indicador de señal RSSI de tres escenarios diferentes y con las tecnologías inalámbricas WI-FI, Bluetooth y Zigbee, cada escenario se compone de distintos niveles de interferencia y tamaño. La base de datos utiliza la técnica de trilateración y *fingerprinting* para el diseño de algoritmos de ML, se utilizaron tres escenarios con diferentes características para determinar el comportamiento del algoritmo de ML en diferentes condiciones ambientales, la base de datos se encuentra disponible en el repositorio de la IEEE [56].

El primer escenario plantea un entorno libre de interferencias entre los WAP y el dispositivo móvil, se toman 49 puntos RM, los cuales conforman la base de datos y se toman 10 registros de prueba al azar para validar el algoritmo. El segundo escenario plantea un entorno con alto niveles de interferencias, se recolectan 16 RM con una distancia mayor entre cada uno en comparación al primer escenario y se recolectan 6 registros al azar dentro del entorno para validación. El tercer escenario es un laboratorio con alta cantidad de ruido debido a interferencias y obstáculos, se toman 40 RM con un patrón alterno entre ellos y se recolectan medidas de 16 registros al azar; en los escenarios se usan tres WAP, la configuración y los puntos de entrenamiento se observan en la Figura 1-19.

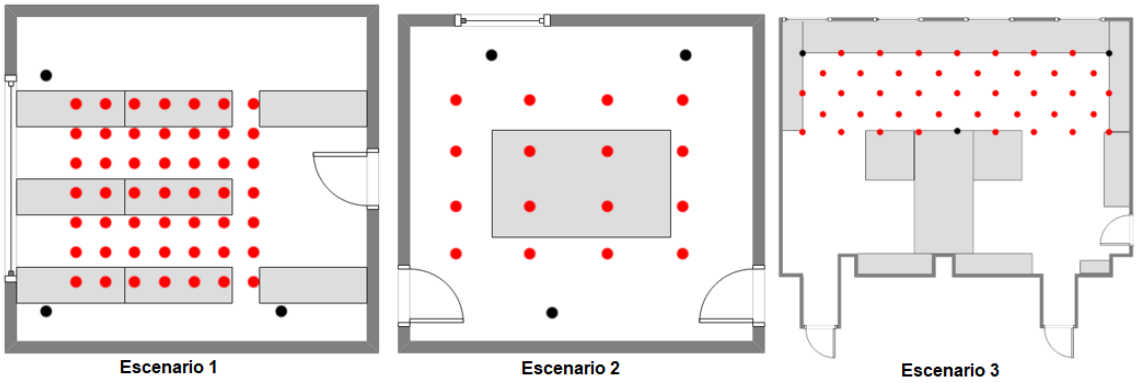

Figura 1-19: Escenarios de entrenamiento.
Adaptada de [56].

1.6.4. Base de datos CSI MAMIMO ultra denso para interiores

La BD contiene medidas de CSI, estos datos fueron recolectados por la Universidad de KU Leuven, las mediciones se tomaron en un cuadricula, usando diferentes arreglos de antenas. La estación base cuenta con 64 antenas y cada una de estas antenas recibe la señal desde cada posición, cada señal de CSI consta de 100 subportadoras con amplitud y fase, estas se encuentran espaciadas uniformemente en frecuencia con características espaciales y temporales.

BDCSI contiene 252.004 registros CSI etiquetados espacialmente utilizando coordenadas bidimensionales cada 5 mm. En la toma de medidas se establecen tres tipos de arreglos de las estaciones base: el primero es un arreglo de 8 x 8 antenas el cual implementa una matriz rectangular uniforme, en segundo lugar, se utilizó el arreglo lineal uniforme de 64 antenas en línea y el ultimo arreglo lleva una distribución alrededor de la cuadricula en un arreglo en pares de 8 antenas. En los arreglos se realizó mediciones en LOS y también en no LOS con la ayuda de una lámina de metal, en la Figura 1-20 se observa la distribución de los arreglos [57].

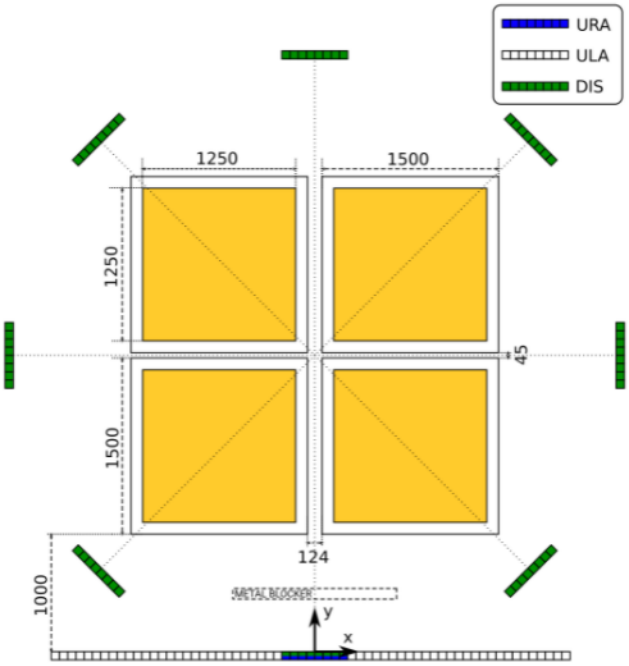

Figura 1-20: Recolección de muestras CSI. Tomada de [57].

1.7. MÉTRICAS DE DESEMPEÑO

En IPS y ML existen métricas que ayudan a evaluar el desempeño del sistema, permitiendo seleccionar los parámetros adecuados del algoritmo. En algoritmos de ML se usa una parte de los registros de la BD en la etapa de validación, en esta etapa se utiliza una cantidad de registros etiquetados para obtener una predicción del modelo entrenado previamente, los valores de etiqueta predichos se comparan con los valores reales, de esta forma se evalúa el desempeño del algoritmo con el uso de distintas métricas. La etapa de validación es de vital importancia en la construcción de un algoritmo de ML, ya que permite saber cómo se comporta el modelo en un entorno real con registros desconocidos.

En IPS las métricas de evaluación más usadas son la exactitud y la complejidad, estas se obtienen de forma similar en la mayoría de investigaciones sobre sistemas de posicionamiento, sin embargo, la precisión en IPS tiene diversas interpretaciones de acuerdo a la naturaleza de los datos y el objetivo del sistema. Teniendo en cuenta las BD disponibles públicamente, los IPS son capaces de determinar la posición de un dispositivo móvil, prediciendo el edificio, piso o coordenada; en la predicción de edificio y piso se usan las métricas de exactitud y complejidad con el fin de analizar

el comportamiento del modelo de predicción basado en clasificación. La predicción de coordenadas se evalúa mediante la exactitud, precisión y complejidad [58] [59].

1.7.1. Tasa de aciertos

En casos de clasificación como la predicción de edificio o piso, la tasa de aciertos es la medida para evaluar la calidad de los modelos de predicción. La tasa de aciertos define la relación entre el número de elementos clasificados correctamente y el número total de elementos de validación, tal cómo se presenta a continuación:

$$\text{Tasa de aciertos} = \frac{\text{Registros clasificados correctamente}}{\text{Total de registros de validación}} \text{ x } 100 \,. \qquad (26)$$

1.7.2. Exactitud

En problemas de IPS de regresión donde se necesita predecir la coordenada de un dispositivo móvil, la exactitud se representa como el error de distancia media, y se mide con la distancia euclidiana entre la coordenada predicha y la coordenada real, esto se hace con todos los registros de validación, su expresión se presenta continuación:

$$\text{Exactitud} = \frac{1}{v}\sum_{j=1}^{v}\sqrt{\left(xp_j - xr_j\right)^2 + \left(yp_j - yr_j\right)^2}\,, \qquad (27)$$

donde (xp_v, yp_v) y (xr_v, yr_v) representan las coordenadas predichas y reales respectivamente y v representa el número de registros de validación. Entre menor sea el valor de exactitud mejor se considera el desempeño del sistema.

1.7.3. Precisión

Para problemas de regresión en IPS la precisión es la dispersión del error de distancia entre las coordenadas predichas y las coordenadas reales, se calcula mediante la desviación estándar. La precisión es una métrica que determina la confiabilidad de un algoritmo de IPS, en condiciones ambientales diversas y cambiantes [60].

1.7.4. Complejidad

La complejidad de un IPS se atribuye a factores como la capacidad informática del algoritmo de posicionamiento y los equipos necesarios para el desarrollo, por lo general, no es fácil deducir una forma de evaluar la complejidad de un IPS, por lo tanto, se considera el tiempo de procesamiento como un buen indicador de

complejidad, en el caso de algoritmos de ML, el tiempo se puede medir tanto en la etapa de entrenamiento como en la etapa de validación.

1.8. DESARROLLO METODOLÓGICO

Para el desarrollo del Trabajo de Grado se optó por utilizar la metodología ágil de Programación Extrema (XP, *Extreme Programming*). Esta metodología se basa en principios básicos como: la capacidad de adaptación a cambios, la retroalimentación y la entrega de un software de calidad, esto se logra mediante el fortalecimiento de las relaciones interpersonales, potenciando el trabajo en equipo e incrementando el conocimiento de los desarrolladores. XP es una metodología ágil utilizada para el desarrollo y gestión de proyectos basados en software, consta de un conjunto de reglas y practicas basadas en cuatro fases o actividades estructurales: planeación, diseño, desarrollo y pruebas [61].

A. Fase de planeación

En esta fase se plantea los requerimientos funcionales y no funcionales del proyecto, luego toda esta información es consignada en historias de usuario con el fin de conocer la funcionalidad y características para el correcto diseño del software.

B. Fase de diseño

Esta fase sirve de guía para la implementación de las historias de usuario por medio de diseños simples o un prototipo funcional del proyecto, detallando el funcionamiento del sistema de forma sencilla y entendible. El diseño sirve a los programadores para afrontar la siguiente fase, por lo tanto, el diseño no debe ser una representación compleja.

C. Fase de implementación

Esta fase recomienda implementar las funcionalidades del prototipo en parejas, así dos personas trabajan con el objetivo de crear código al mismo tiempo para construir una versión de la funcionalidad diseñada previamente, esto disminuye la tasa de errores y mejora el diseño del programa por lo tanto el proyecto final será de mejor calidad. Una parte fundamental de la fase es la implementación de pruebas unitarias, establecidas durante toda la fase de desarrollo.

D. Fase de Pruebas

La fase de pruebas inicia en conjunto con la fase de implementación. Antes de empezar a crear código por parte de los desarrolladores, se debe plantear las pruebas a las cuales se debe someter cada funcionalidad, si el código creado

aprueba de forma eficiente, esta funcionalidad puede ser implementado al código universal del proyecto.

1.8.1. Definición de tareas

En el caso del trabajo de grado se definieron tareas para cada fase, las cuales se presentan en la Figura 1-21, estas tareas ayudan al desarrollo de forma ordenada y sistemática.

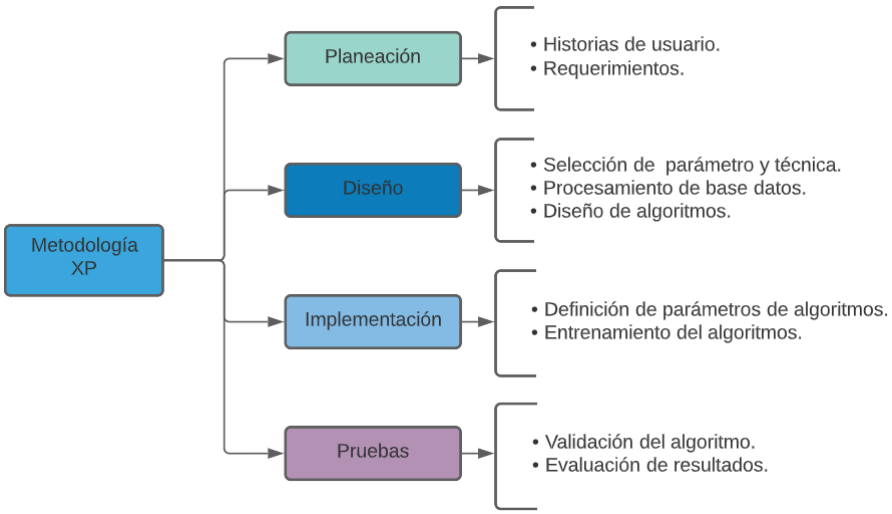

Figura 1-21: Metodología de desarrollo XP.
Por los autores.

La fase de planeación y diseño son consignadas en el capítulo 2, la fase de implementación y pruebas se realizaron en conjunto acorde a la metodología ya que permiten llevar a cabo un desarrollo cíclico como se aprecia en la Figura 1-22.

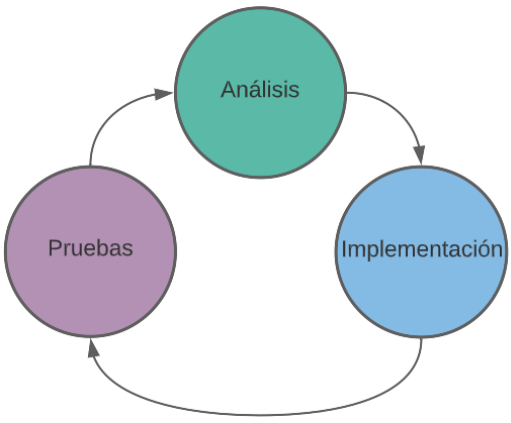

Figura 1-22: Ciclo de desarrollo XP.
Por los autores.

2. FASE DE PLANEACIÓN Y DISEÑO

Este capítulo presenta las fases de planeación y diseño de acuerdo a la metodología XP, se plantean las historias de usuario y los requerimientos para la adaptación del algoritmo propuesto, se realiza la elección de las tecnologías, parámetros y técnicas, con el fin de obtener el diseño del sistema IPS.

2.1. HISTORIAS DE USUARIO

Después de realizar la investigación sobre parámetros, técnicas y algoritmos de ML para IPS se define las historias de usuario con las funcionalidades del algoritmo, facilitando así el entendimiento por parte del equipo de desarrollo. Las historias de usuario están enfocadas a un usuario capaz de manipular el código fuente del modelo de predicción sin necesidad de ser un desarrollador de alto nivel.

Las historias de usuario se presentan mediante las Tablas 2-1, 2-2 y 2-3 en las cuales, en primer lugar, se establece su prioridad, además, el riesgo que conlleva el desarrollo de la historia, el tiempo en semanas necesario para cada historia se representa con los puntos asignados. A continuación, se presentan las historias de usuario:

Historia de Usuario	
Numero: 1	
Nombre historia: Entrenar y validar un algoritmo de ML usando una base de datos.	
Prioridad: Alto	**Riesgo de desarrollo:** Alto
Puntos asignados: 4	
Responsable: Darwin Mejía, Cristhian Pacichana	
Descripción: Se podrá entrenar el algoritmo, realizar el ajuste de hiperparámetros y obtener nuevas predicciones.	

Tabla 2-1: Historia de usuario 1.
Por los autores.

Historia de Usuario	
Numero: 2	
Nombre historia: Cargar una base de datos.	
Prioridad: Medio	**Riesgo de desarrollo:** Medio
Puntos asignados: 2	
Responsable: Darwin Mejía, Cristhian Pacichana	
Descripción: Se podrá seleccionar una de las bases de datos propuestas o utilizar una propia.	

Tabla 2-2: Historia de usuario 2.
Por los autores.

Historia de Usuario	
Numero: 3	
Nombre historia: Evaluar el desempeño del algoritmo propuestos.	
Prioridad: Alto	**Riesgo de desarrollo:** Medio
Puntos asignados: 3	
Responsable: Darwin Mejía, Cristhian Pacichana	
Descripción: Se obtendrá resultados de las métricas de evaluación del desempeño del algoritmo de ML propuesto.	

Tabla 2-3: Historia de usuario 3.
Por los autores.

2.2. REQUERIMIENTOS

En esta sección se exponen los requerimientos funcionales y no funcionales, para el desarrollo del trabajo de grado.

2.2.1. Requerimientos funcionales

- Adaptar un algoritmo de ML al posicionamiento en interiores.
- Cumplir con un valor de precisión para IPS menor a 1 m.

2.2.2. Requerimientos no funcionales

- Elegir el parámetro y técnica adecuados en el diseño e implementación del algoritmo de ML para IPS
- Obtener un tiempo de predicción del algoritmo de ML menor a 1 s.

2.3. DISEÑO

Para el diseño se consideran los componentes de un IPS, representados en la Figura 2-1, a continuación, se presenta una descripción de cada uno de ellos:

- **WAP:** terminales fijos inalámbricos encargados de emitir señal RF, dentro de un entorno interior usando una tecnología como WIFI, Bluetooth o Zigbee.
- **Dispositivo móvil:** encargado de extraer mediciones de la señal RF proveniente de los WAP, mediante un parámetro de posicionamiento para IPS.
- **Servidor**: su función es recibir la información proveniente del dispositivo móvil, posteriormente el estimador de posición se encarga del procesamiento de la información hasta obtener la predicción de la ubicación de un dispositivo móvil, usando una técnica y un algoritmo de posicionamiento.

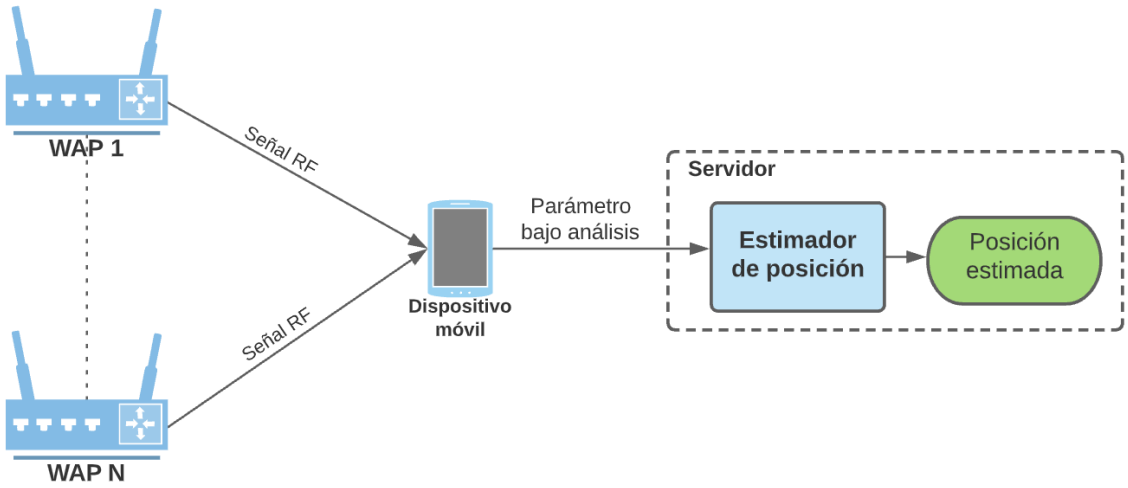

Figura 2-1: Diagrama de bloques de un IPS.
Por los autores.

Para implementar un IPS es necesario definir un parámetro y el estimador de posición, el cual tiene asociado una técnica de posicionamiento como trilateración, triangulación o *fingerprinting*. La elección de la técnica de posicionamiento es un proceso en el cual se tienen en cuenta múltiples variables, como la naturaleza del problema, la infraestructura disponible y el costo.

2.3.1. Análisis y selección de parámetros y técnicas

El diseño de un sistema de posicionamiento en interiores ofrece una serie de alternativas para su desarrollo, con sus respectivas ventajas y desventajas. En la Tabla 2-4 se muestran los parámetros de posicionamiento para IPS más utilizados con sus respectivas características.

Parámetro	AOA	TOA/TDOA	RSSI	CSI
LOS	Si	Si	No	No
Sincronización	No	Si	No	No
Costo	Alto	Alto	Medio	Alto
Tecnología	WIFI Bluetooth	Ultrasonido WIFI	WIFI Bluetooth Zigbee	WIFI
Técnica asociada	Triangulación	Trilateración	Trilateración Fingerprinting	Fingerprinting
Exactitud	Media	Alta	Alta	Alta

Tabla 2-4: Parámetros IPS.
Por los autores.

