

RAE (Resumen Analítico Estructurado)

1. **TIPO DE DOCUMENTO:** Trabajo de grado para optar por el título de Ingeniero De sistemas.
2. **TÍTULO:** Simulador de datos climáticos para el desarrollo de proyectos IoT.
3. **AUTORES:** Camilo Andres Diaz Gomez, Juan Esteban Contreras Diaz, Jhonatan Mauricio Villarreal Corredor.
4. **LUGAR:** Bogotá, D.C
5. **FECHA:** Junio de 2023.
6. **PALABRAS CLAVE:** Redes IoT, Ecosistemas tecnológicos, Sensores, Datos ambientales, Modelo de simulación, Machine Learning, Inteligencia artificial, Predicción de datos.
7. **DESCRIPCIÓN DEL TRABAJO:** El proyecto "Simulador de datos climáticos para proyectos IoT" tiene como objetivo generar un modelo que simule datos IoT mediante programación. Utilizando información climática obtenida de fuentes confiables como Datos Abiertos Colombia y con controles de calidad, se busca facilitar las pruebas de proyectos IoT en una región específica de Colombia. Este simulador empleará técnicas de Machine Learning y datos tabulares extraídos de fuentes abiertas. Entrenando modelos con datos climáticos existentes, se capturarán patrones y relaciones para generar datos simulados que representen el comportamiento real del clima en dicha región. La generación de datos simulados permitirá realizar pruebas y experimentos en proyectos IoT sin depender de datos en tiempo real. Además, se podrán simular diversos escenarios climáticos y ajustar parámetros para evaluar el rendimiento de los proyectos IoT bajo diferentes condiciones.
8. **LÍNEAS DE INVESTIGACIÓN:** Línea de Investigación de la USB: Tecnologías actuales y Sociedad.
9. **METODOLOGÍA:** Es de carácter estadístico - experimental.
10. **CONCLUSIONES:** El modelo de Machine Learning desarrollado para predecir variables climáticas y utilizarlos en el desarrollo de proyectos IoT ha demostrado ser una herramienta eficiente y precisa para analizar y pronosticar el comportamiento del clima. Los resultados obtenidos a través de la evaluación de los diferentes modelos han sido evaluados con diferentes métricas de error y han arrojado resultados cercanos a los datos reales. En general, el modelo de Red Neuronal ha mostrado la mayor precisión en las predicciones, seguido por los modelos de árboles aleatorios y árboles de decisiones, pero la elección del modelo dependerá de los datos específicos y las necesidades del usuario. Este proyecto puede contribuir a la facilitación de pruebas de proyectos IoT en la región de Colombia, lo que podría tener un impacto positivo en la implementación de proyectos de este tipo de proyectos en el país. Además, la metodología y los resultados obtenidos en este proyecto pueden ser utilizados como base para futuras investigaciones relacionadas con la simulación de datos IoT en diferentes contextos y para diferentes aplicaciones.

Simulador de datos climáticos para el desarrollo de proyectos IoT

Camilo Andres Diaz Gomez - 30000050164

Juan Esteban Contreras Diaz - 30000048764

Jhonatan Mauricio Villarreal Corredor - 30000051809

Trabajo de Grado presentado para optar al título de Ingeniero de Sistemas

Asesor: Nikolay Lenin Reyes Jalizev, Magíster (MSc) en Ingeniería Industrial



Universidad de San Buenaventura

Facultad de Ingeniería (Bogotá)

Ingeniería de Sistemas

Bogotá D.C., Colombia

2023

Referencia/Reference	Díaz, C.A; Contreras, J.E y Villarreal, J. M. (2023). <i>Simulador de datos climáticos para el desarrollo de proyectos IoT</i> . [Trabajo de grado profesional].
Estilo/Style:	Universidad de San Buenaventura Bogotá.
APA 7ma ed. (2020)	



Biblioteca Digital (Repositorio)
www.bibliotecadigital.usb.edu.co

Bibliotecas Universidad de San Buenaventura

Biblioteca Fray Alberto Montealegre O.F.M. - Bogotá.

Biblioteca Fray Arturo Calle Restrepo O.F.M. - Medellín, Bello, Armenia, Ibagué.

Departamento de Biblioteca - Cali.

Biblioteca Central Fray Antonio de Marchena – Cartagena.

Universidad de San Buenaventura Colombia - www.usb.edu.co

Bogotá - www.usbbog.edu.co

Medellín - www.usbmed.edu.co

Cali - www.usbcali.edu.co

Cartagena - www.usbctg.edu.co

Editorial Bonaventuriana - www.editorialbonaventuriana.usb.edu.co

Revistas científicas – www.revistas.usb.edu.co

Dedicatoria

Dedicamos nuestra tesis a nuestros padres quienes con su sacrificio y esfuerzo nos dieron la carrera para nuestro futuro contribuyendo al desarrollo personal y profesional, por creer en nuestra capacidad, brindarnos su comprensión, amor y cariño.

De igual manera a nuestros hermanos, quienes nos han apoyado constantemente y nos han inspirado a superarnos cada día y luchar por nuestro futuro.

También a los compañeros, tanto de carrera y en especial, los integrantes y autores de esta tesis, con quienes hemos compartido conocimiento, experiencias, alegrías, tristezas y apoyo que nos ha llevado a cumplir este sueño.

De igual modo, a nuestro tutor de tesis, quien nos ha contribuido con su conocimiento en el desarrollo del y aportando en el proyecto, tanto la documentación

Y, por último, pero no menos importante, a nuestro exprofesor Andres Armando Sánchez Martin, quien fue el que nos dio la idea, contribuyo en el desarrollo de esta y nos acompañó en gran parte del desarrollo del proyecto desde el semillero Cábalas de Software.

Agradecimientos

Inicialmente, doy gracias a Dios por permitirme cumplir este sueño, a mis compañeros, con quienes he compartido experiencias extraordinarias, a mi universidad por transferir en mi todo el conocimiento necesario para llegar a ser el profesional que me apasiona, a mis profesores quienes integralmente hicieron parte de mi formación tanto personal como profesional.

Finalmente, agradezco a quien lee el presente documento, por permitir compartir mis experiencias, investigaciones y conocimiento.

Camilo Andres Diaz Gomez.

Tabla de contenido

Resumen	12
Abstract.....	14
Introducción.....	16
Planteamiento del Problema	17
Antecedentes.....	18
Justificación	21
Objetivos.....	23
Objetivo General.....	23
Objetivos Específicos	23
Marco Teórico	24
Internet of Things (IoT).....	24
Análítica de Datos	34
Simulación	41
Machine Learning.....	50
Metodología.....	53
Compresión del negocio	55
Entendimiento de los datos.....	58
Preparación de los datos	58
Modelado	60
Evaluación	61
Despliegue	62
Resultados.....	66
Análisis exploratorio y desarrollo de los modelos de Machine Learning	66
Resultados obtenidos	96

Conclusiones.....	103
Recomendaciones	105
Referencias	106
Anexos	114

Lista de Tablas

Tabla 1 <i>Etapas para realizar cadena de valor</i>	36
Tabla 2 Cronograma de actividades	63
Tabla 3 <i>Promedio de datos diarios del mes de septiembre</i>	98

Lista de Figuras

Figura 1 <i>Árbol de Problema</i>	22
Figura 2 <i>Modelo general de la arquitectura de una red IoT</i>	28
Figura 3 <i>Funcionamiento de actuadores IoT</i>	32
Figura 4 <i>Cadena de valor</i>	36
Figura 5 <i>Estructura de la metodología CRISP-DM</i>	54
Figura 6 <i>Estadísticas del dataset</i>	67
Figura 7 <i>Código de estaciones</i>	68
Figura 8 <i>Generador de histogramas</i>	68
Figura 9 <i>Diagramas de correlación por variable</i>	69
Figura 10 <i>Calcular matriz de correlación</i>	70
Figura 11 <i>Generador mapa de calor</i>	71
Figura 12 <i>Virtualización de correlaciones</i>	71
Figura 13 <i>Filtración de variables de alta correlación</i>	72
Figura 14 <i>Variables con más relación</i>	73
Figura 15 <i>Variables con menos relación</i>	73
Figura 16 <i>Creación de subconjuntos para entrenamiento</i>	74
Figura 17 <i>Preparación del dataframe para entrenamiento</i>	75
Figura 18 <i>Creación de variables</i>	76
Figura 19 <i>Entrenamiento del modelo</i>	76
Figura 20 <i>Entrenamiento de modelo de regresión lineal</i>	78
Figura 21 <i>Evaluación del modelo</i>	79
Figura 22 <i>Inicializador de contador e iteración de filas</i>	81
Figura 23 <i>Implementación del modelo de Regresión Polinomial y evalúan su precisión mediante el coeficiente de determinación</i>	82
Figura 24 <i>Creación de columnas con nuevos datos</i>	84
Figura 25 <i>Modelo de Machine Learning KNN con regresión</i>	84
Figura 26 <i>Búsqueda del mejor k</i>	85
Figura 27 <i>Obtención del mejor k</i>	86
Figura 28 <i>Entrenamiento del modelo KNN con regresión</i>	86

Figura 29 <i>Modelo de Machine Learning arboles de decisión con regresión</i>	88
Figura 30 <i>Modelo de Machine Learning Random Tree con regresión</i>	89
Figura 31 <i>Estandarizar el conjunto de datos</i>	90
Figura 32 <i>División del conjunto de datos estandarizado</i>	91
Figura 33 <i>Predicción en el conjunto de prueba</i>	91
Figura 34 <i>Modelo de Machine Learning Red Neuronal</i>	94
Figura 35 <i>Calculo el promedio de las diferencias para cada modelo</i>	95
Figura 36 <i>Exportación de dataframe final</i>	96
Figura 37 <i>Representación gráfica de las temperaturas obtenidas por modelo</i>	100
Figura 38 <i>Estructura del sistema</i>	101

Lista de Anexos

Anexo 1. 114

Anexo 2. 116

Anexo 3. 120

Anexo 4. 121

Anexo 5. 122

Anexo 6. 122

Anexo 7. 122

Anexo 8 122

Resumen

Las redes IoT¹, han permitido mejorar diferentes aspectos para la humanidad, con el uso de ecosistemas tecnológicos que, de manera eficiente, recopilen e intercambien información o datos que permitan aumentar mejorar la calidad de vida, aumentar la productividad en las grandes empresas, tener un control más preciso de los procesos, entre muchas funciones. Sin embargo, la implementación de este tipo de redes puede llegar a ser muy costoso, por lo que es preferible hacer un diseño previamente que permita validar y probar el correcto funcionamiento antes de su construcción e implementación.

Ya que este tipo de redes están compuestos por dispositivos que se encargan de recopilar información del exterior o el ambiente que los rodea, llamados sensores, la compra de estos dispositivos conlleva a un gasto monetario elevado, sin embargo, es posible generar los datos ambientales con los que la red podría trabajar, así que al simular los datos que harán funcionar la red, también es posible simular toda una red IoT basada en datos ambientales. Para que los datos sean consistentes entre si es necesario desarrollar un modelo de simulación que genere los datos apropiados teniendo en cuenta los otros aspectos del ambiente que los rodean.

Para lograr una simulación precisa de los datos ambientales, es necesario crear un modelo que tenga en cuenta las diferentes variables del ambiente. En este sentido, el Aprendizaje Automático (Machine Learning), una disciplina de la inteligencia artificial es clave para entrenar el modelo usando datos recopilados de diversas estaciones hidrometeorológicas de la ciudad de Bogotá. Al obtener todos los datos acumulados por dichas estaciones, se podrán relacionar entre sí, considerando otras variables obtenidas de cada estación, como la fecha y hora en la que se extrajeron los datos. De esta manera, se creará un conjunto de datos organizado con el que el modelo será entrenado. Finalmente, el

¹ En español, Internet de las cosas y hace referencia a la interconexión de cosas de uso cotidiano por medio de internet.

modelo será capaz de predecir los datos ambientales en base a la información del entorno que rodea la simulación.

Palabras clave: Redes IoT, Ecosistemas tecnológicos, Sensores, Datos ambientales, Modelo de simulación, Machine Learning, Inteligencia artificial, Predicción de datos.

Abstract

IoT networks have allowed to improve different aspects for humanity, with the use of technological ecosystems that efficiently collect and exchange information or data to improve the quality of life, increase productivity in large companies, have a more precise control of processes, among many functions. However, the implementation of this type of networks can be very costly, so it is preferable to make a previous design that allows to validate and test the correct operation before its construction and implementation.

Since this type of networks are composed of devices that are responsible for collecting information from the outside or the surrounding environment, called sensors, the purchase of these devices leads to a high monetary expenditure, however, it is possible to generate environmental data with which the network could work, so by simulating the data that will make the network work, it is also possible to simulate an entire IoT network based on environmental data. For the data to be consistent with each other it is necessary to develop a simulation model that generates the appropriate data taking into account the other aspects of the surrounding environment.

To achieve an accurate simulation of the environmental data, it is necessary to create a model that takes into account the different variables of the environment. In this sense, Machine Learning, a discipline of artificial intelligence, is key to train the model using data collected from various hydrometeorological stations in the city of Bogota. By obtaining all the data accumulated by these stations, they can be related to each other, considering other variables obtained from each station, such as the date and time at which the data were extracted. In this way, an organized data set will be created with which the model will be trained. Finally, the model will be able to predict the environmental data based on information from the environment surrounding the simulation.

Keywords: IoT networks, technological ecosystems, sensors, environmental data, simulation model, Machine Learning, artificial intelligence, data consistency, data prediction.

Introducción

Con la aparición del internet, el crecimiento tecnológico de las comunicaciones ha avanzado considerablemente, permitiendo así que las personas puedan mantenerse comunicadas fácilmente, además, estos grandes avances han permitido llegar a áreas que, hasta hace algunos años, era innecesaria una comunicación, pero al ver el valor que podían tener los datos, surgió la necesidad de que los objetos como electrodomésticos y dispositivos de uso industrial puedan recopilar e intercambiar entre si información permitiendo expandir aún más los usos que se pueden hacer con el internet y las redes de comunicaciones que lo componen, a esto se le conoce como Internet of Things (IoT) o también como internet de las cosas.

IoT permite que los dispositivos estén integrados en una red, intercambiando datos recopilados entre sí para aportar más beneficios a las personas e incluso las industrias actuales, pero al requerir una correcta integración de dispositivos que cumplan objetivos que aporten a la red los datos necesarios para su correcto funcionamiento y lograr aprovechar al máximo los beneficios que este tipo de redes son capaces de aportar, es importante que estén diseñadas de la mejor manera posible, y así mismo probar que su funcionamiento sea correcto. Sin embargo, los costos de este tipo de redes y los dispositivos que la componen pueden llegar a ser exageradamente caros.

Por esta razón, el proyecto “Simulador de datos climáticos para el desarrollo de proyectos IoT”; se obtendrá un modelo para la generación de datos climáticos, inicialmente temperatura, con ello podrán ser usados para posteriores proyectos donde sea necesario una validación de funcionamiento de dispositivos IoT. Para esto se desarrollarán algoritmos, modelos y pruebas para generar un óptimo funcionamiento del modelo, con ello, llegar al objetivo general y suplir la necesidad de los desarrolladores, investigadores y técnicos de redes IoT que buscan probar sus dispositivos IoT previo a la instalación de las redes en la ciudad de Bogotá para lugares cercanos al Jardín Botánico, Universidad Nacional y a la sede central del IDEAM.

Planteamiento del Problema

En la actualidad, un gran número de dispositivos electrónicos de uso diario, como celulares, computadores, televisores, neveras, aires acondicionados, sensores, entre otros, están conectados a internet con diversos propósitos. A este concepto se le conoce como IoT “Internet of Things”, el cual, “se refiere a la interconexión en red de objetos cotidianos, que a menudo están equipados con inteligencia ubicua” (Xia, Yang, Wang & Vinel, 2012).

Gracias a esto, la conexiones IoT se han enlazado a la vida cotidiana por el avance exponencial y transversal de las tecnologías en la sociedad, que han evolucionado a una velocidad en la que muchas personas, empresas, negocios, entre otros, están adquiriendo estas nuevas tecnologías con el fin de mejorar el rendimiento en diferentes aspectos.

Los beneficios son amplios, adelante en el documento se podrán encontrar, y esto es debido gracias a que los dispositivos IoT están encargados de la obtención de datos y su envío a la nube (cloud), permitiendo la conexión e intercambio de información entre estos objetos.

Según el centro de investigación SAP², estos objetos interconectados están perfectamente integrados a la red de información, lo que hace que se pueda interactuar con los mismos a través de internet, pudiendo consultar o editar su estado a tiempo real (Sandy & Abasolo, 2013).

Actualmente, hay demasiadas personas que se desempeñan en el diseño, instalación y mantenimiento de estas conexiones como desarrolladores, investigadores, técnicos, entre otros en las redes IoT, estos a su vez buscan alguna ayuda para poder desarrollar sus pruebas sin la necesidad de gastar muchos recursos; por esta razón, una de las dificultades recurrentes

² SAP en español (Desarrollo de Programas de Sistemas de Análisis), es uno de los principales productores mundiales de software para la gestión de procesos empresariales.

para la realización de estas conexiones y su posterior análisis es su alto costo económico la cual limitan la optimización, eficiencia y el tiempo en la construcción de estas conexiones que es una de las razones por la que estos proyectos de redes IoT pueden costar más dependiendo del propósito y presupuestos.

Por lo tanto, cuando se desea construir e implementar una red IoT, es necesario previamente hacer un diseño óptimo que cumpla con los objetivos para los que será creada. Es importante planificar correctamente los componentes necesarios que conformaran la red, los costos y el tiempo adecuados para la realización del proyecto IoT. Para ello, en la fase de planificación es importante que las pruebas sean adecuadas y apegadas a la realidad, permitiendo así tomar las mejores decisiones para implementar la red IoT adecuada.

Antecedentes

En la actualidad existe una gran variedad de plataformas y proyectos que están dirigidos a la contextualización, implementación y desarrollo de las redes IoT gracias al gran crecimiento en la actualidad junto al desarrollo de la tecnología. En el mercado se puede encontrar con software. De igual manera, muchos otros proyectos se encuentran dirigidos al desarrollo o implementación de redes IoT, como lo pueden ser:

- *Control y simulación de una planta piloto de laboratorio docente con integración de plataformas IoT para subida de datos a la nube*

Este trabajo de grado es una continuación del anterior TFG, que lleva por nombre Monitorización y Seguimiento de Simuladores de Procesos Industriales con Fines Educativos creado por John Paúl Mayorga Jines. En TFG, el control y la monitorización se realizan en SIMATIC Manager y WinCC Flexible 2008. TFG se divide en dos partes: simular el modelo en la simulación SIMIT y subir los datos a la nube por medio de la plataforma IBM Cloud, tanto en la simulación como en el

modelo real. Los datos del modelo real se cargan a través de la puerta de enlace IoT2040.

- ***Simulación realista de comunicaciones IoT en entornos urbanos***

Pensando en la implementación de ciudades inteligentes, este proyecto utiliza redes IoT para que los dispositivos que integran la red que abarca la ciudad estén interconectados, ya que al tener un bajo consumo energético y un rango de conexión de larga distancia permite ahorrar costes en instalación y mantenimiento.

Este es un trabajo de grado donde se presenta una solución que incluye un simulador de red para medir el rendimiento en las comunicaciones LPWAN mediante un entorno regulado, un motor 3D para la construcción de este entorno y un motor 3D. Incluyendo el trazado de rayos que ayuda a mejorar los patrones de propagación y ofrece los resultados de rendimiento esperados cuando se utiliza la tecnología de red de área amplia (LoRaWAN).

- ***Gestión y análisis de datos de recopilación de dispositivos IoT en cadenas de suministros aplicando Blockchain y Machine Learning***

Este trabajo de grado tiene como objetivo recopilar datos en tiempo real sobre el flujo de mercancías a lo largo de una cadena de suministro. El proyecto hace uso de sensores y dispositivos de seguimiento IoT con el fin de aumentar la eficiencia y reducir los costos operativos. Los gerentes de la cadena de suministro pueden utilizar la simulación para probar diferentes planes de distribución, opciones de envío y niveles de inventario, buscando mejorar el rendimiento global (Guatibonza Solano & Salazar Marin).

- ***Estudio de la implantación de IoT en las redes logísticas de la cadena de suministros***

Este trabajo de grado tiene como objetivo mejorar la visibilidad de las redes logísticas mediante la aplicación de la tecnología del Internet de las Cosas (IoT). En la actualidad, la falta de visibilidad de las mercancías en tránsito genera incertidumbre y dificulta la gestión eficiente de la logística.

Mediante la implementación del IoT, se pretende obtener datos en tiempo real sobre el transporte, lo que permitirá mejorar la coordinación, la gestión de inventarios y la seguridad en la cadena de suministro (Alandí Pajares, 2015).

- ***Análisis de la simulación de dispositivos, circuitos y sistemas electrónicos para Internet de las cosas (IoT)***

Los dispositivos que forman parte de una red IoT se conocen como nodos, los cuales funcionan con energía eléctrica. Es importante tener en cuenta el consumo energético de una red IoT. Por lo tanto, este proyecto de grado tiene como objetivo simular el consumo de energía de los nodos, tomando como referencia la plataforma hardware Cookies desarrollada en el Centro de Electrónica Industrial de la Universidad Politécnica de Madrid.

Este proyecto se ha llevado a cabo mediante modelos de consumo parametrizables, lo que permite al usuario ajustar dichos modelos de acuerdo a las especificaciones de una red IoT de manera precisa.

Con lo anterior, se generó la siguiente pregunta:

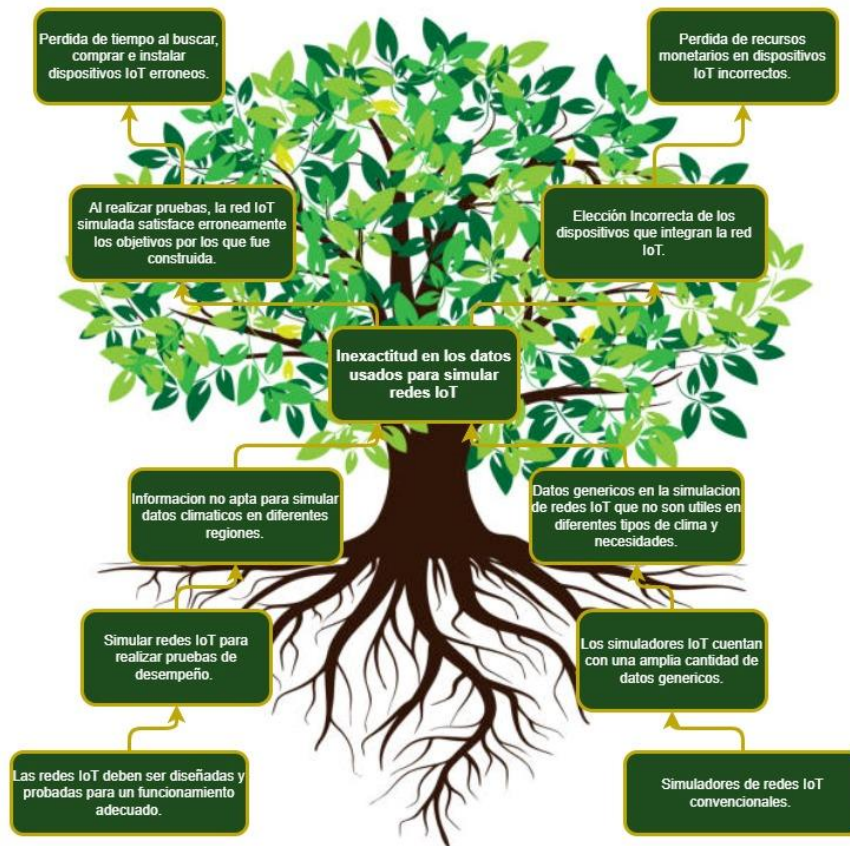
¿Cómo simular datos ambientales mediante el uso de Analítica de Datos para ayudar en la generación de proyectos para modelos IoT?

