Proyección del comportamiento de enlaces en redes inalámbricas LLN mediante series de tiempo aplicando algoritmos de aprendizaje por refuerzo.

Brayan Orlando Rivera Cepeda

Trabajo de Grado para optar el Título de Ingeniero de Sistemas

Director
Pedro Javier Trujillo Tarazona
Magíster en Informática

Codirector
Héctor Niño Quiñonez
Ingeniero de Sistemas
DEA Automatique Informatique Robotique

Universidad Industrial de Santander
Facultad de Ingenierías Fisicomecánicas
Escuela de Ingeniería de Sistemas e Informática
Bucaramanga
2021

Dedicatoria

Quiero dedicar este proyecto de grado principalmente a Dios por proveerme de sabiduría y todos los recursos necesarios para su ejecución.

A mis padres Orlando y Emma, quienes han hecho todo su esfuerzo por ayudarme a cumplir esta meta y me han enseñado los valores y aptitudes de la persona que soy.

A mi nona Luz Minta, quién me ha ofrecido su apoyo incondicional en esta carrera y siempre ha estado muy atenta de mí, y me ha ayudado con su sabiduría y experiencia.

A mi hermana Ingrid y su familia, por su preocupación y buenos deseos que siempre me han demostrado.

A aquellos verdaderos amigos que me han acompañado en esta carrera universitaria desde sus inicios y todos sus momentos difíciles, percances y éxitos: Javier y Johan.

Agradecimientos

Agradezco a la Universidad Industrial de Santander por ser mi segundo hogar, por recibirme como un muchacho de 16 años de campo, y formarme como un profesional lleno de metas y aspiraciones, y unas de ellas cumplidas.

A la Escuela de Ingeniería de Sistemas por su gestión en darme educación e instalaciones de computación de calidad y otros procesos administrativos.

Al profesor Pedro por creer en mi potencial y su dedicación en formarme como uno de los mejores, y por la enseñanza en la disciplina, el orden y siempre hacer las cosas de la mejor manera posible y el conocimiento impartido en mi trabajo de grado.

A mi codirector de proyecto, Héctor Niño, por su colaboración y aporte en las asignaturas dictadas y en mi trabajo de grado.

A todos aquellos profesores y profesoras que han confiado en mis capacidades y me han enseñado todo lo que sé, y me han ayudado a construir conocimiento y una calidad de persona.

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

Pág.

Glosario

Anaconda: ambiente de trabajo local para la ciencia de datos, la cual ayuda a instalar y hacer funcionar aplicaciones y paquetes en diferentes plataformas.

Colab: término usado para referirse a Google colaboratory.

CRAWDAD: sitio web de alojamiento de conjuntos experimentales de datos sobre las comunicaciones en redes inalámbricas.

GitHub: portal web gratuito de almacenamiento de código fuente, enfocado a impartir software libre para toda la comunidad estudiantil y laboral mundial.

Google colaboratory: servicio en la nube y gratuito de compilación de software, el cual permite uso de GPUs y TPUs de Google.

GPU: componente computacional hardware, diseñado especialmente para mejorar el procesamiento gráfico.

IEEE 802.15.4: estándar especializado en redes inalámbricas de área personal, con bajas tasas de transmisión de datos.

PDM: modelamiento matemático que describe la metodología del aprendizaje por refuerzo.

Python: lenguaje de programación de alto nivel, que puede ser usado para el desarrollo web y análisis de datos, que cuenta con alta potencia en el tratamiento de grandes cantidades de datos.

Q-Learning: técnica de aprendizaje por refuerzo que permite solucionar problemas basados en decisiones secuenciales.

SARIMAX: modelo de regresión lineal que utiliza un modelo base ARIMA estacional con factores exógenos.

WFV: estrategia de validación de entrenamiento y evaluación en modelos de aprendizaje automático.

Lista de acrónimos

AR: AutoRegressive

ARIMA: Autoregressive Integrated Moving Average

ARMA: AutoRegressive Moving Average

CRAWDAD: Community Resource for Archiving Wireless Data At Dartmouth

DDPG: Deep Deterministic Policy Gradient

DL: Deep Learning

DPG: Deterministic Policy Gradient

DQN: Deep Q-Network

ETX: Expected Transmission Count

FFD: Full-Function Device

GPU: Graphics Proccessing Unit

GRU: Gated Recurrent Unit

IEEE: Institute of Electrical and Electronics Engineers

kB: KiloBytes

LLN: Low-Power and Lossy Network

LQI: Link Quality Indicator

LRWPAN: Low-Rate Wireless Personal Area Network

LSTM: Long Short-Term Memory

MA: Moving averages

MAE: Mean Absolute Error

MAPE: Mean Absolute Percentage Error

MDP: Markov Decission Process

ML: Machine Learning

MLP: Multi-Layer Perceptron

MSE: Mean-Square Error

RAM: Random Access Memory

RDPG: Recurrent Deterministic Policy Gradient

RFD: Reduced-Function Device

RL: Reinforcement Learning

RMSE: Root-Mean-Square Error

RNN: Recurrent Neural Network

RSSI: Received Signal Strength Indicator

SARIMAX: Seasonal Auto-Regressive Integrated Moving Average with eXogenous factors

SGA: Stochastic Gradient Ascent

WFV: Walk Forward Validation

Resumen

Título: Proyección del comportamiento de enlaces en redes inalámbricas LLN mediante series de tiempo aplicando algoritmos de aprendizaje por refuerzo.[*]

Autor: Brayan Orlando Rivera Cepeda[**]

Palabras Clave: IEEE 802.15.4, series de tiempo, aprendizaje por refuerzo, ARIMA, Monte Carlo, DDPG, RDPG, predicciones en series de tiempo.

Descripción:

Algunos parámetros de enlaces de las redes IEEE 802.15.4 se pueden caracterizar y estudiar mediante series de tiempo, por lo que resulta factible proyectar estos parámetros, como lo son el LQI y el RSSI para evaluar el comportamiento de los enlaces. En este reporte se presenta el uso de algoritmos de aprendizaje por refuerzo para predecir el comportamiento de parámetros de enlaces en redes LLN IEEE 802.15.4 mediante series de tiempo. Se incluyen algunos fundamentos sobre las series de tiempo, algoritmo ARIMA, algoritmos de aprendizaje por refuerzo: Monte Carlo, Monte Carlo con acotación, DDPG y RDPG, métricas de error utilizadas, de enlaces en las redes LLN IEEE 802.15.4, también se muestran detalles alrededor de la estructura y tratamiento de los conjuntos de datos, experimentación, resultados y evaluación. Se concluye que la implementación del algoritmo DDPG aplicando WFV obtuvo los mejores resultados en LQI y RSSI; y para RSSI, este algoritmo DDPG generó las métricas de error más bajas de la experimentación. La utilización de WFV en las implementaciones de ARIMA es poco favorable porque genera mayores métricas de error que cuando no se utiliza. De los algoritmos basados en el método Monte Carlo, el acotado obtuvo menores métricas de error. Los algoritmos RDPG y Monte Carlo obtuvieron las métricas de error más altas de la experimentación.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Pedro Javier Trujillo Tarazona. Mg en Informática. Codirector: Héctor Niño Quiñonez. Ing. de Sistemas DEA AIR.

Abstract

Title: Behavior estimation of the links in LLN wireless networks through time series forecasting applying reinforcement learning algorithms. [*]

Author: Brayan Orlando Rivera Cepeda [**]

Key Words: IEEE 802.15.4, time series, Reinforcement Learning, ARIMA, Monte Carlo, DDPG, RDPG, time series forecasting.

Description:

Some link parameters of networks IEEE 802.15.4 can be characterized and studied using time series. Therefore, it turns out feasible to project link parameters such as LQI and RSSI to assess the efficiency of a network. In this report is presented the use of reinforcement learning algorithms to foretell the behavior of link parameters through time series. This document shows the logical, theoretical and practical concepts of the time series, the ARIMA algorithm, and the implemented reinforcement learning algorithms: Monte Carlo, Monte Carlo with bounding, DDPG and RDPG, LLN 802.15.4 networks, structure and treatment of data sets, experimentation, results and evaluation. It is concluded that the implementation of the DDPG algorithm applying WFV obtained the best results in LQI and RSSI, and for RSSI study case, it obtained the lowest error metrics experimentation results in the project. WFV procedure in ARIMA implementations is unfavorable because it generates higher error metrics than when WFV is not used. Comparing the algorithms based on the Monte Carlo method (Monte Carlo and Monte Carlo with bounding), the bounded one obtained lower error metrics than the other one without bounding. The RDPG and Monte Carlo algorithms obtained the highest error metrics of the experimentation in the Project.

[*] Degree Work

[**] Mechanical-Physical Engineering Faculty. Systems and Computer Engineering School. Director: Pedro Javier Trujillo Tarazona. Mg. in Computer Science. Codirector: Héctor Niño Quiñonez. Systems Eng. DEA AIR.

Introducción

Actualmente, las redes inalámbricas son una herramienta muy importante y esencial en la vida cotidiana y en el desarrollo tecnológico, la ausencia de cables y material físico ayuda a poderlas implementar en lugares anteriormente vistos como inalcanzables. Con el auge del Internet de las cosas, cualquier dispositivo electrónico puede conectarse a una red, ya sea para notificar, recibir información, entre muchas otras cosas.

En estas redes inalámbricas existen parámetros del comportamiento de los enlaces, los cuáles se usan para evaluar el rendimiento de una red en diferentes aspectos: niveles de potencia, recepción de paquetes y niveles de señal, entre otras. Para mejorar el desempeño de estas redes, podría ser conveniente predecir el comportamiento de los parámetros de los enlaces de la red mediante series de tiempo. Métricas tales como el RSSI (*Received Strength Signal Indicator*) y el LQI (*Link Quality Indicator*) contienen información para definir este comportamiento. Para la predicción de estas métricas de enlaces se implementaron algoritmos de aprendizaje por refuerzo: Monte Carlo, Monte Carlo con acotación, DDPG (*Deep Deterministic Policy Gradient*) y RDPG (*Recurrent Deterministic Policy Gradient*); así como una implementación del algoritmo ARIMA (*Autoregressive Integrated Moving Average*), para comparar la eficacia de estos algoritmos en la predicción, usando como elemento de comparación las métricas de error RMSE (*Root-Mean-Square Error*) y MAPE (*Mean Absolute Percentage Error*).

En la sección de series de tiempo se habla acerca de los estados de su estacionalidad, y cómo se proyectan en el tiempo. En las secciones de algoritmos implementados, se describe cada uno teórica y lógicamente, su funcionamiento, diagramas de ejecución y características técnicas. En la sección llamada métricas de error se describen matemáticamente las funciones numéricas

de error aplicadas a las proyecciones generadas. Dentro de la sección Metodología se encuentra la fundamentación teórica de las redes IEEE 802.15.4, la profundización teórica y técnica del aprendizaje por refuerzo, la fuente, estructura y tratamiento de los conjuntos de datos, además de contar con la explicación detallada de las implementaciones WFV (*Walk Forward Validation)* para cada algoritmo. También en la sección Metodología se encuentran la metodología general del proyecto y la información detallada de cada algoritmo y su implementación, el lenguaje de programación y herramientas usadas en los procesos. Por último, se despliegan los resultados de acuerdo con la evaluación de los 260 experimentos ejecutados según las métricas de error, con sus implementaciones WFV y sin WFV por cada algoritmo.

1 Objetivos

1.1 Objetivo General

Evaluación de la predicción del comportamiento de los enlaces en redes LLN tipo IEEE 802.15.4 mediante series de tiempo aplicando algoritmos de aprendizaje por refuerzo.

1.2 Objetivos Específicos

Evaluar un algoritmo ARIMA con modelo SARIMAX (0, 0, 2)x(1, 1, 2, 12) para la proyección del comportamiento de los enlaces en redes IEEE 802.15.4.

Evaluar un algoritmo Monte Carlo para la proyección del comportamiento de los enlaces en redes IEEE 802.15.4.

Evaluar un algoritmo Monte Carlo con acotación en el rango de predicciones, para la proyección del comportamiento de los enlaces en redes IEEE 802.15.4.

Evaluar un algoritmo de Gradiente de Política Determinista Profunda - Deep Deterministic Policy Gradient (DDPG) para la proyección del comportamiento de los enlaces en redes IEEE 802.15.4

Evaluar un algoritmo de Gradiente de política determinista recurrente - Recurrent Deterministic Policy Gradient (RDPG) para la proyección del comportamiento de los enlaces en redes IEEE 802.15.4

2 Cuerpo del Trabajo

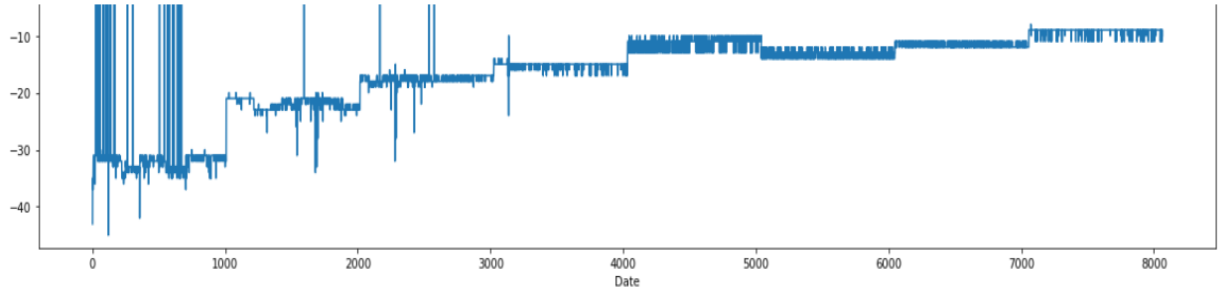
2.1 Marco Referencial

En esta sección se van a definir los conceptos lógicos, teóricos y matemáticos de las series de tiempo, su estacionalidad y proyección, el algoritmo ARIMA con modelo SARIMAX (0, 0, 2)x(1, 1, 2, 12) (Seasonal Auto-Regressive Integrated Moving Average with eXogenous factors), los algoritmos de aprendizaje por refuerzo Monte Carlo, Monte Carlo con acotación, DDPG y RDPG, además se hablará de las redes IEEE 802.15.4 y el conjunto de datos trabajado. También se muestran los conceptos de las métricas de error usadas para la evaluación en la proyección de los algoritmos.

2.1.1 Series de tiempo

Las series de tiempo son una lista de fechas, asignadas con uno o muchos valores numéricos. Es decir, es una secuencia de registros que mantiene ordenes cronológicos regulares. Las series de tiempo son un modo muy estructurado de representar datos y ver su comportamiento con respecto al tiempo, por lo que visualmente son curvas oscilantes. Pueden estar divididas de diferentes maneras: años, meses, días, horas, pero siempre es un intervalo de tiempo fijo en toda la serie (*Series de tiempo (Cadena de suministro)*, s. f.).

Figura 1. *Serie de tiempo RSSI*

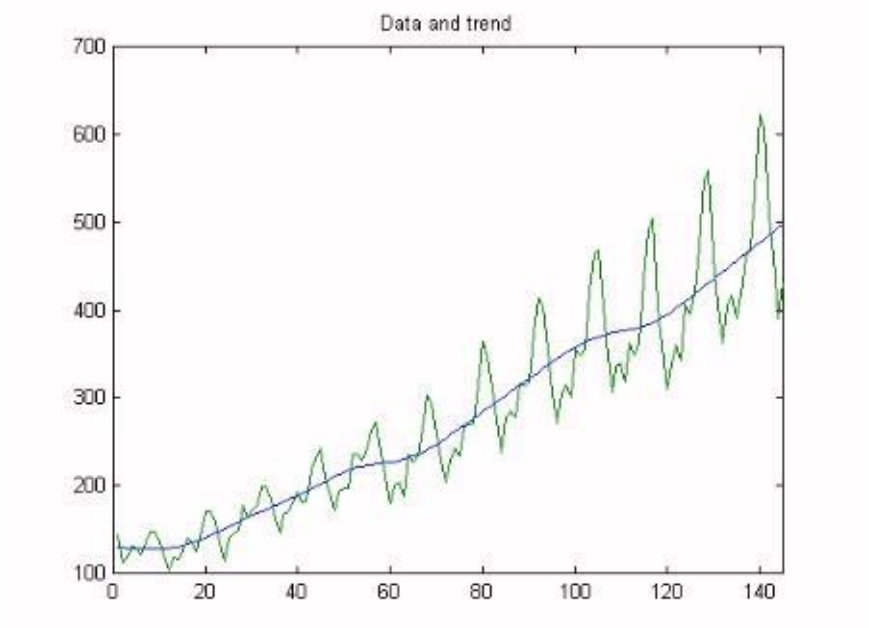


Nota: Serie de tiempo tomada del conjunto experimental de datos RSSI

Una serie de tiempo se puede caracterizar de acuerdo con sus componentes:

1. Tendencia: Es la característica de largo plazo, que muestra la base del crecimiento o decrecimiento de la serie. Si la serie se define como estacionaria, su varianza y su media no varían.