De acuerdo con la información de la Tabla 2-4, un factor clave de RSSI es que no necesita LOS, esto es bastante favorable en escenarios interiores debido a los múltiples obstáculos existentes en el medio, por lo tanto, es el parámetro más viable para el desarrollo del sistema de posicionamiento en interiores, por su bajo costo y por las múltiples tecnologías asociadas.

Otro parámetro destacado es CSI el cual presenta una alta exactitud y un buen desempeño, brinda información detallada del entorno, se basa en señales WIFI y se usa en sistemas robustos. CSI presenta algunos inconvenientes, por ejemplo la necesidad de hardware adicional y además una alta complejidad de implementación, este parámetro presenta una viabilidad media, aun así, se escoge como segundo parámetro para el diseño del sistema IPS [62].

La elección de la técnica depende del parámetro seleccionado; en la Tabla 2-5 se presenta las técnicas asociadas a RSSI o CSI.

Técnica	**Trilateración**	***Fingerprinting***
Hardware adicional	No	No para RSSI, si para CSI
Sincronización	No	No
Complejidad	Media	Alta
Parámetro asociado	RSSI	RSSI, CSI
Exactitud	Media	Alta

Tabla 2-5: Técnicas IPS.
Por los autores.

De acuerdo con la Tabla 2-5, la técnica de trilateración no requiere de sincronización si utiliza RSSI y tiene una exactitud media, en cambio *fingerprinting* no necesita ningún tipo de sincronización, además, su exactitud es alta y utiliza los parámetros RSSI y CSI, estas características evidencian que *fingerprinting* a pesar de tener una alta complejidad ofrece más ventajas frente a trilateración.

La técnica de *fingerprinting* en su primera etapa construye una BD de información con los registros del parámetro elegido, la construcción de la BD permite entrenar y validar los diferentes algoritmos de aprendizaje supervisado.

Además, con el uso de una tecnología inalámbrica como WIFI, Bluetooth o Zigbee, este sistema se adapta fácilmente a la infraestructura inalámbrica existente dentro de las edificaciones, esto significa una disminución en los costos. La técnica utilizada para el diseño del algoritmo de posicionamiento en interiores es *fingerprinting*.

2.3.2. Diseño de *fingerprinting*

Fingerprinting tiene dos fases *offline* y *online*. La fase *offline* recibe mediciones del parámetro RSSI o CSI de las diferentes ubicaciones del entorno, estas señales son procesadas y en cada ubicación crea un RM para almacenarlas dentro de una base de datos, luego se extrae la información necesaria mediante el tratamiento de la base de datos. La fase *online* compara nuevos valores de medición con valores de la base de datos, mediante algoritmos de posicionamiento prediciendo las coordenadas de un dispositivo móvil. El diagrama de bloques de *fingerprinting* se muestra en la Figura 2-2.

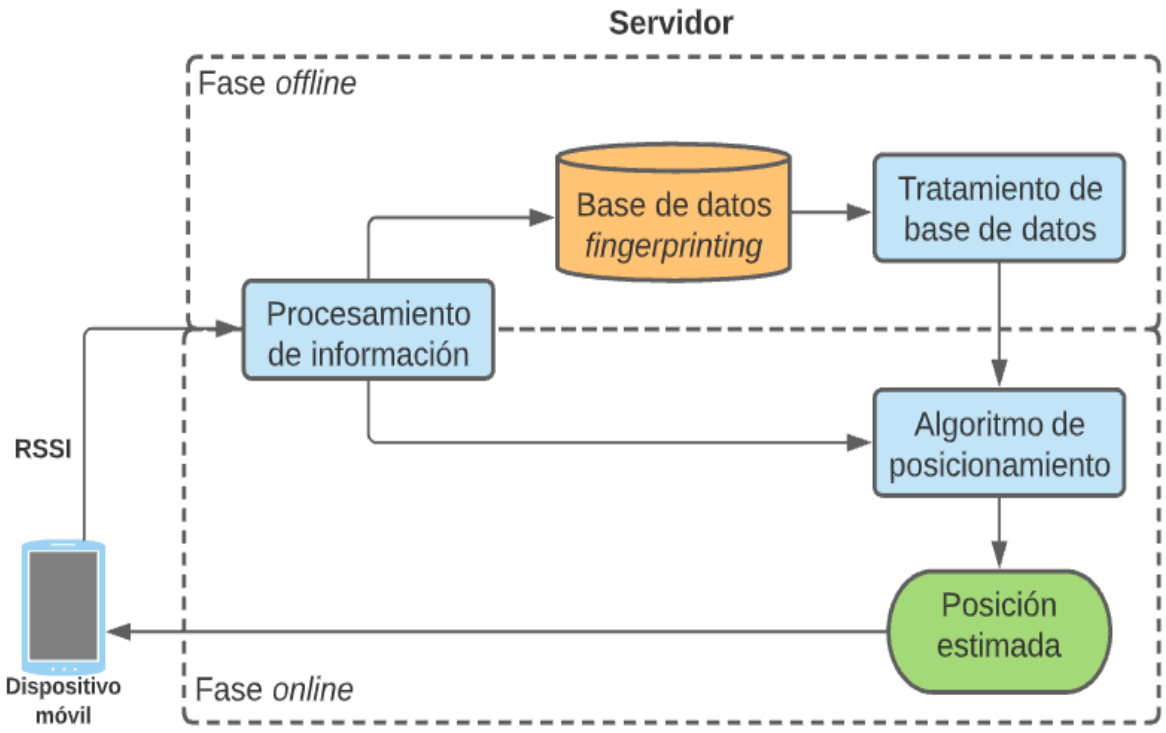

Figura 2-2: Diagrama de bloques técnica fingerprinting.
Por los autores

La elección de los parámetros RSSI y CSI junto con *fingerprinting*, hacen necesario el uso de una base de datos y un algoritmo de posicionamiento; para el trabajo de grado se adaptó un algoritmo ML de aprendizaje supervisado con el fin de analizar el comportamiento de la métrica de precisión y exactitud.

El algoritmo de ML necesita una base de datos con una gran cantidad de registros y características para obtener un modelo de predicción capaz de estimar la posición de un dispositivo móvil, por lo cual se proponen base de datos disponibles públicamente.

2.3.3. Análisis y selección de base de datos

En la fase *offline* de *fingerprinting* se realiza la construcción de una base de datos, con características que permitan al algoritmo estimar la posición de un nuevo dispositivo móvil, en el caso del trabajo de grado las características deben ajustase a los requerimientos de un algoritmo de ML, por lo cual se plantea el uso de conjuntos de datos para ML disponibles públicamente; los más usados en problemas de posicionamiento son, UJI, TAM, BDRSSI y BDCSI, una comparativa de estos se presenta en la Tabla 2-6.

Base de datos	UJI	TAM	BDRSSI	BDCSI
Parámetro	RSSI	RSSI	RSSI	CSI
Número de WAP	520	992	3	64
Número de registros	21048	4684	5894	16128256
Complejidad de procesamiento	Medio	Medio	Bajo	Alto
Construcción de escenario	No	No	Si	Si
Registros de validación	Si	No	Si	No
Etiquetas	Edificio, piso, Coordenadas	Piso, Coordenadas	Coordenadas	Coordenadas

Tabla 2-6: Bases de datos para IPS.
Por los autores.

De acuerdo a la Tabla 2-6, UJI presenta una gran cantidad de registros con varios WAP, múltiples etiquetas y registros de validación, en un escenario adaptado a la infraestructura inalámbrica WIFI en la Universidad Jaume I; TAM contiene mayor número de WAP y menor número de registros en comparación a UJI, además no cuenta con registros de validación, lo cual implica que no es posible evaluar el desempeño del modelo de predicción.

BDRSSI se construye en un escenario establecido, posicionando los WAP WIFI en lugares estratégicos y tomando mediciones ordenadas, además cuenta con datos de validación y gran cantidad de datos de entrenamiento; BDCSI es la BD con mayor número de registros, esto debido a la forma de sus datos. El procesamiento de la información de BDCSI es una tarea compleja siendo su principal desventaja.

UJI y TAM fueron construidas en un escenario de gran área con varios WAP, por el contrario, BDRSSI y BDCSI muestran un escenario de pequeña área con número limitado de WAP en comparación a UJI y TAM, tomando en cuenta lo anterior, se

eligen las bases de datos UJI y BDRSSI, ya que tienen mejores características, lo cual permite evaluar el comportamiento del algoritmo ML en diferentes entornos.

2.3.4. Procesamiento de base de datos

Es necesario realizar un procesamiento para ordenar, clasificar y normalizar los registros de la BD, y obtener una estructura de fácil manejo dentro del entorno del desarrollo.

- **UjiIndoorLoc**

La base de datos de UJI contiene 19937 registros de entrenamiento de los cuales 18 estaban incompletos tanto en los valores RSSI como en las etiquetas, estos registros fueron descartados, obteniendo un total de 19919 registros de entrenamiento, de igual forma en los registros de validación se encontraron datos faltantes, de los 1111 registros, 11 estaban incompletos, estos fueron descartados, obteniendo un total de 1100 registros de validación. La estructura de la base de datos para predecir edificio, piso y coordenadas se presenta en las figuras 2-3, 2-4 y 2-5 respectivamente.

Figura 2-3: a) BD entrenamiento edificio UJI, b) BD Validación edificio UJI. Por los autores.

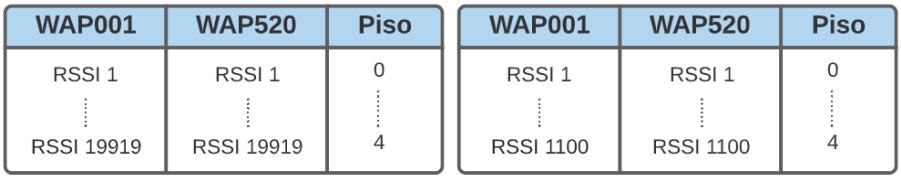

Figura 2-4: a) BD entrenamiento piso UJI, b) BD Validación piso UJI. Por los autores.

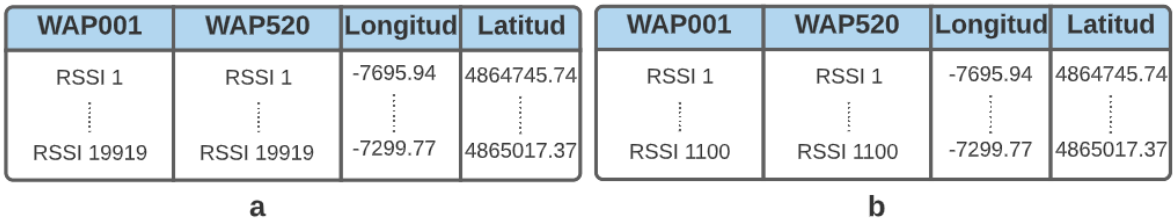

Figura 2-5: a) BD entrenamiento coordenadas UJI, b) BD Validación coordenadas UJI. Por los autores.

- **BDRSSI**

La base de datos disponible en el repositorio de la IEEE contiene información de tres escenarios específicos. El escenario 1 no cuenta con la información de RSSI, por lo cual se descarta, en el caso del escenario 2 no dispone de suficiente información para validación, esto no permite evaluar correctamente los algoritmos de ML, por lo tanto, no se utiliza en el desarrollo del trabajo de grado. En el escenario 3, existen 4160 registros de RSSI de entrenamiento, con 104 datos en cada uno de los 40 RM, se cuenta con 1666 registros de validación con información de RSSI y coordenadas de 16 RM. La estructura de la base de datos del escenario 3 se presenta en la Figura 2-6.

WAP A	WAP C	X	Y
RSSI 1	RSSI 1	0	0
⋮	⋮	⋮	⋮
RSSI 4160	RSSI 4160	9.625	2.492

a

WAP A	WAP C	X	Y
RSSI 1	RSSI 1	0	0
⋮	⋮	⋮	⋮
RSSI 1666	RSSI 1666	9.625	2.492

b

Figura 2-6: a) BD entrenamiento escenario 3, b) BD Validación escenario 3. Por los autores.

Las coordenadas y los valores RSSI son normalizados, en cambio, los valores de etiqueta como edificio y piso no se normalizan, ya que son usados en un problema de clasificación.

UJI y BDRSSI fueron construidas en dos entornos distintos, la primera presenta un escenario de gran área, la cual cuenta con tres edificios, en cada edificio se toman varios RM por piso y cada RM cuenta con mínimo 10 registros RSSI con etiqueta de edificio, piso y coordenadas. Para BDRSSI se presenta un escenario enfocado en un área pequeña y cada RM cuenta con más de 100 registros de RSSI con etiqueta de coordenadas. A pesar de que UJI cuenta con mayor número de puntos de referencia, BDRSSI tiene una mayor cantidad de registros por cada RM, además, la densidad por área de puntos de acceso es mayor para BDRSSI.

2.3.5. Análisis y selección de algoritmos de *machine learning*

El uso de algoritmos de ML en IPS, permite obtener un desempeño favorable en métricas como la exactitud, la precisión y tiempo de procesamiento. Los algoritmos de clasificación actuales que más han contribuido en los sistemas IPS son: KNN, Árbol de decisión, Naive Bayes, SVM y NN.

Para realizar la selección de los algoritmos de ML a emplear en el desarrollo del trabajo de grado se tienen en cuenta las características consignadas en la Tabla 2-7.

Algoritmo / Característica	KNN	Árbol de decisión	Naive Bayes	SVM	NN
Precisión	Media	Media	Media	Alta	Alta
Tiempo de procesamiento	Alta	Media	Alta	Media	Media
Facilidad de implementación	Alta	Alta	Media	Media	Media
Complejidad matemática	Baja	Media	Media	Alta	Alta

Tabla 2-7: Algoritmos de ML para IPS.
Por los autores.

Se selecciona KNN por obtener resultados aceptables en IPS, además, de su fácil implementación en comparación con los demás algoritmos de ML, sin embargo, su principal desventaja es el tiempo de procesamiento en BD de gran tamaño. Por otra parte, se tiene el algoritmo de redes neuronales utilizado ampliamente en la actualidad en IPS por su alta capacidad computacional obteniendo un alto desempeño, siendo una opción adecuada para el desarrollo del sistema IPS [30]. En el trabajo de grado se adaptan los algoritmos KNN y NN, enfocados en resolver el problema de posicionamiento en interiores.

2.3.6. Sistema de posicionamiento con *fingerprinting* y RSSI

En esta sección se describe el funcionamiento general de un sistema IPS basado en ML, para ello se propone un diagrama de flujo presentado en la Figura 2-7 el cual proporciona una visión general del funcionamiento de un sistema IPS.

En la Figura 2-7, se observa el desarrollo paso a paso del sistema IPS con *fingerprinting* hasta obtener el valor de la posición en términos de edificio, piso y coordenadas, según el tipo de BD. El proceso inicia extrayendo las características del parámetro RSSI, luego se determina si el sistema es capaz de estimar la posición, si no lo es, se procede a realizar la fase *offline* de la técnica de *fingerprinting.* La información del parámetro RSSI se usa como registros de entrenamiento para construir una BD. Luego de analizar la BD se extraen las características necesarias para la construcción del modelo de ML en un proceso denominado parametrización en el cual se dividen los registros de entrenamiento y validación.

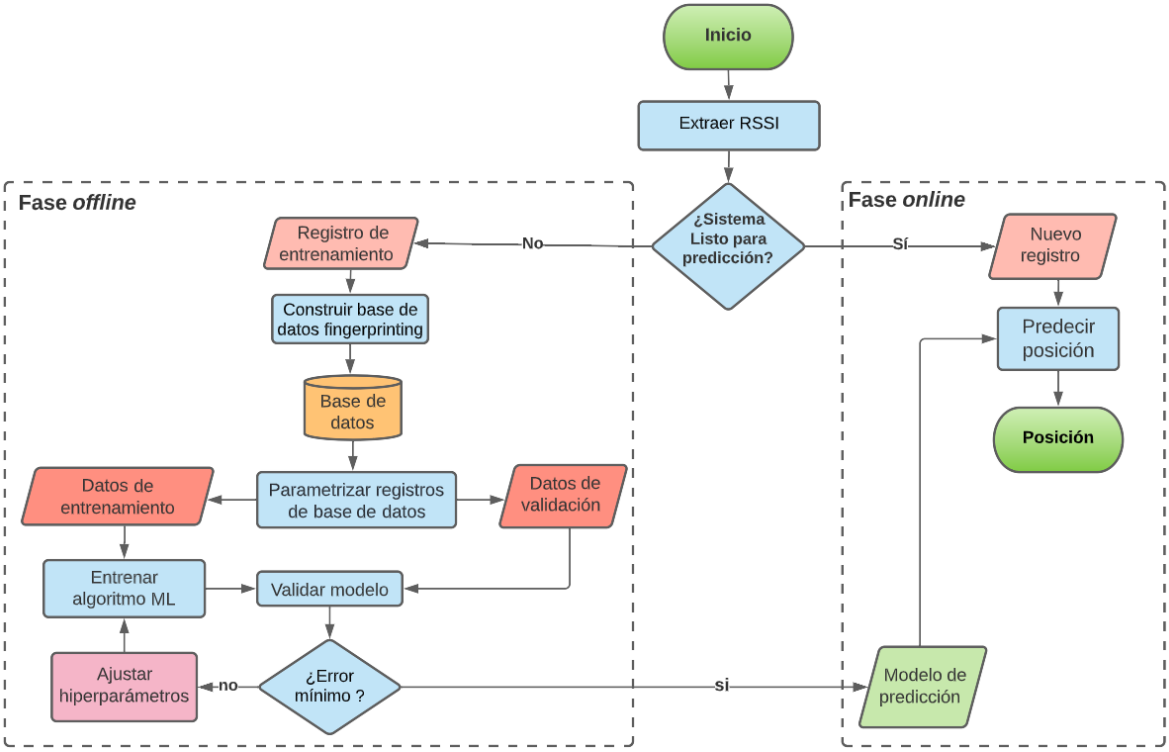

Figura 2-7: Sistema IPS con fingerprinting y ML. Por los autores.

Posteriormente se realiza el entrenamiento del algoritmo y la validación del modelo, si el error en la métrica de desempeño no es el deseado, se ajustan los hiperparámetros, y se realiza una vez más el proceso de entrenamiento y validación, este proceso se repite múltiples veces hasta obtener un error mínimo en la métrica de desempeño[1]. Posteriormente se obtiene un modelo de predicción, el cual se utiliza en la fase online donde junto con nuevos registros RSSI predice una posición.

El trabajo de grado se enfoca en el desarrollo de un algoritmo de ML, orientándose principalmente en los bloques de entrenamiento, validación y ajuste de parámetros. A continuación, se muestran los diagramas de flujo de los algoritmos de ML seleccionados con el fin de proporcionar una visión más clara de su funcionamiento para su posterior implementación.

- **Algoritmo de posicionamiento basado en K vecinos más cercanos**

En la Figura 2-8 se presenta el diagrama de flujo del funcionamiento del algoritmo KNN, el cual usa los datos de entrenamiento y validación para calcular la distancia euclidiana entre estos, y determinar los k vecinos más cercanos, luego por el criterio de decisión se determina la posición de los registros de validación y se comparan

[1] El error mínimo en la métrica de desempeño se obtiene después de entrenar múltiples veces el algoritmo ML con diferentes hiperparámetros, comparando los resultados de cada entrenamiento.

con los valores reales. Si el error es elevado se ajusta el parámetro k y el criterio de decisión y se calcula nuevamente la distancia euclidiana, este proceso se realiza múltiples veces hasta que el error sea mínimo según el criterio de desempeño escogido, luego se obtiene el modelo de predicción que puede ser usado con datos desconocidos.