Justificación

Las redes IoT están conformadas por una variedad de dispositivos, tales como sensores, actuadores, servicios en la nube, entre otros; que trabajan juntos para recopilar y compartir información, y así cumplir con los objetivos establecidos por los usuarios. Por lo tanto, es de vital importancia elegir los dispositivos más adecuados para la red IoT y diseñarla de antemano para poder llevar a cabo pruebas y verificaciones que garanticen su correcto funcionamiento y cumplan con los requisitos de diseño. También es importante probar cada uno de los dispositivos que forman parte de la red para asegurar que estén cumpliendo con su función. Si las pruebas no son satisfactorias, puede ser necesario rediseñar la red desde cero, lo que puede generar pérdidas de recursos, como tiempo y dinero.

Por esto, las redes IoT pueden ser simuladas en emuladores que permitan al usuario diseñar y probar los modelos de red que cumplan con las necesidades de este, pero los datos con los que estas simulaciones trabajan pueden ser genéricos o no aptos para los diversos climas que se pretenden replicar en cada simulación, obteniendo resultados en las pruebas muy inexactos a los climas reales. Es por esto por lo que se utiliza una estructura de árbol de problemas para justificar y explicar el proyecto de manera más clara (ver Figura 1).

Figura 1 *Árbol de Problema*



Fuente: Elaboración propia.

Objetivos

Objetivo General

Realizar un modelo para simular datos IoT mediante un lenguaje de programación para la posterior validación de estos y así facilitar pruebas de proyectos IoT para una región de Colombia.

Objetivos Específicos

- Diseñar los algoritmos de simulación para las diferentes variables climáticas.
- Desarrollar los modelos de simulación de datos de un entorno climático.
- Generar datos e información mediante los algoritmos desarrollados.
- Realizar pruebas funcionales para comprobar la validez de los datos obtenidos.

Marco Teórico

El avance en la tecnología ha permitido la creación de dispositivos inteligentes que se conectan entre sí y con el internet. A este conjunto de dispositivos se le conoce como IoT (Internet de las cosas), una red que permite la interconexión de objetos cotidianos, como vehículos, electrodomésticos, sensores y otros dispositivos electrónicos, que pueden recopilar y transmitir información de manera autónoma.

En conjunto con la analítica de datos, la cual es la disciplina que se encarga de procesar y analizar grandes conjuntos de datos para extraer información relevante y valiosa; esta es usada para el manejo de la gran cantidad de datos obtenida. La simulación que es una técnica que se utiliza para reproducir y analizar situaciones complejas en un entorno controlado y seguro. Y, por último, Machine Learning que es usado para detección de patrones y generación de datos climáticos.

De estos temas estará compuesto el siguiente marco teórico y se busca un entendimiento claro y conciso para el entendimiento de la finalidad final del proyecto.

Internet of Things (IoT)

Internet of Things o también conocido como el Internet de las cosas es un nuevo paradigma del mundo moderno el cual es la conexión de varios nodos IoT que contienen integrados sensores los cuales son los dispositivos, por ejemplo, de recopilación de información, electrodomésticos, celulares, computadores, entre otros que tienen la capacidad de conectarse al internet, que recopilan información de su entorno, por ejemplo, sensores de temperatura, gas, humo, humedad, velocidad del viento, etc. Por otro lado, están los actuadores que son los que reciben como dato de entrada la información proporcionada por los sensores, estos pueden ser enseres domésticos, motores, ventiladores y demás. Esta información transmitida desde el sensor al actuador se hace por medio de las puertas de enlace (gateways) y de esta manera llegar a las plataformas (software) para su respectivo

procesamiento y así suplir el objetivo final de la red. En la actualidad, este concepto es muy ubicuo en el día a día, aunque no sea muy percibido, ya que la mayoría de las cosas de uso diario como lo son los electrodomésticos, celulares, entre otras cosas; lo integran (Xia, Yang, Wang, & Vinel, 2012) (CambioDigital, 2018).

Actualmente, IoT es muy versátil y se puede aplicar a varios usos, por lo que la mayoría de los dispositivos lo integran. Como resultado, la humanidad anticipa un cambio tecnológico donde IoT se volverá cada vez más importante e indispensable para la vida diaria, similar a la situación actual.

Áreas de Aplicación

Las redes IoT tienen una amplia gama de áreas de aplicación en diferentes industrias y dominios, algunas de las cuales incluyen:

Hogares inteligentes

Son usadas para automatizar y controlar electrodomésticos y dispositivos, como termostatos, sistemas de iluminación, cámaras de seguridad, sistemas de entretenimiento, entre otros.

Automatización industrial

Son usadas para monitorear, controlar máquinas y equipos en fábricas, almacenes y demás entornos industriales.

Atención médica

Son usadas para monitorear pacientes de forma remota, recopilar datos de salud de dispositivos portátiles y rastrear equipos médicos en hospitales.

Agricultura

Son usadas para monitorear los niveles de humedad del suelo, la temperatura y otros factores ambientales para optimizar el rendimiento de los cultivos y reducir el uso de agua.

Transporte

Son usadas para rastrear la ubicación y el estado de los vehículos, monitorear el flujo de tráfico y optimizar las operaciones logísticas.

Ciudades inteligentes

Son usadas para monitorear y controlar el alumbrado público, el flujo de tráfico, la gestión de desechos y otros servicios en áreas urbanas.

Estos son solo algunos ejemplos de las muchas áreas de aplicación de las redes IoT. A medida que la tecnología continúa evolucionando y mejorando, podemos esperar ver casos de uso aún más innovadores en el futuro.

Beneficios

La aplicación de IoT es variada y funcional, ya que es versátil para cumplir su objetivo en función de los datos que genera el entorno donde se despliega. Se analiza la red en función de la funcionalidad prevista para cubrir la necesidad deseada (Universidad de Alcalá, 2019).

Por ejemplo, se ha demostrado que la aplicación industrial de IoT minimiza los esfuerzos y los riesgos al tiempo que aumenta la productividad, lo que hace que

los procesos sean más efectivos y eficientes a través de la gestión y el control remotos de la maquinaria. En el estudio Industry 4.0 de Deloitte³, se muestra que la tecnología IoT brinda información sobre el funcionamiento general del trabajo para analizarlo y realizar mejoras, incluido el monitoreo del estado de las máquinas, el control de calidad, el inventario y otros análisis.

El sector agrícola también se ha beneficiado de la tecnología IoT en forma de Smart Agriculture, que combina herramientas tecnológicas para digitalizar y mejorar la productividad agrícola. Los agricultores pueden obtener información actualizada sobre el estado de sus cultivos las 24 horas del día, los 7 días de la semana, y se pueden operar diferentes equipos agrícolas de forma remota para mejorar la cantidad y calidad de los cultivos. Otros sectores como gobierno, transporte, educación, entre otros, también pueden beneficiarse de las redes IoT.

Modelos de Referencia

En el Modelo general de la arquitectura de una red IoT (Ver Figura 2) se puede evidenciar un modelo general estas redes, asimismo, se puede ver dividido en 3 grandes campos. Hay que resaltar que los nombres de cada parte del modelo pueden variar, pero su definición y función serán igual.

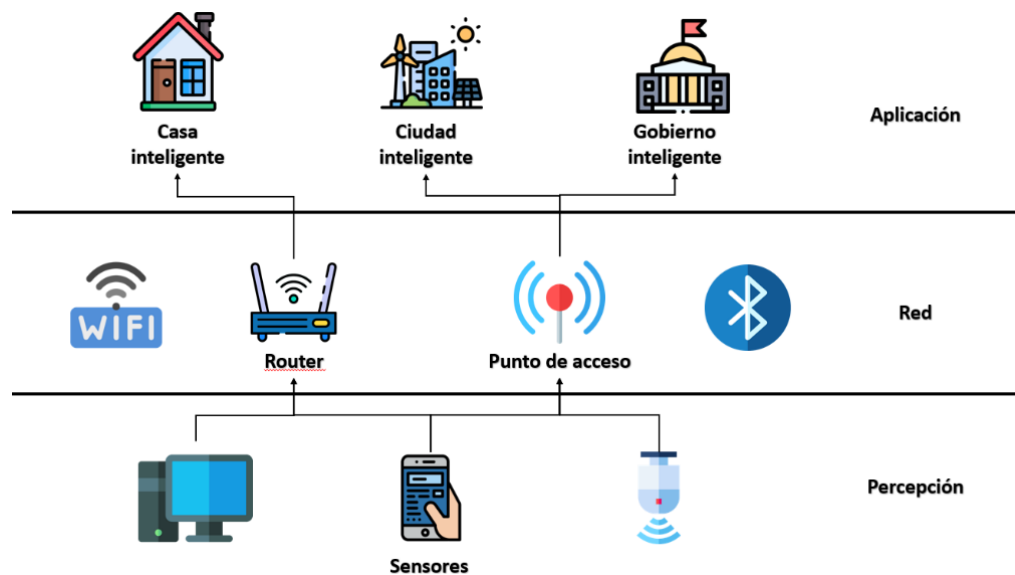
En primer lugar, se puede ver la percepción, el cual contiene todos los sensores que recolectan la información para ser enviada.

Gracias a la percepción esta información se envía a la red o nube, la cual, es la segunda capa, por medio de los gateways que son los encargados de transmitir, con toda la seguridad necesaria, los datos obtenidos entre las dos capas.

³ El informe Industry 4.0 de Deloitte, es un estudio que tiene como objetivo analizar y describir la cuarta revolución industrial que se está produciendo a nivel mundial. Explica como las nuevas tecnologías están y continuaran transformando los negocios en la actualidad.

Finalmente en la tercera capa se encuentran los actuadores o aplicación en donde se recibe toda la información recogida en la capa de percepción a través de la red y gracias a esto actuar dependiendo el requerimiento o el objetivo al que se quiera llegar; como se mencionó anteriormente toda la información es enviada a la red o nube, la cual almacena toda esta información para ser procesada y con su posterior análisis de los datos, estar lista para la aplicación en las “cosas” (Crespo Moreno, 2018).

Figura 2 *Modelo general de la arquitectura de una red IoT*



Fuente: Elaboración propia.

Protocolos

Los protocolos son una guía y/o pasos para saber cómo realizar una acción, de esta manera, los protocolos en IoT son los métodos en la que dos o más componentes se comunican por medio de la red, regulando las condiciones en que se transporta, el direccionamiento, enrutamiento y controles de los datos para que de esta manera se garantice la consistencia y transferencia de la información Machine2Machine (M2M).

Tipos de Protocolos IoT

Con el crecimiento, desarrollo e implementación de dispositivos IoT, se han establecido diferentes protocolos IoT para la gestión de comunicaciones. Para determinar el tipo de protocolo que se debe implementar en una red IoT, se deben considerar los dispositivos interconectados, su función u objetivo y la distancia que recorrerán para la transición de datos. Sin embargo, se utilizan comúnmente dos tipos principales de protocolos:

Protocolos de acceso de red

Se utilizan en la capa inferior para permitir la conexión de dispositivos, que se puede realizar a través de Wi-Fi, Ethernet, 3G, 4G, 5G, etc.

Protocolos de transmisión

Utilizados en la transmisión de datos, que codifica la información enviada a través de las redes.

Hay que recalcar que la elección del protocolo depende de los requisitos específicos de la red a diseñar, los tipos de datos que se transmiten, la cantidad y tipos de dispositivos en la red y la distancia en la que los dispositivos deben transmitir los datos. Algunos de los protocolos comúnmente utilizados en las redes IoT incluyen.

MQTT (Transporte de telemetría de Message Queue Server)

Este protocolo se usa ampliamente en aplicaciones IoT por su baja sobrecarga y distribución eficiente de mensajes. Está diseñado para aplicaciones con redes de bajo ancho de banda y alta latencia.

CoAP (Protocolo de aplicación restringida)

Este protocolo está diseñado para dispositivos y redes con recursos limitados, y se utiliza para transferir datos entre dispositivos.

HTTP (Protocolo de transferencia de hipertexto)

Aunque originalmente se diseñó para la web, HTTP se usa en aplicaciones de IoT por su amplio soporte, compatibilidad y facilidad de uso.

DDS (servicio de distribución de datos)

DDS es un protocolo de mensajería de publicación/suscripción en tiempo real utilizado para sistemas centrados en datos como IoT.

AMQP (Protocolo de cola de mensajes avanzado)

Este protocolo está diseñado para la entrega confiable de mensajes y admite múltiples patrones de mensajería.

ZigBee

Este protocolo se utiliza en redes inalámbricas de baja potencia y está diseñado para aplicaciones con velocidades de datos bajas y un número limitado de dispositivos.

LoRaWAN (red de área amplia de largo alcance)

Este protocolo se utiliza en aplicaciones IoT que requieren comunicación de largo alcance, como las aplicaciones de ciudades inteligentes.

Dispositivos

Como se definió anteriormente, IoT no puede funcionar sin los nodos IoT los cuales son los diferentes dispositivos, en su mayoría físicos, los cuales garantizan el tratamiento de datos rápido, seguros, y eficientes.

Sensores

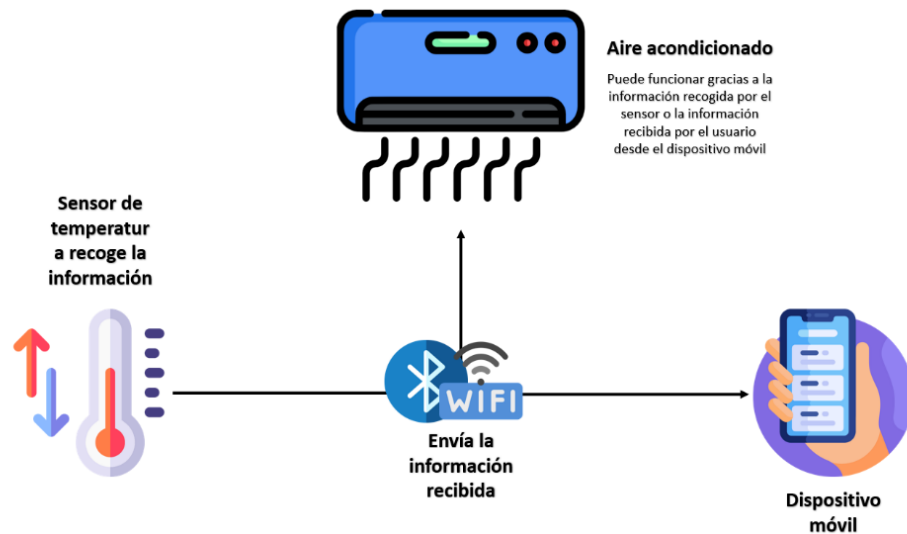
Los sensores son los dispositivos que recogen la información del ambiente en donde se encuentra funcionando o responden a una salida de un sistema, detectando así los cambios que ocurren, de esta manera, por medio de los Gateway es enviada toda la información para luego ser procesada y analizada en la red.

Actuadores

Los actuadores son los dispositivos que reciben o responden a la información producida por los sensores y analizada en la red, todo esto por medio de los gateways, para posteriormente por medio de la aplicación de control, funcionar dependiendo de la situación que en el que esté funcionando.

Por ejemplo, como se podrá observar en la **¡Error! No se encuentra el origen de la referencia.**, un actuador muy común son los aires acondicionados que tiene la capacidad por medio de redes IoT recibir información de la temperatura por medio de un sensor que recoge estos datos, de esta manera, el actuador va a funcionar dependiendo la temperatura y los gustos de la persona que haga uso del aire acondicionado.

Figura 3 *Funcionamiento de actuadores IoT*



Fuente: Elaboración propia.

Gateway

Los gateways son dispositivos intermediarios entre los sensores, actuadores y la red. Generalmente son físicos o software los cuales son los que reciben la información gracias a los sensores para posteriormente enviarla a la red (Crespo Moreno, 2018).

Teléfonos inteligentes y tablets

Estos dispositivos se pueden usar para controlar y monitorear dispositivos IoT de forma remota.

Wearables

Estos dispositivos, como relojes inteligentes y rastreadores de actividad física, recopilan y transmiten datos sobre la actividad física y la salud de un usuario.

Electrodomésticos inteligentes

Estos dispositivos, como termostatos y refrigeradores inteligentes, se pueden controlar y monitorear de forma remota a través de un teléfono inteligente o una tableta.

Sistemas integrados

Estos son sistemas informáticos especializados que están integrados en otros dispositivos, como automóviles o sistemas de seguridad para el hogar, para permitir la funcionalidad de IoT.

Plataformas IoT

Las plataformas IoT son el software en el cual los dispositivos IoT se pueden comunicar o conectar y crear información de valor por medio del entorno digital de la misma plataforma, gracias a esto, el desarrollo y el funcionamiento de la red se produce de la mejor manera. Teniendo en cuenta lo anterior, hoy en día en cuanto a plataformas para el monitoreo de estas conexiones existe una gran variedad. De cierto modo, IoT se está convirtiendo en un pilar para la sociedad y por esta razón han

desarrollado plataformas nuevas y cada día mejores para el buen uso, desarrollo, gestión y mantenimiento de estas (Quiñones Cuenca, González Jaramillo, Torres, & Miguel, 2017).

Analítica de Datos

La analítica de datos es el uso de la información que se puede obtener de manera digital, con el propósito de extraer la mejor información para poder tomar las mejores decisiones (Gibbs, 2012); varios autores vinculan la analítica con el manejo de variables con el uso de algoritmos.

La analítica de datos puede clasificarse en tres grandes categorías: analítica descriptiva, analítica predictiva y analítica prescriptiva (Pusala, Amini, Katukuri, 2016).

La analítica de datos descriptiva

Como estado inicial en el que los diferentes tomadores de decisiones profundizan en los respectivos datos históricos con el fin de detectar patrones de comportamiento en las variables para realizar análisis de correlación (ibertech, 2006).

La analítica de datos predictiva

Donde las empresas con datos registrados anteriormente generan modelos de pronósticos sobre las tendencias y así poder realizar cambios para mejorar (ibertech, 2006).

La analítica de datos prescriptiva

Donde las compañías utilizan modelos de simulación de escenarios, para la optimización de diferentes fuentes de interés (ibertech, 2006).

Áreas de Aplicación de Analítica

Las áreas donde se puede aplicar la analítica son extensas, ya que muchas actividades o procedimientos que realizan es necesario hacer una investigación anteriormente, para poder obtener unos resultados apropiados con el fin de analizar y así poder utilizarlos en un propósito de sacar conclusiones sobre la información tratada; todo esto para tomar las mejores decisiones, verificar teorías y modelos existentes, la clasificación de conjuntos de datos para obtener una relación y utilizarlos en el mejoramiento de campañas (Joyanes Aguilar, 29 de mayo del 2019); muchas de las áreas pueden ser de economía, probabilidad, administración, web, inteligencia artificial etc (Gomez-Aguilar, Garcia-Peñalvo, & Theron, 2014).

Cadena de Valor de Datos

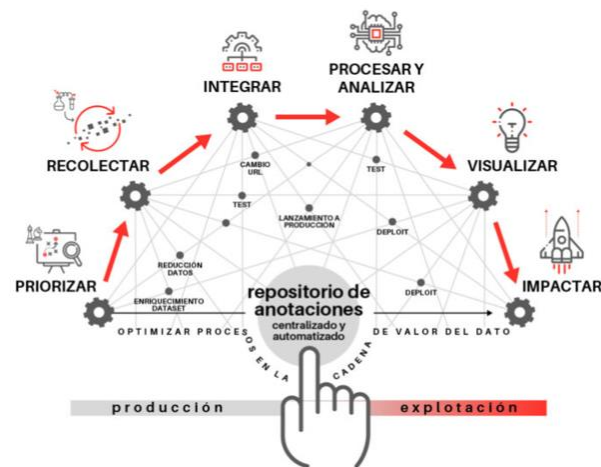
Como su propio nombre indica, son una gran cantidad de datos que representan cierta información (Quintero, 2006); en el cual ciertas empresas ya están destinadas a prestar estos servicios; la cadena tiene varias etapas para su realización las cuales son:

1. Priorizar: Determina por su posición en la lista, el primer conjunto de datos tiene la prioridad más alta.
2. Recolectar: Proceso de recopilación y medición de información sobre variables establecidas de una manera sistemática.
3. Integrar: Combinación de procesos técnicos y de negocio que se utilizan para combinar información de diferentes fuentes.
4. Procesar y analizar: Manipulación de elementos de datos para producir información significativa.

5. Visualizar: Presentación de datos en formato ilustrado o gráfico, el cual permite a los tomadores de decisiones poder optar la mejor opción.
6. Impactar: es el uso que se le pueden dar a los datos frente al impacto que puede tener frente a la empresa u organización.

La cadena de valor proporciona cierto modelo de aplicación el cual permite representar todas las actividades de cualquiera empresa y también proporciona un procedimiento para el desarrollo de ventajas como la debilidad o fortaleza de los datos a tratar, para poder extraer la mayor cantidad posible de información, la identificación de datos relevantes y el planteamiento de estrategias o contingencias para el manejo de los respectivos datos pulidos (ver Figura 4) asimismo en la Tabla 1. Se muestran las etapas para realizar la cadena de valor (Hergert & Morris, 1989)

Figura 4 Cadena de valor



Fuente: (Muñoz, 2019).

Tabla 1 Etapas para realizar cadena de valor

Recopilación	Publicación	Consumo	Impacto
--------------	-------------	---------	---------

Identificar	Analizar	Conectar	Usar
Manipular	Liberación	Incentivar	Cambio
Proceso	Diseminar	Influencia	Reutilizar

Fuente: Elaboración propia.

Datos

Representación simbólica de alguna información o procedimiento, en la cual, puede ser almacenado y analizado para poder realizar ciertas operaciones y así poder generar información adecuada para la toma de decisiones frente al mejoramiento de procesos en los diferentes campos para el obtener mejor desempeño de empresas o establecimientos.

Tipos de datos

Los tipos de datos son una forma de clasificar diferentes valores que se pueden almacenar y manipular en un programa de computadora. La mayoría de los lenguajes de programación proporcionan varios tipos de datos predefinidos, como números enteros, decimales, caracteres y valores booleanos. Cada tipo de datos tiene un conjunto específico de valores que puede aceptar y representar de una manera específica en la memoria de la computadora. Elegir el tipo de datos correcto es muy importante para garantizar que las operaciones realizadas en el programa sean precisas y eficientes.

Estructurados

Son la mayoría de los datos que se pueden encontrar almacenados en una base de datos; se muestran en fila y columnas, tienen definido su longitud en el formato que se encuentran los respectivos datos (Hernandez & Rodriguez, 2008).

No Estructurados

Son datos binarios que no están organizados que no tienen algún valor al utilizarlos hasta que son organizados y almacenados, el cual su manejo es mucho más complejo que en los demás, estos datos no se pueden usar en una base tradicional como son las tablas ya que es imposible poder organizarlos o ajustarlos en filas y columnas estandarizadas, pero se encuentran muchos tipos de datos no estructurados de uso común como archivos PDF, imágenes o archivos de texto (Hassan, Domingo-Ferrer, & Soria-comas, 2018).

Semi – estructurados

Son datos que no son organizados en un repositorio, pero tiene información importante como metadatos (datos que están cerca de los datos) la cual hace que se pueda procesar más fácilmente los datos (Raposo, 2007).

Tipos de analítica

Con el aumento de la cantidad de datos que generan actualmente las organizaciones, los negocios pueden extraer grandes cantidades de información las cuales se pueden utilizar para mejorar a las empresas como predecir qué mes se puede obtener más ganancias.

“Un conjunto de métodos de análisis matemático y estadístico que sirve para identificar patrones de comportamiento, pronósticos, escenarios “que pasaría si”, entre otros” (Davenport y Harrys, 2017).

Descriptiva

Se pueden diferenciar de los otros tipos por medio de la pregunta inicial sea ¿qué sucedió? o ¿Qué está pasando? un resumen del desempeño del total de las

actividades empresariales, el cual permite ver la composición principal dentro de un negocio como por ejemplo observar las ganancias o pérdidas en el mes (cuatro tipos de analítica de retail que todo comercio necesita en 2018, 2018). Consiste en almacenar y realizar información con datos históricos para obtener un análisis del estado actual y anterior del negocio, permite detectar, visualizar, observar e identificar el camino a tomar para la mejor desciño posible.

Diagnostica

Se pueden diferenciar de los otros tipos por medio de la pregunta inicial sea ¿Por qué está pasando? Tiene en cuenta los antecedentes de lo que se quiere analizar para dar un informe más acertado con sus respectivas herramientas y así poder eliminar el problema y así tener los elementos necesarios para que el respectivo análisis en los datos se pueda obtener la causa de los problemas (cuatro tipos de analítica de retail que todo comercio necesita en 2018, 2018).