Figura 2. *Serie de tiempo con tendencia positiva*



Nota: Serie de tiempo con tendencia positiva. Tomado de: http://www.es.lancs.ac.uk/cres/captain/dhrdemo.html

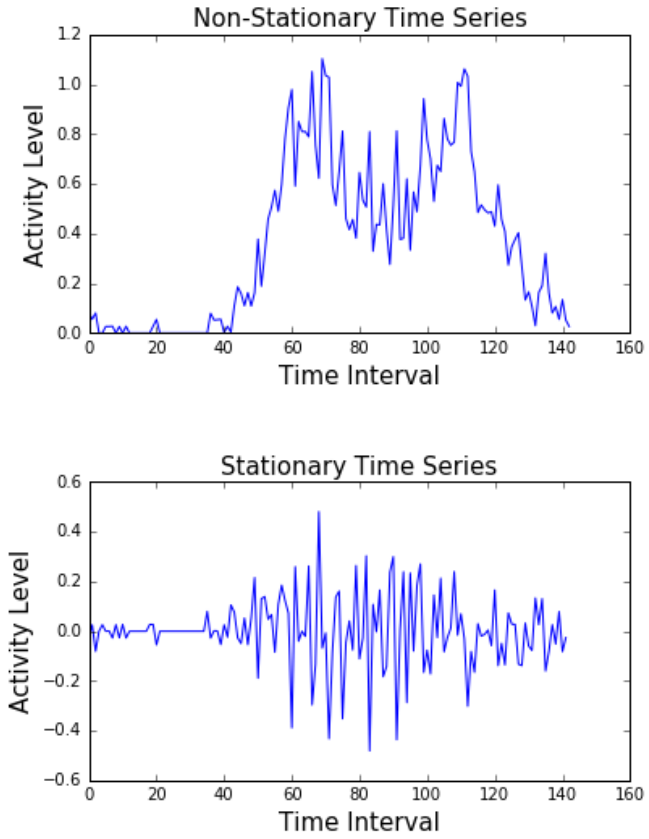
2. Ciclos: Son desviaciones de la tendencia debido a diferentes factores mayormente externos, diferentes a la estacionalidad. Su tiempo y duración no es necesariamente regular.
3. Aleatoriedad: Son los comportamientos variantes o cambios impredecibles no periódicas que se presentan en la serie.
4. Estacionalidad: Es el comportamiento de una serie dentro de un período dado. Las series de tiempo pueden conformar patrones que se repitan en los siguientes períodos (Sáez, 2022).

En cuanto a su estacionalidad, las series de tiempo se clasifican en:

2.1.1.1 Series estacionarias. Una serie es estacionaria cuando no varía a través del tiempo, en palabras técnicas, cuando su media y varianza no fluctúan, y esto se deduce gráficamente en que los valores de la serie oscilan en un medio constante y la variabilidad es la misma durante toda la serie (Villavicencio, s. f.).

2.1.1.2 Series no estacionarias. Son series en las cuales su tendencia y variabilidad cambian con el tiempo. Esos cambios se realizan en la media, y pueden a crecer o decrecer a lo largo del tiempo, entonces la serie no oscila en un valor constante (Villavicencio, s. f.).

Figura 3. *Serie de tiempo estacionaria y no estacionaria*

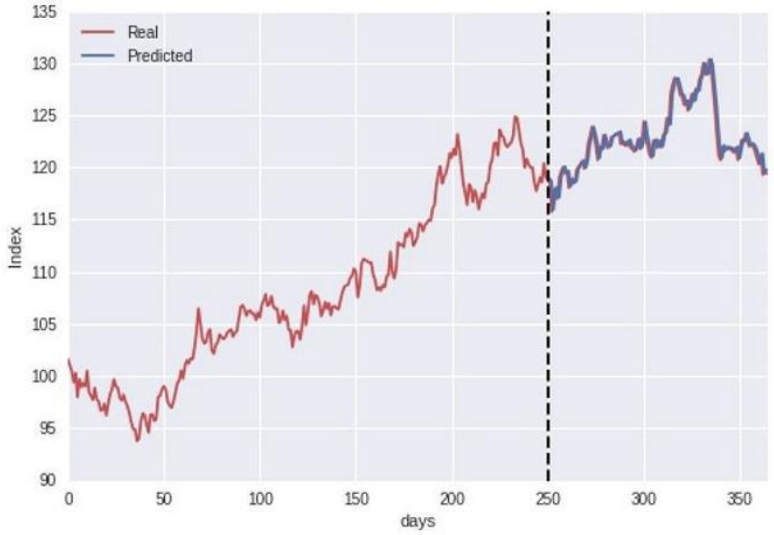


Nota: Figura ilustrativa de una serie de tiempo estacionaria y serie de tiempo no estacionaria. Tomado de: https://www.researchgate.net/figure/Non-stationary-and-stationary-time-series-As-CDR-activities-of-users-are-aggregated-on_fig10_326619835

2.1.1.3 Proyección en las series de tiempo. El pronóstico de una serie de tiempo se basa en proyectar los valores históricos hacia el futuro, siguiendo la línea temporal. Las dos características que lo definen son: El período (horas, días, semanas, meses, etc.), y el horizonte, dado por la cantidad de períodos a proyectar (Sáez, 2022). Para evaluar las predicciones en las series temporales, es necesario dividir el conjunto de datos disponibles a trabajar, en datos de entrenamiento y de prueba, los datos de entrenamiento son los que se ingresarán al modelo para entrenarlo y los datos de prueba, son los que se compararán con las predicciones. Para este caso,

las predicciones deben hacerse dentro del rango del conjunto de datos disponibles, ya que, una predicción a futuro, en sí, no se puede evaluar, por lo que, obteniendo un buen resultado de evaluación entre los datos reales y su predicción, se tiene una mayor probabilidad, que su predicción a futuro sea más precisa (Brownlee, 2017).

Figura 4. *Análisis visual de una predicción de serie de tiempo*



Nota: Figura ilustrativa de una predicción en una serie de tiempo. Tomado de: https://www.kdnuggets.com/2019/05/machine-learning-time-series-forecasting.html

2.1.2 ARIMA

Los procesos ARIMA son los más populares al momento de trabajar con las predicciones en series de tiempo. Éstos son una generalización de los procesos ARMA (*Auto Regresivos y de Medias Móviles*) (Quintana & Jiménez, 2016):

2.1.2.1 Definición. Un modelo es autorregresivo si la variable de un periodo t es proyectada por los históricos de ella misma agregando un término de error. La estadística sustenta que, bajo determinadas condiciones anteriores, toda Yt se puede expresar como una

combinación lineal de sus valores históricos anteriores más un término de error. Estos modelos se denominan AutoRegressive (AR), seguidamente se indica el orden del modelo: AR(1), AR(2), AR(3)… El orden en el modelo dice el número de observaciones retrasadas de las series temporal analizada que interviene en la ecuación, entonces un modelo AR(1) se expresaría de la siguiente forma:

$$Y_t = \emptyset_0 + \emptyset_1 Y_{t-1} + a_t \quad (1)$$

Donde Yt es una observación de la serie en un tiempo específico t dependiente de los $Yt{-}n$ observaciones anteriores, $\emptyset 0$, $\emptyset 1 \ldots \emptyset n \in \mathbb{R}$ y at es llamado ruido blanco (Castillo Gamarra, 2014).

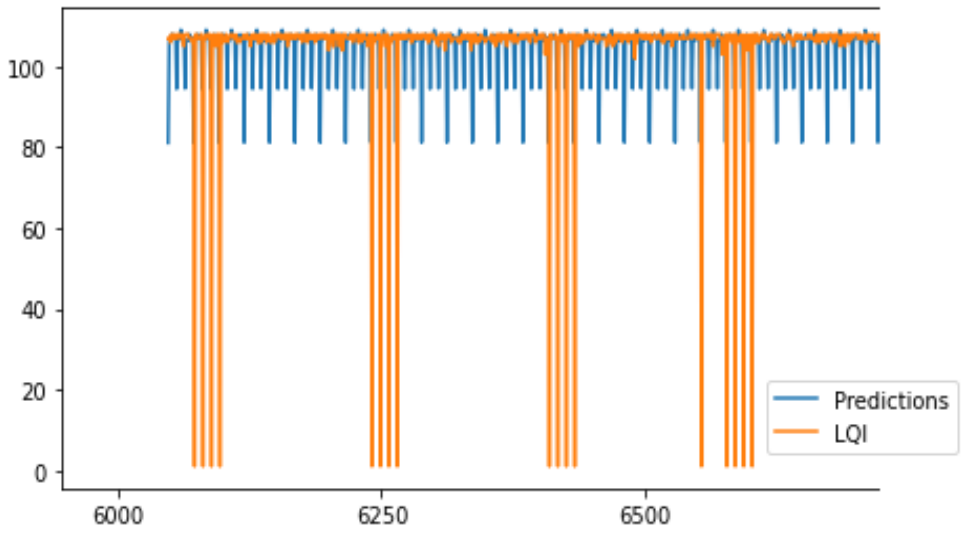
Un modelo de medias móviles es el que muestra el valor de una variable en un periodo t en base a un término independiente y una sucesión de errores precedentes, convenientemente ponderados, se hacen llamar Moving Averages (MA), seguidos de su orden entre paréntesis, por lo que, si hay *q* términos de error, el modelo se expresaría MA(q).

$$Y_t = \mu + a_t + \vartheta_1 a_{t-1} + \vartheta_2 a_{t-2} + \cdots + \vartheta_q a_{q-1} \quad (2)$$

Donde Yt corresponde al valor promedio de las observaciones en un determinado tiempo, $\theta 1, \theta 2, \ldots, \theta q$ y μ son constantes y at es llamado ruido blanco (Castillo Gamarra, 2014).Al igual que los modelos autorregresivos, se suele definir el orden del modelo de medias móviles como MA(1), MA(2), MA(n).

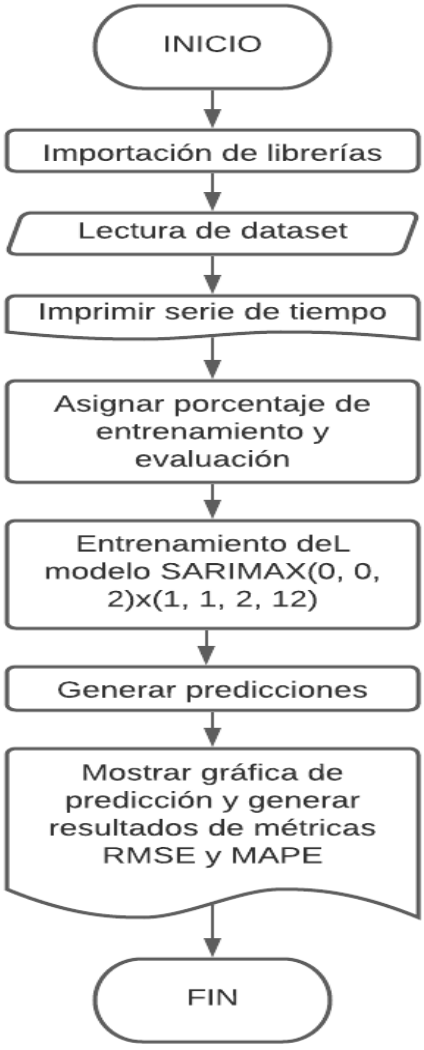
2.1.2.2 Algoritmo Implementado. Existen diferentes tipos de algoritmos ARIMA, el lenguaje de programación Python ofrece un modelo llamado SARIMAX que es esencialmente un modelo de regresión lineal el cuál utiliza un modelo de tipo ARIMA estacional de residuos. Este modelo cuenta con unos parámetros los cuáles son ajustables de las series de tiempo con las que se quiera hacer el trabajo de predicción (p, d, q)x(P, D, Q, M). El conjunto de parámetros(p, d, q) es llamado orden, y hace referencia al orden de la serie de tiempo, el cual también es usado en el modelo ARIMA, y el conjunto de parámetros (P, D, Q, M) es llamado orden estacional y se refiere al orden del componente estacional de la serie de tiempo (Duca, 2021). La función AUTOARIMA de la librería PMDARIMA permite hallar los óptimos valores para cada conjunto de datos. En este proyecto, se implementó la función AUTOARIMA con unos conjuntos de datos, y se escogió el modelo SARIMAX que arrojó mejores resultados al evaluar proyecciones tanto de LQI como de RSSI, el cuál fue SARIMAX (0, 0, 2)x(1, 1, 2, 12). Los parámetros de entrada de este algoritmo son el conjunto de datos a evaluar por proyección, la cantidad de registros para entrenar y la cantidad para evaluar el modelo. Los resultados del algoritmo son las métricas de error generadas en la predicción (RMSE y MAPE), y una gráfica de comparación de los datos originales y los registros generados al intentar predecir los datos, en el rango de datos de evaluación.

Figura 5. *Ejemplo de predicción LQI, algoritmo ARIMA*



Nota: Los registros en azul equivalen a las predicciones del modelo, y los de color naranja, al conjunto de datos originales. Eje x:index del registro LQI. Eje y: valor LQI del registro.

Figura 6. *Algoritmo SARIMAX(0, 0, 2)x(1, 1, 2, 12)*



Nota: Diagrama de flujo o diagrama operacional del algoritmo ARIMA implementado. Cabe recalcar que el algoritmo fue modificado en base a las necesidades del proyecto, el código fuente es tomado de: https://www.geeksforgeeks.org/python-arima-model-for-time-series-forecasting/

2.1.3 Algoritmo de Monte Carlo

El término “Monte Carlo” es usado a menudo para identificar cualquier método de estimación cuya operación involucre un componente aleatorio significativo. El método Monte Carlo solo requiere de experiencia: muestras de secuencias de estados, acciones y recompensas

de la interacción real o simulada con un entorno, pero aun así puede lograr un comportamiento óptimo como de aprendizaje por refuerzo (Gupta, 2021).

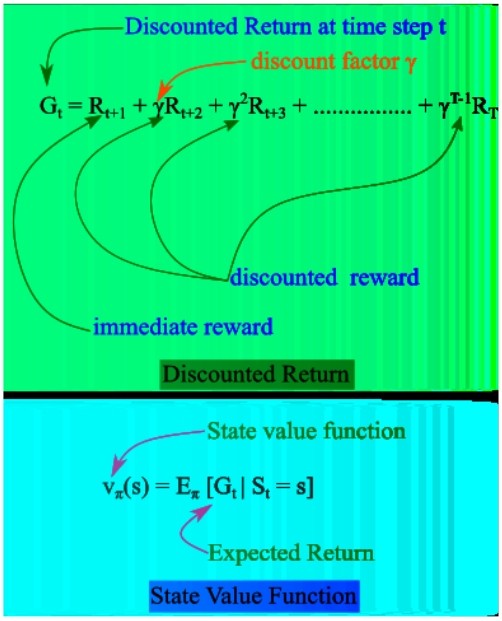
2.1.3.1 Definición. El método de Monte Carlo es un concepto simple, donde el agente aprende sobre los resultados y genera recompensas cuando se interactúa con el entorno, por lo que un algoritmo que contenga este método puede ser considerado un algoritmo de aprendizaje por refuerzo.(Roy, 2020) En este método, el agente genera muestras experimentadas, y luego, basándose en el promedio de su rendimiento, se calcula el valor para un estado, o una acción. Estas son algunas de las características del método Monte Carlo:

- No hay ningún modelo (el agente no conoce las transiciones de estado de Markov Decission Process – MDP).
- El agente aprende de la experiencia en ejemplos.
- Aprende el valor de estado $v\pi(s)$ bajo la política π (recompensa esperada) (Fernández, s. f.) experimentando un rendimiento promedio de todos los episodios muestrales (valor= rendimiento promedio).
- Solo después de un episodio completo, los valores se actualizan (debido a este algoritmo, la convergencia es lenta y la actualización ocurre después de que se completa un episodio), por lo que, al trabajar con series de tiempo, se necesitan hacer muchas predicciones, para lograr obtener mejores resultados. Lo anterior conlleva a que, al aplicar un mayor costo computacional, probablemente se generen mejores resultados.
- Solo se puede utilizar en problemas episódicos.