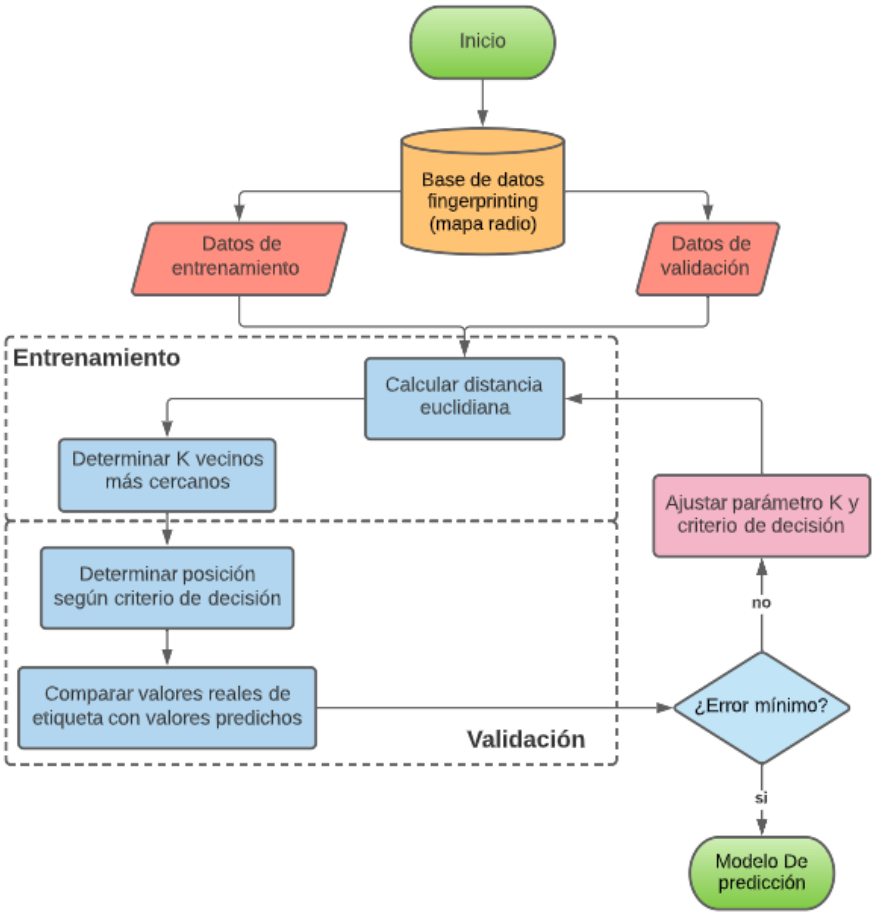

Figura 2-8: Diagrama de flujo algoritmo KNN.
Por los autores.

- **Algoritmo de posicionamiento basado en Redes Neuronales**

Se propone el diseño mostrado en la Figura 2-9, en el algoritmo NN es esencial el uso de una base de datos con suficientes registros de entrenamiento y validación, para obtener un modelo con un buen desempeño, posteriormente se definen los hiperparámetros iniciales para ejecutar el modelo, si el periodo de entrenamiento no termina, se realiza el ajuste de parámetros, el tiempo de entrenamiento se define mediante el número de épocas. Cuando finaliza el entrenamiento se realiza la etapa de validación, en la cual se ejecuta el modelo con los registros de validación y se comparan los valores de las etiquetas reales con los valores predichos, si el error

del modelo no es el esperado se reajustan los hiperparámetros y se entrena nuevamente el modelo, si el error según el criterio de desempeño es mínimo se obtiene un modelo de predicción capaz de predecir nuevos datos.

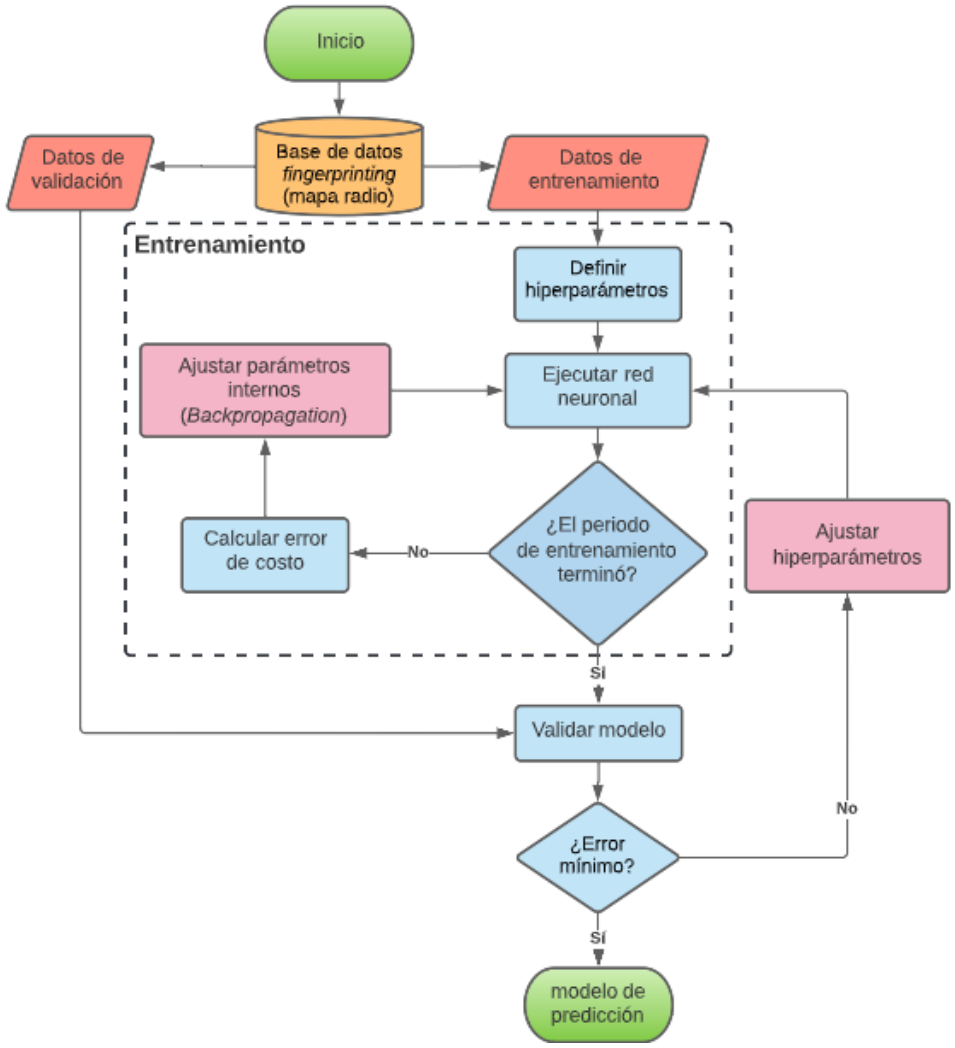

Figura 2-9: Diagrama de flujo algoritmo NN.
Por los autores.

Dentro del campo de las redes neuronales se proponen dos enfoques para la predicción de posición del dispositivo móvil, el primer enfoque utiliza un modelo secuencial denso y el segundo un modelo convolucional, en ambos casos se utiliza el diagrama de flujo de la Figura 2-9, la diferencia se sitúa en el procesamiento de la BD, para el enfoque convolucional se convierten los registros en imágenes de un tamaño definido.

3. FASE DE IMPLEMENTACIÓN Y PRUEBAS

En este capítulo se plantean la fase implementación y pruebas de la metodología XP, en la cual se implementan los algoritmos en el lenguaje de programación Python, se obtiene resultados y se optimizan[1] sus hiperparámetros hasta conseguir un modelo de predicción con un error de distancia aceptable con respecto a la métrica de desempeño.

3.1. IMPLEMENTACIÓN DE ALGORITMOS

KNN y NN se implementan en el software Python usando el entorno de desarrollo de Google Colab [63], siendo la mejor opción ya que ofrece gran variedad de soluciones para el equipo de desarrollo, además, es el lenguaje de referencia para desarrollo de proyectos de AI teniendo a disposición librerías como Tensorflow, Keras y Sklearn. En la primera sección del anexo A se encuentra la guia de implementación de loa algoritmos propuestos.

3.1.1. Implementación algoritmo K vecinos más cercanos

La implementación de KNN se divide en dos etapas: en la primera se almacenan los registros de entrenamiento con sus respectivas etiquetas de clase y en la segunda se realiza la predicción de los datos de validación. Para implementar estas etapas se usa la biblioteca de Sklearn, en la cual es necesario definir el parámetro k y el criterio de decisión.

La construcción software en el entorno de desarrollo de Python se presentan en el pseudocódigo de la Figura 3-1, inicialmente se importan las librerías necesarias para la ejecución del código, luego se carga la base de datos y se realiza el tratamiento de los registros. Se establece un bucle, en el cual se genera el modelo KNN definiendo sus hiperparámetros, posteriormente se entrena el modelo mediante el cálculo de la distancia euclidiana entre los registros de entrenamiento y validación obteniendo los resultados de predicción, estos se comparan con los resultados reales con el fin de evaluar la exactitud, precisión y tiempo de procesamiento.

[1] El proceso de optimización dentro del campo de ML, se refiere a la selección de los valores de hiperparámetros con el menor margen de error de acuerdo a la métrica de desempeño.

```
INICIO
    Importar ('Librerías')
    Cargar ('Base de Datos')
    Proceso ('Tratamiento de datos')
            Separar Registros ('RegEntrenaiento', 'RegValidacion', 'ResReales')
    Repetir
          proceso ('Crear modelo')
                  Definir ('K')
                  Definir ('Criterio de decisión')
          Proceso ('Entrenamiento')
                  Calcular distancia euclidiana('RegEntrenaiento-RegValidacion')
                  Determinar ('K vecinos más cercanos')
                  Determinar ('Posición con criterio de decisión')
                  Obtener ('ResPredichos')
          Proceso ('Validación')
                  Comparar ('ResPredichos', 'ResReales')
                  Obtener ('Exactitud')
                  Obtener ('Precisión')
                  Obtener ('Tiempo de procesamiento')
     Hasta que ('Exactitud', 'Precisión'='Aceptable')
FIN
```

Figura 3-1: Pseudocódigo KNN.
Por los autores.

3.1.2. Implementación algoritmo redes neuronales

La implementación del algoritmo de NN se realiza usando la librería keras, en la cual se definen los modelos en el enfoque secuencial denso y convolucional.

- **Redes neuronales densas**

El modelo secuencial denso recibe en su capa de entrada la información de RSSI de cada WAP de la base de datos, por lo tanto, el número de neuronas de la capa de entrada de la red es igual al número de WAP y el número de neuronas en la capa de salida depende del tipo de problema bajo análisis. En clasificación el número de neuronas debe ser igual al número de clases de la base de datos y en regresión debe ser igual al número de atributos de salida, en el caso de coordenadas se utilizan dos salidas x y y.

El número de capas ocultas y el número de neuronas por cada capa y la función de activación, son hiperparámetros ajustados por el grupo de trabajo. Luego de crear y compilar el modelo de predicción, se realiza el entrenamiento en esta etapa se definen los hiperparámetros que el algoritmo usa para reducir el error de predicción como optimizador, LR y el número de épocas, una vez se obtiene el modelo entrenado se realiza el proceso de validación y se comparan los resultados predichos con los reales para evaluar las métricas de desempeño, en la Figura 3-2 se presenta el pseudocódigo DSNN.

```
INICIO
    Importar ('Librerías')
    Cargar ('Base de Datos')
    Proceso ('Tratamiento de datos')
            Separar Registros ('RegEntrenaiento', 'RegValidacion', 'ResReales')
    Repetir
          Proceso ('Crear modelo')
                  Definir ('Arquitectura')
                  Definir ('Funciones de activación')
          Proceso ('Compilar modelo')
                  Definir ('Optimizador y LR')
          Proceso ('Entrenamiento')
                  Definir ('Número de épocas')
                  Obtener ('Modelo entrenado')
          Proceso ('Validación')
                  Predecir ('Modelo entrenado'+'RegValidacion')
                  Obtener ('ResPredichos')
                  Comparar ('ResPredichos', 'ResReales')
                  Obtener ('Exactitud')
                  Obtener ('Precisión')
                  Obtener ('Tiempo de procesamiento')
     Hasta que ('Exactitud', 'Precisión'='Aceptable')
FIN
```

Figura 3-2: Pseudocódigo DSNN.
Por los autores.

- **Redes neuronales convolucionales**

Después de realizar la implementación del modelo secuencial denso, se implementa un modelo convolucional, por lo cual se convierten los registros de cada base de datos en imágenes usando el método *DeepInsight* [64], con este método se convierten los registros de un vector en una imagen de tamaño $p \times p$, donde p representa el número de pixeles. Los hiperparámetros usados en las capas de convolución son el tamaño del filtro, el tipo de muestreo y su dimensión, una vez la información ha pasado por las capas de convolución, se dirige a una red densa encargada de proporcionar las coordenadas de salida. En la etapa de entrenamiento se definen los mismos hiperparámetros del enfoque secuencial denso. En la Figura 3-3 se muestra el pseudocódigo CNN, la construcción de este es similar a DSNN, puesto que son algoritmos de redes neuronales.

```
INICIO
    Importar ('Librerías')
    Cargar ('Base de Datos')
    Proceso ('Tratamiento de datos')
            Separar Registros ('RegEntrenaiento', 'RegValidacion', 'ResReales')
            Convertir en Imágenes ('RegEntrenaiento', 'RegValidacion')
    Repetir
          Proceso ('Crear modelo')
                  Definir ('Arquitectura')
                  Definir ('Tamaño de filtros')
                  Definir ('Funciones de activación')
          Proceso ('Compilar modelo')
                  Definir ('Optimizador y LR')
          Proceso ('Entrenamiento')
                  Definir ('Número de épocas')
                  Obtener ('Modelo entrenado')
          Proceso ('Validación')
                  Predecir ('Modelo entrenado'+'RegValidacion')
                  Obtener ('ResPredichos')
                  Comparar ('ResPredichos', 'ResReales')
                  Obtener ('Exactitud')
                  Obtener ('Precisión')
                  Obtener ('Tiempo de procesamiento')
     Hasta que ('Exactitud', 'Precisión'='aceptable')
FIN
```

Figura 3-3: Pseudocódigo CNN.
Por los autores.

3.2. ENTRENAMIENTO DE ALGORITMOS

El entrenamiento consiste en ajustar los hiperparámetros y parámetros de los algoritmos hasta obtener modelos de predicción[1], encargados de realizar nuevas predicciones. El ajuste de los hiperparámetros determina el comportamiento de los algoritmos, el proceso de ajuste se realiza múltiples veces hasta obtener los hiperparámetros adecuados.

3.2.1. Entrenamiento algoritmo K vecinos más cercanos

El algoritmo KNN utiliza dos hiperparámetros: el número de los k vecinos más cercanos y el criterio de decisión. Para escoger el parámetro k por lo general se define un número impar mayor a las clases de predicción y se aumenta su valor hasta llegar al valor de k con mejores resultados. El criterio de decisión del algoritmo KNN depende del problema bajo análisis, en casos de clasificación como predicción de edificio o piso, se escoge el criterio por votación, y en casos de regresión para

[1] Un modelo de predicción es la configuración de hiperparámetros y parámetros, con la cual el algoritmo obtuvo un desempeño aceptable en la etapa de entrenamiento.

predecir las coordenadas se elige el criterio de distancia media, los valores de k y el criterio de decisión de cada problema se presentan en la Tabla 3-1.

Problema de predicción	Número de clases	Valores de k	Criterio de decisión
Predicción de edificio	3	5, 7, 9, 11	Votación
Predicción de piso	5	7, 9, 11, 15	Votación
Predicción de coordenadas UJI	-	1, 3, 5, 9, 11, 15	Valor medio
Predicción de coordenadas RSSI *Dataset*	-	1, 3, 5, 9, 11, 15	Valor medio

Tabla 3-1: Hiperparámetros algoritmo KNN.
Por los autores.

En la Tabla 3-1 se muestran los valores de k usados para la predicción de edificio o piso, en la base de datos UJI se inicia con un valor de k impar mayor al número de clases, y se incrementa tres veces de forma secuencial. En el caso de la predicción de coordenadas se establece un valor inicial de k igual a 1 y se incrementa paulatinamente, ya que no se cuenta con un número de clases definidas.

3.2.2. Entrenamiento algoritmo redes neuronales

Los algoritmos NN usan diferentes hiperparámetros, que influyen en su entrenamiento y en la construcción de un modelo final con valores optimizados para establecer el mejor modelo de predicción.

- **Entrenamiento redes neuronales densas**

En DSNN se definen 6 dos hiperparámetros con un valor fijo, estos se presentan en la Tabla 3-2. Las funciones de activación se definen para cada capa, las capas ocultas usan la función Relu, por su capacidad de procesamiento y adaptación a diferentes tipos de registros y problemas, la capa de salida usa una función de activación que depende del tipo de problema a solucionar, en casos de clasificación como la predicción de edificio o piso se usa la función Softmax, y para los problemas de regresión como predecir coordenadas la función *Sigmoid,* la cual brinda una salida lineal entre 0 y 1.

La función de costo se encarga de calcular el error del modelo de predicción y se ajusta al número de clases y al problema bajo análisis. La entropía cruzada es una función utilizada para reflejar la precisión de pronósticos probabilísticos y es ampliamente usada en problemas de clasificación con un número de clases pequeño, por lo cual se usa como función de costo para la predicción de edificio y piso. El error cuadrático medio es una función de costo bastante usada en problemas de regresión, por su versatilidad y gran desempeño con datos poco

ruidosos, debido a estas características es elijada como función de costo para la predicción de coordenadas de ubicación [65].

Problema de predicción	Función de activación	Función de costo
Predicción de edificio	Relu, Softmax	Entropía cruzada
Predicción de piso	Relu, Softmax	Entropía cruzada
Predicción de coordenadas UJI	Relu, Sigmoid	Error cuadrático medio
Predicción de coordenadas BDRSSI	Relu, Sigmoid	Error cuadrático medio

Tabla 3-2: Hiperparámetros fijos algoritmo NN.
Por los autores.

Para elegir un modelo de predicción, primero se deben evaluar diferentes valores de sus hiperparámetros. En DSNN se decide ajustar cuatro hiperparámetros, presentados en la Tabla 3-3. Se definen valores iniciales para cada hiperparámetro, en el caso de la arquitectura de red se usa la regla de la pirámide geométrica para definir el número de neuronas en la capa oculta; el optimizador y LR se definen en conjunto iniciando con el método Adam y el valor de LR de 0.01, estos valores son configurados inicialmente por las librerías de Python para ML, el número de épocas se define en 300 debido al alto número de registros de la base de datos y a la capacidad de procesamiento del sistema [66].

Las pruebas y resultados se obtienen ajustando el valor de cada hiperparámetro. La arquitectura de red se ajusta añadiendo capas ocultas y aumentando el número de neuronas usando la regla de la pirámide geométrica, luego se cambia el tipo de optimizador y se cambia el valor de LR, los optimizadores utilizados son Adam y RMSprop debido a su buen desempeño para IPS. El valor del LR se disminuye paulatinamente para cada optimizador, por último, se aumenta el número de épocas.

Problema de predicción	Arquitectura	Épocas	Optimizador	LR
Predicción de edificio	520 : 40 : 3	300	Adam	0.1
Predicción de piso	520 : 51 : 5 520 : 111 : 23 : 5 520 : 121 : 33 : 5	300	Adam RMSprop	0.1 0.005 0.001 0.0005 0.0001
Predicción de coordenadas UJI	520 : 32 : 2 520 : 81 : 13 : 2 520 : 101 : 33 : 2 520 : 101 : 58 : 33 : 2	300 500		
Predicción de coordenadas RSSI Dataset	3 : 2 : 2 3 : 3 : 2 : 2	300 500		

Tabla 3-3: Hiperparámetros algoritmo NN.
Por los autores.

- **Entrenamiento redes neuronales convolucionales**

A partir de los resultados obtenidos en las métricas de desempeño del modelo secuencial denso, se realiza la implementación del enfoque convolucional. Este enfoque utiliza la configuración de hiperparámetros de mejor desempeño en DSNN, además es necesario definir una serie de hiperparámetros, los cuales están consignados en la Tabla 3-4.