Predictiva

Se pueden diferenciar de los otros tipos por medio de la pregunta inicial sea ¿qué podría pasar? o ¿Qué es lo más probable que pueda pasar? tienen como objetivo identificar la probabilidad que ocurra algo en el futuro y que no perjudique el análisis realizado; estos modelos se suelen utilizar los datos o variables los cuales se puedan realizar predicción y así tomar mejores decisiones por esto es uno de los más importantes (cuatro tipos de analítica de retail que todo comercio necesita en 2018, 2018).

Prescriptiva

Se pueden diferenciar de los otros tipos por medio de la pregunta inicial sea ¿qué deberíamos hacer? o ¿Qué necesito hacer?, entendimiento de lo que ha sucedido,

por qué ha sucedido y un procedimiento en el cual podría suceder con el paso del tiempo, ayudar al usuario a determinar el mejor curso de acción a tomar (cuatro tipos de analítica de retail que todo comercio necesita en 2018, 2018).

Técnicas de Analítica

Muchas veces la base o la raíz que tiene la analítica es entender lo que ha pasado o lo que está pasando en el momento; todo esto para poder adquirir el conocimiento para mejorar las decisiones hacia el futuro, muchas veces se utilizan técnicas de estadística y matemáticas para poder lograr el objetivo (Garcia, 2006).

Análisis exploratorio de datos (EDA): Este es un método para analizar y resumir datos utilizando técnicas gráficas y estadísticas, ayuda a descubrir patrones, relaciones y tendencias en los datos e identificar valores atípicos o datos faltantes (EOI, 2007).

Análisis de regresión: Esta técnica se utiliza para determinar la relación entre una variable dependiente y una o más variables independientes, ayuda a determinar la fuerza y la dirección de las relaciones entre las variables (EOI, 2007).

Análisis de series temporales: Este método se utiliza para analizar datos que cambian con el tiempo y predecir tendencias futuras, ayuda a identificar tendencias y tendencias a lo largo del tiempo y predecir la demanda futura (EOI, 2007).

Análisis de Clustering: Este método se utiliza para organizar datos similares en grupos o grupos. Ayuda a identificar tendencias y piezas de datos que se pueden utilizar para tomar decisiones comerciales (EOI, 2007).

Análisis de texto: Este método se utiliza para analizar y extraer información de grandes cantidades de datos no estructurados, como correos electrónicos, tweets y comentarios en redes sociales, ayudar a identificar temas, ideas y puntos en común (EOI, 2007).

Simulación

La simulación es una representación exacta o casi exacta del comportamiento de un evento o fenómeno, con el fin de obtener el mismo resultado, características, información entre otros, aspectos, consumiendo menos recursos de los que consumiría ejecutar el modelo real, sin necesidad de realizar dicho evento para obtener un análisis o estudio del resultado con una menor inversión (Castellanos Hernández & Chacon Osorio, 2006).

Tipos de Simulación

La simulación puede ser dividida y clasificada según la naturaleza del sistema que será representado en el modelo de simulación, por ejemplo:

Simulación de situaciones

Permite simular una situación física o real y observar su comportamiento, con el fin de extraer información de hechos existentes (Castellanos Hernández & Chacon Osorio, 2006).

Simulación de Realizar Alguna Situación

Son aquellos que permiten experimentar una situación como si el usuario u sujeto estuviera en ella, un simulador de vuelo es un ejemplo de esto, permite al usuario pilotear un avión sin estar en uno realmente (Castellanos Hernández & Chacon Osorio, 2006).

Fases de Estudio de Simulación

Para hacer un modelo de simulación efectivo, es necesario tener en cuenta distintas fases que permitirán la construcción de un modelo eficiente.

Definición de objetivos

Se deben establecer los objetivos que se pretenden conseguir con la simulación, los efectos que causara y las respuestas a responder con este estudio (García Sánchez & Ortega Mier, 2006).

Definición del sistema

Definir los elementos que harán parte del sistema teniendo en cuenta el sistema a emular (García Sánchez & Ortega Mier, 2006).

Elaboración del modelo conceptual

A partir de los objetivos planteados anteriormente se crea un modelo conceptual, el cual debe ser sencillo (solo enfocarse en lo necesario para simular) y específicamente diseñado para cumplir dichos objetivos. El modelo conceptual debe representar sencillez y a su vez representar el realismo del sistema a emular (García Sánchez & Ortega Mier, 2006).

Este modelo conceptual debe ser evaluado y comprobar que refleje fielmente el sistema que se desea emular teniendo en cuenta los objetivos que debe cumplir (Coss Bu, 2003).

Elaboración del sistema comunicativo

Los diseñadores del modelo conceptual son distintos muchas veces a los programadores del simulador. Para su comunicación entre si debe ser eficaz, por esta razón los diagramas de flujo son una opción útil para representar los eventos en el simulador como lo son los datos, el proceso, una decisión un avance en la simulación etc (García Sánchez & Ortega Mier, 2006).

Construcción y verificación de modelo informático

Una vez verificado el modelo conceptual se escoge un lenguaje apto para para la programación del simulador, este lenguaje debe permitir la correcta emulación como fue planeada, se debe escoger el más conveniente (García Sánchez & Ortega Mier, 2006).

Validación final

Una vez construido el modelo de simulación creado anteriormente, es necesario hacer pruebas para verificar su correcto funcionamiento, en las cuales los resultados deberán ser similares a los esperados y si es posible se comparará con los resultados del sistema real al cual se está simulando (Coss Bu, 2003).

Modelos de simulación

Hay diversos modelos de simulación los cuales serán mencionados a continuación, pero nos centraremos más en el modelo Estocástico:

Estático

La simulación no depende del tiempo, por ejemplo, un sistema que se encuentra en un estado de equilibrio o reposo (García Sánchez & Ortega Mier, 2006).

Dinámico

En contraparte a la simulación estática, el modelo dinámico depende del tiempo, sus procesos pueden variar respecto al tiempo que va transcurriendo, por ejemplo, un modelo que represente el incremento poblacional al cabo de n años (García Sánchez & Ortega Mier, 2006).

Determinístico

Su ejecución será siempre igual, el valor de su resultado será el un resultado ya esperado debido a las condiciones iniciales, por lo tanto, no tiene ninguna variable o proceso al azar, por ejemplo, el uso de un dispositivo programado para realizar una tarea determinada al momento de que el usuario oprima un botón (García Sánchez & Ortega Mier, 2006).

Estocástico

A diferencia del modelo determinístico, los procesos estocásticos contienen variables o procesos al azar que cual puede variar el resultado de la simulación, de esta manera funciona todo en la vida real, por ejemplo, un modelo que permita predecir el tiempo que puede tardar en realizarse una transacción bancaria, en la mayoría de veces es posible calcular un tiempo promedio pero su vez hay factores que pueden afectar la velocidad en la que se realiza la transacción, uno de estos factores es el tráfico en la red afectado por la cantidad de usuarios que está realizando una transferencia al mismo tiempo (García Sánchez & Ortega Mier, 2006).

Discreto

Varía dependiendo de sucesos que ocurran en la simulación del modelo y en un tiempo determinado, por ejemplo, en una sala de urgencias los pacientes son remitidos a diferentes especialistas y tratamientos según la incidencia que hayan

sufrido, por lo que el tiempo y otros factores son alterados con respecto a el tipo de atención que dicho paciente requiera (García Sánchez & Ortega Mier, 2006).

Continuo

Tiene un rango de tiempo el cual es previamente establecido. Sin importar los sucesos que ocurran seguirá ejecutándose y sus variables estarán cambiando en cualquier momento, por ejemplo, al comprar productos en internet, la compra será realizada según la disponibilidad del producto, ya que este puede ser vendido por completo a otro cliente en cualquier momento.

Físicos

Se basan en eventos físicos o fenómenos que ocurren y no son posibles de controlar y estudiar, por ejemplo, el cambio climático en una región de Colombia (García Sánchez & Ortega Mier, 2006).

Procesos estocásticos

Los procesos estocásticos es una colección de variables aleatorias infinitas que se basan en el cambio o evolución de una variable con respecto al tiempo o en función de otra variable como lo puede ser la temperatura, el cambio climático entre otras (Myriam Muñoz de ózak, 1991).

Las variables aleatorias que dependen del tiempo son aquellos fenómenos que evoluciona al azar a lo largo del tiempo, el tiempo tomará diferentes valores en dicho conjunto donde la colección de variables aleatorias se verá afectadas por esto tomando diferentes valores según T (Myriam Muñoz de ózak, 1991).

El conjunto de variables que dependen de otra variable son aquellos que su evolución no dependen del tiempo sino de otro fenómeno (Myriam Muñoz de ózak, 1991), estas variables pueden tomar valores directamente proporcionales a este fenómeno no perteneciente al conjunto dicho anteriormente es decir que los valores de las variables aumentarían si el valor del fenómeno aumenta y disminuirá si este disminuye, por otro lado las variables aleatorias que son indirectamente proporcionales al fenómeno harán todo lo contrario si el valor del fenómeno aumenta, el valor de las variables disminuirá y si este disminuye las variables aumentarían.

Variables aleatorias

Las variables aleatorias son parte fundamental de una simulación, ya que los sistemas requieren diferentes tipos de datos no siempre serán los mismos para ejecutar un evento simulado, por esta razón es de vital importancia crear variables aleatorias ya que necesitamos que la simulación sea lo más apegado posible a la realidad (Eduardo García Dunna, 2013).

El modelo que se quiere construir debe estar compuesto de variables aleatorias que interactúen entre sí, para asemejarlo a la realidad (Eduardo García Dunna, 2013). Una variable aleatoria es una representación de un suceso o un número de una parte del evento que se está intentando emular, es decir que un suceso en el sistema puede variar en ese mismo proceso u otro proceso independiente, por ejemplo, si se quiere emular un sensor de temperatura, una de las variables aleatorias será la temperatura ya que esta puede cambiar con el paso del tiempo o ser diferente en otra prueba del simulador (Eduardo García Dunna, 2013).

Existen dos tipos de variables aleatorias, la primera es la variable aleatoria discreta, la cual se caracteriza por ser de solo números enteros es decir no puede tomar valores como 10,97 u otro tipo de número que no es un entero (Eduardo García Dunna, 2013), por otra parte la variable aleatoria continua es más caracterizada por su uso para las mediciones en este caso si puede contener valores decimales, retomando el ejemplo del medidor de

temperatura podemos decir que esta simulación consta de variable aleatoria continua ya que la temperatura es una medida y puede tener parte decimal (Eduardo García Dunna, 2013).

Modelos probabilísticos

Cuando se habla de un modelo probabilístico se hace referencia a un conjunto de datos obtenidos por diversas repeticiones de un evento aleatorio usados para poder predecir el comportamiento de este evento con los mismos o diferentes datos para las futuras repeticiones de dicho evento (Leónardo Darío Bello Parias, 2000), esta serie de repeticiones permiten asemejar el modelo que se está construyendo con datos aleatorios a un conjunto de datos de una población mayor, con esto se hace referencia a la simulación más acercada posible de un evento real mediante la prueba y repetición del modelo que se está simulando.

Existen varios modelos probabilísticos para variables aleatorias:

- Distribución Uniforme.
- Distribución Gamma.
- Distribución Exponencial.
- Distribución Ji-dos.
- Distribución Normal.
- Distribución t Student.
- Distribución F de Sendecos.
- Distribución normal bivalente.

Los modelos probabilísticos son basados en hipótesis y se compone por ecuaciones las cuales relacionan las diversas variables aleatorias (Carlos Gamero Burón, 2015), estos modelos son la representación más viable de una hipótesis para un evento que este compuesto de variables aleatorias por lo cual debe ser rectificado correctamente y probado una y otra vez.

Números pseudoaleatorios

Una simulación, muchas veces se compone de variables aleatorias es decir números al azar, para conseguir esto los números pseudoaleatorios son parte fundamental en este proceso de simulación, su nombre está compuesto de dos palabras, “Pseudo” lo cual significa falso y “aleatorio”, se le denomina falso debido a que es imposible generar números completamente aleatorios, al no ser posible generar números completamente aleatorios los números pseudoaleatorios son creados a partir de algoritmos determinísticos con parámetros de arranque, esto permitirá generar números que se comportaran similarmente a números totalmente aleatorios es decir números sin correlación entre ellos mismos permitiéndonos simular el comportamiento aleatorio de las variables en el evento que queremos simular (Eduardo García Dunna, 2013).

Generación de números de pseudoaleatorios

Para hacer la generación de los números Pseudoaleatorios se debe tomar un espacio o rango lo suficientemente grande para ello, es decir cuente con demasiados números en secuencia para una vida útil prolongada (Eduardo García Dunna, 2013). Es necesario este conjunto tan grande porque al hacer una simulación pequeña se necesitarán un conjunto de números mínimo, pero si se quiere hacer aun mayor este número incrementara, pero al hacer la simulación no puede basarse en solo un resultado para ello es necesaria la simulación una y otra vez con números distintos es por esto por lo que es necesario dicho conjunto lo suficientemente grande para satisfacer esta necesidad (Eduardo García Dunna, 2013).

Para aprobar el uso de estos números el conjunto de números pseudoaleatorios se debe someter a ciertas pruebas que nos permitan comprobar la independencia entre ellos y que estos sean uniformes, para ellos se mencionaran unas pruebas estadísticas para la aprobación de este conjunto se debe asegurar que los números de un conjunto deben ser uniformemente distribuidos lo cual significa que en los subintervalos haya la misma cantidad de números del conjunto, deben ser continuos, la media del

conjunto debe ser equivalente a $\frac{1}{2}$ y la varianza también debe ser $\frac{1}{2}$ (Eduardo García Dunna, 2013).

Ventajas de la simulación

La simulación permite ahorrar recursos para obtener los posibles resultados del comportamiento de un evento (García Sánchez & Ortega Mier, 2006).

A partir de la simulación es posible trabajar mejor los experimentos debido a su mejor manejo en las condiciones de dicho experimento (García Sánchez & Ortega Mier, 2006).

Es posible a partir de la simulación comparar y escoger el sistema más viable dependiendo de una necesidad (García Sánchez & Ortega Mier, 2006).

La simulación puede permitir una mejor comprensión del evento que está simulando (García Sánchez & Ortega Mier, 2006).

Con una simulación es posible hacer diferentes experimentos y su reacción a estos, los cuales no son posibles con el modelo físico el cual se pretende obtener esta nueva información (García Sánchez & Ortega Mier, 2006).

Desventajas de la simulación

Una vez creada la simulación es posible ahorrar tiempo en la obtención de los datos del modelo simulado, pero para crear la simulación lleva tiempo y estudios los cuales no son mayores los recursos que requerirá usar el modelo real (García Sánchez & Ortega Mier, 2006).

La simulación debe ser exacta al modelo real pero aun así se puede generar datos no correctos o no exactos algunas veces a los reales (García Sánchez & Ortega Mier, 2006).

Machine Learning

La creación de las computadoras tiene como objetivo realizar las tareas que las personas no son capaces de hacer o no pueden ejercer de una manera rápida y eficiente, a pesar de que este objetivo ha sido cumplido por las máquinas, nunca han podido tener en ellas una inteligencia al nivel o superior de la mente humana que permita a los dispositivos pensar por sí mismos y tomar decisiones en los diferentes procesos y tareas que ejecutan. Sin embargo, el hombre ha logrado hacer que las máquinas imiten la inteligencia humana, a esto se le conoce como inteligencia artificial (IA) (Oracle, 2022), una característica fundamental del razonamiento humano es el aprendizaje de hechos o experiencias. Esto permite al humano tomar decisiones lógicas sobre las acciones que realiza, permitiendo así obtener mejores resultados de las mismas.

Para lograr que un dispositivo de cómputo sea capaz de aprender, son necesarias grandes cantidades de datos ya que estos son equivalentes al conocimiento necesario para que a partir de algoritmos que tengan la capacidad tomar decisiones y realizar acciones que permitan cambiar y mejorar según el modelo de datos utilizado (Carlemany, 2022), es decir que los algoritmos integrados en un sistema de aprendizaje automático deben tener la capacidad de “alimentarse” de datos con el fin de ser entrenados y de esta manera lograr aprender de ellos encontrando patrones que permitan predecir un comportamiento normal o común entre dichos datos y así tomar decisiones coherentes o acertadas e incluso predecir eventos futuros de manera eficiente. Para lograr el entrenamiento de los algoritmos de Machine Learning, se pueden utilizar tres métodos de aprendizaje: supervisado, no supervisado y por esfuerzo.

Supervisados

El aprendizaje supervisado se lleva a cabo mediante conjuntos de datos que contienen tanto parámetros característicos que toman el rol de preguntas, como parámetros de etiquetas que toman el rol de respuestas (Sandoval, 2018). De esta

manera, el modelo aprende de todas las características que componen al conjunto de datos para interpretar y predecir las respuestas o características faltantes de manera automática y precisa en el futuro. El aprendizaje supervisado se divide en dos tipos de algoritmos: el algoritmo de clasificación y el algoritmo de regresión.

Este tipo de aprendizaje está dividido en dos tipos de algoritmos que son:

Algoritmo de Clasificación

El algoritmo de clasificación se utiliza para clasificar conjuntos de datos gracias a etiquetas de clase categórica. En otras palabras, este algoritmo determina si el dato a predecir pertenece o no a un grupo específico de datos en función de las características que lo componen (Sandoval, 2018).

Algoritmo de Regresión

El algoritmo de regresión, por otro lado, se utiliza para predecir una característica de un elemento, generalmente un número, en función de las demás características que lo componen (Sandoval, 2018).

No Supervisados

Por otro lado, el aprendizaje no supervisado no se basa en la experiencia proporcionada por datos etiquetados, sino que se lleva a cabo a partir de elementos no etiquetados. En este tipo de aprendizaje, se intenta agrupar los datos no estructurados en función de las características que los componen (Dra. Marianella Álvarez Vega, 2020). El aprendizaje no supervisado se divide en dos métodos: agrupamiento y reducción de la dimensionalidad.

Agrupamiento

El agrupamiento se utiliza para interpretar los datos y agruparlos por similitudes en función de las características en las que están compuestos. Esto permite estructurar y organizar los datos que no tienen una etiqueta que los defina en un grupo específico, como ocurre en los algoritmos de clasificación en el aprendizaje supervisado (González, 2015).

Reducción de la dimensionalidad

La reducción de la dimensionalidad se utiliza para seleccionar las características más relevantes o representativas para la clasificación objetivo de los diferentes conjuntos de datos. Al mismo tiempo, se descartan las características menos importantes, lo que permite una mayor eficiencia en el modelo de aprendizaje (Softtek, 2021).

Por esfuerzo

El aprendizaje por esfuerzo (también conocido como aprendizaje activo o aprendizaje semisupervisado) es una técnica de aprendizaje automático en la que el modelo se entrena a partir de un conjunto inicial de datos etiquetados y luego solicita al usuario que proporcione etiquetas adicionales para una selección estratégica de los datos no etiquetados para mejorar el modelo. En lugar de requerir grandes conjuntos de datos etiquetados, el aprendizaje por esfuerzo permite entrenar modelos precisos con una cantidad relativamente pequeña de datos.

Metodología

Esta sección del documento describe los pasos involucrados en el desarrollo del modelo, detallando las actividades desde la documentación y el desarrollo hasta las pruebas. La metodología utilizada es el enfoque CRISP-DM (Cross Industry Standard Process for Data Mining) para la minería de datos.

Las técnicas de ciencia de datos y análisis de datos se originaron en la década de 1990 con el concepto de Knowledge Discovery in Databases (KDD), que implicaba extraer valor de los datos. Hacia finales de los 90, en un esfuerzo por estandarizar los proyectos de ciencia de datos, se desarrollaron dos metodologías: CRISP-DM (Proceso Estándar de la Industria Cruzada para la Minería de Datos) y SEMMA (Muestree, explore, modifique, modele y evalúe).

CRISP-DM (Cross-Industry Standard Process for Data Mining) es una metodología ampliamente utilizada en proyectos de ciencia de datos y minería de datos que proporciona un enfoque estructurado para guiar el desarrollo de un modelo predictivo. La metodología consta de seis fases.

Comprensión del negocio

Comprender los objetivos y requisitos del proyecto, identificar el problema a resolver y definir los objetivos.

Comprensión de datos

Recopilación y exploración de datos, identificación de problemas de calidad de datos y verificación de los datos esperados para los procesos de modelado.

Preparación de datos

Limpieza, transformación y selección de los datos más apropiados para ser utilizados en los modelos.

Modelado

Seleccionar las técnicas de modelado a utilizar, construir y evaluar los diferentes modelos.

Evaluación

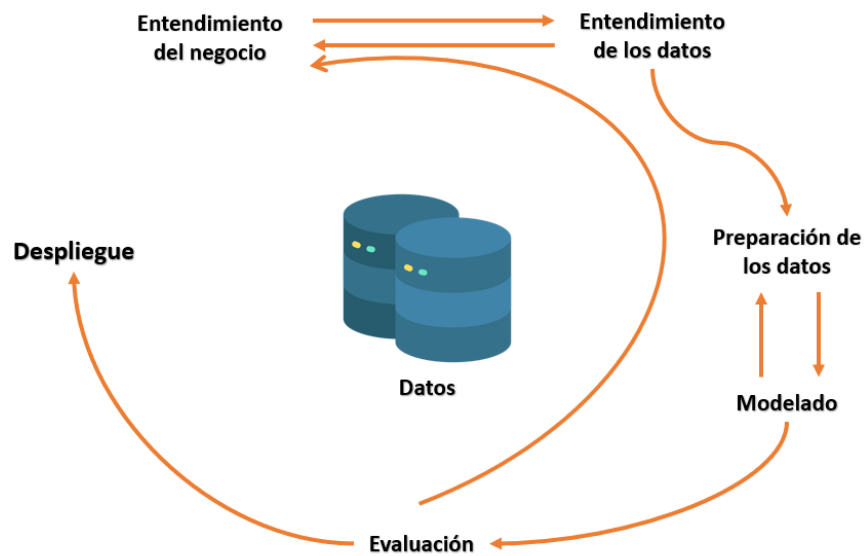
Evaluar la efectividad de los modelos, probar su generalización y asegurarse de que cumpla con los objetivos establecidos con anterioridad.

Despliegue

Integración de los modelos en el proceso del proyecto y seguimiento de su rendimiento.

Para el proyecto actual de simulación de datos ambientales, se seguirá la metodología CRISP-DM para guiar el desarrollo del modelo de simulación. En la fase de comprensión del negocio, se identificarán los objetivos y requisitos del proyecto, tales como simular el comportamiento de un ecosistema específico bajo diferentes escenarios de cambio climático. En la fase de comprensión de datos, se recopilarán y explorarán los datos relacionados con el ecosistema y el clima. En la fase de preparación de datos, se limpiarán, transformarán y seleccionarán los datos para crear el modelo de simulación. En la fase de modelado, se construirá y evaluará el modelo de simulación, seleccionando el más adecuado. En la fase de evaluación, se evaluará y probará la eficacia del modelo de simulación para asegurar que cumpla con los objetivos del proyecto. Por último, en la fase de implementación, se integrará el modelo en el proyecto y se monitoreará su rendimiento.

Figura 5 *Estructura de la metodología CRISP-DM*



Fuente: Elaboración propia.

Compresión del negocio

En esta fase se entenderá el propósito del modelo, contexto del ámbito ambiental y climatológico, terminología, objetivos y búsqueda de la información.

Para esta fase se tuvo previsto unas 5 semanas de investigación, entendimiento de los diferentes temas y búsqueda de los datos.

Tareas

Actividad 1.

Análisis de objetivos: En esta tarea se realizará un análisis detallado de los objetivos del proyecto que se van a llevar a cabo. Se definirán los resultados esperados y se establecerán los criterios para medir el éxito del proyecto.

Actividad 2.

Diseño de justificación y pregunta problema: En esta actividad se realizará una justificación del proyecto o investigación, es decir, se explicarán las razones por las cuales es importante llevar a cabo este proyecto. También se planteará la pregunta problema, que es la cuestión que se buscará responder con el proyecto.

Actividad 3.

Generación de alcances y limitaciones: En esta tarea se establecerán los límites del proyecto, es decir, se definirán las áreas que serán abarcadas y las que no.

Actividad 4.