El rendimiento esperado es igual a la suma descontada de todas las recompensas. En este método, en vez del rendimiento esperado, se usa el rendimiento empírico que el agente ha muestreado según la política (Roy, 2020).

Algunos algoritmos que utilizan el método de Monte Carlo para la predicción de parámetros mediante series de tiempo implementan diferentes variaciones al momento de sus iteraciones, una de ellas es acotar el rango de respuesta de la predicción, si un valor se pasa del rango deseado, se colocará el valor del extremo, para mejorar la calidad de las predicciones y no generar tanto ruido en ellas (Roy, 2020).

Figura 7. *Función de valor de estado y función retorno con descuento Monte Carlo*

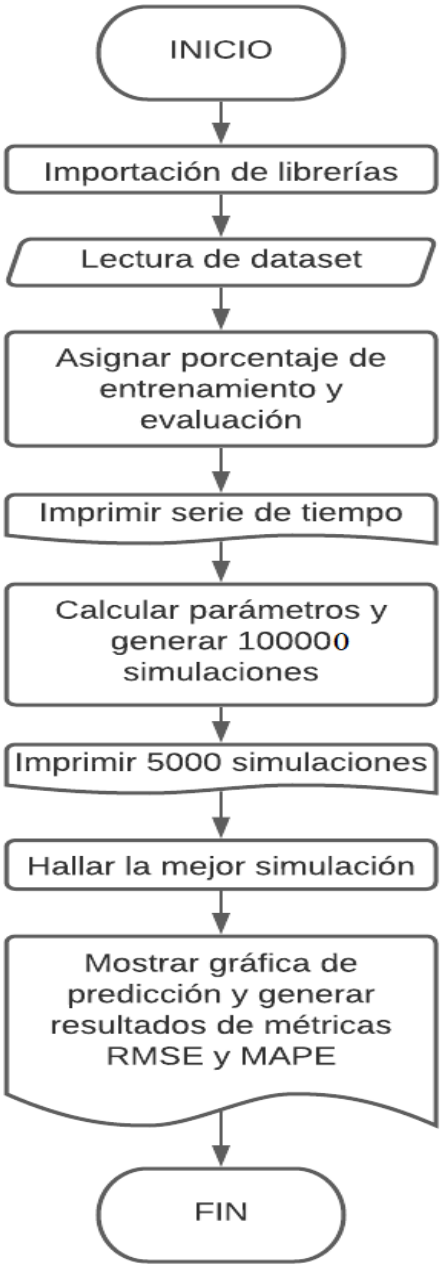


Nota: Tomado de: https://towardsdatascience.com/monte-carlo-learning-b83f75233f92

2.1.3.2 Algoritmo implementado. La implementación aplicada del método de Monte Carlo en un algoritmo para predicción mediante series de tiempo se basa en hallar el logaritmo natural de los registros del conjunto de datos, la mediana del logaritmo natural, la varianza, la

desviación, la derivada y la desviación estándar. A partir de esa información genera registros aleatorios a través del tiempo, este algoritmo genera 100000 series de tiempo, y al final se escoge la serie de tiempo que genere un menor RMSE, y a partir de esta, se muestran sus resultados de evaluación de las métricas RMSE y MAPE, además de una gráfica de la mejor predicción vs los datos originales. Los parámetros de entrada de este algoritmo son el conjunto de datos a evaluar por proyección, la cantidad de registros para entrenar y la cantidad para evaluar el modelo, y la cantidad de iteraciones (series de tiempo generadas, de la cual se escoge la de mejor métrica RMSE). Los resultados del algoritmo son las métricas de error generadas en la predicción (RMSE y MAPE), y una gráfica de comparación de los datos originales y los registros generados al intentar predecir los datos, en el rango de datos de evaluación. Su funcionamiento está descrito en el siguiente diagrama de flujo.

Figura 8. *Algoritmo Monte Carlo*



Nota: Diagrama de flujo o diagrama operacional del algoritmo Monte Carlo implementado, cabe recalcar que el algoritmo fue modificado en base a las necesidades del proyecto, el código fuente es tomado de: https://github.com/macskamancs/Forecasting_StockPrices_MonteCarlo/blob/main/MonteCarlo_Forecasting_StockPrices.ipynb

El orden de complejidad temporal de este algoritmo es de O(n^2), basado en el estudio del código fuente del algoritmo, el cuál muestra los ciclos iterativos anidados que producen este orden de complejidad, y puede ser consultado en el apéndice.

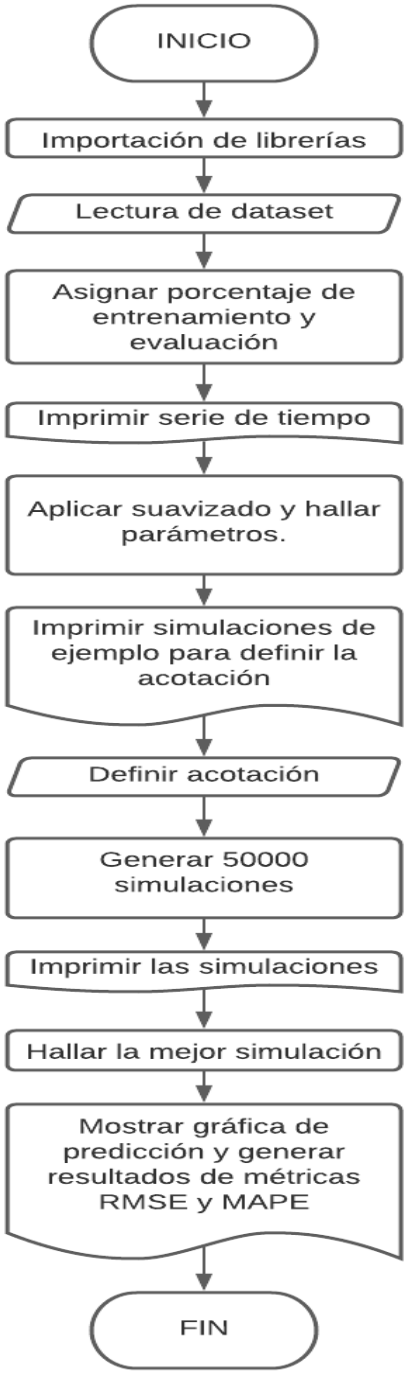
2.1.4 Algoritmo de Monte Carlo con acotación

2.1.4.1 Definición. El algoritmo de Monte Carlo con acotación cumple con todos los fundamentos teóricos de un algoritmo por método Monte Carlo, su diferencia se radica en que, al momento de generar las simulaciones, tiene una acotación en el eje de variación de los valores, y si un valor simulado sobrepasa esta acotación, el valor en este punto será el valor de la acotación.

2.1.4.2 Algoritmo implementado. A diferencia de la anterior implementación de algoritmo Monte Carlo, esta versión no halla el logaritmo natural para poder hallar los parámetros de generación de simulaciones, en cambio, este método aplica una suavización de registros mediante su media, a partir de los nuevos datos suavizados, halla la media y la desviación, se asigna la acotación, la cuál puede ser arriba o debajo de la serie temporal, dependiendo del sentido de la serie en donde exista mayor ruido, esta acotación es asignada manualmente en cada conjunto de datos después de ser analizada visualmente, y ya se procede a realizar las 50000 simulaciones y a imprimirlas en pantalla, y finalmente se escoge la mejor solución de serie de tiempo, a la cual se le aplican las métricas de evaluación RMSE y MAPE además de una gráfica de la mejor predicción vs los datos originales. Los parámetros de entrada de este algoritmo son el conjunto de datos a evaluar por proyección, la cantidad de registros para entrenar y la cantidad para evaluar el modelo, la cantidad de iteraciones (series de tiempo generadas, de la cual se escoge la de mejor métrica RMSE) y el valor de la acotación en unidades de LQI o RSSI respectivamente. Los resultados del algoritmo son las métricas de error generadas en la predicción (RMSE y MAPE), y una gráfica de comparación de los datos originales y los

registros generados al intentar predecir los datos, en el rango de datos de evaluación. Su funcionamiento está descrito en el siguiente diagrama de flujo.

Figura 9. *Algoritmo Monte Carlo con acotación*



Nota: Diagrama de flujo o diagrama operacional del algoritmo Monte Carlo con acotación implementado, cabe recalcar que el algoritmo fue modificado en base a las necesidades del proyecto, el código fuente es tomado de: https://github.com/klameer/monte-carlo-sales-forecasting/blob/main/Monte-Carlo-Sales2.ipynb

El orden de complejidad temporal de este algoritmo es de O(n^2), basado en el estudio del código fuente del algoritmo, el cuál muestra los ciclos iterativos anidados que producen este orden de complejidad, y puede ser consultado en el apéndice.

2.1.5 Algoritmo DDPG

El DDPG es un modelo libre que aprende acciones continuas, combina ideas de DPG (*Deterministic Policy Gradient*) y DQN (*Deep Q-Network*), usa experiencia de repetición y redes de destino de aprendizaje lento de DQN, y se basa en DPG, el cuál puede operar en espacios de acción continua (Team, s. f.).

2.1.5.1 Definición. DPG es un algoritmo de gradiente de política determinista. La idea básica es ese algoritmo de gradiente de política, una distribución de probabilidad paramétrica $\pi_0(a|s) = P[a|s;\theta]$ para representar la política, y debido a que la política es una distribución de probabilidad, entonces la acción a se selecciona aleatoriamente, el llamado gradiente de política estocástico (*【强化学习】DPG, DQN与DDPG*, s. f.). DQN es una generalización a todos los algoritmos de aprendizaje por refuerzo que utilicen redes neuronales para estimar sus parámetros (*Aprendizaje Por Refuerzo*, s. f.). DDPG usa cuatro redes neuronales, una red Q, una red de política determinista, una red objetivo Q y una red de política objetivo.

Parámetros:

$$\theta^{Q}: Red\ Q$$

$$\theta^{U}: Función\ de\ política\ determinística$$

$$\theta^{Q'}: Red\ obejtivo\ Q$$

$$\theta^{w'}: Red\ de\ política\ de\ destino$$

La red Q y la red de políticas son muy similares a un actor crítico de ventaja simple, pero en DDPG, el actor asigna directamente los estados a las acciones en vez de generar la distribución de probabilidad en un espacio de acción.

Las redes de destino son copias retardadas de sus redes originales que van rastreando lentamente las redes que se van aprendiendo, el uso de estas redes de valores objetivo mejora grandemente el aprendizaje.

Para las actualizaciones de la red de política y valor, la red se actualiza de forma similar que en Q-learning, Q se obtiene mediante la ecuación de Bellman:

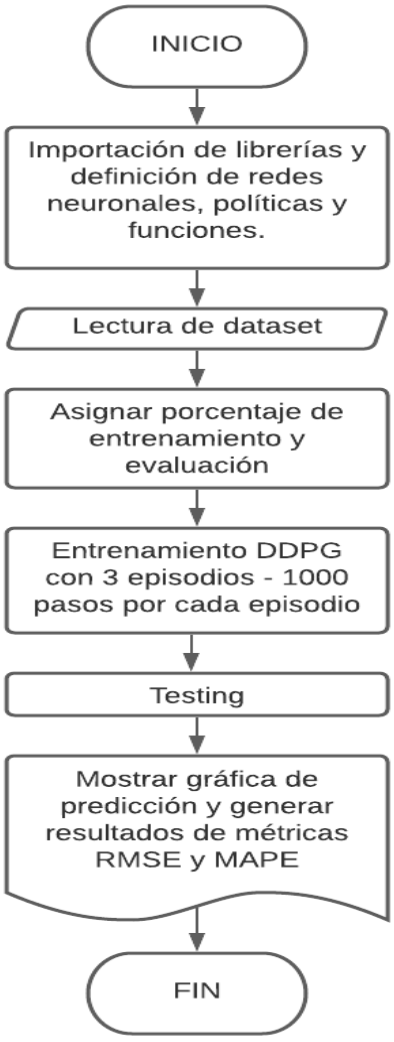
$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}\middle|\theta^{\mu'}\right)\middle|\theta^{Q'}\right) \quad (3)$$

Para la exploración, en el aprendizaje por refuerzo, se realiza mediante la selección probabilística de una acción aleatoria. Para los estados de acción continua, la exploración se realiza agregando ruido a la acción en sí (Yoon, 2019).

2.1.5.1 Algoritmo implementado. El algoritmo DDPG implementado utiliza un búfer de experiencia como dispositivo de memoria para almacenar el historial de transiciones, y aprende

en lotes de muestreo aleatorios del búfer histórico en lugar de aprender en línea. También se establecen redes de destino separadas para el actor y el crítico para debilitar la sobreestimación de la función valor, por lo que, en consecuencia, existen 4 redes neuronales en este DDPG: La red de salida del actor, la red objetivo de salida del actor, la red crítica y la red objetivo (Liu et al., 2020). Los parámetros de entrada de este algoritmo son el conjunto de datos a evaluar por proyección, la cantidad de registros para entrenar y la cantidad para evaluar el modelo, la cantidad de episodios (1000 pasos por episodio), existen otros parámetros que pueden variar, pero son más arraigados a la estructura del DDPG. Los resultados del algoritmo son las métricas de error generadas en la predicción (RMSE y MAPE), y una gráfica de comparación de los datos originales y los registros generados al intentar predecir los datos, en el rango de datos de evaluación. Su funcionamiento está descrito en el siguiente diagrama de flujo.

Figura 10. *Algoritmo DDPG*



Nota: Diagrama de flujo o diagrama operacional del algoritmo DDPG implementado, cabe recalcar que el algoritmo fue modificado en base a las necesidades del proyecto, el código fuente es tomado de: https://github.com/ChefLiutao/Time-series-forecasting-via-deep-reinforcement-learning

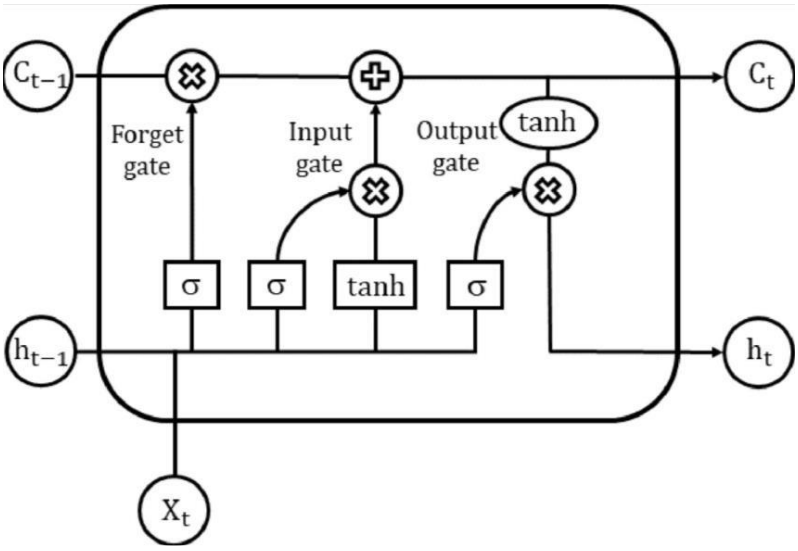
El orden de complejidad temporal de este algoritmo es de O(n^3), basado en el pseudocódigo del algoritmo ofrecido por los autores y desarrolladores del algoritmo (Liu et al., 2020), en el que se detallan las rutinas e instrucciones iterativas anidadas que producen este orden de complejidad temporal.

2.1.6 Algoritmo RDPG

RDPG es una extracción y mejoramiento del método DDPG. El DDPG convencional, en su red actor y red critic, implementa MLP (*Multi-Layer Perceptron*), el cuál consta de redes multicapa conectadas. Una desventaja de MLP es al momento de tratar un problema con linealidad complicada, ya que la estimación de la función de valor de acción no es lo suficiente buena, por esa razón, para solucionar este problema, se propuso el RDPG.

2.1.6.1 Definición. RDPG usa LSTM (*Long-Short Term Memory*) en su red actor y su red critic, lo cual mejora la estimación del valor de cada acción. Como se puede observar en la siguiente figura, LSTM almacena información durante largos períodos de tiempo usando una celda de memoria especialmente diseñada con base a la estructura 3 puertas de RNN (*Recurrent Neural Network*): puerta de entrada, puerta de salida y puerta de olvido. Estas puertas están diseñadas para controlar el flujo de información dentro de cada bloque de memoria.