Este modelo se aplica en la predicción de coordenadas en las bases de datos seleccionadas con sus respectivas configuraciones de hiperparámetros. La transformación de los registros a imágenes se ajusta de acuerdo al número de pixeles, se establece un tamaño de 50 x 50 para no saturar el sistema debido a su alta complejidad, el tamaño del filtro y muestreo se configuran para cada capa de convolución y el valor de estos depende del tamaño de la imagen de entrada. Los valores de la Tabla 3-4 representan la opción más viable para imágenes de un tamaño menor a 100 x 100 pixeles [67].

La arquitectura convolucional aumenta hacia la capa de salida a diferencia del modelo secuencial, con el fin de incrementar sus características. La arquitectura 32:64:128 es ampliamente utilizada por su adaptabilidad a diferentes problemas [68].

Problema de predicción	Pixeles	Tamaño de filtro	Muestreo	Arquitectura convolucional
Predicción de coordenadas	50 X 50	3 x 3	*MaxPooling* (2x2)	32 : 64 : 128

Tabla 3-4: Hiperparámetros algoritmo CNN.
Por los autores.

3.3. VALIDACIÓN DE ALGORITMOS

La validación de los algoritmos permite evaluar cada uno de los modelos de predicción detectando problemas y errores al momento de realizar una predicción, es necesario dividir la base de datos en un conjunto de datos de entrenamiento y validación. El modelo de predicción se entrena con el conjunto de datos de entrenamiento y se observa el error obtenido para el conjunto de validación. La etapa de validación permite observar el comportamiento del modelo de predicción con datos desconocidos, midiendo su desempeño con respecto a una métrica de evaluación.

3.3.1. Pruebas y resultados

De acuerdo a las bases de datos seleccionadas se tienen diferentes problemas de predicción. La base de datos UJI presenta tres tipos de etiquetas, las de edificio y piso forman parte de problemas de clasificación y la de coordenadas se analiza como un problema de regresión, por lo tanto, las pruebas se aplican a cada problema. La base de datos BDRSSI posee la etiqueta de coordenadas en diferentes escenarios, lo cual corresponde a un problema de regresión.

Las pruebas se generan para cada tipo de etiqueta, predecir edificio, piso o coordenadas usando diferentes hiperparámetros en cada algoritmo. Los resultados se obtienen realizando la predicción de los registros de validación con el modelo previamente entrenado, estos resultados se comparan con los valores reales de etiqueta de los registros y se obtienen los valores de cada métrica, este proceso se repite con diferentes hiperparámetros hasta obtener los que más se ajusten a cada problema.

En la predicción de edificio y piso se utiliza la tasa de aciertos y tiempo de procesamiento, además, se analiza la matriz de confusión para visualizar el desempeño de los algoritmos KNN y DSNN y se escoge el modelo de predicción con mejores valores en las métricas propuestas anteriormente.

El uso de la matriz de confusión ayuda a comprender el comportamiento de los algoritmos basados en clasificación, se encarga de comparar los valores reales con los valores predichos de cada clase dentro de una matriz, donde los valores de la diagonal principal representan el número de registros clasificados correctamente y los valores fuera de la diagonal corresponden a los registros clasificados de forma incorrecta.

La predicción de coordenadas en diferentes escenarios de las bases de datos UJI y BDRSSI se evalúan usando la métrica de precisión. El análisis de la precisión de

los modelos de ML propuestos se realiza mediante la desviación estándar de la distribución que más se ajuste a los resultados. Identificar el tipo de distribución ayuda a comprender el comportamiento de una variable, en este caso el error de distancia.

El ajuste de una distribución a un conjunto de datos consiste en encontrar los valores de los parámetros, con los que dicha distribución puede generar los datos bajo análisis. Existen varios métodos para determinar estos parámetros, en el trabajo de grado se aplica el método de Estimación de Máxima Verosimilitud (MLE, *Maximum Likelihood Estimation)* implementado por el módulo *scipy.stats* de Python [69], el cual posee más de 100 distribuciones entre continuas y discretas; el proceso de comparar diferentes distribuciones se realiza mediante el Criterio de Información de Akaike (AIC, *Akaike Information Criterion*) [70]. El valor de AIC se obtiene con la siguiente expresión:

$$AIC = -2\ \log(L) + \mathrm{o}\ \mathrm{u}, \qquad (28)$$

donde, o representa un valor de penalización por cada parámetro de la distribución, es recomendable usar valores desde 2.5 a 4, L es valor de verosimilitud y u el número de parámetros de cada distribución, obtenidos mediante MLE. Entre menor sea el valor de AIC mejor es el ajuste de la distribución, es decir, se selecciona la distribución con el menor valor de AIC, de la cual se calcula la desviación estándar usando los valores de sus propios parámetros, cabe mencionar que el error de distancia no toma valores negativos, por lo cual se tienen en cuenta distribuciones de dominio positivo. En la sección dos del anexo A se encuentran las distribuciones usadas en el método MLE con sus respectivas características.

A continuación, se presentan los modelos de predicción y los resultados de las métricas de precisión, exactitud y tiempo de procesamiento para la predicción de edificio, piso y posición.

3.3.2. Predicción de edificio UJI

Los resultados de la etapa de entrenamiento y validación se dividen para cada algoritmo, evaluando la exactitud de los diferentes hiperparámetros, eligiendo el modelo de cada algoritmo con mejor desempeño en dicha métrica.

- **K vecinos más cercanos**

La Tabla 3-5 contiene los resultados de los diferentes valores k para la predicción de edificio, se evalúa la tasa de aciertos y el tiempo de procesamiento del algoritmo, de acuerdo a estas métricas se selecciona el modelo con mejores resultados.

K	Tasa de aciertos (%)	Tiempo de entrenamiento (s)	Tiempo de validación (s)
5	99.72	0.0147	1.0241
7	99.72	0.0149	1.0649
9	99.54	0.0168	1.0587
11	99.54	0.0163	1.0795

Tabla 3-5: Resultados predicción edificio KNN.
Por los autores.

Los resultados de la Tabla 3-5 muestran a k igual a 5 como el modelo con mejor tasa de aciertos respecto a 9 y 11, mejorando en un 0.18% y menor tiempo de procesamiento en comparación a 7 con un tiempo de entrenamiento y validación menor, teniendo en cuenta lo mencionado anteriormente se elige el valor de k igual a 5 como modelo de predicción del algoritmo KNN.

- **Redes neuronales densas**

Los resultados del algoritmo DSNN para la predicción de edificio se presentan en la Tabla 3-6; la tasa de aciertos obtenida es del 100% y el tiempo de validación es de 0.1 s, estos resultados se obtienen con la configuración inicial de hiperparámetros, por lo cual no se consideró necesario realizar más variaciones.

Tasa de aciertos (%)	Tiempo de entrenamiento (s)	Tiempo de validación (s)
100	624.34	0.1

Tabla 3-6: Resultados predicción de edificio DSNN.
Por los autores.

- **Configuraciones con mejor desempeño**

La combinación de hiperparámetros de cada algoritmo se presenta en la Tabla 3-7, donde KNN usa un valor de k igual a 5 y DSNN utiliza los valores de configuración inicial, con los cuales se obtuvieron resultados aceptables en las métricas de desempeño.

A-KNN	A-DSNN		
k	Arquitectura	Optimizador y LR	Épocas
5	520 : 40 : 3	Adam 0.1	300

Tabla 3-7: Hiperparámetros finales para predicción de edificio.
Por los autores.

La Tabla 3-8 contiene los resultados de las métricas de desempeño, empleando los registros de la base de datos de UJI. Los resultados del modelo DSNN muestran

que los registros de validación se predicen de forma correcta con un porcentaje de 100%, sin errores de predicción, por el contrario, KNN presenta 0.28% de error.

Algoritmos	Tasa de aciertos (%)	Tiempo de entrenamiento (s)	Tiempo de validación (s)
A-KNN	99.72	0.0147	1.0241
A-DSNN	100	624.34	0.1

Tabla 3-8: Resultados predicción de edificio.
Por los autores.

Se tienen mejores resultados de DSNN frente a KNN en la tasa de aciertos obteniendo un 100% de desempeño, lo cual representa, que el modelo es capaz de predecir de forma acertada el edificio en el cual se encuentra un dispositivo móvil. El desempeño del modelo de predicción de DSNN se debe a la gran cantidad de registros de entrenamiento de cada clase.

La matriz de confusión de las Figuras 3-4 y 3-5 permite analizar visualmente los resultados de predicción obtenidos en la clasificación de edificio para KNN y DSNN, donde los porcentajes verticales indican la cantidad de registros clasificados correctamente del total de registros de ese edificio, y los porcentajes horizontales representan la cantidad de registros clasificados correctamente del total de registros predichos en ese edificio.

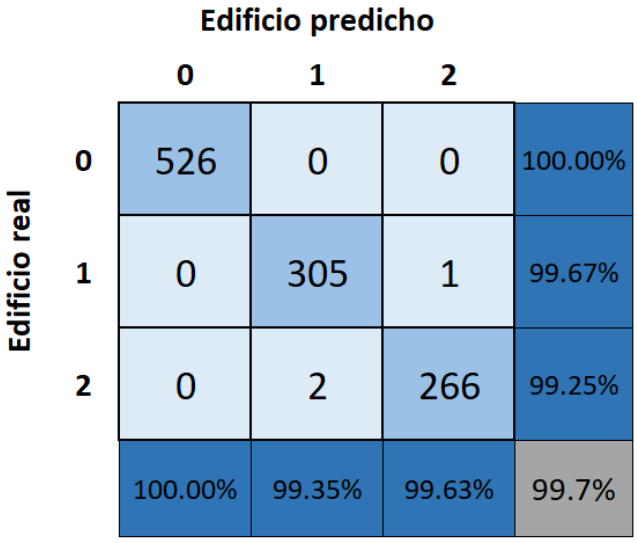

Figura 3-4: Matriz de confusión predicción edificio KNN.
Por los autores.

La Figura 3-4 muestra que en el edificio 0 todos los registros se clasificaron correctamente, en el edificio 1 el 99.35% de los registros predichos pertenecen a ese edificio y el 99.67% de registros del edificio 1 fueron predichos correctamente.

En el edificio 2 se predijeron de forma correcta el 99.25% de los registros y el 99.63% de los registros predichos en el edificio 2 están en dicho edificio.

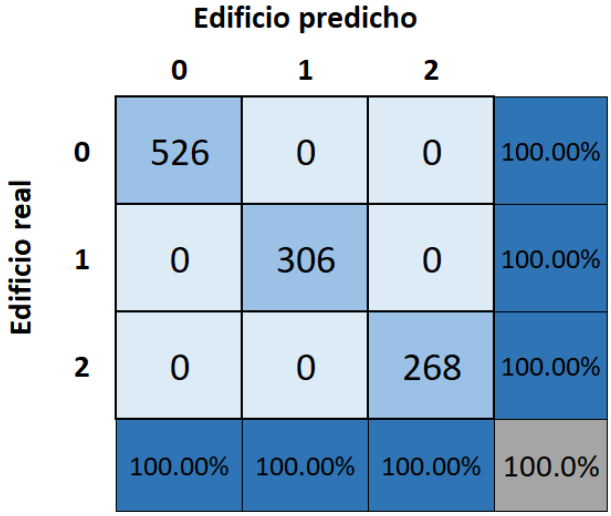

Figura 3-5: Matriz de confusión predicción edificio DSNN. Por los autores.

De acuerdo a la matriz de confusión de la Figura 3-5, los 1100 registros de validación están clasificados correctamente en cada clase. Teniendo en cuenta los resultados de exactitud y matriz de confusión, DSNN presenta un desempeño superior obteniendo mejores resultados con respecto a KNN, gracias a su mayor profundidad de aprendizaje, permitiendo clasificar de forma correcta los registros en el edificio en el cual se encuentren.

El tiempo entrenamiento es una métrica usada para medir la complejidad del algoritmo, los resultados de la Tabla 3-8, muestran a DSNN con un mayor grado de complejidad en la fase de entrenamiento, este factor depende de recursos software y hardware, es decir, el tiempo de entrenamiento disminuye si se tienen programas o equipos computacionales de mayor capacidad de procesamiento. El tiempo de validación tiene una gran importancia en la elección del algoritmo de posicionamiento, este representa el tiempo de predicción del modelo con datos desconocidos, si el tiempo es superior a 1 s el IPS no funciona en tiempo real [60], los resultados de los modelos de predicción evidencian que, a pesar de su complejidad, DSNN tiene un tiempo de validación menor a 1 s por lo cual es más idóneo en IPS de tiempo real.

3.3.3. Predicción de piso UJI

En la predicción de piso se utilizan los algoritmos de KNN y DSNN con diferentes variaciones de sus hiperparámetros, sus resultados se presentan continuación.

- **K vecinos más cercanos**

Los resultados de predicción de piso para el algoritmo KNN se presentan en la Tabla 3-9, donde se muestran las diferentes variaciones de k, el porcentaje de aciertos, tiempo de entrenamiento y validación.

k	Tasa de aciertos (%)	Tiempo de entrenamiento (s)	Tiempo de validación (s)
7	88.54	0.0141	1.0955
9	88.18	0.017	1.0154
11	89	0.0167	1.063
15	88.72	0.0149	1.008
17	89	0.0172	1.124

Tabla 3-9: Resultados predicción de piso KNN.
Por los autores.

Los resultados de k igual a 11 y 17 tienen 89% de aciertos siendo el mejor resultado con respecto a los demás valores, sin embargo, k=11 presenta tiempo de validación menor a k=17, debido a lo expuesto anteriormente, el algoritmo KNN muestra mejores resultados en las métricas de tasa de aciertos y tiempo de procesamiento con k=11.

- **Redes neuronales densas**

Para el algoritmo DSNN, se obtienen resultados de acuerdo a cuatro hiperparámetros, en la Tabla 3-10 se muestran los resultados de las métricas de tasa de aciertos y tiempo de procesamiento de diferentes arquitecturas de red.

Arquitectura		Tasa de aciertos (%)	Tiempo de entrenamiento (s)	Tiempo de validación (s)
1	520 : 51 : 5	94.54	689.73	0.128
2	520 : 111 : 23 : 5	98.54	752.55	0.130
3	520 : 121 : 33 : 5	96.54	752.55	0.197

Tabla 3-10: Resultados arquitectura predicción de piso DSNN.
Por los autores.

El mejor resultado en la tasa de aciertos se presenta en la segunda arquitectura con un valor de 98.54%, superando a las demás arquitecturas en más de 2%, en cuanto al tiempo de procesamiento, la segunda arquitectura a pesar de no poseer el menor tiempo de validación este no supera 1 s, por lo tanto, esta arquitectura se selecciona en la construcción del modelo DSNN.

Después de escoger la arquitectura de DSNN se cambia el optimizador y se ajusta el LR, los resultados de estos hiperparámetros se presentan en la Tabla 3-11.

Optimizador y LR		Tasa de aciertos (%)	Tiempo de entrenamiento (s)	Tiempo de validación (s)
Adam	0.01	98.54	752.55	0.13
	0.005	93.36	742.62	1.194
	0.001	98.36	750.54	0.146
	0.0001	97.72	742.67	0.155
RMSprop	0.01	72.54	682.62	0.147
	0.005	91.09	551.61	0.137
	0.001	97.09	498.19	0.182
	0.0001	96.91	502.48	0.1725

Tabla 3-11: Resultados optimizador y LR predicción de piso DSNN.
Por los autores.

Teniendo en cuenta la información de la Tabla 3-11, el optimizador Adam con un LR de 0.01 tiene una tasa de aciertos de 98.54%, siendo el valor más alto en relación a los demás resultados de exactitud, de igual forma el tiempo de validación es el menor de la Tabla. Los resultados obtenidos corresponden a la variación de tres hiperparámetros para DSNN, por último, se realiza la variación del número de épocas, sus resultados se consignan en la Tabla 3-12.

Épocas	Tasa de aciertos (%)	TIEMPO DE ENTRENAMIENTO (s)	TIEMPO DE VALIDACIÓN (s)
300	98.54	752.55	0.13
500	98.45	1102.46	0.197

Tabla 3-12: Resultados épocas predicción de piso DSNN.
Por los autores.

El número de épocas se establece inicialmente en 300, dando como resultado 98.54% de aciertos, al aumentar el número de épocas el algoritmo no mejora sus resultados en tasa de aciertos, ya que un excesivo número de épocas ocasiona sobreajuste en el algoritmo DSNN. La combinación de hiperparámetros con los cuales se obtuvieron mejores resultados en exactitud y tiempo de procesamiento corresponden a la configuración inicial planteada.

- **Configuraciones con mejor desempeño**

Después de realizar diferentes variaciones de los hiperparámetros, estos se optimizan hasta obtener resultados aceptables con respecto a las métricas de

desempeño. En la predicción de piso los algoritmos KNN y DSNN, presentan un desempeño favorable con los valores de hiperparámetros de la Tabla 3-13.

A-KNN	**A-DSNN**		
k	**Arquitectura**	**Optimizador y LR**	**Épocas**
11	520 : 111 : 23 : 5	Adam 0.01	300

Tabla 3-13: Hiperparámetros finales para predicción de piso. Por los autores.

Los resultados de las métricas de desempeño de cada algoritmo se presentan en la Tabla 3-14, en la cual se evidencian los resultados de tasa de aciertos y tiempo de procesamiento de DSNN y KNN.

Algoritmos	**Tasa de aciertos (%)**	**Tiempo de entrenamiento (s)**	**Tiempo de validación (s)**
A-KNN	89	752.55	0.13
A-DSNN	98.54	750.54	0.146

Tabla 3-14: Resultados predicción de piso. Por los autores.

De los 1100 registros de validación, KNN predijo correctamente el 89% y DSNN predijo correctamente el 98.54% superando a KNN en más de un 9%, lo cual implica que DSNN brinda un mejor desempeño en la métrica de tasa de aciertos. Los resultados de predicción de cada piso se presentan en las matrices de confusión de KNN y DSNN, en las Figuras 3-6 y 3-7.

La matriz de confusión de KNN muestra la clasificación de los registros de cada piso, en el piso 0 el 69.19% de los registros predichos se tomaron en ese piso y el 97.71% de registros del piso 0 se clasificaron correctamente, en el piso 2 el 95.2% de los registros predichos corresponden a ese piso, sin embargo, solo se predijo correctamente el 84.31% de los registros totales; de los registros clasificados en el piso 3 solo el 78.82% pertenecen realmente a ese piso y el 97.56% de los registros del piso 3 están clasificados de forma correcta.

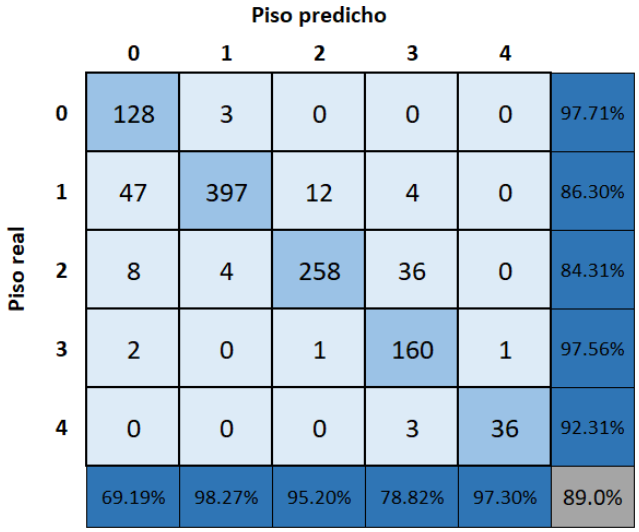

Figura 3-6: Matriz de confusión predicción piso KNN. Por los autores.

En DSNN, el 99.24% de los registros predichos en el piso 0 están en ese piso y el 100% de los registros del piso 0 son predichos correctamente, en el piso 2 se predicen correctamente 96.73% de los registros, de los registros predichos en el piso 2 el 99.33% se encuentran en ese piso; por último, en el piso 3 se clasificaron correctamente el 98.78% de los registros reales y de los registros predichos el 93.64% pertenecen a ese piso.