Marco conceptual: En esta actividad se realizará un análisis detallado de los conceptos teóricos y/o prácticos que se relacionaran con el proyecto o investigación. Se buscará establecer una base teórica sólida que permita entender los conceptos y principios detrás del proyecto.

Actividad 5.

Búsqueda de antecedentes: En esta tarea se buscará información relevante previa que se haya llevado a cabo sobre el tema del proyecto. Se buscará identificar los resultados de investigaciones previas y los enfoques que se han utilizado.

Actividad 6.

Socialización y aprobación del documento: En esta tarea se socializará el documento que se ha generado hasta el momento con las partes interesadas y se buscara obtener su aprobación.

Actividad 7.

Correcciones del documento: En esta tarea se realizarán las correcciones necesarias en el documento en función de los comentarios y sugerencias recibidos durante la socialización y aprobación del documento.

Actividad 8.

Base de conocimiento: En esta tarea se establecerá una base de conocimiento sólida y completa que incluya todos los aspectos relevantes para el desarrollo del proyecto. Esto puede incluir información tanto práctica como teórica. Por ejemplo, es necesario comprender los modelos de Machine Learning que se utilizarán en el proyecto, así como los aspectos prácticos relacionados con la programación, el entrenamiento y la evaluación de estos modelos.

Además, se investigará y analizará la literatura existente para comprender mejor el estado del arte en el área específica del proyecto y estar al tanto de los últimos avances y técnicas.

También se considerará las mejores prácticas en cuanto a metodologías de trabajo y herramientas, con el fin de asegurar una implementación adecuada y eficiente. Será necesario evaluar y seleccionar las herramientas y recursos más apropiados para el proyecto, ya sea en términos de software, hardware o servicios en la nube.

Entendimiento de los datos

En esta fase se entenderán los datos obtenidos después de haberlos obtenido en las fuentes correspondientes, en el caso del proyecto y la fuente más confiable es Datos Abiertos, los cuales, son registros oficiales de estaciones climatológicas del IDEAM a nivel nacional para cada variable a usar en los entrenamientos, de igual manera, contexto del ámbito ambiental y climatológico, terminología, entre otros.

Para esta fase se tuvo previsto unas 4 semanas de investigación, entendimiento de los diferentes temas y búsqueda de los datos.

Tareas

Actividad 9.

Obtención de los datos: En esta actividad se llevará a cabo el proceso de obtención de los datos desde las fuentes correspondientes.

Actividad 10.

Analítica descriptiva de los datos: En esta tarea se analizarán los datos obtenidos y se buscará entender su estructura y contenido. Se aplicarán técnicas de estadística descriptiva para identificar patrones, tendencias y distribuciones de los datos.

Preparación de los datos

Además de comprender los datos, la preparación de datos es uno de los procesos más críticos en la preparación de los modelos. Por lo tanto, se asignó suficiente tiempo para preparar y estandarizar los datos para pasar a la siguiente fase. Esta fase tardó 5 semanas en completarse.

Tareas

Actividad 11.

Estandarización de datos: En esta actividad se llevará a cabo la estandarización de los datos para que puedan ser utilizados de manera uniforme.

Actividad 12.

Obtención de datos extras con respecto a la fecha (año, mes, día y hora): En esta tarea se obtendrán datos adicionales de la fecha y hora para complementar la información ya existente mediante Excel. Esto puede incluir la separación de la fecha y hora en variables separadas, así como la adición de información adicional como el año, mes y día.

Actividad 13.

Carga de datos al sistema de gestión de bases de datos relacional llamado MySQL: En esta tarea se cargarán los datos preparados en una base de datos relacional para facilitar su acceso y manipulación posterior. Se utilizará el sistema de gestión de bases de datos MySQL para esto.

Actividad 14.

Obtención de promedios horarios por variable: En esta tarea se calcularán los promedios horarios para cada variable. Esto permitirá una mejor comprensión de los datos y facilitará su uso en modelos posteriores.

Actividad 15.

Unión de variables para la construcción del conjunto de datos final: En esta tarea se unirán todas las variables preparadas y estandarizadas en un conjunto de datos final que se utilizará en los modelos posteriores.

Actividad 16.

Verificación y carga de datos a Google Collab: En esta tarea se verificará la integridad de los datos y se cargarán en una plataforma de colaboración como Google Collab para facilitar el trabajo colaborativo en el desarrollo de los modelos.

Modelado

Vale la pena señalar que los pasos antes mencionados debían llevarse a cabo con mucho cuidado, atención a los detalles y una comprensión profunda de cada etapa para garantizar un modelado preciso basado en datos de entrada de alta calidad. Por lo tanto, el equipo dedicó 5 semanas a la recopilación de información, comprensión y selección del modelo y lenguaje de programación más adecuados que se alinean con los objetivos del proyecto.

Tareas

Actividad 17.

Entendimiento del conjunto de datos: En esta tarea se utilizará el lenguaje de programación Python para comprender el conjunto de datos preparado en la fase anterior. Se pueden realizar diferentes análisis exploratorios de datos para comprender mejor las variables y la relación entre ellas.

Actividad 18.

Realización de analítica descriptiva a los datos: En esta actividad se llevará a cabo la analítica descriptiva a los datos utilizando Python. Se pueden realizar diferentes estadísticas descriptivas para entender mejor el comportamiento de los datos, como histogramas, boxplots, diagramas de dispersión, diagramas, cuadros de correlación, entre otros.

Actividad 19.

Realización de los diferentes modelos de Machine Learning: En esta tarea se llevará a cabo la construcción de los modelos de Machine Learning utilizando Python. Se pueden utilizar diferentes algoritmos de Machine Learning como Regresión Lineal, KNN, Redes Neuronales, entre otros, para alcanzar los objetivos del proyecto. Esta tarea puede llevar varias semanas dependiendo de la cantidad de modelos que se deseen construir y la complejidad de los algoritmos utilizados. Es importante tener en cuenta que la evaluación de los modelos también es una tarea crítica en esta fase y debe ser realizada cuidadosamente para garantizar un modelado preciso y eficaz.

Evaluación

Si los pasos anteriores se llevaron a cabo correctamente, la evaluación debería arrojar resultados satisfactorios. Dado que esta es una tarea meticulosa, se necesitaron 3 semanas para realizar las pruebas funcionales y las pruebas en tiempo real.

Tareas

Actividad 20.

Pruebas funcionales al código: En esta actividad se realizarán pruebas funcionales al código implementado para garantizar que funcione correctamente.

Actividad 21.

Comparación de datos reales con los datos obtenidos mediante el modelo realizado: En esta actividad se compararán los datos reales con los datos obtenidos a través de los modelos construidos en la fase anterior. Se pueden utilizar diferentes técnicas de evaluación para comparar los datos, como el error cuadrático medio, la correlación, entre otros. Esta tarea es importante para evaluar la precisión del modelo y puede llevar algunos días dependiendo de la cantidad de datos a comparar y la complejidad de las técnicas utilizadas.

Es importante mencionar que en esta fase también se pueden realizar ajustes y mejoras a los modelos en caso de ser necesario para garantizar una mayor precisión en los resultados.

Despliegue

Finalmente, como se menciona en el alcance, la información obtenida del modelo se cargará en una base de datos para su consumo y manipulación. Por lo tanto, fueron necesarias 2 semanas para la construcción de la estructura de la base de datos y la carga de datos.

Tareas

Actividad 22.

Desarrollo de la estructura de la base de datos: En esta actividad se desarrollará la estructura de la base de datos que contendrá los datos obtenidos a través del modelo. Esto incluye definir las tablas y campos necesarios para almacenar los datos de manera eficiente y estructurada.

Después de completar esta actividad, se procedería con la carga de los datos previamente obtenidos con el modelo a la base de datos construida por medio de un script realizado en el lenguaje de programación Python.

Teniendo en cuenta lo anteriormente mencionado, se definieron las siguientes tareas y cronograma realizadas durante todo el proceso.

Tabla 2 Cronograma de actividades

	Ago.				Sept				Oct				Nov			
Actividades	S. 1	S. 2	S. 3	S. 4	S. 5	S. 6	S. 7	S. 8	S. 9	S. 10	S. 11	S. 12	S. 13	S. 14	S. 15	S. 16
Act. #1	X	X														
Act. #2	X	X														
Act. #3		X														
Act. #4		X	X	X												
Act. #5			X	X												
Act. #6				X												
Act. #7				X												
Act. #8				X	X											
Act. #9					X											
Act. #10					X	X										
Act. #11					X	X										
Act. #12						X										
Act. #13							X									
Act. #14							X	X								
Act. #15								X	X							
Act. #16									X							
Act. #17									X	X						

Act. #18										X	X					
Act. #19											X	X	X			
Act. #20													X			
Act. #21													X	X	X	
Act. #22															X	X

Fuente: Elaboración propia.

Resultados

Para dar respuesta a los objetivos planteados en la actual investigación, se clasifican los resultados presentándolos en dos grandes grupos. El primero incluirá la explicación del tratamiento y el entendimiento de los datos, así como el entrenamiento y las pruebas de los modelos; el segundo grupo presentará los resultados obtenidos mediante las pruebas de los modelos.

Análisis exploratorio y desarrollo de los modelos de Machine Learning

Extracción y generación de datos para los entrenamientos de modelos

En primer lugar, se extrajeron los datos de la base de datos oficial del gobierno colombiano, Datos Abiertos Colombia, en formato csv. Luego, se convirtieron a formato xlsx para poder desglosar la columna de fecha de observación en nuevas columnas con los datos separados de las fechas.

Una vez obtenidas las columnas necesarias, se cargó el archivo csv resultante del proceso anterior en Google Collab, el ambiente en el que se desarrollaron los modelos.

Para manipular los datos y trabajar con los modelos, se necesitaron varias librerías que se importaron al inicio del código. Algunas de las librerías generales que se utilizaron fueron Pandas para la manipulación de los DataFrames, Matplotlib para la visualización de los datos, y Scikit-learn para el desarrollo de los modelos de Machine Learning. Si desea obtener más detalles sobre las librerías empleadas (ver Anexo 1).

Dentro de la manipulación de los datos (ver Anexo 2), se tuvo que adaptar las clases de cada columna del DataFrame para evitar errores en el desarrollo de los modelos. Por ejemplo, la columna que contiene la fecha se convirtió a la clase Datetime.

Entendimiento de los datos previamente cargados

La línea de código (Ver Figura 6) `df.describe()` genera un conjunto de estadísticas descriptivas que resumen la tendencia central, la dispersión y la forma de la distribución del conjunto de datos. Estas estadísticas incluyen el recuento de observaciones, la media, la desviación estándar, el valor mínimo, los percentiles (25%, 50% y 75%) y el valor máximo de cada columna numérica en el DataFrame `df`.

Figura 6 *Estadísticas del dataset*

A screenshot of a Jupyter Notebook cell. On the left, there is a green checkmark icon and the text '0s'. To the right, the code '[21] # Estadísticas del dataset' and 'df.describe()' is displayed in a monospaced font.

Fuente: Elaboración propia.

Este código muestra los códigos únicos de las estaciones en el DataFrame `df` (Ver Figura 7). La variable `estaciones` almacena un arreglo con los códigos de estación únicos. Luego, se imprime el arreglo con la función `print()`.

Figura 7 *Códigos de estaciones*

```

✓ [23] # ¿Qué estaciones tenemos?
0s     estaciones=df["CodigoEstacion"].unique()
      print(estaciones)

      [21205012 21206960 21205710]

```

Fuente: Elaboración propia.

Histogramas.

Este código genera un histograma de las columnas, en este caso como ejemplo, temperaturas, del DataFrame 'df' (Ver Figura 8). Primero, se convierte la columna 'Temperatura' en una serie de números enteros y se especifican los intervalos del histograma como un rango de números enteros. Luego se crea el histograma utilizando la biblioteca matplotlib.pyplot. Se establece el título y las etiquetas del eje, y se muestran las marcas del eje x utilizando los intervalos especificados. Finalmente, se muestra el histograma utilizando la función show() de matplotlib.pyplot.

Figura 8 *Generador de histogramas*

```

[ ] # Convertir la columna Temperatura en una serie de números enteros
    temperaturas = df['Temperatura'].astype(int)

    # Especificar los intervalos de los histogramas como un rango de números enteros
    intervalos = range(min(temperaturas), max(temperaturas) + 2)

    # Crear el histograma
    plot.hist(x=temperaturas, bins=intervalos, color='#F9992C', rwidth=0.85)

    # Establecer el título y etiquetas del eje
    plot.title('Histograma de las temperaturas')
    plot.xlabel('Temperaturas')
    plot.ylabel('Frecuencia')
    plot.xticks(intervalos)

    # Mostrar el histograma
    plot.show()

```

Fuente: Elaboración propia.

En el Anexo 4 se demuestra a mejor profundidad el histograma generado de la figura anterior mostrando el comportamiento de la temperatura con respecto a la hora del día.

Diagramas de correlación

De igual manera, se obtuvieron los diagramas de correlación entre cada variable (ver Figura 9), en este caso y nuevamente como ejemplo, la temperatura. Primero, se crea una figura de Matplotlib y un eje utilizando la función `subplots()` de Matplotlib, que devuelve una figura y un eje. Luego, se utiliza la función `plot()` de Pandas para crear un gráfico de dispersión en el eje creado anteriormente, utilizando la columna "Temperatura" como variable en el eje y y la columna "Hora" como variable en el eje x.

Figura 9 *Diagramas de correlación por variable*

```
[36] # Crear una figura de matplotlib y un eje
fig, ax1 = plt.subplots()

# Crear el gráfico de dispersión utilizando la columna "Temperatura" en el eje y, y la columna "Hora" en el eje x
df.plot("Temperatura", "Hora", color='#F9992C', kind="scatter", ax=ax1)

# Realizar ajustes al gráfico utilizando matplotlib
ax1.set_xlabel("Hindfoot length") # Establecer etiqueta del eje x
ax1.tick_params(labelsize=16, pad=8) # Ajustar tamaño y separación de las marcas en los ejes
fig.suptitle('Scatter plot of weight versus hindfoot length', fontsize=15) # Establecer título del gráfico
```

Fuente: Elaboración propia.

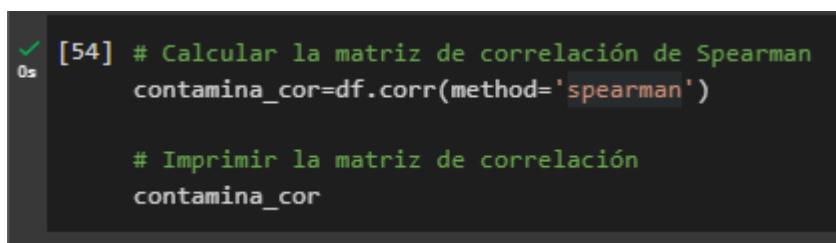
Después, se ajusta el gráfico utilizando funciones de Matplotlib, estableciendo la etiqueta del eje x con `set_xlabel()`, ajustando el tamaño y la separación de las marcas en los ejes con `tick_params()`, y estableciendo el título del gráfico con `suptitle()`. Para observar detalladamente el diagrama de correlación obtenido en la figura anterior ver Anexo 5.

Diagrama de matriz de correlaciones

En las líneas mostradas en la Figura 10 se calcula la matriz de correlación de Spearman entre las diferentes variables en el DataFrame df. La correlación de Spearman es una medida no paramétrica de la correlación entre dos variables. Se utiliza cuando las variables no tienen una distribución normal o cuando los datos son ordinales en lugar de continuos.

La función `corr()` (ver Figura 10) es utilizada para calcular la matriz de correlación, y el argumento `method` se establece en `'spearman'` para calcular la correlación de Spearman en lugar de la correlación de Pearson predeterminada. La matriz resultante es una tabla que muestra la correlación entre todas las combinaciones de pares de variables en el DataFrame.

Figura 10 *Calcular matriz de correlación*

A screenshot of a Jupyter Notebook cell. On the left, there is a green checkmark icon and the text '0s'. The code in the cell is as follows:

```
[54] # Calcular la matriz de correlación de Spearman
      contaminacion_cor=df.corr(method='spearman')

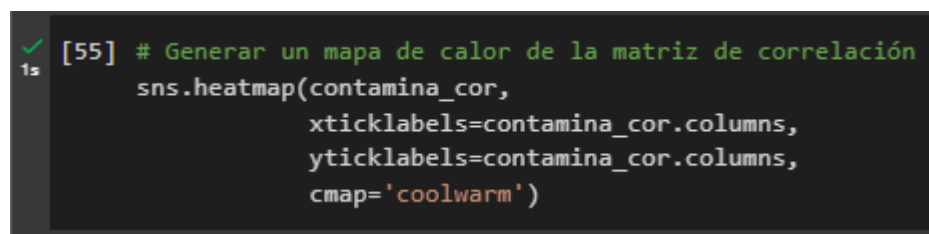
      # Imprimir la matriz de correlación
      contaminacion_cor
```

Fuente: Elaboración propia.

Para generar un mapa de calor de la matriz de correlación `contaminacion_cor` (ver Figura 11), se utiliza la biblioteca `seaborn`. El mapa de calor muestra la relación entre cada par de variables en el conjunto de datos utilizando un gradiente de color. La intensidad del color indica la fuerza y dirección de la correlación entre las variables (Ver Anexo 6).

El argumento `xticklabels` especifica las etiquetas para los marcadores en el eje x, y `yticklabels` especifica las etiquetas para los marcadores en el eje y. El argumento `cmap` especifica la paleta de colores utilizada para el mapa de calor. En este caso, se utiliza la paleta de colores "coolwarm", que va desde azul frío a rojo cálido.

Figura 11 *Generador mapa de calor*

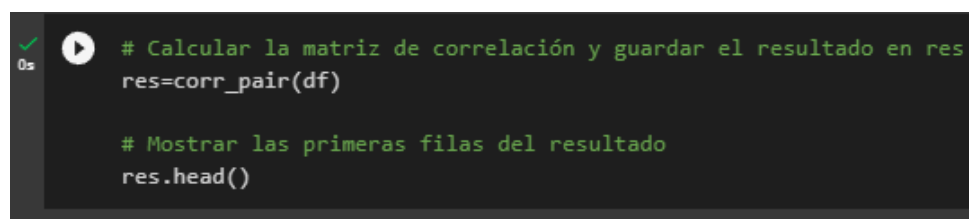


```
[55] # Generar un mapa de calor de la matriz de correlación
sns.heatmap(contamina_cor,
            xticklabels=contamina_cor.columns,
            yticklabels=contamina_cor.columns,
            cmap='coolwarm')
```

Fuente: Elaboración propia.

La función `corr_pair()` se utiliza para calcular la matriz de correlación entre todas las columnas numéricas del DataFrame `df`. La salida es un nuevo DataFrame `res` que contiene las correlaciones entre cada par de columnas. La segunda línea muestra las primeras filas de `res`, lo que permite visualizar algunas de las correlaciones calculadas (ver Figura 12).

Figura 12 *Virtualización de correlaciones*



```
# Calcular la matriz de correlación y guardar el resultado en res
res=corr_pair(df)

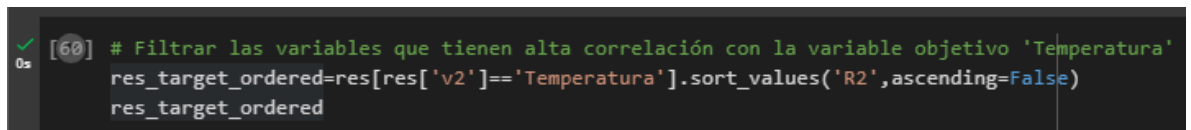
# Mostrar las primeras filas del resultado
res.head()
```

Fuente: Elaboración propia.

El código de la Figura 13 filtra las variables del DataFrame `res` que tienen alta correlación con la variable objetivo 'Temperatura', y las ordena en orden descendente según el valor de correlación. La primera línea de código filtra las filas del DataFrame `res` que tienen el valor 'Temperatura' en la columna 'v2', es decir, filtra las variables que están altamente correlacionadas con la temperatura. Luego, se utiliza el método `sort_values()` para ordenar el DataFrame resultante en orden descendente según el valor de correlación, que está almacenado en la columna 'R2'.

El resultado es un DataFrame llamado 'res_target_ordered' que contiene información sobre las variables del DataFrame original que tienen alta correlación con la temperatura, ordenadas por el valor de correlación.

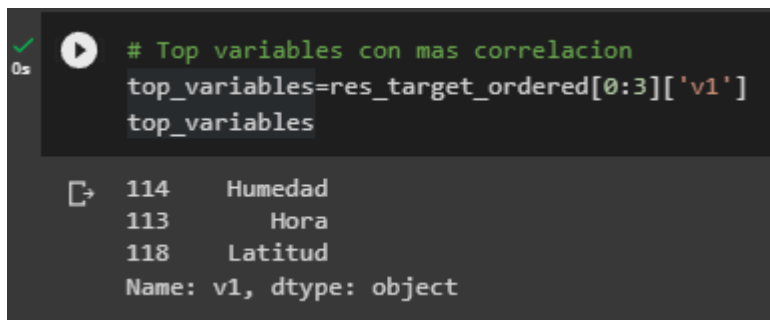
Figura 13 *Filtración de variables de alta correlación*



```
[60] # Filtrar las variables que tienen alta correlación con la variable objetivo 'Temperatura'
res_target_ordered=res[res['v2']=='Temperatura'].sort_values('R2',ascending=False)
res_target_ordered
```

Fuente: Elaboración propia.

Estas líneas en el código de la figura 14, filtran las variables del DataFrame `res` que tienen una alta correlación con la variable objetivo, en este caso, 'Temperatura', ordenándolas de mayor a menor correlación, y mostrando las 3 primeras.

Figura 14 Variables con más relación

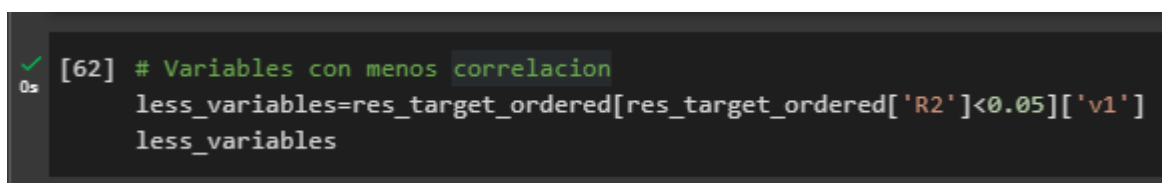
```
✓ 0s # Top variables con mas correlacion
top_variables=res_target_ordered[0:3]['v1']
top_variables
```

114	Humedad
113	Hora
118	Latitud

Name: v1, dtype: object

Fuente: Elaboración propia.

En esta parte del código de la Figura 15, se filtran las variables que tienen una correlación con la variable objetivo ‘Temperatura’ menor al 0.05, es decir, aquellas variables que tienen poca correlación con la variable objetivo. Los resultados se guardan en la variable ‘less_variables’.

Figura 15 Variables con menos relación

```
✓ 0s [62] # Variables con menos correlacion
less_variables=res_target_ordered[res_target_ordered['R2']<0.05]['v1']
less_variables
```

Fuente: Elaboración propia.

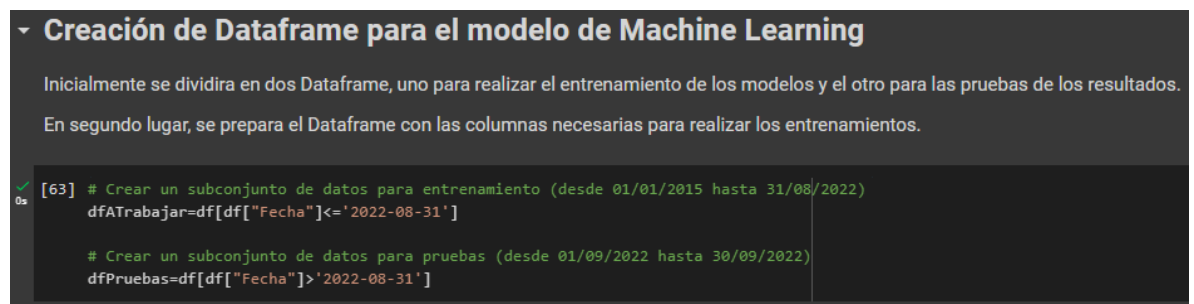
Cargue y entrenamiento de los datos para el entrenamiento de los modelos

Estas líneas de código de la Figura 16 crean dos subconjuntos de datos a partir del conjunto de datos original df.

dfATrabajar contiene los datos que se utilizarán para entrenar un modelo. En este caso, se han seleccionado todas las observaciones que tienen una fecha anterior o igual al 31 de agosto de 2022.

dfPruebas contiene los datos que se utilizarán para probar el modelo. En este caso, se han seleccionado todas las observaciones que tienen una fecha posterior al 31 de agosto de 2022 y anterior o igual al 30 de septiembre de 2022.