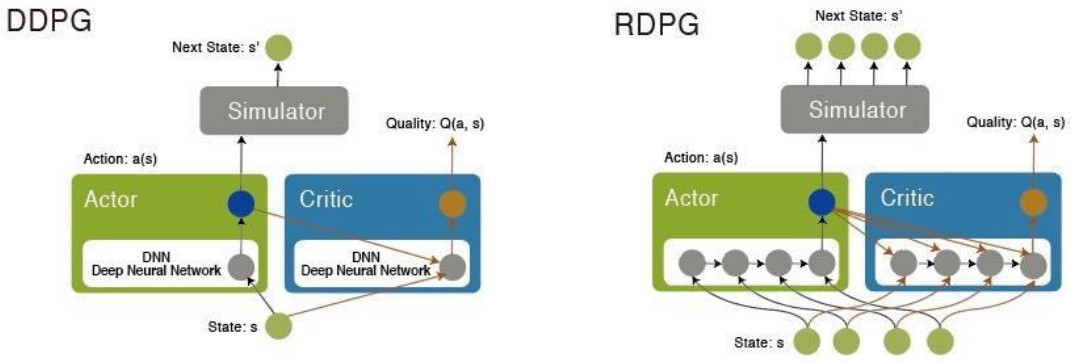
Figura 11. *Diagrama esquemático de LSTM*



Nota: Tomado de: (Liu et al., 2020)

Al usar LSTM en aprendizaje por refuerzo, se le llama Deep Reinforcement Learning, el cuál es una rama del RL. En este caso, al usar RDPG, se puede producir una estimación mucho más precisa de la función de valor de acción que en DDPG, ya que se la red crítica para estimar $Qw(s,a)$ puede agregar observaciones a lo largo del tiempo, por lo que se puede proporcionar un error más preciso y esto lleva a una posible mejor actualización de la red actor (Liu et al., 2020)

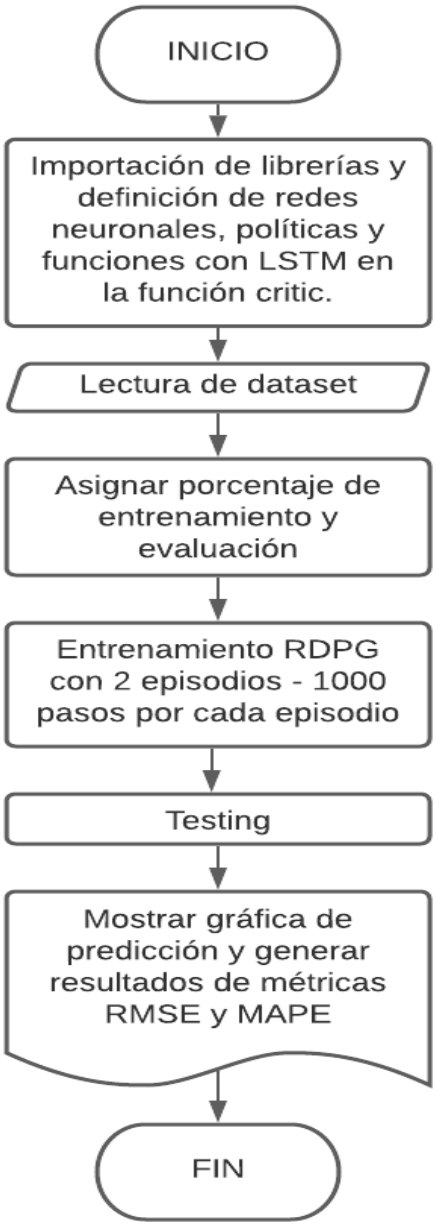
Figura 12. *Comparación entre algoritmos DDPG y RDPG*



Nota: Tomado de: («Making AI Experience Many Trials to Achieve High-Level Behavior», 2018)

2.1.6.1 Algoritmo implementado. Los parámetros de entrada del algoritmo implementado son el conjunto de datos a evaluar por proyección, la cantidad de registros para entrenar y la cantidad para evaluar el modelo, la cantidad de episodios (1000 pasos por episodio). Existen otros parámetros que pueden variar, pero son más arraigados a la estructura del RDPG. Los resultados del algoritmo son las métricas de error generadas en la predicción (RMSE y MAPE), y una gráfica de comparación de los datos originales y los registros generados al intentar predecir los datos, en el rango de datos de evaluación. Su funcionamiento está descrito en el siguiente diagrama de flujo:

Figura 13. *Algoritmo RDPG*



Nota: Diagrama de flujo o diagrama operacional del algoritmo RDPG implementado, cabe recalcar que el algoritmo fue modificado en base a las necesidades del proyecto, el código fuente es tomado de: https://github.com/ChefLiutao/Time-series-forecasting-via-deep-reinforcement-learning

El orden de complejidad temporal de este algoritmo es de O(n^3), basado en el pseudocódigo del algoritmo ofrecido por los autores y desarrolladores del algoritmo (Liu et al., 2020), en el que se detallan las rutinas e instrucciones iterativas anidadas que producen este orden de complejidad temporal.

2.1.7 Métricas de error

Para observar el comportamiento de una red inalámbrica, existen parámetros o también llamadas métricas, las cuales describen su comportamiento, calidad, entre muchos otros factores. Los parámetros llamados LQI (*Link Quality Indicator*) y RSSI (*Received Signal Strength Indicator*) generan información útil dependiendo de la cantidad de paquetes recibidos y perdidos, mostrando una gran variabilidad en ambientes internos, ya que miden cobertura y sensibilidad de los nodos mediante pruebas de recepción de paquetes(RSSI) y niveles de potencia recibida(LQI) (Daboín et al., 2012). Por lo que se concluye que, para este proyecto, la información dada por los parámetros LQI y RSSI es suficiente, debido a que son las que generan más información de acuerdo con el estado de la red inalámbrica, ya que ellos miden la calidad del enlace.

El **LQI** es una métrica que muestra la calidad de la señal del enlace entre un emisor y un receptor, y existen tres niveles en general: bajo, medio o alto («LQI (Link Quality Indicator)», 2016). La estimación del LQI se hace para cada paquete que sea recibido. Además, el LQI es informado a la capa MAC y está disponible para las capas de red (*NWK*) y aplicación (*APL*) para todo tipo de análisis antes de un procedimiento, como, por ejemplo, informarse de los dispositivos en la red para decidir por cuál ruta se va a enviar el mensaje, y la ruta que tenga el LQI más alto, tiene más probabilidades de entregar el mensaje a su destino (Farahani, 2008).

El **RSSI** es una medida que indica la veracidad de un dispositivo para detectar y recibir señales desde un punto específico. Dado el caso que exista una transmisión activa en el tiempo

de muestreo, el RSSI se encarga de medir la potencia de la señal recibida, en el caso que no, el RSSI mide la potencia de la interferencia más el ruido de fondo. La lectura de esta métrica se puede obtener desde cualquier receptor participante de la red. Esta métrica se da en decibeles y sus valores suelen ser negativos entre 0 y -120 (menos ciento veinte) (Yuan et al., 2017).

Para la evaluación de la predicción en las series de tiempo, es importante escoger las métricas con las que se van a analizar las series, en este caso, se escogieron el RMSE (*Root-Mean-Square ERROR*) y el MAPE (*Mean Absolute Percentage Error*). Primero, se escogió RMSE por recomendación de los autores de los algoritmos DDPG y RDPG, además, el RMSE otorga un peso relativamente alto a los errores grandes, esto significa que el RMSE es más útil cuando los errores grandes son indeseables, y en la predicción de series de tiempo (*Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE)*, s. f.), lo que se busca es que sea precisa la proyección. El MAPE se escoge debido a que muestra el nivel de rendimiento en la predicción del modelo que se esté usando, debido a que no se mide en unidades, sino en porcentaje, tiene la ventaja de comparar los resultados de las predicciones entre conjuntos de datos (*3.4 Evaluating forecast accuracy | Forecasting*, s. f.).

El RMSE se deriva del MSE (*Mean-Square Error*), y el MSE representa un valor real xi y un valor predicho $\hat{x}$ en una cantidad de observaciones n:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{4}$$

Es muy útil al momento de predecir, al comparar cada predicción con su respectivo valor real, será una estimación muy importante al momento de evaluar el modelo. Entre más alto sea el

error, significa que no hay buenas aproximaciones a los valores reales. Pero algo que no favorece es si existe una predicción muy desajustada, la cuadratura empeorará el error y puede sesgar el resultado de la métrica para sobreestimar la ineficacia del modelo. Este es un comportamiento particularmente problemático si se tienen datos ruidosos.

Y el RMSE es solo la raíz cuadrada de MSE. La raíz cuadrada se introduce para que las unidades de error sean iguales a las unidades de lo observado.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} = \sqrt{MSE} \tag{5}$$

El MAPE (Mean Absolute Percentage Error), corresponde al promedio del error absoluto entre una observación pronosticada yˆ y la observación real y. Este error es dado en términos porcentuales:

$$MAPE = \frac{\sum_{i=1}^{n} 100|y_i - \hat{y}_i|}{\frac{y_i}{n}} \tag{6}$$

2.2 Metodología

En esta sección se habla de las Redes LLN (*Low-Power and Lossy Networks*), de sus características, del conjunto de datos tomados y sus tratamientos previos. Así mismo se incluye la metodología general de la ejecución del Trabajo de Grado, la metodología del diseño de

experimentos y los resultados generados con métricas de error y análisis de tiempos de ejecución.

Durante el tiempo de ejecución del proyecto de investigación, se propusieron calendarios y maneras de marcar el ritmo en el desarrollo de este, además, se realizaron ejecuciones de prueba previas al diseño de experimentos con el fin de adquirir experticia para lograr eficacia y rendimiento en los experimentos. En cada proceso se llevó a cabo la correspondiente documentación con el fin de sustentar todas las metodologías y resultados mostrados a continuación.

2.2.1 Redes LLN IEEE 802.15.4

Las redes LLN, que en español significa: Redes de bajo consumo y con pérdidas, son redes diseñadas para maximizar la vida útil de la batería en sistemas embebidos, ya que la mayoría son redes inalámbricas (sin conexión cableada), y su fácil integración con otros nodos, con fabricación y costeo de los nodos relativamente más barata que las redes convencionales y su despliegue y funcionamiento a un bajo costo. Las redes LLN comparten las siguientes características:

- Gran cantidad de dispositivos de gama baja: hasta cientos de dispositivos con procesadores de 16 bits y RAM en el orden de Kilobytes (kB).
- Comunicaciones multisalto: son capaces de cubrir grandes áreas con radios de baja potencia requiere comunicaciones multisalto.
- Tamaños de tramas pequeños: el tamaño típico de las tramas es de 127 bytes.
- Bajo consumo de energía: Dispositivos que funcionan con baterías con muy poca intervención humana para recargar o reemplazar baterías, y la vida útil esperada

de la red es de meses o años. Opcionalmente algunas LLN pueden incluir mecanismos de recolección de energía (Garcia Algora et al., 2017).

Las redes IEEE 802.15.4 son un tipo de red LLN. IEEE 802.15.4 es un estándar que define tanto las propiedades físicas como el control de acceso en redes inalámbricas de área personal con niveles bajos de transmisión de datos (*LRWPAN - Low-Rate Wireless Personal Area Network*) y bajo consumo energético (jecrespom, s. f., p. 5). Este sistema cuenta con varios componentes, dos de ellos son el FFD (*Full-Function Device*) y el RFD (*Reduced-Function-Device*). Una red incluirá al menos un FFD como administrador o coordinador. Estos dispositivos son característicos por consumir muy poca energía, tales como interruptores, sensores, alarmas, microcontroladores entre otros (Romero et al., s. f.).

2.2.2 Conjuntos de datos

Para poder evaluar los modelos de aprendizaje por refuerzo, se necesita una buena cantidad de datos y con buena calidad, para solucionar esto, se aprovecharon los conjuntos de datos de parámetros de redes LLN IEEE 802.15.4 disponibles en el repositorio denominado CRAWDAD del Dartmouth College, universidad ubicada en Hanover, New Hampshire, Estados Unidos.

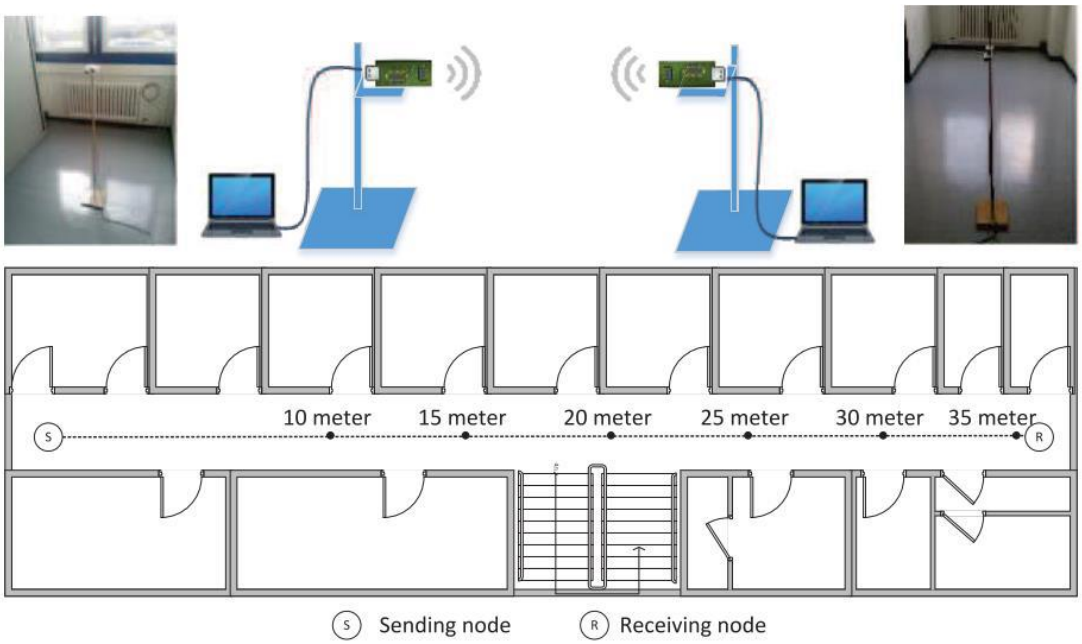
2.2.2.1 Fuente y obtención. CRAWDAD (*Community Resource for Archiving Wireless Data At Dartmouth*) (Henderson & Kotz, 2015) otorga acceso a ficheros de datos, proveídos por (Fu et al., 2015), los cuales están clasificados de la siguiente manera: distance_10m, distance_15m, distance_20m, distance_25m, distance_30m y distance_35m, y cada fichero contiene 12 archivos en formato *txt*. Continuando con la búsqueda de este tipo de datos, se encontró que los mismos datos de CRAWDAD, pero con un tratamiento previo estaban dispuestos y listos para usar en GitHub (Mantilla, 2020/2020), por lo que se tomaron estos datos

para evaluar los modelos, y debidamente en esta sección se explicará su contenido y su pretratamiento.

2.2.2.2 Estructura. Al tomar los conjuntos de datos, estos venían en archivos *.csv*, cada archivo contiene 8064 registros, en el que para cada registro existe información existen dos datos: el índice del registro y su valor correspondiente de LQI y RSSI.

2.2.2.3 Topología de red. Según (Fu et al., 2015) implementaron dos nodos *TelosBK, con TI CC2420 (*chips transmisores IEEE 802.15.4*)* en un espacio de 2 metros de ancho por 40 metros de longitud. En su arquitectura, se mantuvo la línea de visión entre los nodos para evitar interrupciones, lo que permitió la circulación de personas en la ejecución de los experimentos. Variaron la distancia de los nodos, y de esa experimentación tomaron los conjuntos de datos de parámetros de esa red.