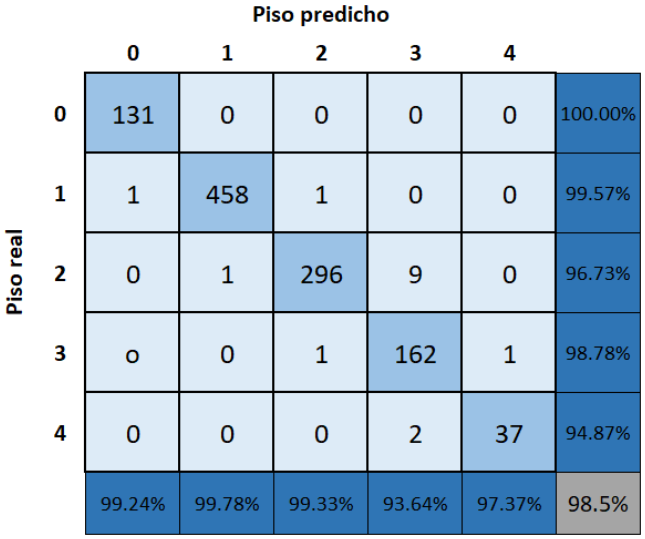

Figura 3-7: Matriz de confusión predicción piso DSNN. Por los autores.

Se evidencia de forma notoria el desempeño superior de DSNN en comparación a KNN, ya que los porcentajes verticales y horizontales de su matriz de confusión son mayores en todas las clases, esto se debe a la capacidad de aprendizaje de DSSN,

obteniendo un mejor desempeño en el momento de predecir el piso en el cual se encuentra un dispositivo móvil.

El tiempo de entrenamiento del algoritmo DSNN es mucho mayor al de KNN debido a la complejidad del algoritmo, sin embargo, el tiempo de validación de DSNN es menor a un 1 s, es decir, el modelo es capaz de predecir el piso en el que se encuentra un dispositivo móvil en tiempo real, esta característica es de gran importancia para IPS actuales. El algoritmo de DSNN es bastante eficiente en métodos de clasificación, debido a la cantidad elevada de registros de la base de datos UJI para cada etiqueta; DSNN puede ser implementado en entornos cerrados con condiciones reales y múltiples interferencias, obteniendo buenos resultados en la tasa de aciertos.

3.3.4. Predicción de coordenadas con UJI

La base de datos UJI proporciona registros con etiquetas de coordenadas, estas son usadas por los algoritmos en la predicción de posición de un dispositivo móvil. Para predecir la posición de un dispositivo móvil se usan coordenadas x, y; es decir la salida del modelo de predicción indica la ubicación de un dispositivo móvil con respecto al sistema de coordenadas utilizado, por lo general, este sistema es propio de cada IPS y se define en la toma de registros de la BD.

La predicción de coordenadas es un problema de regresión. La exactitud representa la distancia promedio a la que se encuentran las coordenadas predichas de las reales. La evaluación de la métrica de precisión se realiza mediante la desviación estándar de la distribución que más se ajuste al error de distancia de los registros de validación, la elección de la distribución se realiza con el método MLE, en el cual se analiza la precisión de las 5 distribuciones con menor valor de AIC. El tiempo de entrenamiento y validación se evalúan de la misma forma usada para la predicción de edificio o piso. En la sección tres del anexo A se muestran los valores de AIC para cada configuración de hiperparámetros de UJI.

Para establecer el mejor modelo de predicción en KNN y DSNN, se analizan las diferentes configuraciones de hiperparámetros, con el fin de encontrar la configuración que presente el mejor equilibrio en exactitud y precisión. En el caso del algoritmo CNN se utiliza la configuración de hiperparámetros con mejor desempeño para DSNN, por lo tanto, no se realizan más variaciones.

- **K vecinos más cercanos**

La evaluación de las métricas de exactitud, precisión y tiempo de procesamiento para el algoritmo KNN, en la predicción de coordenadas de un dispositivo móvil en

base a sus coordenadas se muestran en la Tabla 3-15. Se observa que la exactitud de k=5 es 11.5 m, siendo menor con respecto a los demás resultados.

k	Exactitud (m)	Distribución	Precisión (m)	Tiempo de entrenamiento (s)	Tiempo de validación (s)
1	11.75	Mielke	9.43	0.0196	1.728
3	11.19	Betaprime	10.48	0.0544	1.851
5	11.02	Betaprime	10.33	0.0215	1.833
9	11.08	Lognorm	11.27	0.0144	1.839
11	11.16	Betaprime	11.81	0.0189	1.845
15	11.38	Betaprime	12.39	0.0166	1.847

Tabla 3-15: Resultados predicción de coordenadas KNN UJI.
Por los autores.

Después de obtener los resultados de error de distancia, se aplica el método MLE para encontrar la distribución que más se ajuste según el AIC y el valor de precisión. Los mejores resultados de precisión se alcanzaron con la distribución *Mielke* y Beta Prima con k=1 y k=5 respectivamente. Los resultados de precisión ubican a k=1 con 9.43 m como el mejor valor, superando a k=5 en 0.9 m.

Para elegir el k con mejor desempeño se analiza la exactitud y precisión en conjunto, en este caso se establece a k=1 como el modelo adecuado en la predicción de posición, puesto que la precisión es inferior a 10 m y en cuanto a exactitud aumenta en 0.73 m con respecto al mejor valor. El tiempo de validación de KNN es superior a 1 s, por lo tanto, este algoritmo no es viable en la implementación de IPS en tiempo real.

- **Redes neuronales densas**

En el algoritmo DSNN, se realiza la variación de cuatro hiperparámetros, se elige el valor de cada hiperparámetro con mejores resultados en las métricas de desempeño. Inicialmente se ajusta la arquitectura de red con la cual se obtienen los resultados de la Tabla 3-16.

ARQUITECTURA		EXACTITUD (m)	DISTRIBUCIÓN	PRECISIÓN (m)	TIEMPO DE ENTRENAMIENTO (s)	TIEMPO DE VALIDACIÓN (s)
1	520 : 32 : 2	14.57	Invgauss	10.31	637.11	0.146
2	520 : 81 : 13 : 2	12.25	Recipinvgauss	8.93	682.47	0.15
3	520 : 101 : 33 : 2	11.10	Invgauss	8.73	671.31	0.15
4	520 : 101 : 58 : 33 : 2	10.70	Fatiguelife	8.80	742.54	0.15

Tabla 3-16: Resultados arquitectura predicción de coordenadas DSNN UJI.
Por los autores.

De acuerdo a la Tabla 3-16, la exactitud mejora al aumentar el número de capas ocultas y neuronas, ya que el uso de arquitecturas más robustas genera un aprendizaje más profundo, la cuarta arquitectura tiene una exactitud de 10.7 m siendo el menor valor en comparación a las demás, en el caso de precisión la tercera arquitectura posee el mejor resultado con 8.73 m, sin embargo, su valor de exactitud es de 11.10 m. Para determinar el modelo con mejor desempeño se tiene en cuenta la exactitud y precisión, por lo tanto, la arquitectura 4 presenta una relación más estable entre estas métricas, ya que su valor de precisión de 8.80 m, aumenta en 0.07 m, en comparación a la arquitectura 3 y su valor de exactitud mejora en 0.4 m.

El optimizador y LR se definen después de elegir la arquitectura. Los resultados de estos hiperparámetros se presentan en la Tabla 3-17, los resultados de la primera fila de la tabla corresponden a la configuración inicial de optimizador y LR para DSNN.

Optimizador y LR		Exactitud (m)	Distribución	Precisión (m)	Tiempo de entrenamiento (s)	Tiempo de validación (s)
Adam	0.01	10.70	Fatiguelife	8.80	742.54	0.15
	0.005	14.26	Betaprime	9.82	742.49	0.18
	0.001	13.49	Betaprime	9.65	690.91	0.15
	0.0005	11.68	Betaprime	9.92	742.59	0.15
	0.0001	11.79	Betaprime	9.32	710.78	0.15
RMSprop	0.01	12.87	Invweibull	9.00	502.66	0.17
	0.001	15.93	Invgauss	12.26	502.95	0.17
	0.005	12.67	Exponweib	11.63	502.77	0.16
	0.0005	11.05	Lognorm	8.57	502.65	0.13
	0.0001	11.65	Exponweib	8.70	502.72	0.15

Tabla 3-17: Resultados optimizador y LR predicción de coordenadas DSNN UJI.
Por los autores.

Los mejores resultados de exactitud y precisión se obtienen en la configuración inicial y RMSprop con LR de 0.0005; la exactitud de la configuración inicial es de

10.70 m mejorando en 0.35 m con respecto a RMSprop, no obstante, la precisión en RMSprop es la mejor con un valor de 8.57 m, disminuyendo en 0.23 m en comparación al resultado de Adam con LR de 0.01. Por lo tanto, se define a RMSprop como optimizador con un LR de 0.0005, esto se debe a que presenta una mejor relación entre los resultados de exactitud y precisión. El tiempo de validación de las múltiples configuraciones no supera 1 s, por lo tanto, no influye de forma significativa en la selección de hiperparámetros.

Después de seleccionar los hiperparámetros arquitectura de red, optimizador y LR se aumenta el número de épocas, en la Tabla 3-18 se presentan los resultados. El aumento de épocas a 500 muestra que la exactitud mejora en 0.43 m, sin embargo, la precisión aumenta en 0.53 m, por lo cual no se considera viable aumentar el número de épocas.

ÉPOCAS	EXACTITUD (m)	DISTRIBUCIÓN	PRECISIÓN (m)	TIEMPO DE ENTRENAMIENTO (s)	TIEMPO DE VALIDACIÓN (s)
300	11.05	Lognorm	8.57	502.65	0.13
500	10.62	Fatiguelife	9.10	982.04	0.16

Tabla 3-18: Resultados épocas predicción de coordenadas DSNN UJI.
Por los autores.

Obtener buenos resultados de exactitud no implica que el modelo tenga un mejor desempeño, por lo tanto, se evalúa la métrica de precisión para determinar la dispersión de los datos y medir la confiabilidad del modelo de predicción.

- **Configuraciones con mejor desempeño**

En la Tabla 3-19, se presentan la configuración de hiperparámetros para KNN, DSNN Y CNN, con las cuales se obtuvo el mejor desempeño en la predicción de coordenadas.

Base de datos	A-KNN	A-DSNN			A-CNN	
	k	Arquitectura	Optimizador y LR	Épocas	Tamaño del filtro	Muestreo
UJI	1	520 : 101 : 58 : 33 : 2	RMSprop 0.0005	300	3 x 3	*MaxPooling* (2 x 2)

Tabla 3-19: Hiperparámetros finales para predicción de coordenadas UJI.
Por los autores.

En la Tabla 3-20 se presentan los resultados para la base de datos UJI con valores de exactitud, precisión y tiempo de procesamiento, estos representan el desempeño de las métricas para cada modelo de predicción.

Algoritmo	Exactitud (m)	Distribución	Precisión (m)	Tiempo de entrenamiento (s)	Tiempo de validación (s)
A-KNN	11.75	Mielke	9.43	0.0196	1.72
A-DSNN	11.05	Lognorm	8.57	502.65	0.13
A-CNN	8.39	Lognorm	12.37	3434.31	0.44

Tabla 3-20: Resultados predicción de coordenadas UJI.
Por los autores.

La exactitud de KNN y DSNN tiene un valor de 11.75 m y 11.05 m respectivamente, estos resultados se deben a las características de la BD, ya que fue creada en un escenario de gran área y con baja densidad de puntos de RM, además, se utiliza la estructura inalámbrica existente basada en WIFI de un campus universitario. Para obtener mejores resultados en exactitud se emplea el algoritmo CNN, obteniendo un resultado de 8.39 m, este valor se considera alto en comparación a IPS actuales[1], sin embargo, si se toman en cuenta los trabajos relacionados con la base de datos UJI los valores de exactitud se encuentran dentro de un rango de 8 a 12 m [30] [53] [72]. En UJI el modelo CNN supera a KNN y DSNN en más de 2.5 m, por lo tanto, el uso de algoritmos ML más robustos y con características de implementación más complejas mejoran el desempeño de la exactitud.

El tiempo de validación no debe superar 1 s para realizar predicciones en tiempo real, esta condición se cumple en DSNN y CNN, siendo estos los modelos más adecuados para implementar un IPS. Los resultados de exactitud y tiempo de validación de los algoritmos de ML no permiten determinar un modelo de predicción adecuado, por lo cual se usa la métrica de precisión para medir su calidad y confiabilidad.

Teniendo en cuenta el análisis de precisión de los modelos seleccionados se presentan las gráficas de las distribuciones que se ajustaron a los resultados de error de distancia de cada algoritmo. La Figura 3-8 corresponde al histograma del error de distancia de KNN con las 5 distribuciones con menor AIC ordenadas de menor a mayor en la leyenda (con los nombres de las distribuciones), el eje x representa el error de distancia en rangos de 2 m y el eje y corresponde al porcentaje de registros de validación que se encuentran dentro de un determinado rango de error de distancia. En el caso de KNN la distribucion *Mielke* también conocida como Dagum posee el menor valor de AIC, además, cuenta con una precisión de 9.43 m siendo el mejor resultado.

[1] Los valores de exactitud en IPS actuales es de alrededor de 0.7 m [71]

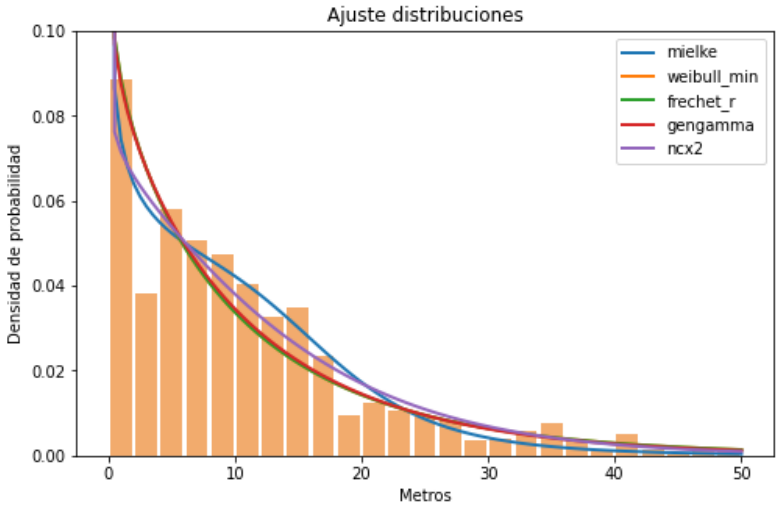

Figura 3-8: Ajuste de distribuciones KNN UJI.
Por los autores.

En DSNN se presentan las 5 distribuciones con menor AIC y el histograma de los resultados del error de distancia en la Figura 3-9, las 3 primeras tienen un AIC similar, por lo tanto, se elige la distribución con la menor precisión, siendo *Lognorm,* con un valor de precisión de 8.57 m.

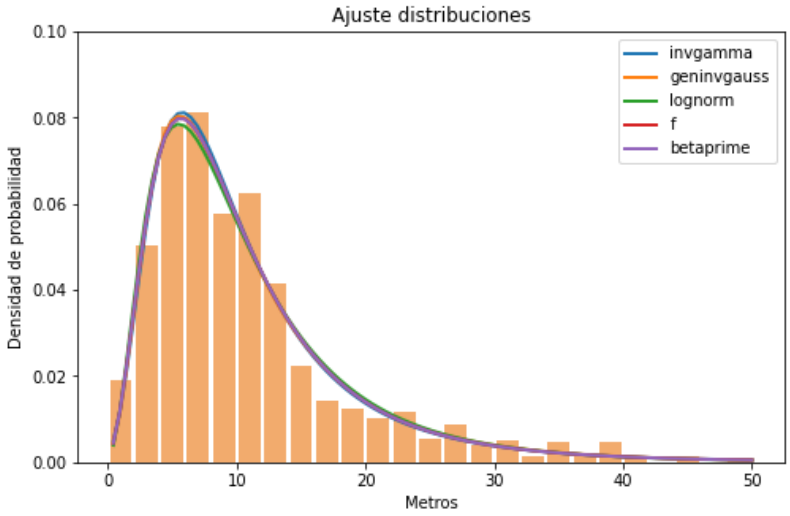

Figura 3-9: Ajuste de distribuciones DSNN UJI.
Por los autores.

Para CNN se obtienen las 5 distribuciones con menor valor de AIC, de acuerdo al histograma del error de distancia de la Figura 3-10, se observa que las distribuciones poseen un comportamiento similar y se superponen, no obstante, la distribución *Lognorm* tiene la mejor precisión con un resultado de 12.37 m.

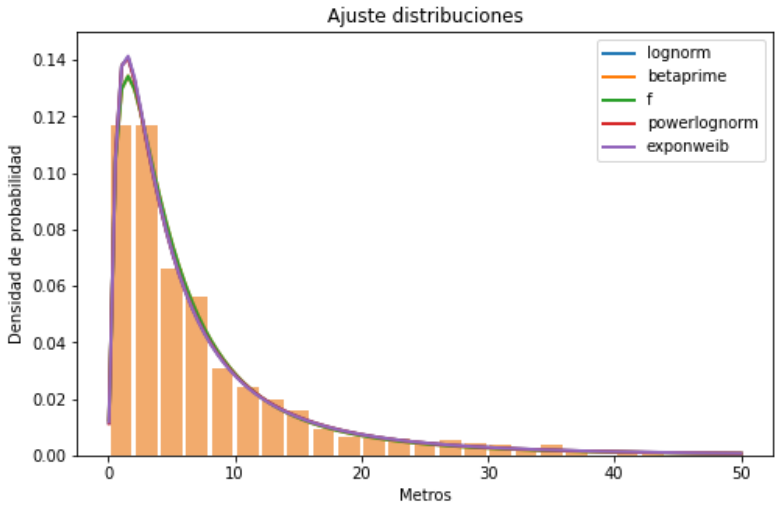

Figura 3-10: Ajuste de distribuciones CNN UJI.
Por los autores.

Teniendo en cuenta los resultados de precisión de los 3 algoritmos, DSNN mejora en relación a KNN y CNN en 0.86 y 3.8 m respectivamente, lo cual indica que obtener un buen resultado en exactitud no implica una mejor precisión. Para seleccionar el modelo de predicción adecuado se analiza las métricas de desempeño, enfocándose principalmente en obtener modelos con una alta precisión, por lo tanto, se elige a DSNN como modelo de predicción, puesto que presenta la mejor precisión con un valor de exactitud aceptable para la base de datos UJI.

3.3.5. Predicción de coordenadas con BDRSSI

La base de datos BDRSSI contiene etiquetas de coordenadas en un escenario de menor área en relación a UJI. El uso de BDRSSI permite analizar el comportamiento de los algoritmos seleccionados en un entorno diferente, con un sistema de coordenadas propio y características que facilitan el posicionamiento.

La evaluación de las métricas de desempeño se realiza de forma similar a UJI, buscando un balance entre exactitud y precisión para establecer el mejor modelo de predicción. A continuación, se presentan los resultados de diferentes configuraciones de hiperparámetros de KNN y DSNN, en el caso de CNN los resultados se analizan en conjunto con el mejor modelo de cada algoritmo. En la sección cuatro del apéndice A se presentan los valores de AIC de las diferentes configuraciones de hiperparámetros de BDRSSI.

- **K vecinos más cercanos**

Los resultados de predicción de KNN se presentan en la Tabla 3-21, en la cual se evalúan valores de k de 1 hasta 19. La exactitud disminuye hasta k=15 con un valor

de 1.47 m, luego permanece estable en 1.48 m, es decir se obtiene el mejor valor de exactitud para k=15.

k	Exactitud (m)	Distribución	Precisión (m)	Tiempo de entrenamiento (s)	Tiempo de validación (s)
1	1.82	Mielke	0.99	0.006	0.0012
3	1.59	Ncx2	0.89	0.0083	0.0023
5	1.52	Exponweib	0.83	0.0057	0.004
9	1.51	Mielke	0.78	0.0102	0.01
11	1.49	Burr	0.77	0.0093	0.004
15	1.47	Burr	0.75	0.0096	0.004
17	1.48	Burr	0.75	0.0339	0.027
19	1.48	Burr	0.75	0.0092	0.026

Tabla 3-21: Resultados predicción de coordenadas KNN BDRSSI.
Por los autores.