Figura 16 *Creación de subconjuntos para entrenamiento*



```

- Creación de Dataframe para el modelo de Machine Learning

Inicialmente se dividirá en dos Dataframe, uno para realizar el entrenamiento de los modelos y el otro para las pruebas de los resultados.
En segundo lugar, se prepara el Dataframe con las columnas necesarias para realizar los entrenamientos.

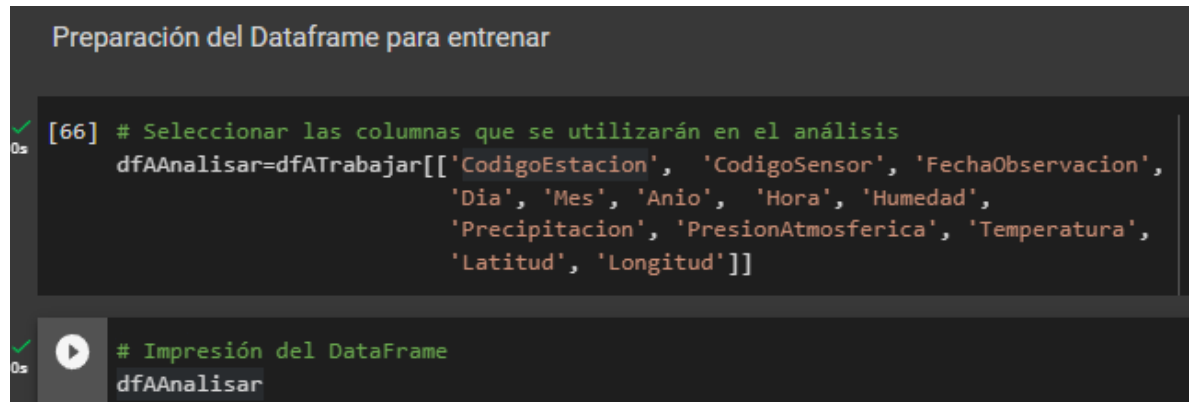
[63] # Crear un subconjunto de datos para entrenamiento (desde 01/01/2015 hasta 31/08/2022)
      dfATrabajar=df[df["Fecha"]<='2022-08-31']

      # Crear un subconjunto de datos para pruebas (desde 01/09/2022 hasta 30/09/2022)
      dfPruebas=df[df["Fecha"]>'2022-08-31']

```

Fuente: Elaboración propia.

En la figura 17 la línea de código selecciona un subconjunto de columnas del DataFrame dfATrabajar para utilizar en el análisis. El nuevo DataFrame se llama dfAAnalizar y contiene las columnas necesarias para entrenar los modelos. Por último, se imprime el DataFrame para comprobarlo.

Figura 17 Preparación del dataframe para entrenamiento


```

Preparación del Dataframe para entrenar

[66] # Seleccionar las columnas que se utilizarán en el análisis
      dfAAnalizar=dfATrabajar[['CodigoEstacion', 'CodigoSensor', 'FechaObservacion',
                              'Dia', 'Mes', 'Anio', 'Hora', 'Humedad',
                              'Precipitacion', 'PresionAtmosferica', 'Temperatura',
                              'Latitud', 'Longitud']]

# Impresión del DataFrame
dfAAnalizar

```

Fuente: Elaboración propia.

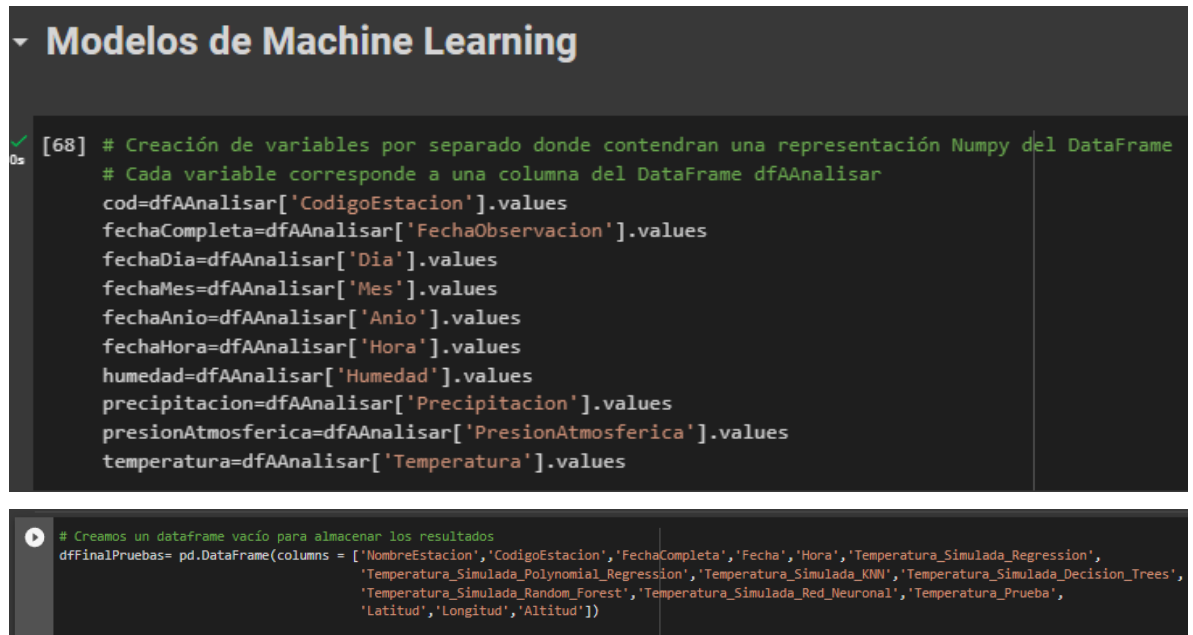
Modelos de Machine Learning

Las líneas de código de la figura 18, crean variables independientes para cada columna del DataFrame dfAAnalizar. Cada variable almacena una representación de la columna correspondiente en forma de un arreglo de valores numpy. Los nombres de las variables se asignan de acuerdo al nombre de la columna del DataFrame.

Luego, se crea un DataFrame vacío llamado dfFinalPruebas (Ver figura 18), que tendrá columnas para almacenar los resultados de la simulación de temperatura utilizando diferentes modelos de aprendizaje automático. El DataFrame tendrá las siguientes columnas:

'NombreEstacion','CodigoEstacion','FechaCompleta','Fecha','Hora','Temperatura_Simulada_Regression','Temperatura_Simulada_Polynomial_Regression','Temperatura_Simulada_KNN','Temperatura_Simulada_Decision_Trees','Temperatura_Simulada_Random_Forest','Temperatura_Simulada_Red_Neuronal','Temperatura_Prueba','Latitud','Longitud','Altitud'.

Figura 18 Creación de variables



```
Modelos de Machine Learning

[68] # Creación de variables por separado donde contendrán una representación Numpy del DataFrame
# Cada variable corresponde a una columna del DataFrame dfAAnalizar
cod=dfAAnalizar['CodigoEstacion'].values
fechaCompleta=dfAAnalizar['FechaObservacion'].values
fechaDia=dfAAnalizar['Dia'].values
fechaMes=dfAAnalizar['Mes'].values
fechaAnio=dfAAnalizar['Anio'].values
fechaHora=dfAAnalizar['Hora'].values
humedad=dfAAnalizar['Humedad'].values
precipitacion=dfAAnalizar['Precipitacion'].values
presionAtmosferica=dfAAnalizar['PresionAtmosferica'].values
temperatura=dfAAnalizar['Temperatura'].values

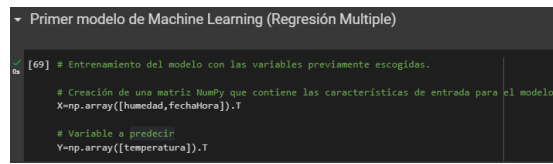
# Creamos un dataframe vacío para almacenar los resultados
dfFinalPruebas= pd.DataFrame(columns = ['NombreEstacion','CodigoEstacion','FechaCompleta','Fecha','Hora','Temperatura_Simulada_Regression',
'Temperatura_Simulada_Polynomial_Regression','Temperatura_Simulada_KNN','Temperatura_Simulada_Decision_Trees',
'Temperatura_Simulada_Random_Forest','Temperatura_Simulada_Red_Neuronal','Temperatura_Prueba',
'Latitud','Longitud','Altitud'])
```

Fuente: Elaboración propia.

Regresión Múltiple

En la Figura 19 las líneas de código se refieren al entrenamiento de un modelo de aprendizaje automático utilizando la regresión lineal como algoritmo de predicción. Las características de entrada del modelo se seleccionan como la humedad y la hora del día, y se utilizan para predecir la temperatura.

Figura 19 Entrenamiento del modelo



```
Primer modelo de Machine Learning (Regresión Multiple)

In [60]: # Entrenamiento del modelo con las variables previamente escogidas.

# Creación de una matriz NumPy que contiene las características de entrada para el modelo.
X=np.array([humedad,fechaHora]).T

# Variable a predecir
Y=np.array([temperatura]).T
```

Fuente: Elaboración propia.

En particular:

- `X=np.array([humedad,fechaHora]).T`: Se crea una matriz NumPy X que contiene las características de entrada del modelo. Aquí, las características son la humedad y la hora del día. Los valores de la humedad y la hora se extraen de las variables humedad y fechaHora, respectivamente, que se crearon previamente.
- `Y=np.array([temperatura]).T`: Se crea una matriz NumPy Y que contiene la variable objetivo a predecir, en este caso la temperatura. Los valores de la temperatura se extraen de la variable temperatura, que también se creó previamente.

Estas variables se utilizan como entrada para el modelo de regresión lineal que se entrena en la figura 20

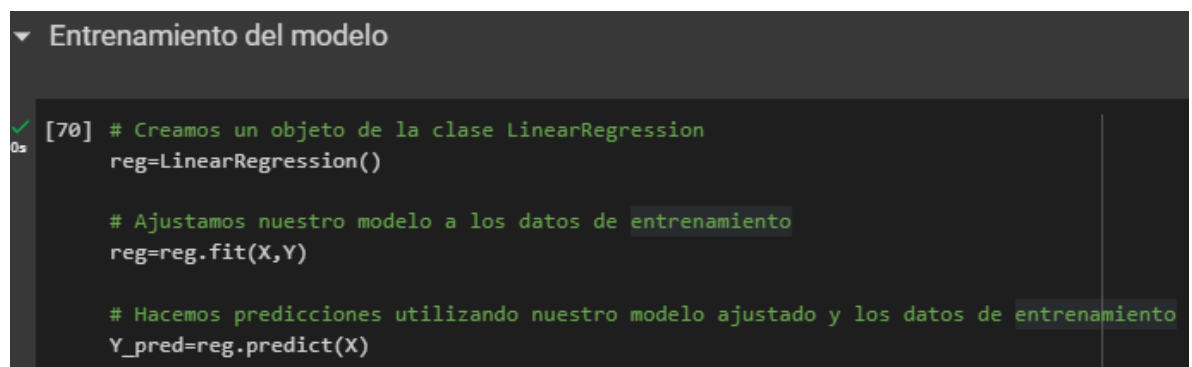
Estas líneas de código corresponden a la implementación del modelo de regresión lineal utilizando la librería scikit-learn. A continuación, se explica cada línea en detalle:

La primera línea de código crea un objeto de la clase `LinearRegression`. Este objeto será utilizado para realizar la regresión lineal.

En la segunda línea, se ajusta el modelo a los datos de entrenamiento. Es decir, se "entrena" el modelo con los datos de entrada X y las etiquetas de salida Y.

En la tercera línea, se hacen predicciones utilizando el modelo ajustado y los datos de entrada X. Estas predicciones se almacenan en la variable Y_pred.

Figura 20 *Entrenamiento de modelo de regresión lineal*



```
▼ Entrenamiento del modelo

[70] # Creamos un objeto de la clase LinearRegression
      reg=LinearRegression()

      # Ajustamos nuestro modelo a los datos de entrenamiento
      reg=reg.fit(X,Y)

      # Hacemos predicciones utilizando nuestro modelo ajustado y los datos de entrenamiento
      Y_pred=reg.predict(X)
```

Fuente: Elaboración propia.

En la figura 21 las líneas de código se realizan para cada resultado de los modelos realizados, por esta razón, no se repetirá más sus capturas, pero se quiere recalcar que se usan en todos, las anteriores líneas de código calculan el error cuadrático medio y el coeficiente de determinación del modelo de regresión lineal entrenado previamente.

Error es la raíz del error cuadrático medio (RMSE) entre las verdaderas observaciones de la temperatura Y las predicciones Y_pred hechas por el modelo de regresión lineal. El cálculo se realiza utilizando la función mean_squared_error de la librería scikit-learn.

R2 es el coeficiente de determinación (R^2), una medida que indica la proporción de la varianza de la variable dependiente explicada por el modelo. Cuanto más cerca de 1 esté este valor, mejor será la capacidad del modelo para explicar la variabilidad de los datos. Se calcula utilizando el método score del objeto reg de la clase LinearRegression.

Los coeficientes del modelo se imprimen utilizando el atributo coef_ del objeto reg. En este caso, se trata de los coeficientes para las variables humedad y fechaHora, que se utilizaron para predecir la temperatura en todos los modelos como se podrá evidenciar más adelante en este documento.

Figura 21 Evaluación del modelo

```
Evaluación del modelo

[74] # Variables obtenidas despues de entrenar el modelo
# Calculamos el error cuadrático medio entre los valores reales (Y) y las predicciones (Y_pred) y se obtiene
# la raíz cuadrada del error cuadrático medio para obtener la raíz del error cuadrático medio (RMSE)
error=np.sqrt(mean_squared_error(Y,Y_pred))

# Calculamos el coeficiente de determinación ( $R^2$ ) utilizando el método score() del objeto reg
# Entre mas cercano a 1 este, mejor sera el resultado.
r2=reg.score(X,Y)

# Impresión de los resultados
print("El error es de: ", error)
print("Valor de  $r^2$ : ", r2)
print("Los coeficientes son: ", reg.coef_)
```

Fuente: Elaboración propia.

En el código de la figura 22 se inicializa un contador i en cero y se itera sobre cada fila del DataFrame de pruebas (dfPruebas). En cada iteración se extraen las variables de interés como el código de la estación, la fecha y hora de la observación, la humedad, la precipitación, la presión atmosférica, la temperatura de prueba, la latitud y la longitud.

Se utiliza el modelo de regresión previamente entrenado para simular la temperatura y se almacena el resultado en la columna 'Temperatura_Simulada_Regression' del DataFrame `dfFinalPruebas`. Luego, se verifica a qué estación pertenece el registro actual y se almacena el registro completo en la fila correspondiente en `dfFinalPruebas`.

Finalmente, se incrementa el contador del índice del DataFrame `dfFinalPruebas` y se repite el proceso para el resto de las filas del DataFrame de pruebas.

Estas líneas de código de igual manera se realizan para cada resultado de los modelos realizados, también, no se repetirá más las capturas, pero se quiere recalcar que se usan en todos y lo que hacen es calcular el coeficiente de determinación (R^2) entre los valores reales de temperatura (almacenados en la columna "Temperatura_Prueba" del dataframe "dfFinalPruebas") y los valores simulados de temperatura utilizando el modelo de regresión (almacenados en la columna "Temperatura_Simulada_Regression" del mismo dataframe). El resultado se almacena en la variable "r2" y se imprime en pantalla. El coeficiente de determinación es una medida de la calidad del ajuste del modelo, y su valor puede oscilar entre 0 y 1. Un valor de 1 indica un ajuste perfecto, mientras que un valor de 0 indica que el modelo no es mejor que una simple predicción basada en la media de los datos. Por lo tanto, cuanto más cercano a 1 sea el valor de R^2 , mejor será la calidad del modelo.

Figura 22 Inicializador de contador e iteración de filas

```

# Inicializamos un contador para el índice del dataframe
i=0

# Iteramos sobre cada fila del dataframe de pruebas
for index, row in dfPruebas.iterrows():

    # Obtenemos las variables por registro del Dataframe que contiene los registros de septiembre
    estacion=row["CodigoEstacion"]
    fechaCompleta=row["FechaObservacion"]
    fecha=row["Fecha"]
    fechaDia=row["Dia"]
    fechaHora=row["Hora"]
    humedad=row["Humedad"]
    precipitacion=row["Precipitacion"]
    presionAtmosferica=row["PresionAtmosferica"]
    temperatura_Prueba=row["Temperatura"]
    latitud=row["Latitud"]
    longitud=row["Longitud"]

    # Simulamos la temperatura utilizando el modelo de regresión
    temperatura_Simulada=reg.predict([[humedad, fechaHora]])
    temperatura_Simulada=temperatura_Simulada[0][0]

    # Almacenamos los resultados en el dataframe de resultados
    if row['CodigoEstacion']==21206960:
        dfFinalPruebas.loc[i]=['IDEAM BOGOTA - AUT [21206960]', estacion, fechaCompleta, fecha, fechaHora,
                                temperatura_Simulada, 0, 0, 0, 0, temperatura_Prueba,
                                latitud, longitud, 2.646]
    elif row['CodigoEstacion']==21205012:
        dfFinalPruebas.loc[i]=['UNIVERSIDAD NACIONAL - AUT [21205012]', estacion, fechaCompleta, fecha, fechaHora,
                                temperatura_Simulada, 0, 0, 0, 0, temperatura_Prueba,
                                latitud, longitud, 2.556]
    else:
        dfFinalPruebas.loc[i]=['JARDIN BOTANICO - AUT [21205710]', estacion, fechaCompleta, fecha, fechaHora,
                                temperatura_Simulada, 0, 0, 0, 0, temperatura_Prueba,
                                latitud, longitud, 2.552]

    # Incrementamos el contador del índice del dataframe
    i=i+1

```

▼ Evaluación de las temperaturas obtenidas por el modelo de regresión

```

# Se calcula el coeficiente de determinación (R²) utilizando los valores reales de temperatura (Temperatura_Prueba)
# y los valores simulados de temperatura utilizando el modelo y se almacena en la variable r2.
r2 = r2_score(dfFinalPruebas['Temperatura_Prueba'], dfFinalPruebas['Temperatura_Simulada_Regression'])
print("El r^2 obtenido despues de las pruebas para el modelo con regresión es de: ", r2)

```

Fuente: Elaboración propia.

Regresión Polinomial

Observando la figura 23, las líneas de código realizan la implementación del modelo de Regresión Polinomial y evalúan su precisión mediante el coeficiente de determinación (R^2). A continuación, se detalla cada una de ellas:

La primera línea divide los datos en conjuntos de entrenamiento y prueba. Se seleccionan las características (humedad y hora) y la variable objetivo (temperatura) de `dfAAnalizar`. `train_test_split` divide los datos aleatoriamente en conjuntos de entrenamiento y prueba en una proporción de 80% y 20%, respectivamente. El parámetro `random_state` se utiliza para garantizar que la división se realice de la misma manera cada vez que se ejecute el código.

La segunda línea realiza el preprocesamiento de datos para la Regresión Polinomial. Se utiliza la clase `PolynomialFeatures` para generar características polinomiales de grado 2 a partir de las características originales.

La tercera línea entrena el modelo de Regresión Polinomial utilizando las características polinomiales de grado 2 generadas y la variable objetivo de temperatura.

La cuarta línea realiza las predicciones sobre los datos de prueba utilizando el modelo de Regresión Polinomial entrenado.

La quinta línea evalúa la precisión del modelo utilizando el coeficiente de determinación (R^2). Se compara la variable objetivo de temperatura real en los datos de prueba con las predicciones generadas por el modelo. Cuanto más cercano a 1 sea el valor de R^2 , mejor será la precisión del modelo en predecir los valores de temperatura.

Figura 23 *Implementación del modelo de Regresión Polinomial y evalúan su precisión mediante el coeficiente de determinación*

```

▼ Segundo modelo de Machine Learning (RandomTree con regresión)

▼ Modelo de Machine Learning (Regresión polinomial)

[84] # Dividir los datos en conjuntos de entrenamiento y prueba
X = dfAAnalizar[['Humedad','Hora']] # Características
y = dfAAnalizar['Temperatura'] # Variable objetivo
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

# Preprocesamiento de datos para la regresión polinomial
poly = PolynomialFeatures(degree=2) # Cambiar el valor de 'degree' según el grado del polinomio que desees
X_poly = poly.fit_transform(X_train)

# Entrenar el modelo de regresión polinomial
regressor = LinearRegression()
regressor.fit(X_poly, Y_train)

# Realizar las predicciones sobre los datos de prueba
X_poly_test = poly.transform(X_test)
Y_pred = regressor.predict(X_poly_test)

# Evaluar la precisión del modelo
r2 = r2_score(Y_test, Y_pred)

```

Fuente: Elaboración propia.

En el código mostrado en la figura 24, el cual también se repite en todos los modelos desde Regresión Polinomial en adelante, recorre el dataframe de pruebas y se realiza la simulación de temperatura utilizando el modelo de Regresión Polinomial que se entrenó previamente. Los valores de temperatura simulados se almacenan en la columna "Temperatura_Simulada_Polynomial_Regression" del dataframe dfFinalPruebas. Para cada registro en el dataframe de pruebas, se accede a la posición correspondiente en la columna de temperatura simulada y se le asigna el valor correspondiente de la predicción del modelo de Regresión Polinomial.

Figura 24 Creación de columnas con nuevos datos

```
✓ [90] # Se crea una nueva columna con los datos obtenidos por Regresión Polinomial
0s
    i=0
    for index, row in dfPruebas.iterrows():
        dfFinalPruebas['Temperatura_Simulada_Polynomial_Regression'].loc[i]=Y_pred[i]
        i=i+1
```

Fuente: Elaboración propia.

En la figura 25 el código permite seleccionar las características (variables predictoras) que se utilizarán para predecir la variable de salida (la temperatura). Las variables predictoras seleccionadas son "Humedad" y "Hora".

Luego, se asigna a la variable Y los valores de la columna "Temperatura" del DataFrame "dfAAnalizar". Esta es la variable que se intentará predecir a partir de las variables predictoras seleccionadas.

Figura 25 Modelo de Machine Learning KNN con regresión

```
▼ Tercer modelo de Machine Learning (KNN con regresión)

✓ [94] # Seleccionar las características (Humedad y Hora) que se utilizarán para predecir la variable de salida
0s      X=dfAAnalizar[['Humedad','Hora']]

      # VVariable a predecir
      Y=dfAAnalizar[['Temperatura']]

▼ Modelo de Machine Learning (KNN)

✓ [97] # Dividir los datos en conjuntos de entrenamiento y prueba
0s      # Se divide X en X_train (conjunto de características de entrenamiento) y X_test (conjunto de características de prueba)
      # Se divide Y en Y_train (conjunto de variables de salida de entrenamiento) y Y_test (conjunto de variables de salida de prueba)
      # test_size=.3 indica que el 30% de los datos se utilizarán para el conjunto de prueba y el 70% para el conjunto de entrenamiento
      # random_state=0 se utiliza para que la división se haga de manera determinista, de tal manera que la división siempre sea la misma
      X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=0)
```

Fuente: Elaboración propia.

Este código de la Figura 26 está dividiendo los datos en conjuntos de entrenamiento y prueba utilizando la función `train_test_split` de la librería `scikit-learn`.

X representa las características que se utilizarán para predecir la variable de salida Y. Y es la variable de salida que se quiere predecir.

`Test_size=0.2` indica que el 20% de los datos se utilizarán para el conjunto de prueba y el 70% para el conjunto de entrenamiento. `random_state=0` se utiliza para que la división se haga de manera determinista, de tal manera que la división siempre sea la misma.

X_train y Y_train son los conjuntos de características de entrenamiento y variables de salida de entrenamiento, respectivamente. X_test y Y_test son los conjuntos de características de prueba y variables de salida de prueba, respectivamente.

