ULPGC
Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería Informática

Modelado predictivo y visualización de la calidad del aire en la ciudad de Nueva York mediante fusión de datos abiertos, sobre plataforma big data.

TÍTULACIÓN: Grado en Ingeniería Informática.

AUTOR: María De Los Ángeles Rodríguez Felipe.

TUTORIZADO POR:

Javier Jesús Sánchez Medina.

Carolina Peña Alonso.

Julio de 2021.

Agradecimientos

Llegar hasta aquí no fue sencillo, durante este camino tuve el apoyo de mi familia y mis amigos. Quiero agradecer a Katherine y a Héctor por leer mi proyecto, a pesar de que no tuviesen la menor idea de informática, y por acompañarme durante estos 4 años de carrera.

A mi madre, mi abuela y mi primita, que soportaron mis malos humores durante toda mi carrera y que aun así me siguen queriendo. A mi padre que, aunque sea en la distancia, siempre ha estado presente y a quien extraño por seguir este sueño loco de mudarme de país.

A todos mis amigos y familiares que no nombraré porque no es suficiente con un folio de agradecimiento, a ustedes que siempre han creído en mí y me han brindado su amor y cariño.

A mis tutores que me guiaron y me ayudaron a lo largo de este trabajo.

Un especial agradecimiento a Juan Manuel que me apoyó, me ayudó y me motivó a seguir adelante cuando menos ganas tenía, esta experiencia no hubiera sido la misma sin tu compañía

Resumen

La monitorización y gestión de la calidad del aire en las ciudades desarrolladas es un elemento crítico de las condiciones de salubridad de estas, y un aspecto en el que cada vez más ciudades deben dedicar recursos y esfuerzos continuados. Existen evidencias de que el tráfico y las condiciones climatológicas tienen un impacto claro en las condiciones del aire.

El propósito de este trabajo es, aprovechando la plataforma de datos abiertos de la ciudad de Nueva York y otras fuentes, desarrollar modelos predictivos de la contaminación del aire. Para ello, será necesario una fase de data ingestion donde se recojan los datos y posteriormente, la generación de un modelo predictivo cuyos resultados serán representados en una sencilla página web.

Abstract

The monitoring and management of air quality in developed cities is a critical element of the health conditions of these, and an aspect in which more and more cities must dedicate resources and continuous efforts. There is evidence that traffic and weather conditions have a clear impact on air conditions.

The purpose of this work is, taking advantage of the open data platform of the city of New York and other sources, develop a predictive model of air pollution. To achieve this, a data ingestion phase in which data is collected will be necessary. Once this phase is completed, a predictive model will be generated whose results will be represented on a simple web page.

Índice general

Índice de figuras

Índice de gráficas

Índice de algoritmos

Capítulo 1
Introducción

Respirar es un proceso fisiológico indispensable para la vida. La real academia española define respirar como el proceso de "Absorber el aire, por pulmones, branquias, tráquea, etc., tomando parte de las sustancias que lo componen, y expelerlo modificado". (Real Academia Española, 2021)

> "Se llama respiración al proceso mediante el cual los seres vivos intercambian gases con el medio externo. Consiste en la entrada de oxígeno al cuerpo de un ser vivo y la salida de dióxido de carbono de este mismo. Es indispensable para la vida de los organismos aeróbicos. Dependiendo del tipo de órgano encargado del proceso, la respiración puede ser pulmonar, como en los mamíferos; traqueal, en los artrópodos; branquial, en los peces; o cutánea, en los anélidos. El intercambio puede producirse con el aire atmosférico, como ocurre en las aves y mamíferos, o tener lugar en el medio acuático que también contiene oxígeno y dióxido de carbono disuelto." (Patiño Restrepo, Celis Rodríguez, & Díaz Cortés, 2015)

La forma de la respiración humana es la respiración pulmonar, que implica inhalar aire rico en oxígeno del entorno externo a los pulmones y exhalar aire rico en dióxido de carbono de los pulmones al mundo exterior.

Pero ¿qué sucede cuando un acto tan elemental y básico como el de respirar, se convierte en una acción peligrosa para la salud de los seres humanos?

Según estudios realizados por la Organización Mundial de la Salud (OMS), la calidad del aire tiene efectos en la salud de los seres humanos. Dependiendo de los niveles de concentración de los distintos contaminantes del aire, la exposición constante a estos puede ocasionar distintos efectos nocivos en la salud. (OMS, 2021)

El aire cambia constantemente. Existen muchos factores que alteran la composición de éste, como son la emisión de CO_2 y otros gases de efecto invernadero derivados del desarrollo urbano, la reducción de biomasa vegetal debido a cambios

en los usos del suelo, el cambio climático, fenómenos derivados de una explosión demográfica mundial que contribuyen a la globalización y aceleración de procesos de producción y consumo, y por tanto a la quema de combustibles fósiles para el transporte, entre otros. Todos estos cambios pueden afectar drásticamente al tiempo y al clima y, por ende, a la salud de los seres humanos y de los ecosistemas. (OMM, 2009)

Según la Organización Mundial de la Salud,

> "Se estima que la contaminación ambiental del aire, tanto en las ciudades como en las zonas rurales, fue causa de 4,2 millones de muertes prematuras en todo el mundo por año; esta mortalidad se debe a la exposición a partículas pequeñas de 2,5 micrones o menos de diámetro (PM2.5), que causan enfermedades cardiovasculares y respiratorias, y cáncer." (OMS, 2021)

Una forma de proteger la salud de la población es monitorear continuamente los índices de concentración de partículas nocivas en el aire.