La precisión de KNN mejora hasta k=15 con un valor de 0.75 m, para esta métrica los resultados están por debajo a 1 m, esto indica que tiene una baja dispersión en el error de distancia y un modelo mucho más confiable. Los mejores valores de exactitud y precisión se presentan con k=15, el tiempo de validación no se toma como criterio de decisión para elegir un k, puesto que su valor no es superior a 1 s.

- **Redes neuronales densas**

En DSNN se plantean dos arquitecturas, en la Tabla 3-22 se muestran los resultados de las métricas de desempeño. La exactitud de la arquitectura 1 es de 1.25 m superando en 0.06 m a la arquitectura 2, debido a las limitadas características de entrada de la base de datos, ya que al usar una arquitectura más robusta se genera un efecto de sobreajuste, lo cual provoca un mayor error de distancia.

ARQUITECTURA		EXACTITUD (m)	DISTRIBUCIÓN	PRECISIÓN (m)	TIEMPO DE ENTRENAMIENTO (s)	TIEMPO DE VALIDACIÓN (s)
1	3 : 2 : 2	1.25	Mielke	0.56	142.49	0.24
2	3 : 3 : 2 : 2	1.31	Mielke	0.56	106.66	0.23

Tabla 3-22: Resultados arquitectura de predicción de coordenadas DSNN BDRSSI.
Por los autores.

El valor de precisión de las arquitecturas es igual de 0.56 m, por lo tanto, no influye en la selección de la arquitectura, además, el tiempo de validación no es superior a 1 s. En este caso, la selección de arquitectura se realiza tomando en cuenta únicamente el valor de exactitud, siendo la arquitectura 1 la más adecuada.

Después de seleccionar la arquitectura 1 se ajusta el optimizador y LR, los resultados de estos hiperparámetros se presentan en la Tabla 3-23. Los resultados de la configuración inicial se consignan en la primera fila.

Optimizador y LR		Exactitud (m)	Distribución	Precisión (m)	Tiempo de entrenamiento (s)	Tiempo de validación (s)
Adam	0.01	1.25	Mielke	0.56	142.49	0.24
	0.005	1.27	Burr	0.56	101.34	0.22
	0.001	2.56	Exponweib	1.28	102.20	0.14
	0.0005	2.56	Exponweib	1.27	142.29	0.16
	0.0001	1.35	Foldnorm	0.62	142.32	0.21
RMSprop	0.01	1.25	Mielke	0.53	109.83	0.15
	0.005	1.30	Mielke	0.53	142.38	0.15
	0.001	1.27	Rice	0.54	142.39	0.14
	0.0005	1.29	Burr	0.58	109.86	0.22
	0.0001	1.41	Gompertz	0.70	142.35	0.24

Tabla 3-23: Resultados optimizador y LR predicción de coordenadas DSNN BDRSSI. Por los autores.

De acuerdo a la Tabla 3-23 la mejor configuración de optimizador y LR es RMSprop con LR de 0.01, dado que tiene los mejores resultados en cuanto a precisión (0.53 m) y exactitud (1.25 m).

Por último, se incrementa el número de épocas de 300 a 500, los resultados se muestran en la Tabla 3-24, se evidencia que la exactitud y precisión aumentan cuando el número de épocas es mayor, por lo tanto, se define el número de épocas en 300.

ÉPOCAS	EXACTITUD (m)	DISTRIBUCIÓN	PRECISIÓN (m)	TIEMPO DE ENTRENAMIENTO (s)	TIEMPO DE VALIDACIÓN (s)
300	1.25	Mielke	0.53	109.83	0.15
500	1.32	Mielke	0.64	192.23	0.15

Tabla 3-24: Resultados épocas predicción de coordenadas DSNN BDRSSI. Por los autores.

- **Configuraciones con mejor desempeño**

La configuración de hiperparámetros con mejor desempeño de cada algoritmo se presentan en la Tabla 3-25 en la cual se muestra el valor k en KNN, los cuatro hiperparámetros de DSNN y los parámetros adicionales de CNN.

Base de datos	A-KNN	A-DSNN			A-CNN	
	k	Arquitectura	Optimizador y LR	Épocas	Tamaño del filtro	Muestreo
BDRSSI	15	3 : 2 : 2	RMSprop 0.01	300	3 x 3	*MaxPooling* (2x2)

Tabla 3-25: Hiperparámetros finales para predicción de coordenadas BDRSSI. Por los autores.

Los resultados de validación de la base de datos BDRSSI se exponen en la Tabla 3-26, la cual contiene los resultados del mejor modelo de predicción de cada algoritmo, las métricas se evalúan de forma similar a la base de datos UJI.

Algoritmo	Exactitud (m)	Distribución	Precisión (m)	Tiempo de entrenamiento (s)	Tiempo de validación (s)
A-KNN	1.47	Burr	0.75	0.0096	0.004
A-DSNN	1.25	Mielke	0.53	109.83	0.15
A-CNN	2.53	Mielke	1.39	322.75	0.27

Tabla 3-26: Resultados predicción de coordenadas BDRSSI. Por los autores.

El modelo DSNN alcanza un valor de exactitud de 1.25 m, mostrando resultados superiores en comparación a KNN y CNN, cabe destacar que el margen de error de exactitud no supera 3 m, puesto que los registros de entrenamiento y validación son tomados en un escenario controlado y diseñado específicamente para IPS. Los valores de exactitud para BDRSSI toman valores menores a 3 m, siendo mejores a comparación de UJI, estos resultados aún no se consideran buenos en IPS actuales, sin embargo, los trabajos relacionados con esta base de datos presentan una exactitud entre 1,5 y 3.8 m [73] [74], teniendo en cuenta lo anterior se consideran los resultados de los algoritmos como aceptables.

El tiempo de validación no supera 1 s en ninguno de los modelos de predicción, es decir, son capaces de realizar predicciones en tiempo real, cumpliendo el requerimiento de IPS actuales. Los resultados de exactitud y tiempo de validación no son suficientes para determinar el modelo de predicción para IPS, por esta razón es necesario analizar la precisión de los algoritmos.

La evaluación de precisión se determina mediante la desviación estándar de las 5 distribuciones con menor valor de AIC, en la Figura 3-11 se presenta el histograma del error de distancia y las 5 distribuciones para KNN. El mejor valor de precisión es 0.75 m obtenido con la distribución *Burr*, la cual posee el menor valor de AIC.

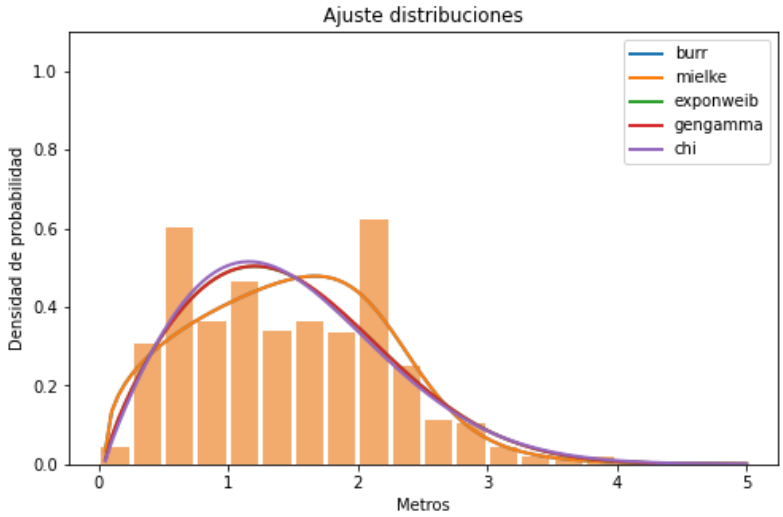

Figura 3-11: Ajuste de distribuciones KNN BDRSSI.
Por los autores.

En DSNN, la distribución *Mielke* tiene el menor valor de AIC y presenta un valor de precisión de 0.53 m, siendo el mejor resultado, en la Figura 3-12 se muestran las 5 distribuciones que más se ajustan al histograma de error de distancia.

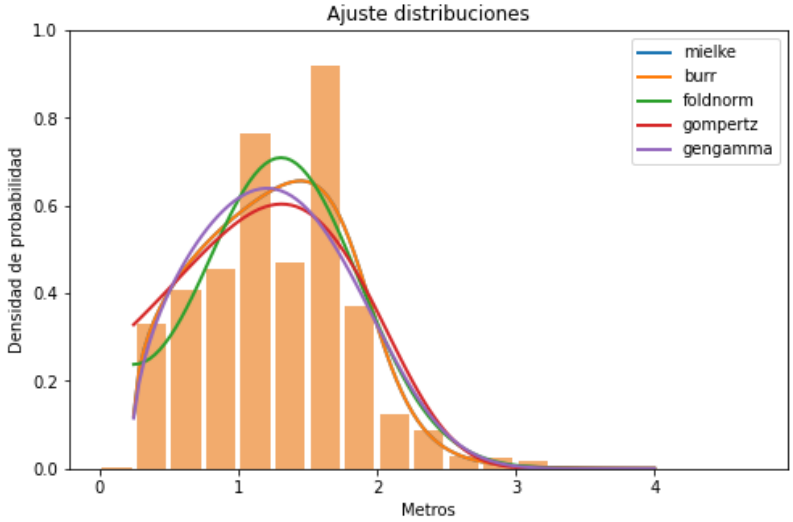

Figura 3-12: Ajuste de distribuciones DSNN BDRSSI.
Por los autores.

El histograma de error de distancia de CNN de la Figura 3-13, muestra una mayor dispersión en comparación con KNN y DSNN, esto se refleja en una precisión de 1.39 m, correspondiente a la distribución *Mielke*, la cual tiene el menor valor de AIC.

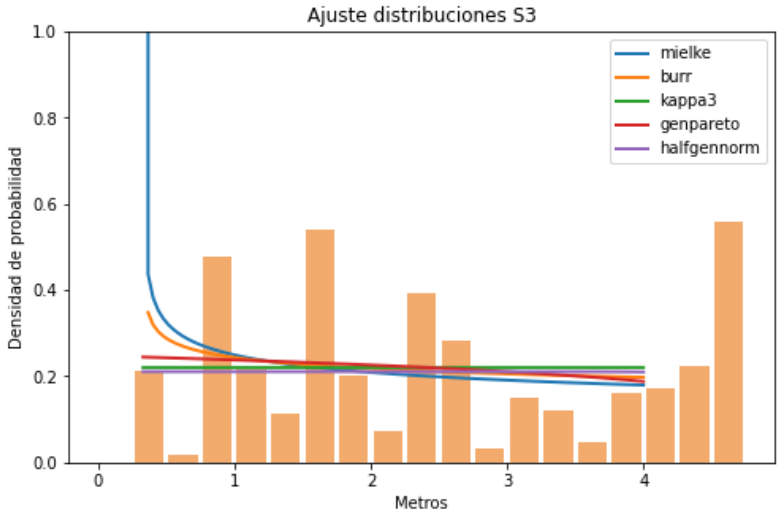

Figura 3-13: Ajuste de distribuciones CNN BDRSSI.
Por los autores.

Los resultados de precisión de los algoritmos muestran mejores valores con DSNN, superando a KNN en 0.22 m y a CNN en 0.86 m, lo cual demuestra que los resultados dependen de las características de la base de datos y del tipo de escenario. Se elige a DSNN como el modelo de predicción para BDRSSI, considerando su buen desempeño en las métricas de exactitud y precisión.

El modelo CNN para la base de datos BDRSSI, no presenta buenos resultados en las métricas de exactitud y precisión, debido a la falta de características para la conversión de registros a imágenes y el bajo número de puntos de acceso, siendo difícil obtener imágenes detalladas para realizar una buena predicción.

3.4. MODELO DE PREDICCIÓN FINAL

El desempeño de un IPS con la técnica *fingerprinting* está estrechamente relacionado con el entorno y la construcción de su base de datos, características como la densidad de RM, su posición y la cantidad de registros RSSI, son factores cruciales en la implementación de algoritmos de ML capaces de aprender de los datos y predecir una posición.

De acuerdo al análisis de resultados en las métricas de desempeño, los modelos de predicción se comportan diferente para cada BD, puesto que los algoritmos de ML crean sus modelos de predicción según el tipo de información. Los entornos de las bases de datos seleccionadas difieren en muchos aspectos, otorgando a BDRSSI características que ayudan a reducir el error de predicción, esto influye en los resultados de las métricas de desempeño, si se analizan los resultados para las dos

bases de datos la exactitud y precisión en BDRSSI es mejor a comparación de UJI, en todos los algoritmos.

En consecuencia, es válido proponer un modelo de predicción para cada base de datos, que se ajuste a sus características y cumpla con los requerimientos planteados en el trabajo de grado; para la base de datos UJI se presenta el modelo de la Figura 3-14. La predicción de edificio, piso y coordenadas se realizan de forma paralela con el fin de no incrementar el tiempo de procesamiento, además, cada red neuronal necesita la información de RSSI de los 520 WAP.

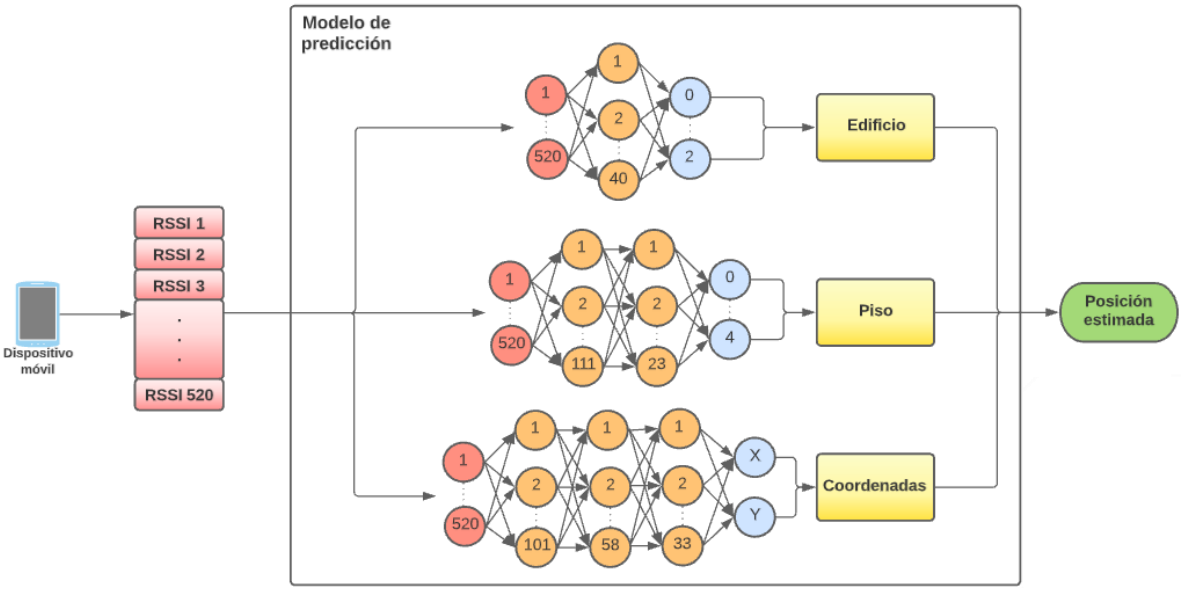

Figura 3-14: Modelo de predicción UJI.
Por los autores

Para la predicción de etiquetas de edificio y piso se elige los modelos de DSNN, debido a los resultados obtenidos en la etapa de validación en los cuales mostraron un desempeño superior a KNN en las métricas de exactitud y tiempo de validación. Para la predicción de coordenadas se utiliza el modelo DSNN, el cual a pesar de su complejidad y tiempo de entrenamiento tiene un mejor desempeño en comparación a KNN y CNN en exactitud y precisión.

Para comparar modelos de predicción en IPS, es necesario usar el mismo escenario o base de datos, para UJI se han desarrollado diversos estudios donde se evalúa el desempeño de diferentes algoritmos de ML, estos utilizan la exactitud como métrica principal, los resultados presentan una exactitud de predicción de edificio entre el 99 y 100%, de piso entre el 96 y 98%, y de coordenadas entre 8 y 12 metros, los resultados mencionados anteriormente respaldan el modelo final propuesto.

El modelo propuesto para BDRSSI se presenta en la Figura 3-15, este modelo se enfoca en la predicción de coordenadas, debido a que la base de datos no cuenta con información adicional de etiquetas; los resultados del modelo de DSNN tienen un mejor desempeño en las métricas de precisión y exactitud con respecto a KNN y CNN.

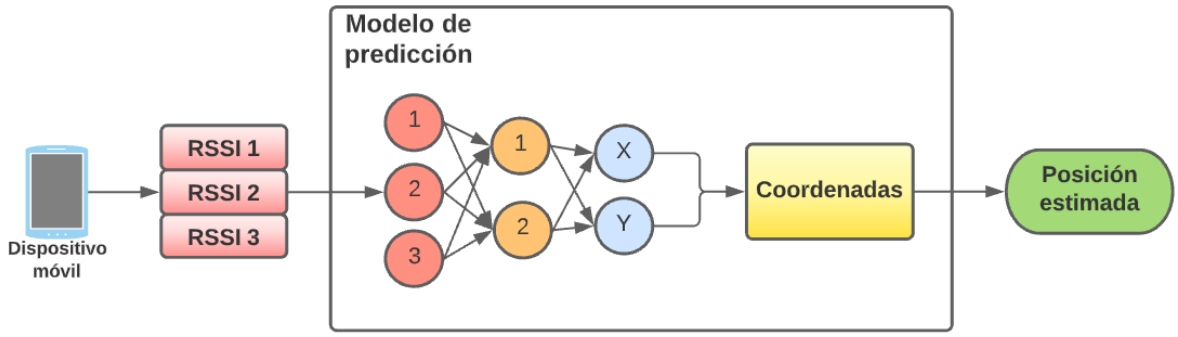

Figura 3-15: Modelo de predicción BDRSSI.
Por los autores

Resultados de exactitud para diferentes algoritmos de ML utilizando la base de datos BDRSSI, presentan un valor de exactitud para el escenario 3 entre 1.6 y 3.8 m. La exactitud del modelo propuesto se encuentra dentro de los rangos aceptables para la base de datos BDRSSI.

En diversas investigaciones sobre IPS en las cuales se usan algoritmos de ML, se plantea únicamente la exactitud como métrica de desempeño, sin evaluar la precisión, la cual es considerada un indicador clave para medir la calidad y confiabilidad en modelos de predicción, dejando muchas dudas sobre el funcionamiento de los IPS. El trabajo de grado propone la precisión como métrica principal de desempeño, evaluando su comportamiento mediante la implementación de algoritmos de ML, además, se usa KNN como punto de comparación para evaluar si el uso de algoritmos de ML más complejos como DSNN y CNN mejoran el desempeño del IPS. Los resultados muestran una mejor precisión cuando se usan algoritmos de ML de mayor complejidad, no obstante, el desempeño de los modelos de predicción depende de las características de la base de datos, ya que si estas no son suficientemente robustas el algoritmo no es capaz de aprender de forma adecuada.

4. CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se presentan las conclusiones que surgen del trabajo de grado y trabajos futuros en el campo del posicionamiento en interiores con Machine Learning.

4.1. CONCLUSIONES

A. Adaptar algoritmos ML depende de factores como la complejidad matemática, el número de parámetros, la cantidad de información y las herramientas software disponibles para su implementación. En el trabajo de grado se proponen los algoritmos KNN, DSNN y CNN adaptados a un sistema de posicionamiento en interiores usando el lenguaje de programación Python, puesto que ofrece gran variedad de librerías y funciones facilitando el procesamiento de datos y el desarrollo de modelos de ML. DSNN Y CNN necesitan de gran capacidad computacional para realizar su entrenamiento, por lo tanto, se recomienda usar unidades de procesamiento gráfico.