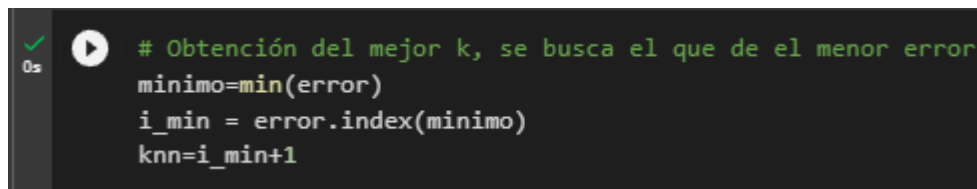
Figura 26 *Búsqueda del mejor k*

```
[98] # Búsqueda del mejor K para el modelo.  
# Se prepara una lista con valores del 1 al 100, se prueban todos estos y se obtiene  
# su error cuadrático medio por cada K  
error=[]  
for i in range(1, 101, 1):  
    clf=KNeighborsRegressor(i)  
    clf.fit(X_train,Y_train)  
    Y_pred=clf.predict(X_test)  
    error.append(mean_squared_error(Y_test,Y_pred))
```

Fuente: Elaboración propia.

Este fragmento de código de la Figura 27 realiza una búsqueda del mejor valor de k para el modelo K-Nearest Neighbors (KNN) utilizando el método de validación cruzada con una partición 80/20 entre conjuntos de entrenamiento y prueba. Para hacerlo, se entrena un modelo KNN con diferentes valores de k (del 1 al 100) y se evalúa su desempeño en el conjunto de prueba mediante el cálculo del error cuadrático medio (MSE). Por último, el error cuadrático medio se almacena en una lista llamada `error`.

Figura 27 *Obtención del mejor k*

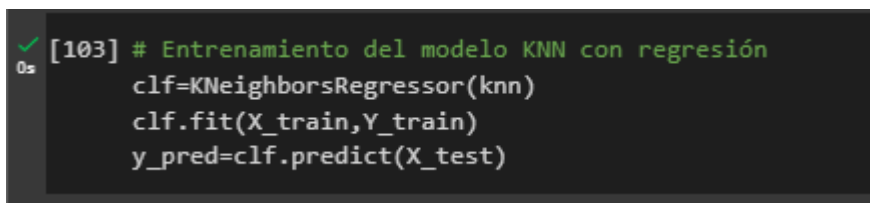


```
✓ 0s # Obtención del mejor k, se busca el que de el menor error
minimo=min(error)
i_min = error.index(minimo)
knn=i_min+1
```

Fuente: Elaboración propia.

Este código de la Figura 28 encuentra el valor de k óptimo para el modelo de regresión k-NN. El error cuadrático medio se calcula para valores de k desde 1 hasta 100 y se almacenan en la lista "`error`". Luego se busca el valor mínimo de la lista de errores y se determina su índice en la lista. Como los índices en Python comienzan en 0 y no en 1, se agrega 1 al índice para obtener el valor óptimo de k .

Figura 28 *Entrenamiento del modelo KNN con regresión*



```
✓ 0s [103] # Entrenamiento del modelo KNN con regresión
clf=KNeighborsRegressor(knn)
clf.fit(X_train,Y_train)
y_pred=clf.predict(X_test)
```

Fuente: Elaboración propia.

Este código está realizando el entrenamiento del modelo K-Nearest Neighbors (KNN) con regresión. Primero, se instancia el modelo de regresión KNN con el valor de k obtenido en la búsqueda del mejor k. Luego, se entrena el modelo con los datos de entrenamiento (X_train e Y_train) utilizando el método fit. Finalmente, se realiza la predicción sobre los datos de prueba (X_test) y se almacenan en la variable y_pred.

Arboles de decisión con regresión

En el código presentado en la figura 29 permite realizar:

1. Seleccionar las características que se utilizarán para predecir la variable de salida. En este caso, se seleccionan todas las columnas del DataFrame dfAAnalizar excepto Temperatura y FechaObservacion. Estas características se almacenan en la variable X.
2. Definir la variable objetivo, que es la columna Temperatura del DataFrame dfAAnalizar. Esta variable se almacena en la variable y.
3. Dividir los datos en conjuntos de entrenamiento y prueba utilizando la función train_test_split de scikit-learn. El conjunto de características de entrenamiento se almacena en la variable X_train, el conjunto de características de prueba se almacena en la variable X_test, el conjunto de variables de salida de entrenamiento se almacena en la variable y_train y el conjunto de variables de salida de prueba se almacena en la variable Y_test. La división se hace de tal manera que el 20% de los datos se utiliza para el conjunto de prueba y el 80% para el conjunto de

entrenamiento. Además, se establece una semilla (`random_state=0`) para hacer la división de manera determinista.

4. Definir un modelo de árboles de decisión utilizando la clase `DecisionTreeRegressor` de `scikit-learn`. Se establece una semilla aleatoria (`random_state=3`) para que los resultados sean reproducibles.
5. Entrenar el modelo de árboles de decisión con los datos de entrenamiento utilizando el método `fit` de la clase `DecisionTreeRegressor`.
6. Realizar las predicciones sobre los datos de prueba utilizando el método `predict` del modelo entrenado. Las predicciones se almacenan en la variable `Y_pred`.

Figura 29 *Modelo de Machine Learning arboles de decisión con regresión*

```
▼ Cuarto modelo de Machine Learning (Arboles de desiciones con regresión)

✓ [117] # Dividir los datos en conjuntos de entrenamiento y prueba
0s      X = dfAAnalizar.drop(["Temperatura", "FechaObservacion"], axis=1) # Características
        y = dfAAnalizar['Temperatura'] # Variable objetivo
        X_train, X_test, y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

        # Entrenar el modelo de árboles de decisión
        regressor = DecisionTreeRegressor(random_state=3)
        regressor.fit(X_train, y_train)

        # Realizar las predicciones sobre los datos de prueba
        Y_pred = regressor.predict(X_test)
```

Fuente: Elaboración propia.

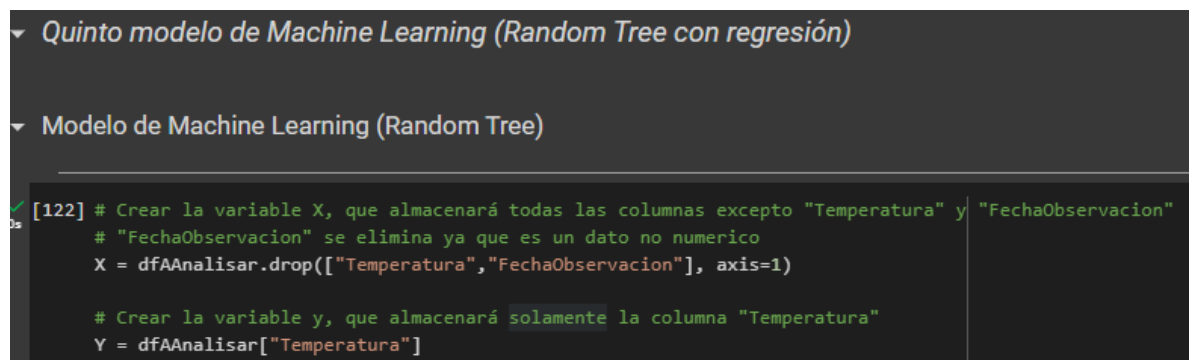
Random Tree con regresión

En la Figura 30 se observan las líneas de código que se utilizan para preparar los datos para un modelo de regresión.

En la primera línea, se crea la variable X, que contendrá todas las columnas del conjunto de datos original excepto "Temperatura" y "FechaObservacion". Esto se hace utilizando el método drop() de pandas para eliminar estas dos columnas del DataFrame original.

En la segunda línea, se crea la variable y, que contendrá solamente la columna "Temperatura" del DataFrame original. Esta columna será la variable objetivo que el modelo intentará predecir en función de las demás características del conjunto de datos.

Figura 30 *Modelo de Machine Learning Random Tree con regresión*



```
Quinto modelo de Machine Learning (Random Tree con regresión)

Modelo de Machine Learning (Random Tree)

[122] # Crear la variable X, que almacenará todas las columnas excepto "Temperatura" y "FechaObservacion"
      # "FechaObservacion" se elimina ya que es un dato no numerico
      X = dfAAnalizar.drop(["Temperatura", "FechaObservacion"], axis=1)

      # Crear la variable y, que almacenará solamente la columna "Temperatura"
      Y = dfAAnalizar["Temperatura"]
```

Fuente: Elaboración propia.

Las líneas de código de la Figura 31 crean un objeto scaler de la clase StandardScaler y lo utilizan para estandarizar el conjunto de datos X. La estandarización de los datos significa que se resta la media de cada característica y se divide por su desviación estándar para que todas las características tengan una media

de cero y una desviación estándar de uno. Esto es importante porque muchas técnicas de aprendizaje automático asumen que las características están en la misma escala.

En la segunda línea de código, se llama al método `fit_transform` del objeto `scaler` pasando `X` como argumento. Este método ajusta el scaler a `X` y luego aplica la transformación de estandarización a `X`, almacenando el resultado en `X_scaled`.

Figura 31 Estandarizar el conjunto de datos

```
[125] # Crear un objeto scaler de la clase StandardScaler
      scaler = StandardScaler()

      # Usar el objeto scaler para estandarizar el conjunto de datos X
      # La función "fit_transform" de StandardScaler estandariza las características de X,
      # es decir, resta la media y divide por la desviación estándar. El resultado se almacena en X_scaled.
      # La función "fit" ajusta el scaler a X y la función "transform" lo aplica a X.
      X_scaled = scaler.fit_transform(X)
```

Fuente: Elaboración propia.

En la Figura 32 se observa líneas de código donde utilizan la función `train_test_split` de `sklearn.model_selection` para dividir el conjunto de datos estandarizado `X_scaled` y la variable objetivo `Y` en conjuntos de entrenamiento y prueba. El argumento `test_size=0.20` indica que el 20% del conjunto de datos se reservará para la prueba. El argumento `random_state=42` se utiliza para establecer una semilla aleatoria y garantizar que la división de datos sea reproducible. `X_scaled` se divide en `X_train` (80%) y `X_test` (20%), y `Y` se divide en `Y_train` (80%) y `Y_test` (20%). Los conjuntos de entrenamiento se utilizarán para entrenar el modelo y los conjuntos de prueba se utilizarán para evaluar la precisión del modelo.

Figura 32 *División del conjunto de datos estandarizado*

```
[ ] # Importar la función train_test_split de sklearn.model_selection
    from sklearn.model_selection import train_test_split

    # Dividir el conjunto de datos estandarizado X_scaled en conjuntos de entrenamiento y prueba,
    # y separar la variable objetivo Y en los conjuntos de entrenamiento y prueba correspondientes
    # La función "train_test_split" de sklearn.model_selection divide los datos en conjuntos de entrenamiento y prueba.
    # El argumento "test_size=0.20" indica que el 20% del conjunto de datos se reservará para la prueba.
    # El argumento "random_state=42" se utiliza para establecer una semilla aleatoria y garantizar que la división de datos sea reproducible.
    # X_scaled se divide en X_train (80%) y X_test (20%), y Y se divide en Y_train (80%) y Y_test (20%).

    X_train, X_test, Y_train, Y_test = train_test_split(X_scaled, Y, test_size=0.20, random_state=42)
```

Fuente: Elaboración propia.

Estas líneas de código están creando un modelo de regresión aleatoria (Random Forest Regressor) y entrenándolo, utilizando el conjunto de datos de entrenamiento (X_train y Y_train).

En la primera línea, se crea un objeto de la clase RandomForestRegressor y se especifica el número de árboles de decisión que se utilizarán en el modelo (n_estimators=100).

En la segunda línea, se entrena el modelo utilizando el método fit(), que ajusta el modelo a los datos de entrenamiento (X_train y Y_train). Durante el entrenamiento, el modelo ajusta los parámetros de cada árbol de decisión para minimizar el error en la predicción de la variable objetivo (Y_train) utilizando las características de entrada (X_train).

Finalmente, se realiza una predicción en el conjunto de prueba X_test utilizando el método predict() del objeto modelo y se guarda el resultado en y_pred (ver Figura 33).

Figura 33 *Predicción en el conjunto de prueba*

```

✓ [133] # Crea el clasificador
1m      model = RandomForestRegressor(n_estimators=100)

      # Entrena el modelo usando el conjunto de entrenamiento
      model.fit(X_train, Y_train)

      # Predicción en el conjunto de prueba
      y_pred = model.predict(X_test)

```

Fuente: Elaboración propia.

Red Neuronal

Estas líneas de código tienen como objetivo entrenar y evaluar un modelo de Red Neuronal para predecir la temperatura en función de la hora del día y la humedad. A continuación, se detallan las acciones que se realizan en cada una de ellas:

`X = dfAAnalizar[['Hora', 'Humedad']]` y `y = dfAAnalizar['Temperatura']`: Se separan los datos de entrada (X) y la salida (y) del conjunto de datos `dfAAnalizar`, donde X contiene las columnas "Hora" y "Humedad" y y contiene la columna "Temperatura".

`X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)`: Se divide el conjunto de datos en conjuntos de entrenamiento (`X_train` y `y_train`) y de prueba (`X_test` y `y_test`) utilizando la función `train_test_split` de `scikit-learn`. La variable `test_size` indica que el 30% de los datos se utilizarán para la prueba y `random_state` se utiliza para obtener los mismos resultados en cada ejecución del script.

`Scaler = StandardScaler()`: Se crea un objeto de tipo `StandardScaler`, que se utilizará para escalar los datos de entrada.

`X_train = scaler.fit_transform(X_train)` y `X_test = scaler.transform(X_test)`:

Los datos de entrada se escalan utilizando el objeto `scaler` creado anteriormente.

`Best_r2_score = 0`: Se inicializa la variable `best_r2_score` con un valor de cero, que se utilizará para almacenar el mejor valor de R cuadrado (r^2) obtenido durante el entrenamiento del modelo.

`For hidden_layer_sizes in [(100,), (50, 50), (25, 25, 25)]::` Se inicia un bucle `for` que recorre diferentes configuraciones de capas ocultas en la Red Neuronal. En este caso se prueban 3 configuraciones diferentes, cada una con una cantidad diferente de capas ocultas.

`For alpha in [0.001, 0.01, 0.1]::` Se inicia otro bucle `for` que recorre diferentes valores de `alpha`, que se utiliza como un parámetro de regularización para evitar el sobreajuste del modelo.

`For learning_rate_init in [0.001, 0.01, 0.1]::` Se inicia otro bucle `for` que recorre diferentes valores de `learning_rate_init`, que se utiliza para ajustar la tasa de aprendizaje de la Red Neuronal.

`Model = MLPRegressor(hidden_layer_sizes=hidden_layer_sizes, alpha=alpha, learning_rate_init=learning_rate_init, random_state=42)`: Se crea un objeto de tipo `MLPRegressor` (regresor de Red Neuronal multicapa) con los parámetros especificados en los bucles anteriores.

`Model.fit(X_train, y_train)`: Se entrena el modelo utilizando los datos de entrenamiento.

$Y_{pred} = \text{model.predict}(X_{test})$ y $r2 = r2_score(y_{test}, y_{pred})$: Se utilizan los datos de prueba para evaluar el modelo y se calcula el valor de R cuadrado ($r2$) para medir su desempeño (ver Figura 34)

Figura 34 *Modelo de Machine Learning Red Neuronal*

```
▼ Sexto modelo de Machine Learning (Red Neuronal)

[112] from sklearn.neural_network import MLPRegressor
      from sklearn.preprocessing import StandardScaler
      from sklearn.model_selection import train_test_split
      from sklearn.metrics import r2_score

      # Separar los datos de entrada y salida
      X = dfAAnalizar[['Hora', 'Humedad']]
      y = dfAAnalizar['Temperatura']

      # Dividir los datos en conjuntos de entrenamiento y prueba
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

      # Escalar los datos de entrada
      scaler = StandardScaler()
      X_train = scaler.fit_transform(X_train)
      X_test = scaler.transform(X_test)

      # Entrenar el modelo de red neuronal con diferentes configuraciones
      best_r2_score = 0
      for hidden_layer_sizes in [(100,), (50, 50), (25, 25, 25)]:
          for alpha in [0.001, 0.01, 0.1]:
              for learning_rate_init in [0.001, 0.01, 0.1]:
                  # Crear el modelo
                  model = MLPRegressor(hidden_layer_sizes=hidden_layer_sizes, alpha=alpha, learning_rate_init=learning_rate_init, random_state=42)

                  # Entrenar el modelo
                  model.fit(X_train, y_train)

                  # Evaluar el modelo en el conjunto de prueba
                  y_pred = model.predict(X_test)
                  r2 = r2_score(y_test, y_pred)

                  # Actualizar la mejor configuración encontrada hasta ahora
                  if r2 > best_r2_score:
                      best_model = model
                      best_r2_score = r2
                      best_params = {'hidden_layer_sizes': hidden_layer_sizes, 'alpha': alpha, 'learning_rate_init': learning_rate_init}

      # Imprimir la mejor configuración encontrada y su desempeño
      print(f"Mejor configuración: {best_params}")
      print(f"R^2 score: {best_r2_score}")
```

Fuente: Elaboración propia.

Pruebas

Estas líneas de código tienen como objetivo analizar el desempeño de los diferentes modelos de Machine Learning para obtener la temperatura, comparando sus resultados con los datos de prueba.

La primera línea convierte la columna "Fecha" de la variable `dfFinalPruebas` en formato `datetime`. Luego, se ordena el dataframe por fecha y hora.

A continuación, se crean varias columnas nuevas en el dataframe `dfFinalPruebas` que representan las diferencias entre las temperaturas simuladas por cada modelo de Machine Learning y las temperaturas de prueba.

Finalmente, se calcula el promedio de las diferencias para cada modelo y se imprime en la consola para comparar los resultados. Esto permitirá determinar cuál de los modelos tiene un mejor desempeño en términos de precisión en la predicción de la temperatura (ver Figura 35)

Figura 35 *Calculo el promedio de las diferencias para cada modelo*

```
dfFinalPruebas['Fecha'] = pd.to_datetime(dfFinalPruebas['Fecha'], format="%Y/%m/%d")
dfFinalPruebas=dfFinalPruebas.sort_values(['Fecha', 'Hora'])

# Obtención de las diferencias entre las temperaturas obtenidas por cada modelo.
dfFinalPruebas['Diferencia_Regression']=dfFinalPruebas['Temperatura_Simulada_Regression']-dfFinalPruebas['Temperatura_Prueba']
dfFinalPruebas['Diferencia_Polynomial_Regression']=dfFinalPruebas['Temperatura_Simulada_Polynomial_Regression']-dfFinalPruebas['Temperatura_Prueba']
dfFinalPruebas['Diferencia_KNN']=dfFinalPruebas['Temperatura_Simulada_KNN']-dfFinalPruebas['Temperatura_Prueba']
dfFinalPruebas['Diferencia_Decision_Trees']=dfFinalPruebas['Temperatura_Simulada_Decision_Trees']-dfFinalPruebas['Temperatura_Prueba']
dfFinalPruebas['Diferencia_Random_Forest']=dfFinalPruebas['Temperatura_Simulada_Random_Forest']-dfFinalPruebas['Temperatura_Prueba']
dfFinalPruebas['Diferencia_Red_Neuronal']=dfFinalPruebas['Temperatura_Simulada_Red_Neuronal']-dfFinalPruebas['Temperatura_Prueba']

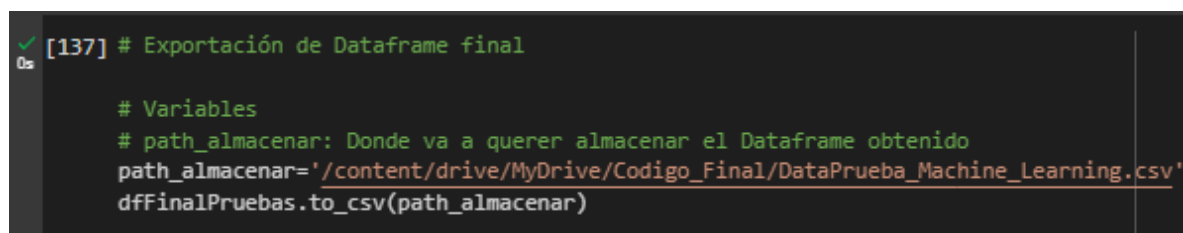
# Promedio de error por modelo realizado.
print("Promedio de diferencia con Regresión", dfFinalPruebas['Diferencia_Regression'].mean())
print("Promedio de diferencia con Regresión polinomial", dfFinalPruebas['Diferencia_Polynomial_Regression'].mean())
print("Promedio de diferencia con KNN", dfFinalPruebas['Diferencia_KNN'].mean())
print("Promedio de diferencia con Arboles de desiciones", dfFinalPruebas['Diferencia_Decision_Trees'].mean())
print("Promedio de diferencia con Random Forest", dfFinalPruebas['Diferencia_Random_Forest'].mean())
print("Promedio de diferencia con Red Neuronal", dfFinalPruebas['Diferencia_Red_Neuronal'].mean())
```

Fuente: Elaboración propia.

Estas líneas de código tienen como objetivo exportar el dataframe dfFinalPruebas a un archivo CSV en la ubicación especificada por la variable path_almacenar. Esto permitirá guardar los resultados de los análisis realizados y compartirlos con otros usuarios o utilizarlos en futuros análisis.

La función to_csv() de Pandas convierte el dataframe en un archivo CSV y lo almacena en la ubicación especificada por path_almacenar (ver Figura 36)

Figura 36 Exportación de dataframe final



```
[137] # Exportación de Dataframe final

# Variables
# path_almacenar: Donde va a querer almacenar el Dataframe obtenido
path_almacenar='/content/drive/MyDrive/Codigo_Final/DataPrueba_Machine_Learning.csv'
dfFinalPruebas.to_csv(path_almacenar)
```

Fuente: Elaboración propia.

Resultados obtenidos

En segundo lugar, con relación a los resultados, se obtuvo un rendimiento satisfactorio. Es importante destacar que, aunque se generaron los datos de ejemplo para la variable "Temperatura", se pueden generar variables adicionales utilizando el mismo proceso.

Después de diseñar, programar y probar los modelos de aprendizaje automático, se lograron resultados satisfactorios gracias a la gran cantidad de datos insertados en el modelo. Estos datos incluyen información desde el año 2015 hasta agosto de 2022. Cabe destacar que los modelos de aprendizaje automático proporcionan mejores resultados cuanto mayor sea la cantidad de datos utilizados para entrenarlos.

Inicialmente, los modelos estaban siendo entrenados con datos horarios de las variables con las que tenían más correlación: humedad y hora. Esto tiene sentido gracias a un informe proporcionado por el IDEAM denominado "Características Climatológicas de Ciudades Principales y Municipios Turísticos" desde el año 2018 hasta julio del 2022.

Durante este lapso, se obtuvieron resultados satisfactorios, pero no como se esperaba, ya que había una diferencia promedio de 1.06°C centígrados. Por esta razón, se decidió procesar una mayor cantidad de datos, tal como se mencionó anteriormente, y gracias a esta decisión se lograron mejores resultados en los datos generados de temperaturas para el mes de septiembre de 2022 (el mes en el que se realizaron las pruebas). El modelo que dio los mejores resultados tuvo un promedio de 0.962, lo cual posiblemente se debió a que se le pasaron las variables que tenían una mejor correlación con la variable a predecir.

Se llevaron a cabo pruebas utilizando tanto datos en tiempo real como datos previamente procesados para entrenar modelos de Machine Learning. Se presentarán a continuación los resultados obtenidos mediante la comparación de estos datos y el análisis en tiempo real.

Para iniciar, se generaron promedios diarios correspondientes al mes de septiembre por cada modelo utilizando la columna de temperatura real. A partir de la Tabla 3, el promedio de datos diarios del mes de septiembre) presenta los resultados de diferentes técnicas de modelado aplicadas a la generación de temperaturas en un período de 30 días. Las técnicas de modelado incluyen Regresión, Regresión Polinomial, K-Nearest Neighbors (KNN), Árboles de decisiones, Bosque Aleatorio y la Red Neuronal.

Cada fila representa un día en particular y las columnas corresponden a las diferentes técnicas de modelado y la temperatura real registrada para ese día. Los valores en las celdas de cada columna son los datos realizados por la técnica correspondiente para la temperatura real en ese día.

En general, se puede observar que la regresión y la Red Neuronal logran generar temperaturas con mayor similitud y precisión a la temperatura real.