Figura 14. *Topología de red fuente del conjunto de datos*



Nota: Imagen tomada de: Experimental Study for Multi-layer Parameter Configuration of WSN Links - (Fu et al., 2015) - *https://ieeexplore.ieee.org/document/7164923*

2.2.2.4 Tratamientos aplicados previamente. Según (Mantilla, 2020/2020) los datos fueron sometidos a un filtro de tratamiento, el cual se basó en reemplazar los registros de datos de RSSI con el valor de 0 (cero) por -1 (menos uno), y también reemplazar los registros de datos LQI con el valor de 0 (cero) por 1 (uno), todo con el propósito de evitar indeterminaciones en las ecuaciones de las métricas de error. Además, por cada archivo *.txt* obtenido de CRAWDAD, se ejecutó un script en el lenguaje *Python,* el cual automatizó el proceso de extracción de los 300 registros de metadatos, y se transformaron en series de tiempo de una variable, seguidamente, las series de tiempo se almacenaron en formato *.csv.* Los nombres de los archivos seleccionados se componen de la siguiente manera: *raw_data_runX_Y.csv,* donde *X* es el número del archivo original tomado de CRAWDAD y equivale al número de la columna de la extracción correspondiente a la columna de las 300 posibles para cada registro.

Tabla 1. *Series de tiempo resultantes LQI*

Nombre de fichero original en CRAWDAD	**Series de tiempo extraídas en formato *.csv***	**Cantidad de series de tiempo generadas**
Distance_10m	raw_data_run1--1	1
Distance_15m	raw_data_run1--1	1
Distance_20m	raw_data_run1--1	11
	raw_data_run1--19	
	raw_data_run1--20	
	raw_data_run1--21	
	raw_data_run1--22	
	raw_data_run1--23	
	raw_data_run1--24	
	raw_data_run1--25	
	raw_data_run1--26	
	raw_data_run1--27	
	raw_data_run1--28	
Distance_25m		0
Distance_30m		0
Distance_35m		0
		Total: 13

Tabla 2. *Series de tiempo resultantes RSSI*

Nombre de fichero original en CRAWDAD	**Series de tiempo extraídas en formato *.csv***	**Cantidad de series de tiempo generadas**
Distance_10m	raw_data_run3--1	1
Distance_15m	raw_data_run2--1	1
Distance_20m	raw_data_run2--1	11
	raw_data_run2—2	
	raw_data_run2--3	
	raw_data_run4--1	
	raw_data_run4--2	
	raw_data_run5—1	
	raw_data_run5--2	
	raw_data_run5--3	
	raw_data_run7—1	
	raw_data_run8—1	
	raw_data_run8--2	
Distance_25m		0
Distance_30m		0
Distance_35m		0
		Total: 13

2.2.2.5 Tratamiento de datos para aplicación de DDPG y RDPG. Principalmente, los datos tomados de (Mantilla, 2020/2020), contienen dos columnas por archivo *.csv* que equivale a una serie de tiempo como en la siguiente tabla:

Tabla 3. *Muestra de conjunto de datos RSSI 10m*

Date	
1	-43
2	-35
3	-35
4	-35
5	-37

Pero debidamente a la estructura y ejecución de los algoritmos DDPG y RDPG, la columna llamada *Date* se suprimió, ya que los algoritmos cuentan con su propia indexación, por lo tanto, se crearon réplicas modificadas de los archivos *.csv*, en donde los nombres de los archivos replicados se componen de la siguiente manera: *raw_data_runX_Y - copua.csv*, donde *X* es el número del archivo original tomado de CRAWDAD y equivale al número de la columna de la extracción correspondiente a la columna de las 300 posibles para cada registro, y *copia* significa que es una réplica del archivo original, pero sin la columna *Date*, de la siguiente manera:

Tabla 4. *Muestra de conjunto de datos para uso en DDPG y RDPG RSSI 10m*

-43
-35
-35
-35
-37

2.2.2.6 Resumen de resultados. Con la finalidad de tener una visión y estudio más claro de los conjuntos de datos, se realizó para todos los 26 (veintiséis) archivos con series de tiempo, análisis de las siguientes métricas que muestran su comportamiento: Varianza y desviación estándar, mediante un script en el lenguaje *Python*.

Tabla 5. *Medidas de dispersión datos LQI*

Serie de tiempo	Varianza	Desviación estándar
10m: Raw_data_run1—1	40.95568	6.39966
15m: Raw_data_run1—1	622.21239	24.94418
20m: Raw_data_run1—1	291.95389	17.08665
20m: Raw_data_run1—19	410.10160	20.25096
20m: Raw_data_run1—20	776.74249	27.87010
20m: Raw_data_run1—21	418.25159	20.45120
20m: Raw_data_run1—22	618.72678	24.87421
20m: Raw_data_run1—23	416.77124	20.41497
20m: Raw_data_run1—24	628.08698	25.06166
20m: Raw_data_run1—25	425.98727	20.63945
20m: Raw_data_run1—26	630.12692	25.10232
20m: Raw_data_run1—27	439.68614	20.96869
20m: Raw_data_run1—28	618.32138	24.86606
Valores medios	487.53264	21.45616

Tabla 6. *Medidas de dispersión datos RSSI*

Serie de tiempo	Varianza	Desviación estándar
10m: Raw_data_run3—1	50.74551	7.12358
15m: Raw_data_run2—1	81.15486	9.00859
20m: Raw_data_run2—1	53.35278	7.30429
20m: Raw_data_run2—2	54.89671	7.40923
20m: Raw_data_run2—3	57.92205	7.61065
20m: Raw_data_run4—1	65.91824	8.11900
20m: Raw_data_run4—2	72.99561	8.54374
20m: Raw_data_run5—1	72.15530	8.49442
20m: Raw_data_run5—2	72.11263	8.49191
20m: Raw_data_run5—3	80.15678	8.95303
20m: Raw_data_run7—1	63.33900	7.95858
20m: Raw_data_run8—1	72.66760	8.52452
20m: Raw_data_run8—2	72.64361	8.52312
Valores medios	66.92774	8.15881

Por lo tanto, los resultados muestran acerca del conjunto de datos que: Hay un total de 13 conjuntos de datos para la métrica LQI y 13 conjuntos de datos para la métrica RSSI, para un total de 26 juegos de datos. Para la evaluación de los algoritmos DDPG y RDPG, se aplicó una duplicación de los juegos de datos y una modificación de sus columnas, por lo que, sumado con los 26 juegos de datos anteriores, da un total de 52 juegos de datos. Según la tabla 7, mediante el

estudio de métricas de dispersión para los juegos de datos, se ve que los datos LQI son 2.62981 veces más dispersos que los datos RSSI, según sus unidades correspondientes, basados en la media de la desviación estándar de cada uno de los conjuntos de datos.

Tabla 7. *Comparación medidas de desviación entre datos LQI y RSSI*

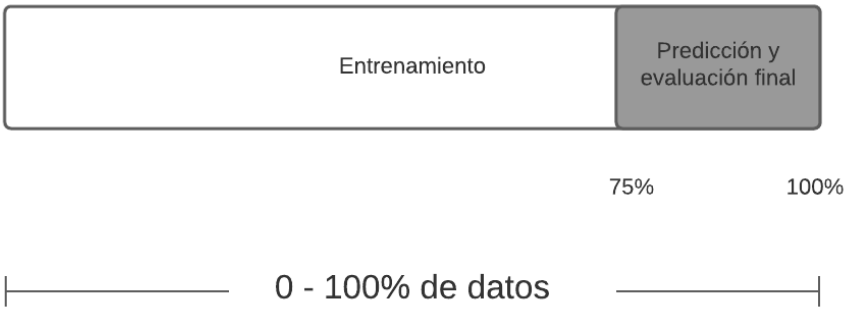
Métrica	Varianza	Desviación estándar
LQI	487.53264	21.45616
RSSI	66.92774	8.15881
Cociente	7.28446	2.62981

2.2.3 Modelos de entrenamiento y evaluación

Uno de los parámetros más importantes en el momento de implementar y evaluar cualquier modelo de aprendizaje automático (Machine Learning ML) es el porcentaje de datos designados a entrenar el modelo, y la cantidad de datos destinados a evaluar el modelo. En esta sección se muestran los dos modelos implementados en los algoritmos para el diseño de experimentos realizado, su teoría y aspectos lógicos y técnicos.

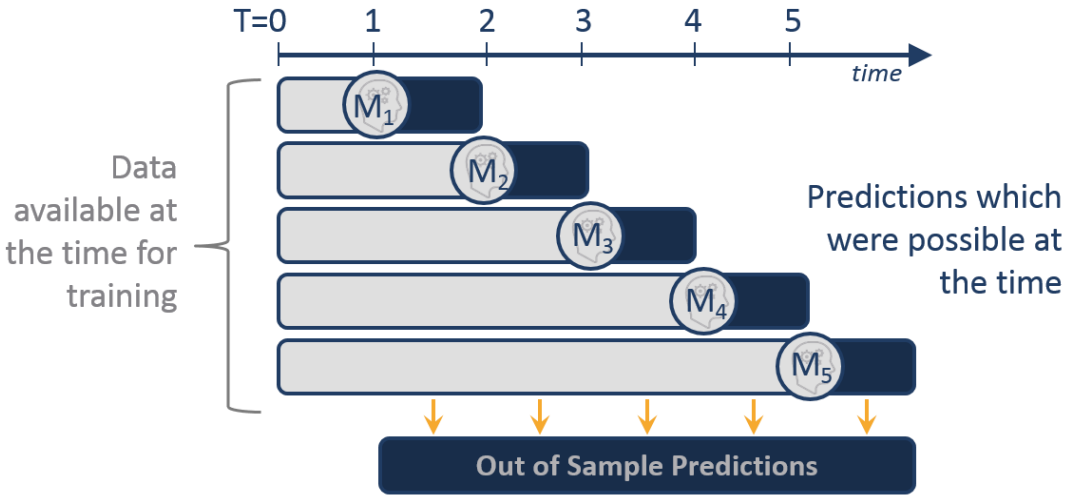
2.2.3.1 Modelo propuesto 75 – 25. Llegar a encontrar la distribución óptima entre el conjunto de datos de entrenamiento y de evaluación es todo un reto. Una de las distribuciones o modelo más usado es el 75 – 25, el cuál consta en asignar el primer 75% de los datos de la serie de tiempo en orden cronológico para el entrenamiento del modelo, y el siguiente 25% de los datos para la predicción de acuerdo con su entrenamiento, esta predicción se evalúa comparativamente con el último 25% de los datos originales.

Figura 15. *Ilustración del modelo propuesto 75 - 25*



2.2.3.2 Walk Forward Validation (WFV). Una estrategia de validación para poder encontrar los mejores resultados del modelo a evaluar, es el *Walk Forward Validation (WFV),* el cual consiste en secuencias de avance, entrenando periódicamente el modelo mediante ventanas de datos, con los propios resultados de predicción del modelo en esas ventanas anteriores (*Stock Prediction with ML: Walk-forward Modeling — The Alpha Scientist*, s. f.).

Figura 16. *Ilustración general del WFV*



Nota: Imagen tomada de: (Stock Prediction with ML: Walk-forward Modeling — The Alpha Scientist, s. f.) - https://alphascientist.com/walk_forward_model_building.html

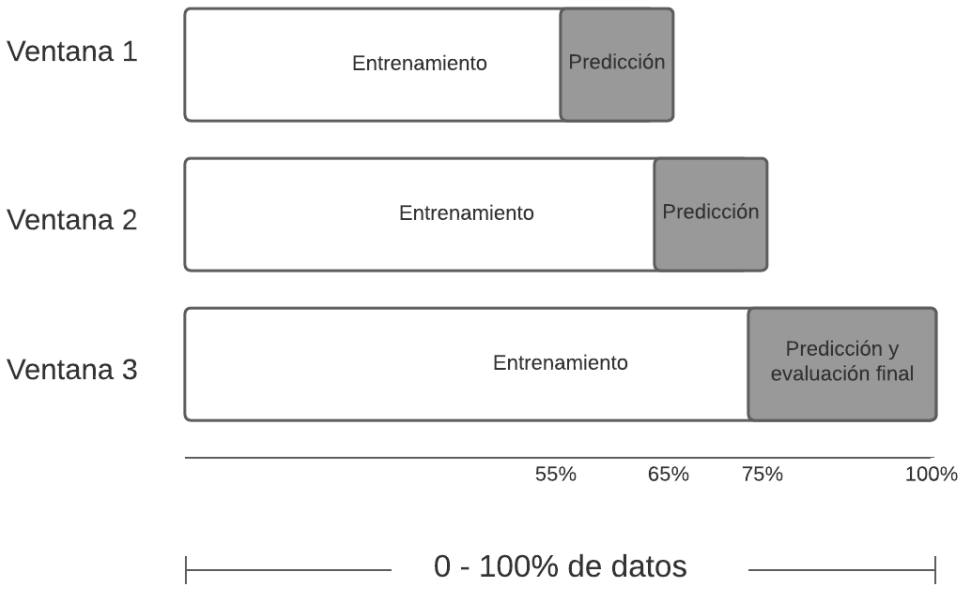
La secuencia de entrenamiento se basa en los siguientes pasos:

1. Dividir la cantidad de datos disponibles en ventanas con la longitud conveniente.
2. Comenzar a entrenar con la primera ventana de datos, predecir la próxima ventana.
3. Almacenar las predicciones como valores reales.
4. Expandir la ventana con las anteriores predicciones y luego repetir el proceso con las siguientes ventanas (paso 2).
5. La evaluación final del modelo puede ser con un 100% de los datos, o a conveniencia.

En la fase previa de conocimiento y apropiamiento de los algoritmos y conjuntos de datos, se notaron los mejores resultados mediante la siguiente distribución del WFV:

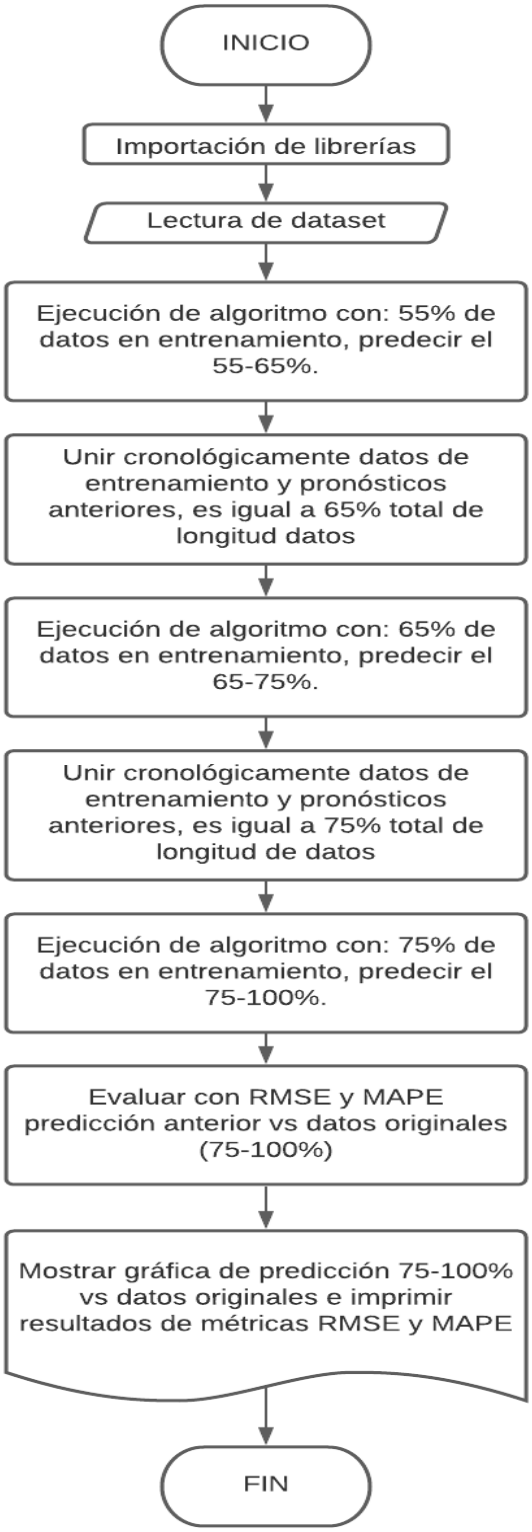
- Iteración 1: Entrenamiento: 0-55% -- Predicción: 55-65%
- Iteración 2: Entrenamiento: unión de entrenamiento de iteración 1 más predicción de iteración 1 -- Predicción: 65-75%
- Iteración 3: Entrenamiento: unión de entrenamiento de iteración 2 más predicción Iteración 2 -- Predicción y evaluación con respecto a datos originales: 75-100%

Figura 17. *Ilustración del modelo WFV aplicado*



Para la implementación de este modelo en los algoritmos a evaluar, se tuvo en cuenta los límites de las ventanas de los datos, el entrenamiento y predicción de cada ventana y las siguientes iteraciones, así como lo muestra el siguiente diagrama de flujo general:

Figura 18. *Implementación general de WFV*



Nota: Las características de cada algoritmo se pueden ver en las figuras 6 a 13.