B. En la actualidad existen múltiples algoritmos para IPS, se adaptó a KNN, DSNN y CNN, de acuerdo a la investigación sobre algoritmos de ML para IPS. KNN tiene buenos resultados de precisión y exactitud en bases de datos con baja cantidad de registros, DSNN es uno de los algoritmos más implementados a IPS por sus buenos resultados en grandes cantidades de registros y CNN permite mejorar significativamente el error de posición al aumentar las características de la base de datos.

C. En la base de datos UJI para casos de clasificación como la predicción de edificio y piso, el algoritmo DSNN presentó resultados en tasa de aciertos de 100% y 98.54% y KNN de 99.72% y 89%, siendo DSNN el algoritmo con mejor desempeño, esto se debe a la gran cantidad de registros y un número bajo de etiquetas a clasificar, por lo tanto, se obtiene un modelo de predicción que puede ser aplicado en IPS para predecir el edificio y piso en el que se encuentra un dispositivo móvil.

D. En el trabajo de grado se optó por utilizar diferentes bases de datos para evaluar el desempeño de los algoritmos ML seleccionados en la predicción de coordenadas. Los resultados de exactitud y precisión de KNN, DSNN y CNN en la base de datos UJI presentan un margen de error de predicción de

coordenadas mayor a BDRSSI en más de 6 m, debido a que la capacidad de aprendizaje de los modelos de predicción depende directamente del tamaño del entorno, el número de puntos de acceso por área, y la densidad de RM. Los algoritmos de posicionamiento usando BDRSSI presentan mejores resultados de exactitud y precisión, ya que los escenarios son construidos estratégicamente en condiciones establecidas para el posicionamiento en interiores.

E. Los resultados del tiempo de predicción de los algoritmos DSNN y CNN son menores a 1 s en las dos bases de datos, por lo tanto, se cumple el requerimiento de tiempo establecido para predecir una posición desconocida en las diferentes configuraciones de los modelos. Disminuir el tiempo de predicción permite obtener un IPS adaptable a sistemas de navegación en interiores en tiempo real.

F. Los resultados de los algoritmos se evalúan mediante las métricas de precisión y exactitud con el fin de obtener un análisis más detallado en el momento de seleccionar el modelo de predicción adecuado para estimar la posición de un dispositivo móvil. En UJI el modelo de predicción CNN presenta una exactitud de 8.39 m, superando a KNN y DSNN. Para BDRSSI, DSNN tiene una exactitud de 1.25 m siendo mejor en comparación a los modelos KNN y CNN. Lo anterior demuestra que el uso de algoritmos de ML más complejos y con mejor capacidad de aprendizaje mejoran la exactitud de un IPS, si las características de la base de datos permiten realizar un buen entrenamiento.

G. El resultado de precisión del modelo de predicción DSNN en UJI es de 8.57 m, siendo el mejor en relación a KNN y CNN; en BDRSSI, el modelo DSNN tiene una precisión de 0.53 m, superando a los otros modelos. Los resultados de exactitud y precisión no tienen una relación proporcional ya que el error de distancia puede tener un valor medio cercano a 0 m pero una alta dispersión, esto se evidencia en los resultados de modelo CNN con UJI en la predicción de coordenadas con exactitud igual a 8.39 m y precisión igual a 12.37 m, por esta razón es indispensable evaluar la precisión de los IPS, para medir la confiabilidad del sistema al predecir la posición de un dispositivo móvil.

4.2. TRABAJOS FUTUROS

A continuación, se describen posibles trabajos futuros que pueden mejorar el desempeño de IPS con el uso de conceptos basados en Machine Learning, teniendo como base este trabajo de grado.

- Optimizar los algoritmos propuestos en el trabajo de grado utilizando filtros u otros mecanismos de regularización, que permitan mejorar las métricas de desempeño disminuyendo el margen de error para posicionamiento en interiores.

- Construir una base de datos con la técnica *fingerprinting* para posicionamiento en interiores utilizando la infraestructura y tecnología inalámbrica de la FIET, que abarque diferentes pisos y salones, con el fin de ser utilizada como referente para evaluación y comparación de nuevos algoritmos de ML.

- Diseñar nuevos algoritmos de ML para posicionamiento en interiores y evaluar su desempeño de acuerdo a las métricas precisión y exactitud utilizando las bases de datos propuestas en este trabajo de grado, la base de datos UJI y BDRSSI y comparar con los resultados obtenidos en los algoritmos de KNN, DSNN y CNN.

5. REFERENCIAS

[1] M. Cruz and J. Sandí, “Sistemas y tecnologías que facilitan el posicionamiento Indoor,” *Pensamiento Actual*, vol. 17, no. 29, pp. 132–144, Dec. 2017, doi: 10.15517/pa.v17i29.31585.

[2] F. Berrios, “Evaluación de modelos de aprendizaje automático para posicionamiento indoor utilizando bluetooth low energy,” Universidad técnica Federico Santa María, Santiago-Chile, 2018.

[3] H. Obeidat, W. Shuaieb, O. Obeidat, and R. Abd-Alhameed, “A Review of Indoor Localization Techniques and Wireless Technologies,” *Wireless Pers Commun*, Jul. 2021, doi: 10.1007/s11277-021-08209-5.

[4] R. Mautz, “Indoor Positioning Technologies,” ETH Zurich, 2012. [Online]. Available: ttps://doi.org/10.3929/ethz-a-007313554

[5] Y. Ying, T. Qingping, L. Panpan, and Y. Hai, “Study on Wifi Indoor Location Techniques Based on Android,” Dec. 2017, doi: 10.1109/ICCSEC.2017.8446775.

[6] N. Maung and W. Zaw, “Comparative Study of RSS-based Indoor Positioning Techniques on Two Different Wi-Fi Frequency Bands,” Jun. 2020, doi: 10.1109 / ECTI-CON49241.2020.9158211.

[7] J. Carranza and F. Abad, “Estudio de un sistema de posicionamiento para interiores,” Universitat Politècnica de València, 2018.

[8] J. Jiménez, O. Cánovas, and Félix Garcia, “Localización en interiores con redes de sensores mediante técnicas de fingerprinting,” Universidad De Murcia, 2019.

[9] Y. Cheng and T. Zhou, “UWB Indoor Positioning Algorithm Based on TDOA Technology,” Jan. 2020.

[10] X. Zhou, C. Xu, J. He, and J. Wan, “A Cross-region Wireless-synchronization - based TDOA Method for Indoor Positioning Applications,” Jul. 2019, doi: 10.1109/WOCC.2019.8770637.

[11] J. Piedra, “Sistema de posicionamiento móvil para interiores vía WIFI,” Universitat Oberta de Catalunya, 2016.

[12] K. Gettapola, R. Ranaweera, G. Godaliyadda, and M. Imara, “Location based fingerprinting techniques for indoor positioning,” presented at the 2017 6th National Conference on Technology and Management (NCTM), Jan. 2017. doi: 10.1109/NCTM.2017.7872849.

[13] G. Escudero, J. Hwang, and J. Park, “An Indoor Positioning Method using IEEE 802.11 Channel State Information,” *Journal of Electrical Engineering and Technology*, May 2017, doi: 10.5370/JEET.2017.12.3.1286.

[14] L. Wen *et al.*, “Survey on CSI-based Indoor Positioning Systems and Recent Advances,” Nov. 2019, doi: 10.1109/IPIN.2019.8911774.

[15] C. Yogita, I. Aleksandar, S. Aruna, and J. Sanjay, “CSI-MIMO: Indoor Wi-Fi fingerprinting system,” Oct. 2014, doi: 10.1109/LCN.2014.6925773.

[16] M. Elgwad, E. Ashry, and I. Bassen, “Wi-Fi based indoor localization using trilateration and fingerprinting methods,” *International Conference on*

Aerospace Sciences and Aviation Technology, vol. 18, no. 18, pp. 1–20, Apr. 2019, doi: 10.1088/1757-899X/610/1/012072.

[17] J. Marín, "MODELO DE PROPAGACIÓN EMPÍRICO PARA PREDICCIÓN DE PÉRDIDAS DE POTENCIA EN SEÑALES INALÁMBRICAS BAJO EL ESTÁNDAR IEEE 802.11B/G," *Revista GTI*, vol. 8, no. 20, Art. no. 20, 2009.

[18] N. Pakanon, M. Chamchoy, and P. Supanakoon, "Study on Accuracy of Trilateration Method for Indoor Positioning with BLE Beacons," Jul. 2020, doi: 10.1109/ICEAST50382.2020.9165464.

[19] E. Teoman and T. Ovatman, "Trilateration in Indoor Positioning with an Uncertain Reference Point," May 2019, doi: 10.1109/ICNSC.2019.8743240.

[20] T. Khanh, V. Nguyen, X. Pham, and E. Huh, "Wi-Fi indoor positioning and navigation: a cloudlet-based cloud computing approach," *Hum. Cent. Comput. Inf. Sci.*, Jul. 2020, doi: 10.1186/s13673-020-00236-8.

[21] Y. Yin, C. Song, M. Li, and Q. Niu, "A CSI-Based Indoor Fingerprinting Localization with Model Integration Approach," *Sensors*, Art. no. 13, Jan. 2019, doi: 10.3390/s19132998.

[22] V. Honkavirta, T. Perala, S. Ali, and R. Piche, "A comparative survey of WLAN location fingerprinting methods," presented at the Navigation and Communication 2009 6th Workshop on Positioning, Mar. 2009. doi: 10.1109/WPNC.2009.4907834.

[23] X. Wang, L. Gao, and S. Mao, "CSI Phase Fingerprinting for Indoor Localization With a Deep Learning Approach," *IEEE Internet of Things Journal*, Dec. 2016, doi: 10.1109/JIOT.2016.2558659.

[24] J. McCarthy, "Artificial Intelligence, Logic and Formalizing Common Sense," in *Philosophical Logic and Artificial Intelligence*, R. H. Thomason, Ed. Dordrecht: Springer Netherlands, 1989, pp. 161–190. doi: 10.1007/978-94-009-2448-2_6.

[25] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. CreateSpace Independent Publishing Platform, 2016.

[26] R. López de Mántaras, "Algunas reflexiones sobre el presente y futuro de la Inteligencia Artificial," 2015.

[27] L. Rouhiainen, *Inteligencia artificial, 101 cosas que debes saber hoy sobre nuestro futuro.*, Editorial Planeta. 2018.

[28] D. Hinestroza Ramírez, "El Machine Learning a través de los tiempos, y los aportes a la humanidad," Jun. 2018, [Online]. Available: https://hdl.handle.net/10901/17289

[29] J. Ang, A. Mirzal, H. Haron, and H. Hamed, "Supervised, Unsupervised, and Semi-Supervised Feature Selection: A Review on Gene Selection," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Sep. 2016, doi: 10.1109/TCBB.2015.2478454.

[30] S. Bozkurt, G. Elibol, S. Gunal, and U. Yayan, "A comparative study on machine learning algorithms for indoor positioning," in *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, Sep. 2015, pp. 1–8. doi: 10.1109/INISTA.2015.7276725.

[31] Z. Liu, X. Luo, and T. He, “Indoor positioning system based on the improved W-KNN algorithm,” in *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Mar. 2017, pp. 1355–1359. doi: 10.1109/IAEAC.2017.8054235.

[32] “The top 10 ML algorithms for data science in 5 minutes.” https://www.educative.io/blog/top-10-ml-algorithms-for-data-science-in-5-minutes

[33] X. Ge and Z. Qu, “Optimization WIFI indoor positioning KNN algorithm location-based fingerprint,” in *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Aug. 2016, pp. 135–137. doi: 10.1109/ICSESS.2016.7883033.

[34] N. Avinash, “Python Decision Tree Classification with Scikit-Learn DecisionTreeClassifier,” *DataCamp Community*, Dec. 28, 2018. https://www.datacamp.com/community/tutorials/decision-tree-classification-python

[35] A. Navada, A. N. Ansari, S. Patil, and B. A. Sonkamble, “Overview of use of decision tree algorithms in machine learning,” presented at the 2011 IEEE Control and System Graduate Research Colloquium, Jun. 2011. doi: 10.1109/ICSGRC.2011.5991826.

[36] J. Yim, “Introducing a decision tree-based indoor positioning technique,” *Expert Systems with Applications*, Feb. 2008, doi: 10.1016/j.eswa.2006.12.028.

[37] A. Norén, “Árbol de decisión en Machine Learning,” *sitiobigdata.com*, Dec. 14, 2019. https://sitiobigdata.com/2019/12/14/arbol-de-decision-en-machine-learning-parte-1/

[38] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: an update,” *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, Nov. 2009, doi: 10.1145/1656274.1656278.

[39] J. Ma, X. Li, X. Tao, and J. Lu, “Cluster filtered KNN: A WLAN-based indoor positioning scheme,” in *2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks*, Jun. 2008, pp. 1–8. doi: 10.1109/WOWMOM.2008.4594840.

[40] K. Zia, H. Iram, M. Aziz-ul-Haq, and A. Zia, “Comparative study of classification techniques for indoor localization of mobile devices,” in *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, Nov. 2018, pp. 1–5. doi: 10.1109/ATNAC.2018.8615220.

[41] Z. Wu, Q. Xu, J. Li, C. Fu, Q. Xuan, and Y. Xiang, “Passive Indoor Localization Based on CSI and Naive Bayes Classification,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1566–1577, Sep. 2018, doi: 10.1109/TSMC.2017.2679725.

[42] S. Zhang, J. Guo, W. Wang, and J. Hu, “Indoor 2.5D Positioning of WiFi Based on SVM,” in *2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS)*, Mar. 2018, pp. 1–7. doi: 10.1109/UPINLBS.2018.8559903.

[43] S. Premanand, “Beginners Guide to Support Vector Machine,” *Analytics Vidhya*, Jun. 2021. https://www.analyticsvidhya.com/blog/2021/06/support-vector-machine-better-understanding/

[44] J. B. Kristensen, M. Massanet Ginard, O. K. Jensen, and M. Shen, “Non-Line-of-Sight Identification for UWB Indoor Positioning Systems using Support Vector Machines,” in *2019 IEEE MTT-S International Wireless Symposium (IWS)*, May 2019, pp. 1–3. doi: 10.1109/IEEE-IWS.2019.8804072.

[45] B. A. Labinghisa and D. M. Lee, “Neural network-based indoor localization system with enhanced virtual access points,” *J Supercomput*, doi: 10.1007/s11227-020-03272-4.

[46] A. Cavdar and K. Türk, “Artificial Neural Network Based Indoor Positioning in Visible Light Communication Systems,” presented at the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Sep. 2018. doi: 10.1109/IDAP.2018.8620763.

[47] J. J. Hopfield, “Artificial neural networks,” *IEEE Circuits and Devices Magazine*, vol. 4, no. 5, pp. 3–10, Sep. 1988, doi: 10.1109/101.8118.

[48] B. YEGNANARAYANA, *ARTIFICIAL NEURAL NETWORKS*. PHI Learning Pvt. Ltd., 2009.

[49] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” *arXiv:1511.08458 [cs]*, Dec. 2015, Accessed: Nov. 17, 2021. [Online]. Available: http://arxiv.org/abs/1511.08458

[50] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, Aug. 2017, pp. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.

[51] M. I. C. Rachmatullah, J. Santoso, and K. Surendro, “A Novel Approach in Determining Neural Networks Architecture to Classify Data With Large Number of Attributes,” *IEEE Access*, pp. 204728–204743, 2020, doi: 10.1109/ACCESS.2020.3036853.

[52] J. Feng and S. Lu, “Performance Analysis of Various Activation Functions in Artificial Neural Networks,” *Journal of Physics: Conference Series*, Jun. 2019, doi: 10.1088/1742-6596/1237/2/022030.

[53] X. Song *et al.*, “A Novel Convolutional Neural Network Based Indoor Localization Framework With WiFi Fingerprinting,” *IEEE Access*, 2019, doi: 10.1109/ACCESS.2019.2933921.

[54] G. H. Apostolo, I. G. B. Sampaio, and J. Viterbo, “Feature selection on database optimization for Wi-Fi fingerprint indoor positioning,” *Procedia Computer Science*, Jan. 2019, doi: 10.1016/j.procs.2019.09.180.

[55] E. S. Lohan, J. Torres-Sospedra, H. Leppäkoski, P. Richter, Z. Peng, and J. Huerta, “Wi-Fi Crowdsourced Fingerprinting Dataset for Indoor Positioning,” *Data*, vol. 2, no. 4, Art. no. 4, Dec. 2017, doi: 10.3390/data2040032.

[56] P. SPACHOS, “RSSI Dataset for Indoor Localization Fingerprinting.” IEEE, Apr. 17, 2020.

[57] A. P. Guevara, S. De Bast, and S. Pollin, “Massive MIMO: A Measurement-Based Analysis of MR Power Distribution,” in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, Dec. 2020, pp. 1–6. doi: 10.1109/GLOBECOM42002.2020.9322283.

[58] G. S. Handelman *et al.*, “Peering Into the Black Box of Artificial Intelligence: Evaluation Metrics of Machine Learning Methods,” *American Journal of Roentgenology*, Jan. 2019, doi: 10.2214/AJR.18.20224.

[59] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of Wireless Indoor Positioning Techniques and Systems,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Nov. 2007, doi: 10.1109/TSMCC.2007.905750.

[60] G. Oguntala, R. Alhameed, S. Jones, J. Noras, M. Patwary, and J. Rodriguez, “Indoor location identification technologies for real-time IoT-based applications: An inclusive survey,” *Computer Science Review*, vol. 30, pp. 55–79, Nov. 2018, doi: 10.1016/j.cosrev.2018.09.001.

[61] R. Pressman, *Ingenieria del Software. Un Enfoque Practico*, 7th ed. University of Connecticut: McGraw-Hill, 2010.

[62] H. Zhang, H. Du, Q. Ye, and C. Liu, “Utilizing CSI and RSSI to Achieve High-Precision Outdoor Positioning: A Deep Learning Approach,” in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6. doi: 10.1109/ICC.2019.8761305.

[63] M. J. Nelson and A. K. Hoover, “Notes on Using Google Colaboratory in AI Education,” New York, NY, USA, Jun. 2020. doi: 10.1145/3341525.3393997.

[64] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, and T. Tsunoda, “DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture,” *Scientific Reports*, vol. 9, no. 1, Art. no. 1, doi: 10.1038/s41598-019-47765-6.

[65] N. Zhang, S.-L. Shen, A. Zhou, and Y.-S. Xu, “Investigation on Performance of Neural Networks Using Quadratic Relative Error Cost Function,” *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2930520.

[66] H. Li, J. Li, X. Guan, B. Liang, Y. Lai, and X. Luo, “Research on Overfitting of Deep Learning,” presented at the 2019 15th International Conference on Computational Intelligence and Security (CIS), Dec. 2019. doi: 10.1109/CIS.2019.00025.

[67] S. Liu and W. Deng, “Very deep convolutional neural network based image classification using small training sample size,” in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Nov. 2015, pp. 730–734. doi: 10.1109/ACPR.2015.7486599.

[68] J. Jang and S. Hong, “Indoor Localization with WiFi Fingerprinting Using Convolutional Neural Network,” presented at the 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), Jul. 2018. doi: 10.1109/ICUFN.2018.8436598.

[69] I. J. Myung, “Tutorial on maximum likelihood estimation,” *Journal of Mathematical Psychology*, Feb. 2003, doi: 10.1016/S0022-2496(02)00028-7.

[70] J. E. Cavanaugh and A. A. Neath, “The Akaike information criterion: Background, derivation, properties, application, interpretation, and refinements,” *WIREs Computational Statistics*, vol. 11, no. 3, p. e1460, 2019, doi: 10.1002/wics.1460.

[71] F. Zafari, A. Gkelias, and K. Leung, “A Survey of Indoor Localization Systems and Technologies,” *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019, doi: 10.1109/COMST.2019.2911558.

[72] M. Ibrahim, M. Torki, and M. ElNainay, “CNN based Indoor Localization using RSS Time-Series,” in *2018 IEEE Symposium on Computers and Communications (ISCC)*, Jun. 2018, pp. 01044–01049. doi: 10.1109/ISCC.2018.8538530.