Tabla 3 *Promedio de datos diarios del mes de septiembre*

Dia	Regresión lineal	Regresión Polinomial	KNN	Arboles de decisiones	Bosque aleatorio	Red Neuronal	Temperatura real
1	18,479	14,663	14,817	14,620	14,155	14,219	12,448
2	14,544	15,734	15,728	15,652	15,144	15,154	14,406
3	15,456	14,560	14,475	14,377	15,212	15,247	13,725
4	15,169	15,266	15,344	15,235	14,850	14,776	12,654
5	14,848	14,815	14,743	14,643	15,530	15,493	13,441
6	16,358	14,681	14,589	14,481	14,542	14,656	14,221
7	16,241	14,782	14,751	14,613	14,771	14,735	14,904
8	15,355	14,653	14,778	14,611	15,044	14,974	14,772
9	15,604	14,541	14,761	14,760	15,383	15,441	13,271
10	17,137	16,810	17,384	17,394	16,599	16,592	12,060
11	16,657	15,056	15,075	15,015	14,006	13,963	14,567
12	15,890	14,937	14,615	14,399	14,895	14,727	9,119
13	13,682	14,718	14,852	14,689	14,946	14,999	14,621
14	16,794	14,623	14,634	14,641	15,001	14,890	14,981
15	14,264	15,217	15,225	15,291	14,616	14,568	15,580
16	17,972	14,078	14,174	14,245	16,104	16,094	11,923
17	16,051	15,241	15,459	15,407	15,612	15,654	14,363
18	15,492	15,157	15,228	15,168	14,467	14,664	13,238
19	14,592	15,321	15,325	15,161	14,469	14,439	14,150
20	13,931	15,069	14,882	14,905	14,921	14,982	10,878
21	15,450	14,851	14,875	14,903	14,413	14,287	13,812

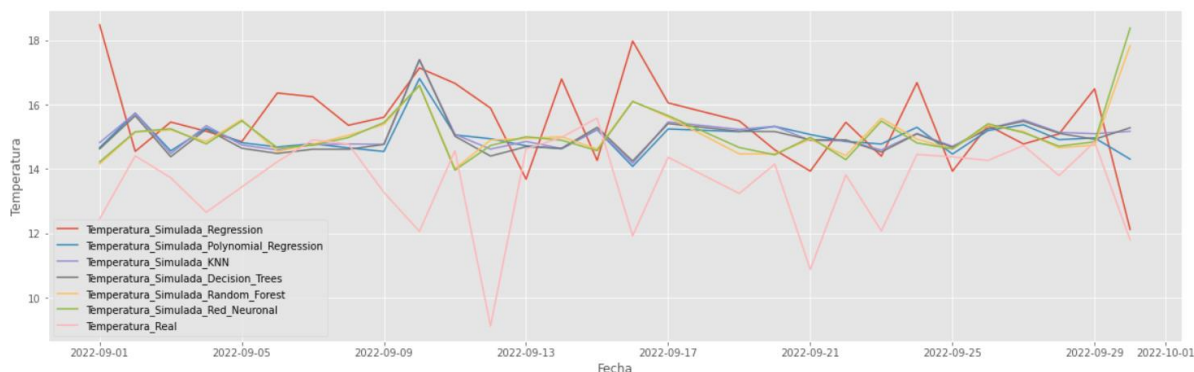
22	14,395	14,775	14,589	14,525	15,576	15,486	12,074
23	16,680	15,294	15,100	15,090	14,925	14,808	14,449
24	13,929	14,461	14,703	14,668	14,619	14,622	14,375
25	15,350	15,193	15,255	15,248	15,354	15,403	14,264
26	14,775	15,372	15,525	15,480	15,163	15,127	14,728
27	15,088	14,908	15,135	15,107	14,653	14,705	13,794
28	16,488	14,964	15,094	14,920	14,740	14,842	14,837
29	12,119	14,304	15,168	15,278	17,832	18,375	11,800
30	18,479	14,663	14,817	14,620	14,155	14,219	12,448

Fuente: Elaboración propia.

De igual manera, se representó la información de la Tabla 3, donde se demuestran representadas las temperaturas obtenidas.

La grafica de la Figura 36, representa los resultados de cada día de los 30 días del mes de septiembre, de esta manera, se puede observar que los modelos de presentan resultados variados en la generación de la temperatura real. Algunos modelos se desempeñan mejor en ciertos días que en otros, y algunos incluso presentan valores extremos, por ejemplo, la temperatura obtenida por medio de Regresión lineal el 16 de septiembre, estos resultados pueden ser considerados como anomalías.

Figura 37 Representación gráfica de las temperaturas obtenidas por modelo



Fuente: Elaboración propia.

Es necesario realizar un análisis más detallado de los resultados para determinar qué modelo es el más adecuado para la generación de registros para la variable deseada. Se deben considerar aspectos como la precisión de la predicción, la velocidad de procesamiento y el costo del modelo.

Se realizaron cálculos de diferencia entre los datos horarios obtenidos y los datos del conjunto de datos para cada uno de los modelos de Machine Learning. Se calcularon los promedios de estas diferencias, obteniendo como resultado un valor de 1.535 para el modelo de regresión, 1.010937637470246 para el modelo de Regresión Polinomial, 1.057 para el modelo KNN, 0.990 para árboles de decisiones, 0.966 para bosque aleatorio y 0.962 para el modelo de Red Neuronal.

Estos resultados sugieren que el modelo de Red Neuronal tiene una menor discrepancia con el conjunto de datos original en comparación con los demás modelos. Sin embargo, es importante tener en cuenta que una discrepancia mayor no necesariamente indica una menor precisión del modelo en la predicción de la temperatura real.

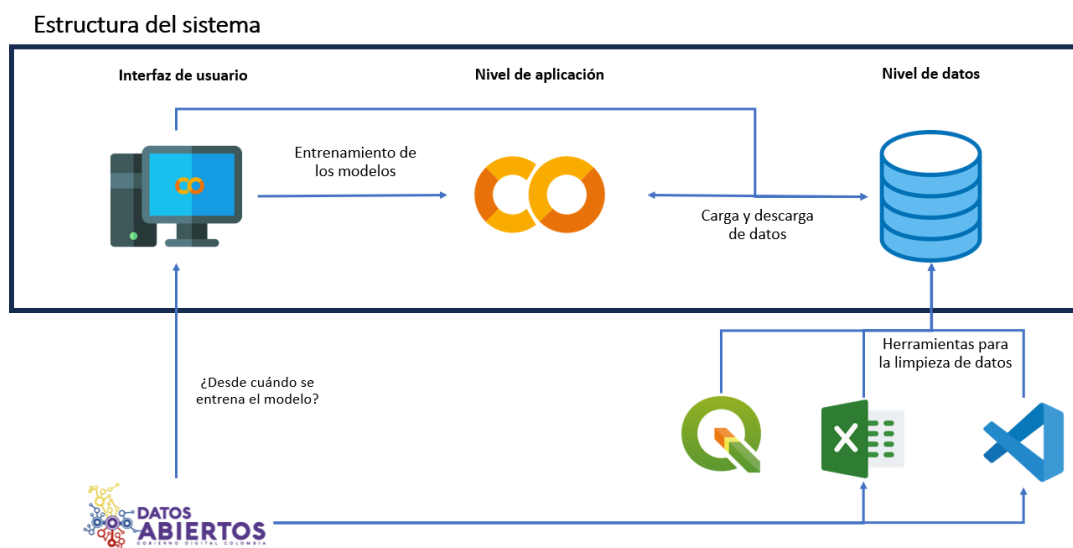
Es necesario realizar una evaluación más detallada de los resultados y considerar otros factores, como la precisión de la predicción, el tiempo de procesamiento y la eficiencia del modelo en diferentes condiciones climáticas.

De igual manera, se realizaron pruebas en tiempo real para saber cómo se acercaban los resultados a la fecha actual en la que se estaban realizando las pruebas. Gracias a esto se obtuvieron resultados muy gratificantes (ver Anexo 3).

Por último, pensando en el mejor entendimiento del proceso y explicación para entrenar los modelos decidimos realizar un video; este se puede encontrar en los anexos (ver Anexos 7 y 8) donde se encontrará el link del video subido a la plataforma YouTube y el repositorio con todos los códigos programados para el óptimo funcionamiento del sistema

Y, finalmente, la estructura del sistema (ver Figura 38) se pensó de tal manera que fuera fácil para el usuario llevar a cabo y/o iniciar desde cero los entrenamientos de los modelos.

Figura 38 Estructura del sistema



Fuente: Elaboración propia

Como se puede observar, el usuario interactuará con Google Colab teniendo en cuenta que los datos están listos para el entrenamiento. Sin embargo, en caso de que no sea así, el usuario podrá realizar consultas a Datos Abiertos y llevar a cabo la limpieza de los mismos utilizando las herramientas usadas e indicadas en la anterior imagen (Figura 38).

Conclusiones

Se ha llevado a cabo con éxito el desarrollo de un modelo de simulación de datos IoT mediante un lenguaje de programación para la validación de proyectos en la región del centro de Bogotá D.C, Colombia. Los objetivos específicos planteados en este proyecto se lograron satisfactoriamente, incluyendo el diseño de algoritmos de simulación para los diferentes sensores, el desarrollo de modelos de simulación, la generación de datos e información mediante los algoritmos desarrollados y la realización de pruebas funcionales para comprobar la validez de los datos simulados.

El modelo de Machine Learning desarrollado para predecir variables climáticas y utilizarlos en el desarrollo de proyectos IoT ha demostrado ser una herramienta eficiente y precisa para analizar y pronosticar el comportamiento del clima. Los resultados obtenidos a través de la evaluación de los diferentes modelos han sido evaluados con diferentes métricas de error y han arrojado resultados cercanos a los datos reales. En general, el modelo de Red Neuronal ha mostrado la mayor precisión en las predicciones, seguido por los modelos de árboles aleatorios y árboles de decisiones, pero la elección del modelo dependerá de los datos específicos y las necesidades del usuario.

En cuanto a los resultados obtenidos diariamente por cada modelo, se evidencia que el modelo de regresión presenta una mayor diferencia promedio en comparación con los otros modelos evaluados. Por otro lado, los modelos de la Red Neuronal, bosque aleatorio, árboles de decisión presentan una menor diferencia promedio, lo que sugiere que son modelos más precisos y confiables en la simulación de datos IoT.

Este proyecto puede contribuir a la facilitación de pruebas de proyectos IoT en la región de Colombia, lo que podría tener un impacto positivo en la implementación de proyectos de este tipo de proyectos en el país. Además, la metodología y los resultados obtenidos en este proyecto pueden ser utilizados como base para futuras investigaciones

relacionadas con la simulación de datos IoT en diferentes contextos y para diferentes aplicaciones.

En conclusión, el modelo de Machine Learning desarrollado en este proyecto tiene un gran potencial para mejorar la eficiencia y precisión en proyectos IoT y podría ser una herramienta valiosa en la planificación del desarrollo sostenible y la gestión del clima en general. Los resultados obtenidos indican que los modelos desarrollados en este proyecto pueden ser utilizados para proporcionar pronósticos precisos del clima y realizar un entorno simulado de datos climáticos, lo que puede ser especialmente importante en el contexto de proyectos IoT, generando un impacto positivo en la implementación de proyectos IoT en el país, donde la precisión en la toma de decisiones es fundamental.

Recomendaciones

Para entrenar modelos de Machine Learning de manera efectiva, es fundamental tener en cuenta algunas recomendaciones clave.

En primer lugar, es importante tener en cuenta que los modelos de Machine Learning funcionan mejor cuando se trabaja con una gran cantidad de datos de alta calidad. Por lo tanto, es fundamental obtener una gran cantidad de datos para entrenar el modelo y hacer que sea más preciso.

Sin embargo, la cantidad de datos por sí sola no es suficiente. También es necesario prestar atención a la calidad de los datos y asegurarse de que estén limpios y procesados de manera adecuada. Esto es especialmente importante en el caso de la obtención de datos climáticos, donde los sensores pueden fallar y dar valores atípicos. Por lo tanto, una buena calidad, procesamiento y elección de datos es la segunda recomendación clave para entrenar modelos de Machine Learning de manera efectiva.

A pesar de la importancia de los dos pasos anteriores, es importante tener en cuenta que no serían posibles en gran medida si la máquina en la que se trabajará para procesar los datos y entrenar el modelo es limitada en recursos. Por esta razón, se recomienda hacer todo el proceso en una máquina virtual proporcionada por una plataforma de servicios de computación en la nube como GCP, Azure, AWS, entre otros. Esto permitirá que el proceso sea más rápido y eficiente, lo que a su vez aumentará la precisión del modelo.

Referencias

- Semle, A. (Septiembre de 2016). Protocolos IIoT para considerar. *AADECA Revista*, 32-35. Recuperado el 25 de Marzo de 2020, de https://editores-srl.com.ar/sites/default/files/aa2_semle_protocolos_ilot.pdf
- Ahrenholz, J., Danilov, C., Herderson, T., & Kim, J. (2008). A real-time network emulator. In Military Communications Conference. *MILCON*, 1-7.
- Alandí Pajares, A. (2015). *Estudio de la implantación de Internet de las Cosas, en las redes Logísticas de la Cadena de suministros*. valencia.
- Amazon Web Services. (2023). *¿Qué es IoT?* Obtenido de ¿Qué es el Internet de las cosas (IoT)?: <https://aws.amazon.com/es/what-is/iot/#:~:text=El%20término%20IoT%2C%20o%20Internet,como%20entre%20los%20propios%20dispositivos>
- Amine Khelif, M., Lorandel, J., Romain, O., Regnery, M., & Baheux, D. (2019). A Versatile Emulator of MitM for the identification of vulnerabilities of IoT devices, a case of study: smartphones. En A. f. Machinery (Ed.), *3rd International Conference on Future Networks and Distributed Systems*, (págs. 1-6). Paris. Recuperado el 2 de Abril de 2020, de <https://dl.acm.org/doi/pdf/10.1145/3341325.3342019>
- AprendiendoArduino. (17 de Noviembre de 2018). *Protocolos IoT Capa Aplicación*. Recuperado el 14 de Abril de 2020, de Aprendiendo Arduino: <https://aprendiendoarduino.wordpress.com/2018/11/17/protocolos-iot-capa-aplicacion/>
- AWS. (s.f.). *IoT Device Simulator*. Obtenido de IoT Device Simulator: <https://aws.amazon.com/es/solutions/implementations/iot-device-simulator/>
- Barbara IoT. (29 de Abril de 2021). *barbara*. Obtenido de Protocolos de comunicación en IoT que deberías conocer: <https://barbaraiot.com/blog/protocolos-iot-que-deberias-conocer/>
- CambioDigital*. (12 de diciembre de 2018). Recuperado el 13 de marzo de 2020, de IoT: Qué necesitan saber los profesionales de la red: <https://cambiodigital-ol.com/2018/12/iot-que-necesitan-saber-los-profesionales-de-la-red/>

- Carlemany, U. (19 de 7 de 2022). *universitatcarlemany*. Obtenido de <https://www.universitatcarlemany.com/actualidad/cual-es-la-diferencia-entre-inteligencia-artificial-y-machine-learning#:~:text=La%20IA%2C%20se%20puede%20decir,all%C3%A1%20de%20a%20inteligencia%20humana>.
- Carlos Gamero Burón, J. L. (2015). *Modelos probabilísticos para Variables aleatorias continuas*. Malaga, España.
- Castellanos Hernández, W. E., & Chacon Osorio, M. E. (17 de Abril de 2006). Utilización de herramientas software para el modelado y la simulación de redes de comunicaciones. *GTI*, V(11), 74-75. Recuperado el 26 de Marzo de 2020, de <https://revistas.uis.edu.co/index.php/revistagti/article/view/1624/2014>
- Corso, C., & Lorena, C. (2009). *Aplicacion de algoritmos de clasificacion sepervisan y no supervisada usando Weka*. cordoba: Universidad Tecnologi Nacional.
- Coss Bu, R. (2003). *Simulación un enfoque practico*. Monterrey, Mexico: Limusa. Recuperado el 25 de Marzo de 2020, de <https://books.google.es/books?hl=es&lr=&id=iY6dI3E0FNUC&oi=fnd&pg=PA11&dq=simulacion&ots=uKV85h0Scu&sig=fMdlmFTXdSYn3HghHIZ7HbFXQhg#v=onepage&q=simulacion&f=false>
- Crespo Moreno, J. E. (11 de Noviembre de 2018). *Aprendiendo Arduino*. Recuperado el 2 de Abril de 2020, de Arquitecturas IoT: <https://aprendiendoarduino.wordpress.com/2018/11/11/arquitecturas-iot/>
- cuatro tipos de analítica de retail que todo comercio necesita en 2018. (7 de Agosto de 2018). *Analitica de retail*. Obtenido de Analitica de retail: <http://analiticaderetail.com/tipos-de-analitica-de-retail/>
- Deloitte. (s.f.). *IoT - Internet Of Things*. Obtenido de Deloitte: <https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html>
- Diaz, T., Escriba, F., & Murgui, M. (2002). La base de datos BD. *MORES. Revista de Economia Aplicada*, 165-184.

- Dra. Marianella Álvarez Vega, D. L. (8 de 8 de 2020). *Revista Medica Sinergia*. Obtenido de <https://www.revistamedicasinergia.com/index.php/rms/article/view/557/923>
- Dunkles, A., Schmidt, O., Finne, N., Erikson, J., Osterlind, F., Tsiftes, N., & Durvy, M. (2011). *The contiki os: The operating system for the internet of things*. Obtenido de Online: <http://www.contikios.org>
- editorial, E. (1 de Noviembre de 2018). *REPORTEGIGITAL*. Obtenido de REPORTEGIGITAL: <https://reportedigital.com/cloud/analitica-de-datos/>
- Eduardo García Dunna, H. G. (2013). *Simulación y análisis de sistemas con ProModel*. Naucalpan de Juárez, Estado de México, México : PEARSON. Obtenido de <https://jrvargas.files.wordpress.com/2015/04/libro-simulacion-y-analisis-de-sistemas-2da-edicion.pdf>
- EOI. (2007). *TÉCNICAS ANALÍTICAS*. (Ley de Propiedad Intelectual del 17 de noviembre de 1987 y Reales Decretos).
- Gan, S. (2017). *An IoT simulator in NS3 and a key-based authentication architecture for IoT devices using blockchain*. Tesis, Instituto Indio de Tecnología Kanpur, Ciencias informáticas e ingeniería, Kanpur. Recuperado el 2 de Abril de 2020, de https://security.cse.iitk.ac.in/sites/default/files/12807624_0.pdf
- García Sánchez, Á., & Ortega Mier, M. (2006). *Introducción a la simulación de sistemas discretos*. Recuperado el 25 de Marzo de 2020, de http://www.iol.etsii.upm.es/arch/intro_simulacion.pdf
- Garcia, P. (2006). *TECNICAS DE ANALITICA*.
- Gibbs, G. (2012). *Análisis de datos en investigaciones cualitativas*. Ediciones Morata.
- Gomez-Aguilar, D., Garcia-Peñalvo, F., & Theron, R. (2014). Analítica visual en learning. *El profesional de la informática*, 23(3).
- González, F. A. (22 de 2 de 2015). Modelos de aprendizaje computacional en. *Revista Colombiana de Reumatología*, págs. 77-78.
- Guatibonza Solano, D. M., & Salazar Marin, K. V. (2022). *Gestión y análisis de datos recopilados de dispositivos IoT en cadenas de suministro aplicando Blockchain y Machine Learning*. Bogota.

- Hardwarelibre*. (s.f.). Obtenido de Hardwarelibre: <https://www.hwlibre.com/iotify-servicio-web-desarrolladores-hardware-libre/#:~:text=As%C3%AD%20hace%20poco%20hemos%20conocido,compatible%20con%20cualquier%20hardware%20libre>.
- Hassan, F., Domingo-Ferrer, J., & Soria-comas, J. (2018). Anominacion de datos no estrucutrados a traves del reconocimiento de entidades nominadas. *Actas de la XV Reunni Espaola sobre Criptologa y Seguridad de la informcin-RECSI*, 102-106.
- Hergert, M., & Morris, D. (1989). Datos contables para el analisis de la cadena de valor . *Diario de gestion estrategica*, 10(29,175-188.
- Hernandez Perez, A. (2013). *Datos abiertos y repositorios de datos* . nuevo reto para los bibliotecarios.
- Hernandez, C., & Rodriguez, J. (2008). Preprocesamiento de datos estructurados. *Vinculos*, 27-48.
- Huang, Y., Wang, L., Hou, Y., Zhang, W., & Zhang, Y. (2018). *A prototype IOT based wireless sensor network for traffic information monitoring*. *International Journal of Pavement Research & Technology*.
- ibertech*. (45 de 06 de 2006). Obtenido de ibertech: <https://www.ibertech.org/analitica-descriptiva-predictiva-y-prescriptiva/>
- ibertech*. (18 de 06 de 2006). Obtenido de ibertech: <https://www.ibertech.org/analitica-descriptiva-predictiva-y-prescriptiva/>
- IBM* . (s.f.). Obtenido de IBM : <https://www.ibm.com/es-es/cloud?>
- Isaac Lera, C. G. (2019). *YAFS*. Palma. Obtenido de <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8758823>
- itop*. (20 de agosto de 2018). Recuperado el 13 de marzo de 2020, de IoT: origen, importancia en el presente y perspectiva de futuro: <https://www.itop.es/blog/item/iot-origen-importancia-en-el-presente-y-perspectiva-de-futuro.html>
- Joyanes Aguilar, L. (29 de mayo del 2019). *Inteligencia de negocios y anlitica de datos*. Bogota: Alfaomega.
- Leónardo Darío Bello Parias, L. C. (2000). *Libro de estadística descriptiva*. Medellin, Antioquia, Colombia: Editorial Amistad ISBN. Obtenido de

http://aprendeenlinea.udea.edu.co/lms/moodle/pluginfile.php/128508/mod_resource/content/0/Tema_4/Modelos_probabilisticos_Caucasia.pdf

Levis, P., Lee, N., Welsh, M., & Culler, D. (2003). *TOSSIM: Accurate and scalable simulation of entire Tinyos applications*. In Proceedings of the 1st international conference on Embedded networked sensor systems.ACM.

Loukides, M. (2011). *¿Que es la ciencia de datos?* O'Reilly Media, Inc.

Mäkinen, A. (2016). *Emulation of IoT Devices*. Espoo. Obtenido de https://aaltodoc.aalto.fi/bitstream/handle/123456789/23951/master_M%c3%a4kinen_Alli_2016.pdf?sequence=1&isAllowed=y

Martin Iglesia, M. (Febrero de 2019). *Archivo Digital UPM*. Obtenido de Análisis de la simulación de dispositivos, circuitos y sistemas electrónicos para Internet de las cosas (IoT): <https://oa.upm.es/54136/>

MathWorks. (s.f.). Obtenido de MathWorks: <https://www.mathworks.com/solutions/internet-of-things.html>

Mehmood, T. (s.f.). *COOJA Network Simulator*. Islamabad. Obtenido de <https://arxiv.org/ftp/arxiv/papers/1712/1712.08303.pdf>

Micrsoft. (s.f.). *Micrsoft*. Obtenido de <https://azure.microsoft.com/es-es/overview/machine-learning-algorithms/#overview>

Muñoz, G. (14 de septiembre de 2019). *¿Dónde está Avinash cuando se le necesita?* Obtenido de *¿Dónde está Avinash cuando se le necesita?:* <http://dondeestaavinashcuandoselenecesita.com/herramientas/>

Myriam Muñoz de ózak, S. F. (1991). PROCESOS ESTOCÁSTICOS CON DOS PARÁMETROS I. 72-74. Obtenido de <https://revistas.unal.edu.co/index.php/estad/article/viewFile/9955/10486>

NEO.LCC. (s.f.). *Protocolos de transporte*. Recuperado el 14 de Abril de 2020, de Herramientas WEB para la enseñanza de protocolos de comunicación: <http://neo.lcc.uma.es/evirtual/cdd/tutorial/transporte/protrans.html>

Nigro, O., Corti, D., & Terren, D. (2004). *Knowledge Discovery in Databases*. Un proceso centrado ene el usuario. In VI Woorkshop de investigadores en Ciencias de lamComputacion.