El tiempo de ejecución de cada algoritmo con su implementación de WFV es aproximadamente de 3 veces mayor que el tiempo dado que se encuentra en las especificaciones de cada algoritmo (sección 2.1.2 a sección 2.1.6), además, a continuación, se muestra una tabla comparativa del orden de complejidad temporal y tiempos aproximados de ejecución de cada una de las implementaciones de los algoritmos RL.

Tabla 8. *Costos computacionales de algoritmos RL implementados*

Implementación	Plataforma de ejecución	Orden de complejidad temporal estimado	Tiempo aproximado de ejecución (minutos)
Monte Carlo	Jupyter notebook	O(n^2)	8
Monte Carlo WFV	Jupyter notebook	O(n^2)	25
Monte Carlo con acotación	Jupyter notebook	O(n^2)	15
Monte Carlo con acotación WFV	Jupyter notebook	O(n^2)	45
DDPG	Google colaboratory	O(n^3)	5
DDPG WFV	Google colaboratory	O(n^3)	15
RDPG	Google colaboratory	O(n^3)	6
RDPG WFV	Google colaboratory	O(n^3)	20

El algoritmo ARIMA cuenta con un tiempo aproximado de ejecución de 3 minutos, y su implementación WFV, consta de un tiempo de 5 minutos de ejecución.

2.2.4 Ejecución del diseño de experimentos

Para la evaluación de las predicciones de los parámetros LQI y RSSI de redes IEEE 802.15.4 y teniendo en cuenta los modelos de entrenamiento y evaluación: modelo propuesto 75-25 y WFV, se realizó el siguiente diseño de experimentos:

Tabla 9. *Planteamiento del diseño de experimentos*

Algoritmo	***Walk Forward Validation***			**Sin *Walk Forward Validation***			**Total**
	Número de implementaciones			**Número de implementaciones**			
	LQI	**RSSI**	**TOTAL**	**LQI**	**RSSI**	**TOTAL**	
ARIMA	13	13	26	13	13	26	52
Monte Carlo	13	13	26	13	13	26	52
Monte Carlo con acotación	13	13	26	13	13	26	52
DDPG	13	13	26	13	13	26	52
RDPG	13	13	26	13	13	26	52
TOTAL			130			130	260

2.2.4.1 Experimentos RSSI. Se comenzó la experimentación con los datos de la métrica RSSI con la estrategia 75-25 (también la llamaremos: estrategia sin *Walk Forward Validation*). Se prepararon todas las herramientas para la ejecución de estos experimentos, *jupyter notebook, Google Colaboratory*, y se comenzaron a ejecutar según el orden mostrado en la tabla 8 para los experimentos RSSI sin WFV: 13 experimentos con ARIMA, 13 experimentos con Monte Carlo, 13 experimentos con Monte Carlo con acotación, 13 experimentos con DDPG y 13 experimentos con RDPG. Se ejecutaron las celdas de importe de librerías, así como se seleccionó el nombre

del archivo de la serie de tiempo a evaluar, los cuales se encontraban en una carpeta con todos los conjuntos de datos preparados para la evaluación RSSI sin WFV. Seguidamente se ejecutaron todas las celdas siguientes, y al final del *notebook* se encontraron los resultados de las métricas MSE, RMSE y MAPE, de las cuales se tomaron el RMSE y el MAPE con 5 dígitos de significancia después de la coma, además de encontrar una gráfica ilustrativa de la predicción del modelo evaluado vs los datos reales. Las tablas de resultados se encuentran en el apartado de resultados de la sección 2.2.6.

2.2.4.2 Experimentos RSSI WFV. Para esta experimentación RSSI, igual que en las demás, se prepararon todas las herramientas para la ejecución de estos experimentos, *jupyter notebook, Google Colaboratory*, y se comenzaron a ejecutar según el orden mostrado en la tabla 8 para los experimentos RSSI WFV: 13 experimentos con ARIMA, 13 experimentos con Monte Carlo, 13 experimentos con Monte Carlo con acotación, 13 experimentos con DDPG y 13 experimentos con RDPG. A diferencia de los experimentos sin WFV, para el modelo implementado de WFV en los algoritmos DDPG y RDPG, por cada ventana de entrenamiento y predicción, al indexar los registros de entrenamiento con los proyectados, se exportaban en un archivo *.csv* , debido a que los modelos mencionados no pueden ser entrenados 2 (dos) veces en el mismo script, por lo que para los experimentos RSSI WFV con DDPG y RDPG cada algoritmo se dividió en 3 scripts, cumpliendo el siguiente funcionamiento:

Primer Script: Recibe datos del 0-55%, entrena el modelo, realiza predicciones del 55-65%, une los datos de entrenamiento con los proyectados, para un total de 0-65% del total de datos, y los exporta en archivo *.csv*.

Segundo Script: Recibe los datos exportados por el script anterior (0-65%), entrena el modelo, realiza predicciones del 65-75%, une los datos de entrenamiento con los proyectados, para un total de 0-75% del total de datos, y los exporta en archivo *.csv*.

Tercer Script: Recibe los datos exportados por el script anterior (0-75%), entrena el modelo, realiza predicciones del 75-100% y aplica y muestra los resultados de las métricas de predicción MSE, RMSE y MAPE, de las cuales se tomaron el RMSE y el MAPE con 5 dígitos de significancia después de la coma, además de encontrar una gráfica ilustrativa de la predicción del modelo evaluado vs los datos reales. Las tablas de resultados se encuentran en el apartado de resultados de la sección 2.2.6.

2.2.4.3 Experimentos LQI. Continuando la experimentación, esta vez con los datos de la métrica LQI sin WFV, Se prepararon todas las herramientas para la ejecución de estos experimentos, *jupyter notebook*, *Google Colaboratory*, y se comenzaron a ejecutar según el orden mostrado en la tabla 8 para los experimentos LQI WFV: 13 experimentos con ARIMA, 13 experimentos con Monte Carlo, 13 experimentos con Monte Carlo con acotación, 13 experimentos con DDPG y 13 experimentos con RDPG. Se ejecutaron las celdas de importe de librerías, así como se seleccionó el nombre del archivo de la serie de tiempo a evaluar, los cuales se encontraban en una carpeta con todos los conjuntos de datos preparados para la evaluación LQI WFV. Seguidamente se ejecutaron todas las celdas siguientes, y al final del notebook se encontraron los resultados de las métricas MSE, RMSE y MAPE, de las cuales se tomaron el RMSE y el MAPE con 5 dígitos de significancia después de la coma, además de encontrar una gráfica ilustrativa de la predicción del modelo evaluado vs los datos reales. Las tablas de resultados se encuentran en el apartado de resultados de la sección 2.2.6.

2.2.4.4 Experimentos LQI WFV. Para esta experimentación LQI, repitiendo el protocolo de experimentación, se prepararon todas las herramientas para la ejecución de estos experimentos, *jupyter notebook, Google Colaboratory,* y se comenzaron a ejecutar según el orden mostrado en la tabla 8 para los experimentos LQI WFV: 13 experimentos con ARIMA, 13 experimentos con Monte Carlo, 13 experimentos con Monte Carlo con acotación, 13 experimentos con DDPG y 13 experimentos con RDPG. Para los experimentos de LQI WFV con DDPG y RDPG cada algoritmo se dividió en 3 scripts, cumpliendo el siguiente funcionamiento:

Primer Script: Recibe datos del 0-55%, entrena el modelo, realiza predicciones del 55-65%, une los datos de entrenamiento con los proyectados, para un total de 0-65% del total de datos, y los exporta en archivo *.csv*.

Segundo Script: Recibe los datos exportados por el script anterior (0-65%), entrena el modelo, realiza predicciones del 65-75%, une los datos de entrenamiento con los proyectados, para un total de 0-75% del total de datos, y los exporta en archivo *.csv*.

Tercer Script: Recibe los datos exportados por el script anterior (0-75%), entrena el modelo, realiza predicciones del 75-100% y aplica y muestra los resultados de las métricas de predicción MSE, RMSE y MAPE, de las cuales se tomaron el RMSE y el MAPE con 5 dígitos de significancia después de la coma, además de encontrar una gráfica ilustrativa de la predicción del modelo evaluado vs los datos reales. Las tablas de resultados se encuentran en el apartado de resultados de la sección 2.2.6.

2.2.5 Tecnologías usadas

2.2.5.1 Entornos y lenguajes de desarrollo. Python es un lenguaje de programación muy poderoso de alto nivel, el cuál es muy utilizado en la analítica de datos, gracias a su potencia al trabajar grandes cantidades de datos, la mayoría de las aplicaciones de ML, DL y RL

se realizan en Python, aunque también existen otros lenguajes usados como R y Matlab, aunque no son tan populares como Python. Todos los algoritmos se trabajaron en su versión base 3.0. Los algoritmos contienen, en su mayoría, funciones, por lo que en el desarrollo del código se hacen llamados a estas para su ejecución.

Google Colaboratory también llamado "*Colab*" es un entorno de compilación virtual de Google que permite ejecutar y programar en Python desde un navegador con las ventajas de no requerir configuración, dar acceso gratuito limitado a GPUs (*Graphics Processing Unit*) y permitir contenido fácilmente. Se accedió a esta cuenta de Google *Colab* mediante una cuenta personal de Gmail. El sitio *Colab* no es una página estática, se trata de un entorno interactivo llamado cuaderno de *Colab* en el que se puede escribir y ejecutar código. En esta herramienta online se pueden trabajar todo tipo de algoritmos de ML (*Machine Learning*), tales como DL (*Deep Learning*), RL (*Reinforcement Learning*), entre otros (*Google Colaboratory*, s. f.). Para la ejecución de los experimentos de este proyecto, se usó *Colab* en horas de la madrugada y mañana (Hora colombiana), con el fin de evitar intermitencia y bajo rendimiento en la compilación de los códigos, debido a que los servidores de *Colab* son abiertos al público, en algunas ocasiones sufren de congestión por la cantidad de usuarios al mismo tiempo. En esta plataforma se implementaron y ejecutaron los algoritmos DDPG y RDPG con los modelos WFV y sin WFV debido a su gran tiempo de ejecución computacional y a los mejores recursos de alojamiento y ejecución que ofrece esta herramienta gratuita.

Jupyter Notebook es un compilador de Python y otros lenguajes de programación, tiene versiones online como la misma plataforma *Jupyter Notebook Online, Google Colaboratory*, entre otras, como además versiones de compilación local para diferentes sistemas operativos. En este proyecto se ejecutaron los algoritmos ARIMA, Monte Carlo y Monte Carlo con acotación,

con sus implementaciones WFV y sin WFV en Jupyter Notebook local, proveído para la plataforma Windows mediante *Anaconda*.

2.2.5.2 Tratamiento y visualización de datos. Durante la implementación de los algoritmos y sus estrategias de entrenamiento y evaluación, es necesario estar revisando visualmente los resultados parciales y completos de los algoritmos que se están ejecutando y las estructuras y comportamientos de los conjuntos de datos que están siendo utilizados, para ello, se usaron las siguientes herramientas:

Pandas es una librería del lenguaje Python, que es de gran utilidad para tomar los conjuntos de datos de diferentes formatos, *txt, csv, xls* y llevarlos al entorno de programación del lenguaje. También es útil para trabajar las series de tiempo y trabajar con grandes estructuras de datos.

Numpy es también otra librería del lenguaje Python, que permite trabajar con arreglos, operar entre ellos, y diferentes funcionalidades, las cuales fueron muy útiles en el desarrollo de la fase de experimentación.

Matplotlib se usó para la generación de gráficos de las series de tiempo, tanto de los datos originales como de las predicciones por cada algoritmo, además es útil para diagramas de cajas, visualizar correlaciones, entre muchas otras funcionalidades que ofrece esta librería de Python.

Excel es una herramienta de *Microsoft Office* muy útil para el tratamiento y visualización de datos, en esta herramienta se modificaron los conjuntos de datos para convertirlos a *.csv* junto con la herramienta **Bloc de notas,** además que las dos fueron muy útiles al momento de modificar los registros para poder ser usados en los algoritmos de DDPG y RDPG y el trabajo en diferentes scripts de sus implementaciones en la estrategia de WFV.

2.2.5.3 Aprendizaje por refuerzo. Para la implementación de los algoritmos de aprendizaje por refuerzo se usaron librerías y funciones especiales de código abierto, para más información sobre ellas, existe abundante información en los sitios web oficiales de Python.

Statsmodels es una librería que ayuda a implementar modelos estadísticos, de esta librería se obtuvo el modelo *SARIMAX*, además de ser junto con *SKlearn* el módulo encargado de obtener las métricas de evaluación de todos los experimentos realizados.

SKlearn es muy usada para hallar métricas en modelos de ML, además de ayudar a implementarlos.

Scipy es una biblioteca libre y de código abierto, la cual contiene módulos de optimización, integración, interpolación, entre otros. Para nuestro proyecto, Scipy se utilizó principalmente para generar distribuciones de números randómicos en los algoritmos de Monte Carlo y Monte Carlo con acotación.

TensorFlow es una biblioteca de software de uso abierto al público muy útil para la computación numérica, fue muy usada en los algoritmos DDPG y RDPG, en RDPG principalmente para la implementación de las *LSTM*, y en ella fue que se escribieron los códigos de estos algoritmos.

2.2.6 Resultados

En este contenido se clasificaron los resultados dependiendo de la métrica y la estrategia de entrenamiento y evaluación usada de la siguiente manera:

- Resultados RSSI
- Resultados RSSI WFV
- Resultados LQI
- Resultados LQI WFV

En cada sección se muestran las tablas de resultados de métricas de error de predicción de los 5 algoritmos utilizados con un conjunto de datos, por lo que al estar disponibles 13 conjuntos de datos LQI y 13 conjuntos de datos LQI para la implementación sin WFV, y 13 conjuntos de datos LQI y 13 conjuntos de datos LQI para la implementación WFV, se tiene un total de 52 conjuntos de datos, que equivale a principalmente 52 tablas de resultados en general. Además, en cada sección se adaptaron 3 tablas de valores mínimos, valores máximos y una tabla de promedios (en base a RMSE), lo que es igual a 12 tablas en general, para un total de 64 tablas de resultados. En la subsección de comparación, se muestran tablas ordenadas de los algoritmos con mejores resultados, clasificadas por métrica y estrategia de entrenamiento y evaluación.

Nota: Para comparar y evaluar resultados en diferentes conjuntos de datos, se usó la métrica MAPE, la cuál es porcentual, lo que ayuda a evaluar entre diferentes implementaciones con diferentes tipos de datos.