[73] S. Sadowski, P. Spachos, and K. N. Plataniotis, “Memoryless Techniques and Wireless Technologies for Indoor Localization With the Internet of Things,” *IEEE Internet of Things Journal*, Nov. 2020, doi: 10.1109/JIOT.2020.2992651.

[74] N. Singh, S. Choe, and R. Punmiya, “Machine Learning Based Indoor Localization Using Wi-Fi RSSI Fingerprints: An Overview,” *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3111083.

ANEXO A: Resultados de los algoritmos de posicionamiento

A continuación, se presenta la guía de implementación y las tablas con las distribuciones y los resultados de AIC para las diferentes configuraciones de hiperparámetros para los algoritmos KNN, DSNN y CNN.

1. Guia de implementación.

Para las fases de implementación y pruebas se crean diferentes *scripts*, cada uno contiene el código necesario para ejecutar un algoritmo. Se presentan 10 scripts cada uno para cada problema y algoritmo, se realiza esto para disminuir el tiempo de procesamiento además de llevar un orden de la ejecución y resultados, el nombre de cada script se muestra en la Figura A-1, donde cada uno representa el problema de predicción, el algoritmo y la base de datos que usa.

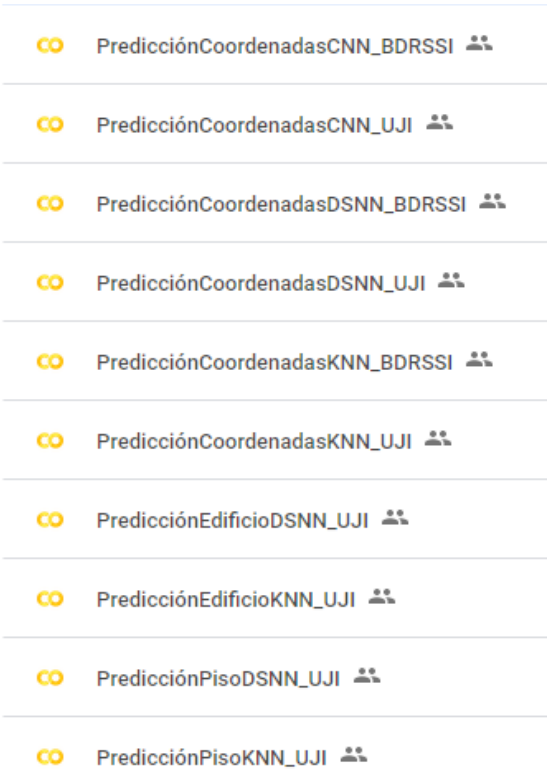

Figura A-1: Scrpts Google colab.
Por los autores.

Los archivos se presentan dentro de una carpeta en drive, la cual contiene los registros de las bases de datos ordenados, así como se muestra en la Figura A-2.

Figura A-2: Carpetas de bases de datos.
Por los autores.

Los archivos Excel de las bases de datos deben ser cargados a *colab* con la opción remarcada en azul en la Figura A-3. Para la ejecución del código no es necesario la instalación de programas adicionales.

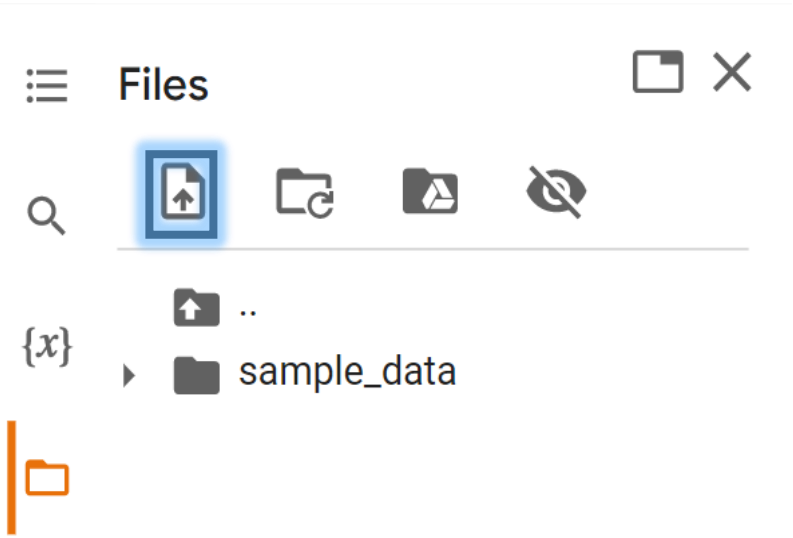

Figura A-3: Cargar bases de datos.
Por los autores.

El código de cada *script* se encuentra debidamente comentado explicando cada sección del código. Los archivos se encuentras disponibles en el link: https://drive.google.com/drive/folders/17WXvkIOQTI4w116wxbtCag8bOgqwSXLI?usp=sharing

2. Distribuciones y características para el método MLE

DISTRIBUCIÓN (PYTHON)		DISTRIBUCIÓN	No PARÁMETROS	DISTRIBUCIÓN (PYTHON)		DISTRIBUCIÓN	No PARÁMETROS
1	Alpha	Alpha	3	26	Halfgennorm	Media normal generalizada	3
2	Betaprime	Beta prima	4	27	Halflogistic	Semi-logística	2
3	Burr	Singh-Madala Tipo III	4	28	Halfnorm	Media normal	2
4	Burr12	Singh-Madala tipo XII	4	29	Invgamma	Gamma invertida	3
5	Chi	Chi	3	30	Invgauss	Gauss invertida	3
6	Chi2	Chi cuadrado	3	31	Invweibull	Weibull invertida	3
7	Erlang	Erlang	3	32	Kappa3	kappa	3
8	Expon	Exponencial	2	33	Kstwobing	Kolmogorov-Smirnov	2
9	Exponpow	Potencia exponencial	3	34	Levy	levy	2
10	Exponweib	Weibull exponencial	4	35	Loglaplace	Log-Laplace	3
11	F	Fsher-Snedecor	4	36	Lognorm	Log-normal	3
12	Fatiguelife	Birnbaum-Saunders	3	37	Lomax	Pareto de segunda clase	3
13	Fisk	Log-logística	3	38	Maxwell	Maxwell	2
14	Foldcauchy	Cauchy plegada	3	39	Mielke	Dagum	4
15	Foldnorm	Normal plegada	3	40	Nakagami	Nakagami	3
16	Frechet_r	Fréchet	3	41	Ncf	F no central	5
17	Gamma	Gamma	3	42	Ncx2	Chi2 no central	4
18	Genexpon	Exponencial generalizada	5	43	Powerlognorm	Log-normal de potencia	4
19	Gengamma	Gamma generalizada	4	44	Rayleigh	Rayleigh	2
20	Genhalflogistic	Logistica media generalizada	3	45	Recipinvgauss	Gauss invertida recíproca	3
21	Geninvgauss	Gaussiana inversa generalizada	4	46	Rice	Rician	3
22	Genpareto	Pareto generalizada	3	47	Truncexpon	Exponencial truncada	3
23	Gilbrat	Gilbrat	2	48	Wald	Wald	2
24	Gompertz	Gompertz	3	49	Weibull_min	Weibull minima	3
25	Halfcauchy	Cauchy media	2				

Tabla A-1: Distribuciones Método MLE.
Por los autores.

3. Resultados AIC y precisión UJI

- **KNN**

La Tabla A-2 presenta los resultados de precisión de las 5 distribuciones con menor AIC de los diferentes valores de k que se tuvieron en cuenta en el ajuste de hiperparámetros del algoritmo de posicionamiento basado en KNN en la predicción de coordenadas con la BD UJI.

k	Distribución	AIC	Precisión (m)
1	Mielke	7533.76	9.43
	Weibull_min	7576.79	12.61
	Frechet_r	7576.79	12.61
	Gengamma	7582.08	11.87
	Ncx2	7597.10	10.83
3	Mielke	7405.01	12.16
	Burr12	7411.72	10.54
	Betaprime	7421.74	10.48
	Lognorm	7434.84	10.91
	Gengamma	7435.81	10.38
5	Burr12	7360.69	10.59
	Mielke	7361.92	13.54
	Burr	7391.93	13.54
	Betaprime	7365.91	10.33
	F	7365.91	10.33
9	Ncf	7339.10	12.08
	Burr12	7340.65	13.08
	Betaprime	7342.12	11.79
	F	7342.12	11.79
	Lognorm	7343.31	11.27
11	Burr12	7359.63	13.25
	Betaprime	7362.45	11.81
	F	7362.45	11.81
	Burr	7362.45	19.35
	Mielke	7362.55	19.35
15	Mielke	7381.98	18.37
	Burr	7381.98	18.37
	Burr12	7382.05	13.94
	Betaprime	7387.33	12.39
	F	7387.33	12.39

Tabla A-2: Distribuciones, AIC y precisión para KNN UJI. Por los autores.

- **DSNN**

Para el algoritmo de posicionamiento basado en DSNN se presentan los resultados de precisión de las 5 distribuciones con menor AIC de los hiperparámetros que fueron ajustados con la BD UJI. En la Tabla A-3 se muestran los resultados para las diferentes arquitecturas, la Tabla A-4 contiene los resultados de las configuraciones de optimizador y LR por último en la Tabla A-5 se muestra el número de épocas con sus respectivos resultados.

ARQUITECTURA		DISTRIBUCIÓN	AIC	PRECISIÓN (m)
1	520 : 32 : 2	Lognorm	7733.74	10.60
		Invgauss	7734.73	10.31
		Exponweib	7735.36	10.47
		Geninvgauss	7735.49	10.50
		Betaprime	7735.97	10.66
2	520 : 81 : 13 : 2	Ncf	7424.00	9.32
		Invgauss	7424.74	9.11
		Fatiguelife	7425.49	9.00
		Recipinvgauss	7426.54	8.93
		Lognorm	7426.57	9.45
3	520 : 101 : 33 : 2	Invgauss	7254.69	8.73
		Exponweib	7255.65	8.77
		Lognorm	7255.86	9.12
		Fatiguelife	7255.86	8.58
		Betaprime	7256.27	8.91
4	520 : 101 : 58 : 33 : 2	Invgauss	7222.76	9.00
		Fatiguelife	7223.93	8.80
		Exponweib	7224.48	9.06
		Geninvgauss	7224.76	9.00
		Lognorm	7225.13	9.53

Tabla A-3: Distribuciones, AIC y precisión para arquitectura DSNN UJI. Por los autores.

OPTIMIZADOR Y LR		DISTRIBUCIÓN	AIC	PRECISIÓN (m)	OPTIMIZADOR Y LR		DISTRIBUCIÓN	AIC	PRECISIÓN (m)
Adam	0.01	Invgauss	7222.76	9.00	RMSprop	0.01	Burr	7405.54	10.27
		Fatiguelife	7223.93	8.80			Mielke	7405.54	10.27
		Exponweib	7224.48	9.06			Fisk	7405.69	10.58
		Geninvgauss	7224.76	9.00			Burr12	7407.08	10.08
		Lognorm	7225.13	9.53			Invweibull	7411.22	9.00
	0.005	Ncf	7707.00	10.00		0.005	Lognorm	8011.16	12.74
		Burr12	7709.21	10.06			Burr12	8012.20	14.17
		Betaprime	7710.05	9.82			Betaprime	8012.50	12.82
		F	7710.05	9.82			F	8012.50	12.82
		Exponweib	7710.41	9.77			Invgauss	8012.59	12.26
	0.001	Burr12	7637.85	9.89		0.001	Ncf	7644.60	12.38
		Burr	7640.54	11.21			Betaprime	7650.65	12.49
		Mielke	7640.34	11.21			F	7650.65	12.49
		Betaprime	7640.66	9.65			Exponweib	7650.89	11.63
		F	7640.66	9.65			Lognorm	7651.05	12.54
	0.0005	Betaprime	7431.31	9.92		0.0005	Invgamma	7161.18	9.36
		F	7431.31	9.92			Geninvgauss	7161.28	8.71
		Exponweib	7431.67	9.68			Lognorm	7161.41	8.57
		Lognorm	7431.71	10.10			F	7162.56	9.07
		Burr	7431.81	10.69			Betaprime	7162.56	9.06
	0.0001	Burr12	7386.08	9.87		0.0001	Exponweib	7369.09	8.70
		Lognorm	7386.96	9.42			Gengamma	7369.23	8.70
		Betaprime	7387.10	9.32			F	7369.46	8.73
		F	7387.10	9.32			Betaprime	7369.46	8.73
		Exponweib	7387.63	9.18			Recipinvgauss	7669.55	8.82

Tabla A-4: Distribuciones, AIC y precisión para optimizador y LR de DSNN UJI.
Por los autores.

ÉPOCAS	DISTRIBUCIÓN	AIC	PRECISIÓN (m)
300	Invgamma	7161.18	9.36
	Geninvgauss	7161.28	8.71
	Lognorm	7161.41	8.57
	F	7162.56	9.07
	Betaprime	7162.56	9.06
500	Fatiguelife	7216.82	9.10
	Invgauss	7216.83	9.36
	Geninvgauss	7218.25	9.22
	Recipinvgauss	7218.68	8.91
	Powerlognorm	7219.41	9.42

Tabla A-5: Distribuciones, AIC y precisión para épocas de DSNN UJI. Por los autores.

- **CNN**

Los resultados de precisión de las 5 distribuciones con menor valor de AIC del algoritmo de posicionamiento basado en CNN se muestran en la Tabla A-6.

DISTRIBUCIÓN	AIC	PRECISIÓN (m)
Lognorm	6682.66	12.37
Betaprime	6683.58	18.84
F	6683.58	18.84
Powerlognorm	6684.65	12.30
Exponweib	6685.34	12.29

Tabla A-6: Distribuciones, AIC y precisión para CNN UJI. Por los autores.

4. Resultados AIC y precisión BDRSSI

- **KNN**

La Tabla A-7 presenta los resultados de precisión de las 5 distribuciones con menor AIC de los diferentes valores de k en el ajuste de hiperparámetros del algoritmo de posicionamiento basado en KNN en la predicción de coordenadas con BDRSSI.

k	Distribución	AIC	Precisión (m)
1	Ncx2	3929.50	1.35
	Chi	4160.21	1.17
	Gengamma	4192.76	1.13
	Mielke	4254.00	0.99
	Exponpow	4255.18	1.23
3	Ncx2	4117.85	0.89
	Chi	4177.22	0.92
	Nakagami	4157.22	0.92
	Exponweib	4158.64	0.92
	Gengamma	4158.96	0.92
5	Chi	3903.21	0.83
	Nakagami	3903.21	0.83
	Exponweib	3903.43	0.83
	Gengamma	3903.90	0.83
	Mielke	3905.34	0.82
9	Mielke	3780.57	0.78
	Burr	3780.57	0.78
	Chi	3784.12	0.79
	Nakagami	3784.12	0.79
	Rayleigh	3784.20	0.78
11	Burr	3703.05	0.77
	Mielke	3703.05	0.77
	Exponweib	3723.41	0.78
	Gengamma	3723.51	0.78
	Chi	3726.72	0.79
15	Burr	3660.98	0.75
	Mielke	3660.98	0.75
	Exponweib	3672.60	0.76
	Gengamma	3673.19	0.76
	Chi	3673.57	0.77
17	Burr	3659.79	0.75
	Mielke	3659.79	0.75
	Rayleigh	3661.21	0.76
	Rice	3662.53	0.75
	Chi	3663.17	0.76

Tabla A-7: Distribuciones, AIC y precisión para KNN BDRSSI. Por los autores.

- **DSNN**

Para el algoritmo de posicionamiento basado en DSNN se presentan los resultados de precisión de las 5 distribuciones con menor AIC de los hiperparámetros que fueron ajustados con BDRSSI. En la Tabla A-8 se muestran los resultados para las diferentes arquitecturas, la Tabla A-9 contiene los resultados de las configuraciones de optimizador y LR por último en la Tabla A-10 se muestra el número de épocas con sus respectivos resultados.

ARQUITECTURA		DISTRIBUCIÓN	AIC	PRECISIÓN (m)
1	3 : 2 : 2	Mielke	2703.50	0.55
		Burr	2703.50	0.55
		Foldnorm	2722.10	0.55
		Rice	2735.41	0.55
		Gengamma	2740.88	0.56
2	3 : 3 : 2 : 2	Mielke	2676.40	0.55
		Burr	2676.40	0.55
		Foldnorm	2731.34	0.55
		Rice	2737.42	0.56
		Gengamma	2742.26	0.56

Tabla A-8: Distribuciones, AIC y precisión para arquitectura DSNN BDRSSI. Por los autores.

OPTIMIZADOR Y LR		DISTRIBUCIÓN	AIC	PRECISIÓN (m)	OPTIMIZADOR Y LR		DISTRIBUCIÓN	AIC	PRECISIÓN (m)
Adam	0.01	Mielke	2703.50	0.55	RMSprop	0.01	Mielke	2536.24	0.53
		Burr	2703.50	0.55			Burr	2536.24	0.53
		Foldnorm	2722.10	0.55			Foldnorm	2546.84	0.54
		Rice	2735.41	0.55			Kappa3	2605.05	0.57
		Gengamma	2740.88	0.56			Gompertz	2611.58	0.57
	0.005	Burr	2737.14	0.56		0.005	Mielke	2559.96	0.53
		Mielke	2737.14	0.56			Burr	2559.96	0.53
		Rice	2759.43	0.56			Foldnorm	2565.83	0.54
		Gengamma	2761.80	0.56			Gengamma	2635.60	0.55
		Exponweib	2762.27	0.56			Exponweib	2640.13	0.55
	0.001	Exponweib	984.11	1.28		0.001	Rice	2596.79	0.54
		F	1570.73	4.13			Exponweib	2597.27	0.54
		Nakagami	1897.19	1.77			Gengamma	2597.51	0.54
		Gengamma	2454.72	1.36			Weibull_min	2598.89	0.53
		Mielke	2515.48	1.29			Frechet_r	2598.89	0.53
	0.0005	F	1812.46	2.77		0.0005	Burr	2733.52	0.58
		Exponweib	2100.74	1.27			Mielke	2733.52	0.58
		Ncx2	2151.42	1.71			Gengamma	2738.12	0.57
		Nakagami	2643.33	1.34			Exponweib	2739.95	0.57
		Chi	2657.88	1.36			Exponpow	2745.85	0.58
	0.0001	Foldnorm	2960.69	0.62		0.0001	Kappa3	3265.91	0.71
		Mielke	2985.07	0.63			Gompertz	3284.51	0.70
		Burr	2985.07	0.63			Foldnorm	3286.63	0.70
		Gompertz	2985.78	0.63			Halfgennorm	3288.67	0.72
		Gengamma	2997.24	0.63			Gengamma	3290.54	0.72

Tabla A-9: Distribuciones, AIC y precisión para optimizador y LR de DSNN BDRSSI.
Por los autores.

ÉPOCAS	DISTRIBUCIÓN	AIC	PRECISIÓN (m)
300	Mielke	2536.24	0.53
	Burr	2536.24	0.53
	Foldnorm	2546.84	0.54
	Kappa3	2605.05	0.57
	Gompertz	2611.58	0.57
500	Mielke	3010.62	0.64
	Burr	3010.62	0.64
	Kappa3	3045.88	0.65
	Foldnorm	3020.80	0.64
	Gengamma	3021.62	0.64

Tabla A-10: Distribuciones, AIC y precisión para épocas de DSNN BDRSSI. Por los autores.

- **CNN**

Los resultados de precisión de las 5 distribuciones con menor valor de AIC del algoritmo de posicionamiento basado en CNN se muestran en la Tabla A-11.

DISTRIBUCIÓN	AIC	PRECISIÓN (m)
Mielke	5116.74	1.39
Burr	5126.16	1.34
Kappa3	5154.71	1.31
Genpareto	5219.01	1.35
Halfgennorm	5257.94	1.38

Tabla A-11: Distribuciones, AIC y precisión para CNN BDRSSI. Por los autores.