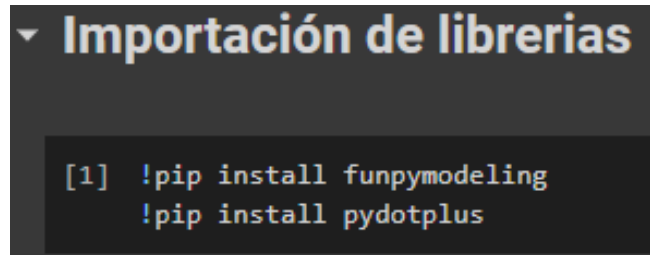
- NSNAM. (2011). *NS-3 Network Simulator*. Recuperado el 2 de Abril de 2020, de NS-3: <https://www.nsnam.org/>
- Oracle. (2022). *www.oracle.com*. Obtenido de <https://www.oracle.com/co/artificial-intelligence/what-is-ai/>
- Peña, S. (2017). *Análisis de Datos*. Bogota D.C: Areandino.
- Prado, J. (s.f.). *VALTX*. Obtenido de VALTX: <https://www.valtx.pe/blog/que-es-la-analitica-de-datos-y-como-puede-impactar-positivamente-en-tu-negocio>
- Quintero, J. (2006). La cadena de valor : Una herramienta de pensamiento estrategico. *Telos*, 14.
- Quiñones Cuenca, M., González Jaramillo, V., Torres, R., & Miguel , J. (2017). *Sistema de Monitoreo de variables medioambientales usando una red de sensores inalámbricos y plataformas de Internet de las Cosas*. Universidad Técnica Particular de Loja, Departamento de Ingeniería, Loja. Recuperado el 14 de Abril de 2020, de http://scielo.senescyt.gob.ec/scielo.php?script=sci_arttext&pid=S1390-65422017000100329
- Raposo, J. (2007). *Técnicas de mantenimiento automatico de programas envoltorio para fuentes de datos web semiestructuradas*. Coruña: Doctoral dissertation.
- Riverder Technologies. (2017). *opnet simulator*. Obtenido de <https://www.riverbed.com/in/products/steelcentral/opnet.html>
- Robles Solano, D. (Marzo de 2021). *Repositorio Digital*. Obtenido de Control y simulación de una planta piloto de laboratorio docente con integración de plataformas IoT para subida de datos a la nube: <https://repositorio.upct.es/handle/10317/9259>
- Rodríguez Moreno, E. S., & López Ordoñez, V. F. (2017). *Diseño e implementación de un sistema inteligente para un edificio*. Tesis de grado, Universidad Francisco Jose de Caldas, Facultad de Ingeniería, Bogotá. Recuperado el 25 de Marzo de 2020
- Roldán Carrasco, Á. (2007). *Emulador de Gameboy para dispositivos móviles*. Tesis, Escuela Superior de Ingeniería Informática, Departamento de Informática, Ciudad Real. Recuperado el 25 de Marzo de 2020, de <https://www.esi.uclm.es/www/cglez/downloads/pfc/pfcaroldan.pdf>

- Ruz Nieto, A. (2021). *Repositorio Digital*. Obtenido de Simulación realista de comunicaciones IoT en entornos urbanos: <https://repositorio.upct.es/handle/10317/9646>
- Sánchez Martín, A. A., Barreto Santamaría, L. E., Ochoa Ortiz, J. J., & Villanueva Navarro, S. E. (2019). *Emulador para desarrollo de proyectos IoT y analíticas*. PREGUNTAR, Universidad Distrital Francisco José de Caldas, Bogotá. Recuperado el 13 de marzo de 2020
- Sandoval, L. J. (16 de 4 de 2018). ALGORITMOS DE APRENDIZAJE AUTOMÁTICO PARA ANÁLISIS Y. *ITCA FEPADE*, págs. 36-40.
- Sandy E. Abasolo, M. A. (2013). *Evaluación del modelo de referencia de “Internet of Things” (IoT), mediante la implantación de arquitecturas basadas en plataformas comerciales, open hardware y conectividad IPv6*.
- Snoke, J., & HalanLarochelle. (2012). Practica y optimizacion de algoritmos de aprendizaje automatico. *Avances en sistemas de procesamiento de informacion neuronal*, 2951-2959.
- Softtek. (2 de 9 de 2021). *softtek.eu*. Obtenido de La Reducción de Dimensionalidad en el Machine Learning: <https://softtek.eu/tech-magazine/artificial-intelligence/la-reduccion-de-dimensionalidad-en-el-machine-learning/>
- Tetcos. (2017). *Netsim emulator*. Obtenido de <http://tetcos.com/>
- Timarán-Pereira. (2016). *The Process of Knowledge Discovery on Databases* . Bogota : Ediciones .
- Torres Bataller, J. (2016). *Desarrollo de una solucion para la simulacion de entornos IoT*.
- Universidad de Alcalá. (2019). *¿Por qué actualmente es tan importante el IoT?* Recuperado el 13 de marzo de 2020, de Máster en industria 4.0: <https://www.masterindustria40.com/importancia-iot-master/>
- Varga, A. (2016). *In Modeling and tools for network simulation*. Berlin,Heidelberg: Springer.
- Xia, F., Yang, L., Wang, L., & Vinel, A. (2012). *Internet of Things*. International journal of communication systems. doi:10.1002/dac.2417

Yacchirema Vargas, D. C., & Palau Salvador, C. E. (s.f.). *Smart IoT Gateway For Heterogeneous Devices Interoperability* (Octava ed., Vol. 14). IEEE Latin America Transactions. doi:10.1109/TLA.2016.7786378

Anexos

Anexo 1. Importación de librerías



```
▼ Importación de librerías

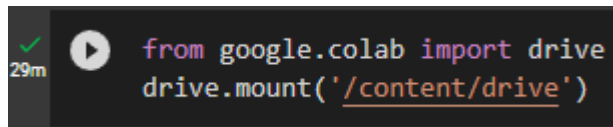
[1] !pip install funpymodeling
    !pip install pydotplus
```

Las anteriores líneas de código utilizan el comando pip para instalar dos paquetes de Python: funpymodeling y pydotplus.

- Pip es un administrador de paquetes para Python que permite instalar, actualizar y desinstalar paquetes de software en un entorno de Python.
- Funpymodeling es un paquete de Python que proporciona una variedad de herramientas para la exploración y visualización de datos, preprocesamiento, modelado y evaluación de modelos de aprendizaje automático.
- Pydotplus es una biblioteca de Python para generar diagramas de grafos utilizando el lenguaje de descripción de gráficos DOT. En el contexto del aprendizaje automático, esta biblioteca se utiliza a menudo para visualizar los árboles de decisión generados por algunos algoritmos de aprendizaje automático.

Las siguientes líneas de código importan el módulo "drive" del paquete "google.colab" y montan el contenido de Google Drive en el entorno de ejecución de Colab. Google Drive es un servicio de almacenamiento en la nube proporcionado por Google, y Colab es un servicio de Google que permite ejecutar código de Python en la nube. La función

"drive.mount()" permite acceder a los archivos de Google Drive desde el entorno de ejecución de Colab.



Luego, las líneas de código mostradas a continuación, importan diversas librerías de Python que se utilizarán en el análisis y visualización de datos. Primero se importan las librerías básicas de manipulación y exploración de datos: pandas, datetime y numpy. Luego se importan las librerías de visualización de datos: seaborn y matplotlib, que permiten crear gráficos y visualizaciones en 2D y 3D. También se importa Axes3D y cm para visualización en 3D, y se establece el tamaño y el estilo de las figuras con plt.rcParams. Finalmente, se importa "corr_pair" de la librería funpymodeling, que es una herramienta para explorar correlaciones entre variables en un conjunto de datos.

```
[9] # =====
# Importación de librerías necesarias para manipulación y exploración de datos
# =====
import pandas as pd
from datetime import datetime
import numpy as np
import pandas as pd
# =====
# Importación de librerías necesarias para la visualización de datos para Python, permite crear gráficos y visualizaciones en 2D y 3D.
# =====
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
# =====
# Importación de librerías necesarias para el análisis de datos, en este caso, para hallar correlaciones.
# =====
from funpymodeling.exploratory import corr_pair
# =====
```

A continuación, las líneas de código importan diferentes librerías de Python necesarias para trabajar con modelos de regresión y validación de datos.

- LinearRegression, PolynomialFeatures, KNeighborsRegressor, DecisionTreeRegressor, RandomForestRegressor, y MLPRegressor son

modelos de regresión de la librería sklearn que se utilizan para ajustar los datos y predecir valores.

- R2_score, mean_squared_error, y train_test_split son funciones de sklearn que se utilizan para evaluar los modelos y dividir los datos en conjuntos de entrenamiento y prueba.
- StandardScaler y MinMaxScaler son herramientas de preprocesamiento de datos que se utilizan para estandarizar y normalizar los datos.
- Pandas_profiling es una herramienta que se utiliza para generar informes de perfilamiento de datos.
- Tensorflow es una biblioteca de aprendizaje automático de código abierto que se utiliza para crear y entrenar modelos de aprendizaje profundo.

```
# =====  
# Importación necesarias para los modelos en general y para regresiones  
# =====  
from sklearn.linear_model import LinearRegression  
from sklearn.metrics import r2_score  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import mean_squared_error  
from sklearn.preprocessing import StandardScaler, MinMaxScaler  
# =====  
from sklearn.preprocessing import PolynomialFeatures  
# =====  
# Importación de librerías necesarias para modelo KNN  
# =====  
from sklearn.neighbors import KNeighborsRegressor  
# =====  
# Importación de librerías necesarias para modelo de Árboles de Decisión  
# =====  
from sklearn.tree import DecisionTreeRegressor  
# =====  
# Importación de librerías necesarias para modelo de Bosques Aleatorios  
# =====  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import accuracy_score  
import pandas_profiling  
# =====  
# Importación de librerías necesarias para modelo de Red Neuronal  
# =====  
from sklearn.neural_network import MLPRegressor  
# =====  
# Importación de librerías necesarias para la validación de datos obtenidos  
# =====  
import tensorflow as tf
```

Anexo 2. Apertura de los datos para los entrenamientos

Apertura del CSV

```
[12] # =====  
# Se define la ruta del archivo CSV que se va a leer y se almacena en la variable 'path_csv'  
path_csv='/content/drive/MyDrive/Codigo_Final/Data_2015-2022.csv'  
  
# Se utiliza la biblioteca pandas para leer el archivo CSV en un DataFrame llamado 'df'.  
# La opción 'encoding' se establece en 'windows-1252' para especificar la codificación del archivo CSV.  
df= pd.read_csv(path_csv, encoding='windows-1252')  
# =====
```

Estas líneas de código se encargan de leer un archivo CSV llamado "Data_2015-2022.csv" desde la ruta especificada en la variable "path_csv". La biblioteca pandas es utilizada para leer el archivo CSV y cargar los datos en un DataFrame llamado "df". La opción "encoding" se establece en "windows-1252" para asegurarse de que la codificación del archivo CSV sea la correcta.

Preparación de la tabla o arreglo del dataset

```
[13] # Se cambia el nombre de las columnas del DataFrame 'df' por los nombres especificados en la lista  
df.columns=['CodigoEstacion', 'CodigoSensor', 'FechaObservacion', 'Dia', 'Mes', 'Año', 'Fecha', 'Hora', 'Humedad',  
            'Humedad_UnidadMedida', 'Precipitacion', 'Precipitacion_UnidadMedida', 'PresionAtmosferica',  
            'PresionAtmosferica_UnidadMedida', 'Temperatura', 'Temperatura_UnidadMedida',  
            'NombreEstacion', 'Departamento', 'Municipio', 'ZonaHidrografica', 'Latitud', 'Longitud']
```

En siguiente lugar el anterior código renombra las columnas del DataFrame df para tener nombres más claros y descriptivos. La primera línea crea una lista con los nombres de las columnas que queremos asignar al DataFrame. La segunda línea utiliza el método columns de Pandas para cambiar los nombres de las columnas del DataFrame a los nombres de la lista. En este caso, las nuevas columnas se corresponden con diferentes variables relacionadas con mediciones de estaciones meteorológicas y geográficas en Colombia.

```
# Se imprime los tipos de datos de cada columna del DataFrame 'df'  
df.dtypes
```

Estas líneas de código imprimen los tipos de datos de cada columna en el DataFrame 'df'. Esto es útil para asegurarse de que los tipos de datos son los correctos y para identificar posibles problemas con los datos que pueden requerir una limpieza o una conversión de tipo de datos.

```
✓ [19] # Se transforma la clase de columna Fecha a Datetime
1s df['Fecha'] = pd.to_datetime(df['Fecha'], format="%m/%d/%Y")

# Se obtiene el numero de dia que corresponde al año
df['Dia'] = pd.to_datetime(df['Fecha']).dt.strftime("%j")
df['Dia'] = df['Dia'].astype('int')
```

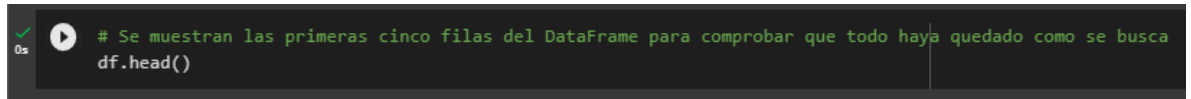
Estas líneas de código transforman la columna 'Fecha' del DataFrame 'df' al formato de fecha/hora de Pandas utilizando la función `pd.to_datetime()`. Después, se usa la función `dt.strftime()` para obtener el número de día del año en formato de cadena y se almacena en la columna 'Dia' del DataFrame 'df'. Finalmente, se cambia el tipo de datos de la columna 'Dia' a 'int' utilizando la función `astype()`.

```
✓ 0s # Se inserta la columna 'Dia' almacenada en la posición 5 (índice 4) del DataFrame 'df'
# Se inserta la columna 'Dia' almacenada en la posición 5 (índice 4) del DataFrame 'df'
columna_Almacenada = df.pop('Dia')
df.insert(5, 'Dia', columna_Almacenada)
```

Estas líneas de código mueven la columna 'Dia' del DataFrame 'df' desde su ubicación actual hasta la posición 5 (índice 4) del DataFrame. Primero, la columna 'Dia' se almacena en la variable 'columna_Almacenada' utilizando el método 'pop()', que elimina y devuelve el elemento en la posición especificada (en este caso, la columna 'Dia' del DataFrame). Luego, la columna 'Dia' se inserta en la posición 5 del DataFrame utilizando el método 'insert()'.

```
✓ 0s # Se ordena el DataFrame por la columna 'FechaObservacion' en orden ascendente
df=df.sort_values(by='FechaObservacion')
```

Estas líneas de código ordenan el DataFrame `df` en orden ascendente según los valores en la columna `'FechaObservacion'`. El método `sort_values()` se utiliza para ordenar los valores en un DataFrame según una o más columnas especificadas. En este caso, solo se ha especificado una columna para ordenar. La función devuelve un nuevo DataFrame ordenado, que se almacena en la misma variable `df`.

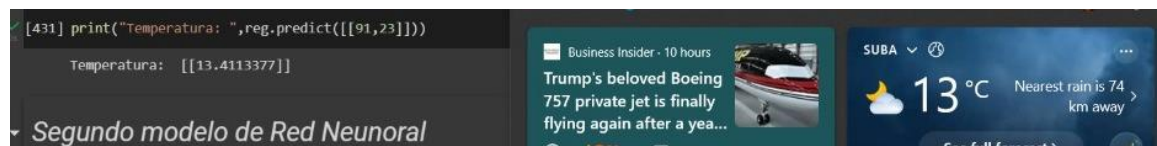
A screenshot of a code editor with a dark background. On the left, there is a green checkmark icon and a play button icon. The code is written in green text. The first line is a comment: `# Se muestran las primeras cinco filas del DataFrame para comprobar que todo haya quedado como se busca`. The second line is the code `df.head()`.

```
0s # Se muestran las primeras cinco filas del DataFrame para comprobar que todo haya quedado como se busca
df.head()
```

Esta línea de código permite mostrar las primeras cinco filas del DataFrame `'df'`.

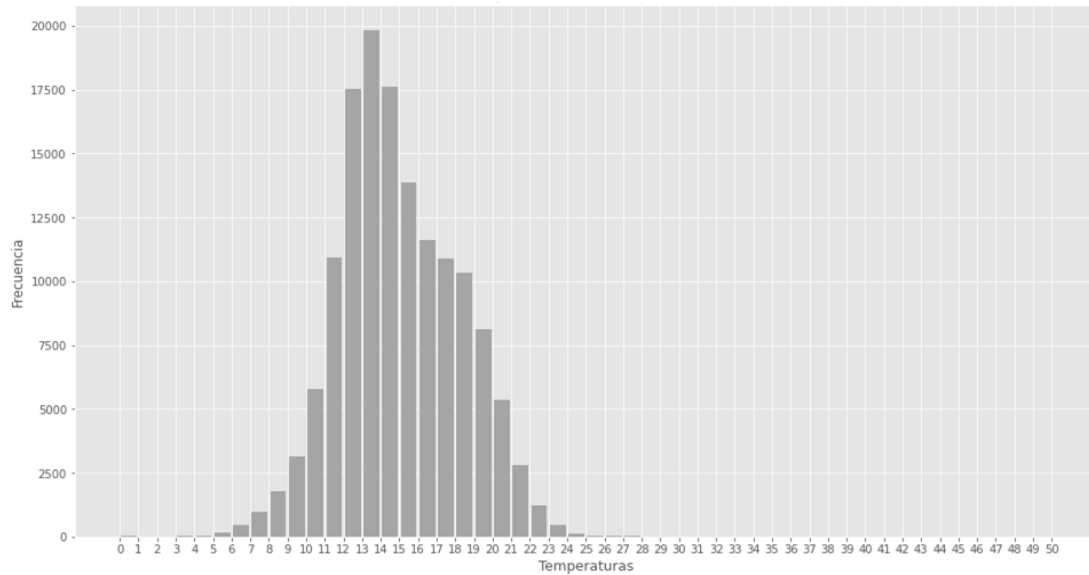
Anexo 3. Resultados en tiempo real

La siguiente imagen fue una prueba realizada el 1 de octubre de 2022 a las 11:00 pm, obteniendo una diferencia de 0.41°C con la temperatura proporcionada por Microsoft. Esto destaca la gran precisión obtenida después de entrenar el modelo. Para obtener resultados más actuales, es necesario entrenar los modelos con una mayor cantidad de datos hasta la fecha actual.



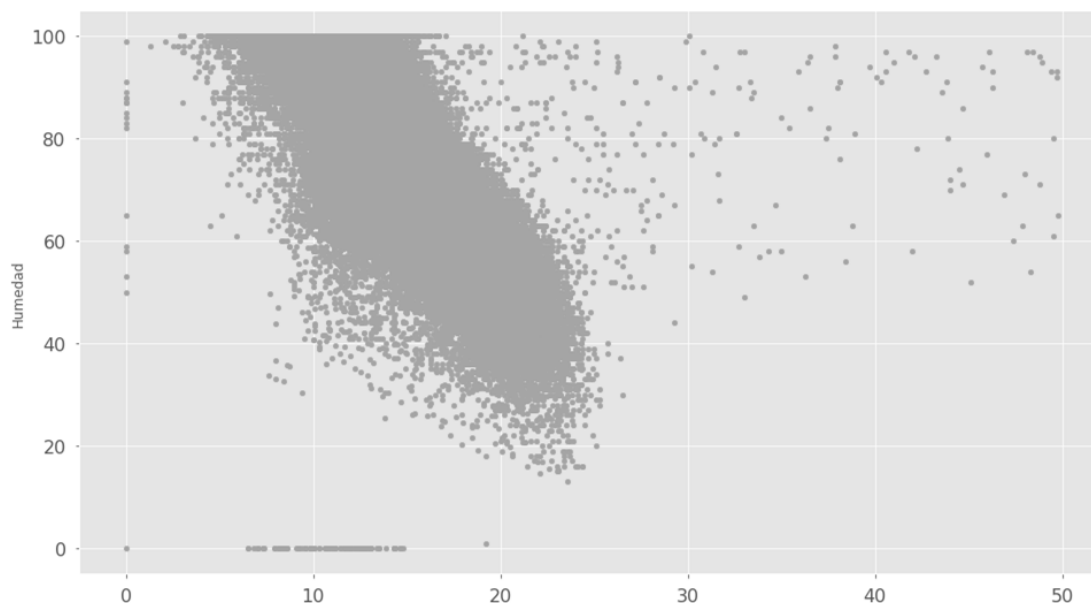
Anexo 4. Histograma de temperaturas

El histograma de temperaturas permite representar el comportamiento en general de esta variable, como se puede evidenciar una campana de Gauss, en lo que permite deducir las temperaturas promedio dentro del dataset, siendo esta, entre 12°C a 15°C.



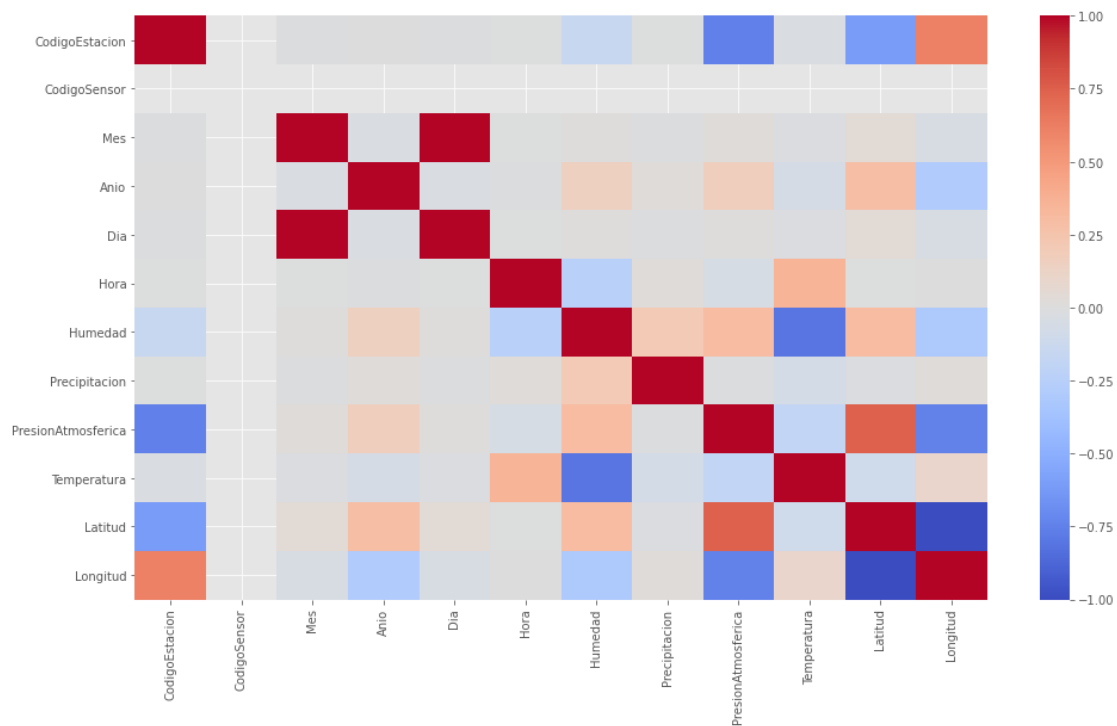
Anexo 5. Diagrama de correlación entre humedad y temperatura

En el diagrama de correlación permite mostrar que la relación entre humedad es inversa con respecto a la temperatura, ya que a menor humedad las temperaturas son mayores.



Anexo 6. Diagrama matriz de correlación

El Diagrama matriz de correlación, se puede evidenciar todas las variables que se están trabajando y la correlación de las variables utilizadas dando a entender que se pueden tener eventos cuando dos variables llegan a uno son altamente correlacionadas (misma dirección) y cuando el valor es menos uno da a entender que es una correlación perfecta pero negativa (direcciones opuestas) como ejemplo la temperatura y humedad.



Anexo 7. Video explicativo del proceso de entrenamiento de los modelos programados

A continuación, encontrará el video explicativo del paso a paso para el entrenamiento de los diferentes modelos programados:



Fuente: Elaboración propia.

Anexo 8. Repositorio del proyecto

En el siguiente Link encontrará el repositorio con todos los códigos del proceso de ETL con su posterior cargue y entrenamiento de los modelos:
<https://github.com/CamiloAndresDG/NIOTE>.