2.2.6.1 Resultados RSSI.

Tabla 10. *Resultados RSSI 10M - raw_data_run3--1*

Juego de datos: **raw_data_run3--1**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 10M	3.21691	30.45741%	9.76894	92.59853%	4.37529	40.38285%	0.66867	5.98773%	16.44834	100%

Tabla 11. *Resultados RSSI 15M - raw_data_run2--1*

Juego de datos: **raw_data_run2--1**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 15M	2.43715	15.08638%	15.50263	90.38495%	2.03100	11.25664%	0.44491	1.67739%	16.39410	100%

Tabla 12. *Resultados RSSI 20M - raw_data_run2--1*

Juego de datos: **raw_data_run2--1**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	2.24867	13.00864%	13.43000	80.00799%	4.17406	7.98408%	0.56442	3.32004%	19.16092	100%

Tabla 13. *Resultados RSSI 20M - raw_data_run2--2*

Juego de datos: **raw_data_run2--2**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	2.40345	18.40889%	15.46988	96.02412%	2.12467	12.52408%	0.99624	6.68115%	14.17286	98.57160%

Tabla 14. *Resultados RSSI 20M - raw_data_run2--3*

Juego de datos: **raw_data_run2--3**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	2.56788	24.42266%	15.70900	97.91166%	2.29420	17.14751%	1.42750	14.13443%	15.36491	100%

Tabla 15. *Resultados RSSI 20M - raw_data_run4--1*

Juego de datos: **raw_data_run4--1**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	2.30965	19.26691%	11.35468	93.06172%	3.83806	31.74531%	2.29928	16.49389%	18.14865	100%

Tabla 16. *Resultados RSSI 20M - raw_data_run4--2*

Juego de datos: **raw_data_run4--2**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	3.06086	51.75288%	11.65627	100%	4.39551	70.12148%	3.03820	52.09469%	6.28322	64.43409%

Tabla 17. *Resultados RSSI 20M - raw_data_run5--1*

Juego de datos: **raw_data_run5--1**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	2.81189	22.13066%	9.74295	76.54349%	4.27536	37.58958%	0.57608	3.75000%	18.41348	100%

Tabla 18. *Resultados RSSI 20M - raw_data_run5--2*

Juego de datos: **raw_data_run5--2**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	2.77937	22.03635%	10.89510	94.33683%	4.25757	37.43554%	0.51473	2.51510%	11.44742	100%

Tabla 19. *Resultados RSSI 20M - raw_data_run5--3*

Juego de datos: **raw_data_run5--3**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	3.49376	61.74070%	10.93758	99.13312%	4.81170	76.85677%	2.29376	45.82602%	13.65401	100%

Tabla 20. *Resultados RSSI 20M - raw_data_run7--1*

Juego de datos: **raw_data_run7--1**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	2.81927	25.09478%	9.37291	81.50322%	4.40416	40.96031%	1.47941	13.55476%	12.28702	100%

Tabla 21. *Resultados RSSI 20M - raw_data_run8--1*

Juego de datos: **raw_data_run8--1**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	2.64691	23.78332%	10.45085	91.35545%	4.38332	40.76966%	0.62048	5.61594%	19.72747	100%

Tabla 22. *Resultados RSSI 20M - raw_data_run8--2*

Juego de datos: **raw_data_run8--2**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	2.64925	22.48083%	10.64949	94.03216%	4.35301	39.39367%	0.57286	4.63891%	18.15004	100%

Valores mínimos en base a RMSE

Tabla 23. *Resultados mínimos RSSI*

Juego de datos: **Mínimos**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI	2.24867	13.00864%	9.37291	81.50322%	2.03100	11.25664%	0.44491	1.67739%	6.28322	64.43409%

Valores máximos en base a RMSE

Tabla 24. *Resultados máximos RSSI*

Juego de datos: **Máximos**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI	3.49376	61.74070%	11.65627	100%	4.81170	76.85677%	2.29928	16.49389%	19.72747	100%

Promedios en base a RMSE

Tabla 25. *Resultados promedios RSSI*

Juego de datos: **Promedio**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI	2,72624	26,89772%	11,91848	91,29948%	3,82445	35,70519%	1,19204	13,56079%	15,35788	97,15428%

2.2.6.2 Resultados RSSI WFV.

Tabla 26. *Resultados RSSI WFV 10M - raw_data_run3--1*

Juego de datos: **raw_data_run3--1**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 10M	1.68346	14.63194%	10.40737	100%	4.41612	41.33314%	0.58914	4.99735%	26.06609	100%

Tabla 27. *Resultados RSSI WFV 15M - raw_data_run2--1*

Juego de datos: **raw_data_run2--1**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 15M	4.46251	27.61556%	16.64039	99.99999%	2.62464	12.75499%	0.43717	1.56891%	29.90523	100%

Tabla 28. *Resultados RSSI WFV 20M - raw_data_run2--1*

Juego de datos: **raw_data_run2--1**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	1.43414	6.81179%	15.96674	99.7452%	2.37218	8.97790%	0.45242	2.19279%	22.53049	100%

Tabla 29. Resultados RSSI WFV 20M - raw_data_run2--2

Juego de datos: **raw_data_run2--2**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	1.70745	12.47475%	15.97620	100%	2.45440	13.10778%	1.01796	7.27647%	24.09491	100%

Tabla 30. *Resultados RSSI WFV 20M - raw_data_run2--3*

Juego de datos: **raw_data_run2--3**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	2.00350	18.99927%	15.94355	100%	2.54104	18.39375%	1.41086	13.24067%	27.76309	100%

Tabla 31. *Resultados RSSI WFV 20M - raw_data_run4--1*

Juego de datos: **raw_data_run4--1**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	3.91456	33.91662%	11.87887	100%	3.77570	31.79019%	2.19596	16.48773%	26.67837	100%

Tabla 32. *Resultados RSSI WFV 20M - raw_data_run4--2*

Juego de datos: **raw_data_run4--2**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	4.48953	77.14540%	11.75246	100%	4.35788	72.18702%	2.99163	52.10623%	29.79776	100%

Tabla 33. *Resultados RSSI WFV 20M - raw_data_run5--1*

Juego de datos: **raw_data_run5--1**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	4.48788	41.13181%	10.51309	93.38854%	4.26522	37.85984%	0.50698	2.71562%	28.75068	100%

Tabla 34. *Resultados RSSI WFV 20M - raw_data_run5--2*

Juego de datos: **raw_data_run5--2**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	4.49170	41.18273%	11.19127	100%	4.24617	37.44182%	0.59970	4.05241%	20.81212	100%

Tabla 35. *Resultados RSSI WFV 20M - raw_data_run5--3*

Juego de datos: **raw_data_run5--3**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	4.94184	84.44162%	11.00164	100%	4.82333	78.62906%	2.10836	38.96698%	26.36169	100%

Tabla 36. *Resultados RSSI WFV 20M - raw_data_run7--1*

Juego de datos: **raw_data_run7--1**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	4.70592	45.55112%	10.19483	92.47911%	4.40556	41.16844%	0.62697	4.06779%	22.19636	100%

Tabla 37. *Resultados RSSI WFV 20M - raw_data_run8--1*

Juego de datos: **raw_data_run8--1**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	4.65053	45.47634%	10.91764	99.99999%	4.38580	41.12902%	0.55793	3.46847%	25.82388	100%

Tabla 38. *Resultados RSSI WFV 20M - raw_data_run8--2*

Juego de datos: **raw_data_run8--2**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI 20M	4.69244	44.14232%	10.93473	100%	4.35342	39.51548%	0.59148	5.46100%	21.87040	100%

Valores mínimos en base a RMSE

Tabla 39. *Resultados mínimos RSSI WFV*

Juego de datos: **Mínimos**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI	1.43414	6.81179%	10.19483	92.47911%	2.37218	8.97790%	0.43717	1.56891%	20.81212	100%

Valores máximos en base a RMSE

Tabla 40. *Resultados máximos RSSI WFV*

Juego de datos: **Máximos**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI	4.94184	84.44162%	16.64039	99.99999%	4.82333	78.62906%	2.99163	52.10623%	29.90523	100%

Promedios en base a RMSE

Tabla 41. *Resultados promedios RSSI WFV*

Juego de datos: **Promedio**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
RSSI	3,66657	37,96317%	12,56298	98,89329%	3,77088	36,48372%	1,08358	12,04634%	25,58854	100%

2.2.6.3 Resultados LQI.

Tabla 42. *Resultados LQI 10M - raw_data_run1--1*

Juego de datos: **raw_data_run1--1**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 10M	1.33248	0.92646%	102.57147	94.99795%	3.74165	2.81926%	1.26962	0.87439%	104.31067	97.89696%

Tabla 43. *Resultados LQI 15M - raw_data_run1--1*

Juego de datos: **raw_data_run1--1**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 15M	1.71321	1.23773%	104.60596	99.03233%	20.95385	16.84399%	1.62390	1.14766%	46.95891	44.57496%

Tabla 44. *Resultados LQI 20M - raw_data_run1--1*

Juego de datos: **raw_data_run1--1**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	3.30462	2.35591%	103.75452	98.39739%	14.61106	11.28763%	1.97976	1.44680%	82.56728	78.59061%

Tabla 45. *Resultados LQI 20M - raw_data_run1--19*

Juego de datos: **raw_data_run1--19**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	4.05379	7.55196%	104.24850	98.97659%	16.80452	17.08197%	3.46476	6.95390%	88.37984	84.87839%

Tabla 46. *Resultados LQI 20M - raw_data_run1--20*

Juego de datos: **raw_data_run1--20**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	22.99032	100%	102.71754	100%	25.86660	100%	21.82820	100%	88.88585	100%

Tabla 47. *Resultados LQI 20M - raw_data_run1--21*

Juego de datos: **raw_data_run1--21**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	6.15063	28.47112%	104.15428	98.95484%	17.26186	35.09787%	5.84054	26.91324%	63.08463	69.64883%

Tabla 48. *Resultados LQI 20M - raw_data_run1--22*

Juego de datos: **raw_data_run1--22**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	17.65789	100%	103.34400	100%	22.69303	100%	16.76637	100%	85.23612	100%

Tabla 49. *Resultados LQI 20M - raw_data_run1--23*

Juego de datos: **raw_data_run1--23**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	3.12720	2.22939%	104.20918	98.91196%	16.93851	13.28980%	2.08603	1.38949%	78.47717	74.69082%

Tabla 50. *Resultados LQI 20M - raw_data_run1--24*

Juego de datos: **raw_data_run1--24**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	17.60396	100%	103.37793	99.45563%	22.64897	100%	16.38632	100%	95.12617	100%

Tabla 51. *Resultados LQI 20M - raw_data_run1--25*

Juego de datos: **raw_data_run1-25**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	4.65436	12.76640%	104.16240	98.95564%	17.41457	22.00191%	4.65146	12.74605%	70.11800	70.02882%

Tabla 52. *Resultados LQI 20M - raw_data_run1--26*

Juego de datos: **raw_data_run1--26**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	17.68026	100%	103.31574	100%	22.81807	100%	17.11737	100%	72.53956	100%

Tabla 53. *Resultados LQI 20M - raw_data_run1--27*

Juego de datos: **raw_data_run1--27**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	5.13706	17.92188%	104.03758	98.79226%	17.48360	25.54128%	5.47458	18.11905%	72.05198	73.20348%

Tabla 54. *Resultados LQI 20M - raw_data_run1--28*

Juego de datos: **raw_data_run1--28**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	18.07983	100%	103.31890	99.41439%	22.65832	100%	17.13129	100%	83.07011	100%

Valores mínimos en base a RMSE

Tabla 55. *Resultados mínimos LQI*

Juego de datos: **Mínimos**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI	1.33248	0.92646%	102.71754	100%	3.74165	2.81926%	1.26962	0.87439%	46.95891	44.57496%

Valores máximos en base a RMSE

Tabla 56. *Resultados máximos LQI*

Juego de datos: **Máximos**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI	22.99032	100%	104.60596	99.03233%	25.86660	100%	21.82820	100%	104.31067	97.89696%

Promedios en base a RMSE

Tabla 57. *Resultados promedios LQI*

Juego de datos: **Promedio**	ARIMA sin WFV		Monte Carlo sin WFV		Monte Carlo con acotación sin WFV		DDPG sin WFV		RDPG sin WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI	9,49889	44,11237%	103,67830	98,91453%	18,60727	49,53567%	8,89386	43,81466%	79,29279	84,11637%

2.2.6.4 Resultados LQI WFV.

Tabla 58. *Resultados LQI WFV 10M - raw_data_run1--1*

Juego de datos: **raw_data_run1--1**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 10M	1.36443	0.88920%	106.55127	100%	3.80227	2.84162%	1.25951	0.9%	70.16911	65.84452%

Tabla 59. *Resultados LQI WFV 15M - raw_data_run1--1*

Juego de datos: **raw_data_run1--1**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 15M	1.73462	1.24571%	105.27208	100%	21.09155	17.09056%	1.63898	1.20318%	85.06114	80.79199%

Tabla 60. *Resultados LQI WFV 20M - raw_data_run1--1*

Juego de datos: **raw_data_run1--1**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	3.27784	2.34979%	105.02460	100%	14.83130	11.44560%	2.29575	1.52822%	79.95881	76.11024%

Tabla 61. *Resultados LQI WFV 20M - raw_data_run1--19*

Juego de datos: **raw_data_run1--19**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	3.97899	7.51932%	105.00946	100%	17.10136	17.08770%	3.17324	6.54169%	84.47134	81.36756%

Tabla 62. *Resultados LQI WFV 20M - raw_data_run1--20*

Juego de datos: **raw_data_run1--20**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	24.66240	100%	105.00000	100%	25.79679	100%	21.08292	100%	75.10382	100%

Tabla 63. *Resultados LQI WFV 20M - raw_data_run1--21*

Juego de datos: **raw_data_run1--21**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	6.20513	28.58737%	104.90815	100%	17.52501	35.50204%	5.67925	26.98936%	85.84375	85.99341%

Tabla 64. *Resultados LQI WFV 20M - raw_data_run1--22*

Juego de datos: **raw_data_run1--22**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	18.71879	100%	105.00000	100%	22.65204	100%	16.71757	100%	70.03926	100%

Tabla 65. *Resultados LQI WFV 20M - raw_data_run1--23*

Juego de datos: **raw_data_run1--23**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	3.13734	2.23320%	105.03444	100%	17.10628	13.37205%	1.88862	1.41316%	66.73932	63.45427%

Tabla 66. *Resultados LQI WFV 20M - raw_data_run1--24*

Juego de datos: **raw_data_run1--24**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	18.72036	100%	105.00000	100%	22.74291	100%	16.45754	100%	73.19367	100%

Tabla 67. *Resultados LQI WFV 20M - raw_data_run1--25*

Juego de datos: **raw_data_run1--25**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	4.64253	12.77747%	104.93798	100%	17.59663	23.57780%	4.11705	11.88586%	60.45100	61.74437%

Tabla 68. *Resultados LQI WFV 20M - raw_data_run1--26*

Juego de datos: **raw_data_run1--26**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	18.84065	100%	105.00000	100%	22.84533	100%	16.98652	100%	68.32104	100%

Tabla 69. *Resultados LQI WFV 20M - raw_data_run1--27*

Juego de datos: **raw_data_run1--27**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	5.15992	17.95377%	104.96691	100%	17.89558	25.02071%	4.68570	16.91523%	72.05198	73.20348%

Tabla 70. *Resultados LQI WFV 20M - raw_data_run1--28*

Juego de datos: **raw_data_run1--28**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI 20M	18.97232	100%	105.00000	100%	22.48868	100%	17.24328	100%	83.07011	100%

Valores mínimos en base a RMSE

Tabla 71. *Resultados mínimos LQI WFV*

Juego de datos: **Mínimos**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI	1.36443	0.88920%	104.90815	100%	3.80227	2.84162%	1.25951	0.9%	60.45100	61.74437%

Valores máximos en base a RMSE

Tabla 72. *Resultados máximos LQI WFV*

Juego de datos: **Máximos**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI	24.66240	100%	106.55127	100%	25.79679	100%	21.08292	100%	85.84375	85.99341%

Promedios en base a RMSE

Tabla 73. *Resultados promedios LQI WFV*

Juego de datos: **Promedio**	ARIMA WFV		Monte Carlo WFV		Monte Carlo con acotación WFV		DDPG WFV		RDPG WFV	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
LQI	9,95502	44,11967%	105,13114	100%	18,72890	49,68754%	8,70968	43,64436%	74,95956	83,73152%

2.2.6.5 Comparación. Las tablas 74 a 80 muestran comparaciones de las métricas de error en las implementaciones de los algoritmos de predicción de LQI y RSSI.

Tabla 74. *Comparativa de resultados RSSI sin WFV en base a RMSE*

Algoritmo	RMSE promedio
DDPG sin WFV	1,19204
ARIMA sin WFV	2,72624
Monte Carlo con acotación sin WFV	3,82445
Monte Carlo sin WFV	11,91848
RDPG sin WFV	15,35788

Tabla 75. *Comparativa de resultados RSSI WFV en base a RMSE*

Algoritmo	RMSE promedio
DDPG WFV	1,08358
ARIMA WFV	3,66657
Monte Carlo con acotación WFV	3,77088
Monte Carlo WFV	12,56298
RDPG WFV	25,58854

Tabla 76. *Comparativa de resultados RSSI en base a RMSE*

Algoritmo	RMSE promedio	MAPE promedio
DDPG WFV	1,08358	12,04634%
DDPG sin WFV	1,19204	13,56079%
ARIMA sin WFV	2,72624	37,96317%
ARIMA WFV	3,66657	26,89772%
Monte Carlo con acotación WFV	3,77088	36,48372%
Monte Carlo con acotación sin WFV	3,82445	35,70519%
Monte Carlo sin WFV	11,91848	91,29948%
Monte Carlo WFV	12,56298	98,89329%
RDPG sin WFV	15,35788	97,15428%
RDPG WFV	25,58854	100%

Tabla 77. *Comparativa de resultados LQI sin WFV en base a RMSE*

Algoritmo	**RMSE promedio**
DDPG sin WFV	8,89386
ARIMA sin WFV	9,49889
Monte Carlo con acotación sin WFV	18,60727
RDPG sin WFV	79,29279
Monte Carlo sin WFV	103,67830

Tabla 78. *Comparativa de resultados LQI WFV en base a RMSE*

Algoritmo	**RMSE promedio**
DDPG WFV	8,70968
ARIMA WFV	9,95502
Monte Carlo con acotación WFV	18,72890
RDPG WFV	74,95956
Monte Carlo WFV	105,13114

Tabla 79. *Comparativa de resultados LQI en base a RMSE*

Algoritmo	RMSE promedio	MAPE promedio
DDPG WFV	8,70968	43,64436%
DDPG sin WFV	8,89386	43,81466%
ARIMA sin WFV	9,49889	44,11237%
ARIMA WFV	9,95502	44,11967%
Monte Carlo con acotación sin WFV	18,60727	49,53567%
Monte Carlo con acotación WFV	18,72890	49,68754%
RDPG WFV	74,95956	83,73152%
RDPG sin WFV	79,29279	84,11637%
Monte Carlo sin WFV	103,67830	98,91453%
Monte Carlo WFV	105,13114	100%

Tabla 80. *Comparativa de mejores resultados en general en base a MAPE*

Algoritmo	MAPE promedio
RSSI – DDPG WFV	12,04634%
LQI – DDPG WFV	43,64436%

2.2.6.6 Discusión. A partir de las tablas 7, 76 y 79, se encuentra que los datos iniciales de RSSI tienen una menor varianza que los datos LQI, y que los resultados de las métricas RMSE y MAPE en los experimentos, tienden a ser mejores para RSSI que para LQI. Además, los datos iniciales con mayor varianza, al momento de ser proyectados, pierden similitud con los datos originales, y esto se visualiza en el aumento de la métrica de error MAPE.

El algoritmo DDPG aplicando WFV, es el algoritmo más eficiente en la proyección de LQI y RSSI, según la tabla 76 y 79, ya que obtiene las 2 menores métricas de error RMSE. A su vez, según la tabla 80, la implementación DDPG WFV con el conjunto de datos RSSI obtuvo los mejores resultados, al obtener un MAPE casi 3 veces menor que LQI.

La estrategia de entrenamiento y evaluación WFV, no favorece a las implementaciones para LQI y RSSI del algoritmo ARIMA, ya que, los resultados de las tablas 76 y 79 muestran que la estrategia 75-25 obtiene mejores resultados.

Entre los algoritmos implementados de aprendizaje por refuerzo DDPG, RDPG, Monte Carlo y Monte Carlo con acotación y sus aplicaciones WFV, según las tablas 76 y 79, solamente el algoritmo DDPG, obtuvo mejores resultados que el algoritmo ARIMA, el cual es tomado como referencia en la proyección de series de tiempo. A su vez, el de menor tiempo de ejecución entre estos algoritmos es el DDPG sin WFV.

Estudiando los algoritmos basados en el método Monte Carlo (Monte Carlo y Monte Carlo con acotación) y sus implementaciones WFV, el algoritmo de Monte Carlo con acotación obtiene menores resultados de las métricas de error RMSE y MAPE, según lo indican las tablas 76 y 79.

Al observar los resultados de las implementaciones de las estrategias de entrenamiento y evaluación, se nota que el impacto del WFV es neutro, ya que en comparación con la estrategia

75-25, según las tablas 76 y 79, 4 de 10 comparaciones de resultados entre WFV y 75-25 mediante la métrica RMSE, son a favor de las implementaciones usando WFV, al obtener valores de error más bajos. Continuando con el WFV, este modelo tuvo un impacto neutro en cuanto a resultados, pero su tiempo de ejecución computacional es aproximadamente *n* veces mayor que la implementación 75-25 (véase tabla 8), siendo *n* el número de ventanas de datos usadas para sus entrenamientos y predicciones de WFV.

Al estudiar la función f(n) (Velocidad de crecimiento u orden de complejidad temporal) y su aproximación en los algoritmos implementados (ver tabla 8), se observa que los algoritmos DDPG y RDPG tienen un orden de complejidad mayor (O(n^3)) que los algoritmos Monte Carlo y Monte Carlo con acotación (O(n^2)), pero DDPG y RDPG cuentan con menores tiempos de ejecución que los algoritmos Monte Carlo, esto es debido a que al momento de hallar f(n) se eliminan constantes multiplicativas y aditivas asociadas al número de iteraciones de sus funcionalidades, las cuales para los algoritmos Monte Carlo, según su descripción (secciones 2.1.3 y 2.1.4) son en el rango de diez miles, y para DDPG y RDPG, están en el rango de miles.

Según la tabla comparativa 8, los tiempos de ejecución de los algoritmos implementados de menor a mayor están dados en el siguiente orden: DDPG, RDPG, Monte Carlo, DDPG WFV, Monte Carlo con acotación, RDPG WFV, Monte Carlo WFV, Monte Carlo con acotación WFV. El tiempo de ejecución promedio aproximado del algoritmo ARIMA es de 3 minutos y su implementación WFV de unos 5 minutos, por lo que al comparar el algoritmo más rápido de los RL implementados (DDPG, 5 minutos) contra ARIMA, el algoritmo ARIMA se clasifica como el más rápido en proyectar predicciones de series de tiempo. Además, se considera que el algoritmo de Monte Carlo con acotación WFV es el que demora más tiempo en predecir las series de tiempo (45 minutos).

Para las implementaciones 75-25 (sin WFV), con los conjuntos de datos LQI y RSSI, el algoritmo con la métrica de error RMSE más baja es el algoritmo DDPG, seguido de ARIMA, como lo indican las tablas 74 y 77.

Los algoritmos con errores más altos en la experimentación, según su estrategia de entrenamiento y evaluación y el conjunto de datos usado se clasifican de la siguiente manera según las tablas 74, 75, 77 y 78: error más alto para LQI WFV: Monte Carlo; error más alto para RSSI WFV: RDPG; error más alto para LQI sin WFV: Monte Carlo; error más alto para RSSI sin WFV: RDPG.

3 Conclusiones

A menor varianza de los datos iniciales LQI y RSSI, menor es la tasa de error en los resultados de los experimentos.

Al aumentar la varianza de un conjunto de datos, el comportamiento de las predicciones pierde similitud con respecto a los datos originales, lo que produce un aumento en la métrica de error MAPE.

La implementación del algoritmo DDPG con WFV alimentado con datos RSSI, tiene las métricas de error más bajas.

El algoritmo DDPG aplicando WFV, es el algoritmo con mejor predicción en la proyección de LQI y RSSI.

La estrategia de entrenamiento y evaluación WFV, no favorece a las implementaciones para LQI y RSSI del algoritmo ARIMA.

Entre los algoritmos de aprendizaje por refuerzo DDPG, RDPG, Monte Carlo y Monte Carlo con acotación, y sus implementaciones WFV, solamente el algoritmo DDPG obtuvo mejores resultados que el algoritmo ARIMA en la proyección de LQI y RSSI.

Entre los algoritmos de aprendizaje por refuerzo DDPG, RDPG, Monte Carlo y Monte Carlo con acotación, y sus implementaciones WFV, el que se ejecutó en menor tiempo es el DDPG sin WFV.

Entre los algoritmos Monte Carlo y Monte Carlo con acotación y sus implementaciones WFV, el algoritmo de Monte Carlo con acotación genera menores métricas de error para la proyección de LQI y RSSI.

Sin considerar las tasas de error, el algoritmo que ejecuta las proyecciones de RSSI y LQI en menos tiempo es ARIMA sin WFV.

El algoritmo que obtuvo tasas más bajas de error entre las implementaciones sin WFV es DDPG para la proyección de LQI y RSSI.

El algoritmo que obtuvo los valores más altos de error entre las implementaciones WFV es Monte Carlo para la proyección de LQI.

El algoritmo que obtuvo los valores más altos de error entre las implementaciones WFV es RDPG para la proyección de RSSI.

El algoritmo que obtuvo los valores más altos de error entre las implementaciones sin WFV es Monte Carlo para la proyección de LQI.

El algoritmo que obtuvo los valores más altos de error entre las implementaciones sin WFV es RDPG para la proyección de RSSI.

4 Recomendaciones

Se recomienda:

La exploración, implementación y evaluación de otros algoritmos de aprendizaje por refuerzo tales como SGA (*Stochastic Gradient Ascent*), GRU (*Gated Recurrent Unit*) y DQN (*Deep Q-Network*).

La experimentación y análisis de los algoritmos utilizados en este trabajo de grado (Monte Carlo, Monte Carlo con acotación, DDPG, RDPG) para la proyección de otras métricas de los enlaces, como ETX.

La experimentación y análisis de los algoritmos utilizados en este trabajo de grado (Monte Carlo, Monte Carlo con acotación, DDPG, RDPG) con mayor cantidad de ventanas y con más diversidad en los tamaños de ventanas en la estrategia WFV, para evaluar la relación costo beneficio entre el tiempo de ejecución y la calidad de las proyecciones.

.

Referencias Bibliográficas

Evaluating forecast accuracy | Forecasting: Principles and Practice (2nd ed). (s. f.). Recuperado 21 de junio de 2021, de https://Otexts.com/fpp2/

Aprendizaje por refuerzo: DeepQ-Learning. (s. f.). Víctor Noriega. Recuperado 7 de junio de 2021, de https://victornoriega.github.io/aprendizaje-refuerzo/

Brownlee, J. (2017, marzo 30). Simple Time Series Forecasting Models to Test So That You Don't Fool Yourself. *Machine Learning Mastery*. https://machinelearningmastery.com/simple-time-series-forecasting-models/

Castillo Gamarra, J. E. (2014). Modelación de la volatilidad del índice general de la Bolsa de Valores de Lima, periodo 2009-2011. *Universidad Nacional Agraria La Molina*. http://repositorio.lamolina.edu.pe/handle/UNALM/2276

Daboín, Á., Verde, G., Anzola, F. T., & Gharbi, T. (2012). Medición de RSSI, LQI y pruebas de cobertura para diferentes escenarios de propagación en una red inalámbrica de sensores. *Revista Digital de Investigación y Postgrado*, *2*(1), 3.

Duca, A. L. (2021, septiembre 10). *Understanding the Seasonal Order of the SARIMA Model*. Medium. https://towardsdatascience.com/understanding-the-seasonal-order-of-the-sarima-model-ebef613e40fa

Farahani, S. (2008). Chapter 3—ZigBee and IEEE 802.15.4 Protocol Layers. En S. Farahani (Ed.), *ZigBee Wireless Networks and Transceivers* (pp. 33-135). Newnes. https://doi.org/10.1016/B978-0-7506-8393-7.00003-0

Fernández, G. C. (s. f.). *Reinforcement learning como reacción frente a anomalías en la red*. 102.

Fu, S., Zhang, Y., Jiang, Y., Hu, C., Shih, C.-Y., & Marrón, P. J. (2015). Experimental Study for Multi-layer Parameter Configuration of WSN Links. *2015 IEEE 35th International Conference on Distributed Computing Systems*, 369-378. https://doi.org/10.1109/ICDCS.2015.45

Garcia Algora, C. M., Alfonso Reguera, V., Deligiannis, N., & Steenhaut, K. (2017). Review and Classification of Multichannel MAC Protocols for Low-Power and Lossy Networks. *IEEE Access*, *5*, 19536-19561. https://doi.org/10.1109/ACCESS.2017.2748178

Google Colaboratory. (s. f.). Recuperado 8 de junio de 2021, de https://colab.research.google.com/notebooks/welcome.ipynb?hl=es

Gupta, M. (2021, mayo 30). *Monte Carlo for Reinforcement Learning with example*. Medium. https://medium.com/data-science-in-your-pocket/monte-carlo-for-reinforcement-learning-with-example-1754439dd628

Henderson, T., & Kotz, D. (2015). Data Citation Practices in the CRAWDAD Wireless Network Data Archive. *D-Lib Magazine*, *21*(1/2). https://doi.org/10.1045/january2015-henderson

jecrespom. (s. f.). *IEEE 802.15.4*. Aprendiendo Arduino. Recuperado 13 de abril de 2021, de https://aprendiendoarduino.wordpress.com/tag/ieee-802-15-4/

Liu, T., Tan, Z., Xu, C., Chen, H., & Li, Z. (2020). Study on deep reinforcement learning techniques for building energy consumption forecasting. *Energy and Buildings*, *208*, 109675. https://doi.org/10.1016/j.enbuild.2019.109675

LQI (Link Quality Indicator). (2016, noviembre 16). *bou-man.es*. https://bou-man.es/productos/ikusi-telecontrol/opciones-de-personalizacion/lqi-link-quality-indicator/

Making AI Experience Many Trials to Achieve High-level Behavior. (2018, octubre 26). 人工知能事業｜株式会社グリッド. https://gridpredict.jp/our_service_en/making-ai-experience-many-trials-to-achieve-high-level-behavior/

Mantilla, J. (2020). *Juandmantilla/Prediccion-LQI-RSSI-series-de-tiempo* [Jupyter Notebook]. https://github.com/juandmantilla/Prediccion-LQI-RSSI-series-de-tiempo (Original work published 2020)

Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). (s. f.). Recuperado 9 de junio de 2021, de http://www.eumetrain.org/data/4/451/english/msg/ver_cont_var/uos3/uos3_ko1.htm

Quintana, M. J. G., & Jiménez, S. A. M. (2016). Modelos de series de tiempo aplicados a los expedientes de la Comisión de Derechos Humanos del Distrito Federal. *Economía Informa*, *398*, 89-99. https://doi.org/10.1016/j.ecin.2016.04.007

Romero, C. A. V., Jaimes, J. E. B., Carolina, D., & González, P. (s. f.). *La Tecnología ZigBee estudio de las características de la capa física*. 10.

Roy, B. (2020, febrero 23). *Monte Carlo Learning*. Medium. https://towardsdatascience.com/monte-carlo-learning-b83f75233f92

Sáez, J. I. L. (2022). *Análisis de Series de Tiempo*. 50.

Series de tiempo (Cadena de suministro). (s. f.). Recuperado 16 de septiembre de 2021, de https://www.lokad.com/es/series-de-tiempo-en-cadena-de-suministro

Stock Prediction with ML: Walk-forward Modeling—The Alpha Scientist. (s. f.). Recuperado 21 de junio de 2021, de https://alphascientist.com/walk_forward_model_building.html

Team, K. (s. f.). *Keras documentation: Deep Deterministic Policy Gradient (DDPG)*. Recuperado 12 de mayo de 2021, de https://keras.io/examples/rl/ddpg_pendulum/

Villavicencio, J. (s. f.). *Introducción a Series de Tiempo*. 33.

Yoon, C. (2019, mayo 23). *Deep Deterministic Policy Gradients Explained*. Medium. https://towardsdatascience.com/deep-deterministic-policy-gradients-explained-2d94655a9b7b

Yuan, D., Kanhere, S. S., & Hollick, M. (2017). Instrumenting Wireless Sensor Networks—A survey on the metrics that matter. *Pervasive and Mobile Computing*, *37*, 45-62. https://doi.org/10.1016/j.pmcj.2016.10.001

【强化学习】DPG，DQN与DDPG. (s. f.). 知乎专栏. Recuperado 7 de junio de 2021, de https://zhuanlan.zhihu.com/p/337976595

Apéndice

Apéndice A. Repositorio del proyecto

Para la construcción continua de conocimiento, y siguiendo las normas éticas de software libre para todos, se ha decidido compartir un repositorio en la plataforma *GitHub* con lo siguiente: Códigos fuentes de algoritmos implementados, conjuntos de datos, scripts de evaluación de características de los datos e instrucciones para el uso de estas herramientas, con el fin de que sea reutilizado y sirva de ayuda y motivación en futuros avances del aprendizaje por refuerzo.

Enlace del repositorio: https://github.com/BitanCepeda/Reinforcement-Learning-for-time-series-forecasting

Apéndice B. Repositorio de algoritmos base DDPG y RDPG

Con el fin de dar reconocimiento a los autores de los algoritmos DDPG y RDPG indispensables para este trabajo de grado y posibles terceros que hayan adaptado su funcionamiento para ser funcionales en la predicción de series de tiempo, se comparte el repositorio en la plataforma *GitHub* donde se encuentra el código fuente base del cual se adaptó las implementaciones estudiadas en este trabajo de grado.

Enlace del repositorio: https://github.com/ChefLiutao/Time-series-forecasting-via-deep-reinforcement-learning