En función de la calidad del aire, la OMS establece unas directrices sobre éste, en las que se indican los límites máximos recomendados de exposición a contaminantes en el aire a los que un ser humano puede estar expuesto sin sufrir daños en su salud.

Este índice se calcula para 4 contaminantes: dióxido de azufre (SO_2), dióxido de nitrógeno (NO_2), ozono (O_3) y partículas suspendidas (PM).

El Departamento de Salud local de la ciudad de Nueva York señala que existen cantidades importantes de contaminantes en el aire en concentraciones considerables, principalmente en carreteras y áreas de mayor tráfico como es la zona de Manhattan. Dicha investigación señaló que la zona de Manhattan presentó peores condiciones en la calidad del aire que el resto de las zonas de la ciudad. (Moscoso, 2010)

La Organización Meteorológica Mundial (OMM) indica que la calidad del aire depende considerablemente de la ubicación. En sitios con altitudes más bajas, durante el verano, cuando la temperatura aumenta, lleva a que se utilicen más los aires acondicionados, agravando la contaminación. Mientras que, en sitios de latitudes altas, durante el invierno se pueden ocasionar neblinas de humo provocadas por un proceso

denominado inversión del calor, o de la temperatura, que retiene el aire frío cerca del suelo, donde queda estancado, atrapando a los contaminantes. (OMM, 2009)

Además, la OMM (2009) señala que la topografía también influye.

> "Una ciudad situada en un valle experimentará los efectos de la inversión del calor, dado que los contaminantes quedan retenidos en la cuenca, por la falta de vientos suficientemente fuertes que les permitan pasar sobre el terreno montañoso. Sin embargo, en las inmediaciones de un gran lago o de un océano, las brisas constantes pueden hacer que los contaminantes desaparezcan sin demora." (OMM, 2009)

Dependiendo de la zona geográfica, la calidad del aire puede variar considerablemente. Ciudades como Manhattan cuentan con grandes edificios y rascacielos que impiden una buena circulación del aire. Además, con los millones de personas y automóviles que cruzan a diario las calles de Manhattan, es más probable que los niveles de contaminación sean más elevados que en zonas más despejadas y con menos tráfico.

Además, el asfalto y los grandes edificios generan una gran cantidad de calor al absorber la radiación durante el día y liberarlo durante la noche, lo que puede intensificar la producción de ozono al nivel del suelo, que es uno de los principales contaminantes del clima. En los centros de las grandes ciudades se tienden a alcanzar temperaturas entre 0.5 y 6º C más calurosas que las zonas suburbanas adyacentes y las zonas rurales. (OMM, 2009)

La OMM (2009) señala que, a medida que más científicos recopilan datos sobre el aire, se hace más estrecho el vínculo que existe entre la calidad del aire y el sistema tiempo-clima. Factores como el viento, la lluvia, la nieve, la luz solar y la temperatura controlan el transporte y la duración de los agentes contaminantes en todo el mundo. La OMM indica que, cuanto mejor se entienda la relación tiempo-clima, se podrá predecir con mayor acierto la distribución de partículas y gases atmosféricos que pueden ser nocivos y los propios contaminantes que repercuten sobre este sistema.

Es evidente que factores como el clima y el tráfico están estrechamente relacionados con la calidad del aire. Debido a que Manhattan es la parte de Nueva York que se ve más afectada por las altas concentraciones de contaminantes en el aire.

Se plantea, como propósito de este proyecto crear un modelo de predicción de la calidad del aire, basado en el tráfico en el área de estudio.

En primer lugar, se identificarán las fuentes de datos que se utilizarán para proporcionar información al modelo predictivo. Posteriormente, se analizarán las distintas tecnologías aportadas por la Inteligencia Artificial (IA) para encontrar la más adecuada a los objetivos de este proyecto.

Una vez identificadas las fuentes de los datos y las tecnologías a utilizar, se recogerán los datos en una única estructura y se entrenarán diferentes modelos predictivos. A continuación, se observará el rendimiento de estos modelos y se seleccionará el que se considere más apropiado para generar las predicciones. Finalmente, los resultados se mostrarán en una página web simple.

Capítulo 2

Estado actual y objetivos iniciales

2.1 Situación actual del tema relacionado con el TFT

En la actualidad, la monitorización y seguimiento de los niveles de contaminación en el aire es un tema de gran importancia, en especial en las grandes ciudades, ya que éstas son las más afectadas.

La OMS (2021) indica que, dependiendo de los niveles de concentración de los distintos contaminantes del aire, la exposición a éstos puede tener efectos nocivos en la salud de los seres humanos.

Las partículas suspendidas (PM) afectan a más personas que cualquier otro contaminante. Si tienen un diámetro igual o menor de 10 micrones (PM10) pueden atravesar y alojarse profundamente dentro de los pulmones, y si son menores o iguales a 2,5 micrones (PM2.5) pueden atravesar la barrera pulmonar y entrar en el sistema sanguíneo. La exposición crónica a partículas contribuye al riesgo de desarrollar enfermedades cardiovasculares y respiratorias, así como cáncer de pulmón. (OMS, 2021).

Un estudio realizado recientemente por el Instituto de Tecnología de Massachusetts (MIT), en la ciudad de Nueva York, indicó tras 120 días de observación enfocándose en los niveles de concentración del PM2.5 en la ciudad, que los neoyorquinos que vivían en la zona de Manhattan estaban expuestos a niveles de contaminación más elevados que en el resto de los barrios de la ciudad. (Nyhan, y otros, 2016)

En la siguiente figura (véase **¡Error! No se encuentra el origen de la r eferencia.**) se puede observar cómo la concentración de PM2.5 en Manhattan y sus alrededores es mucho más alta que en otras partes de la ciudad.

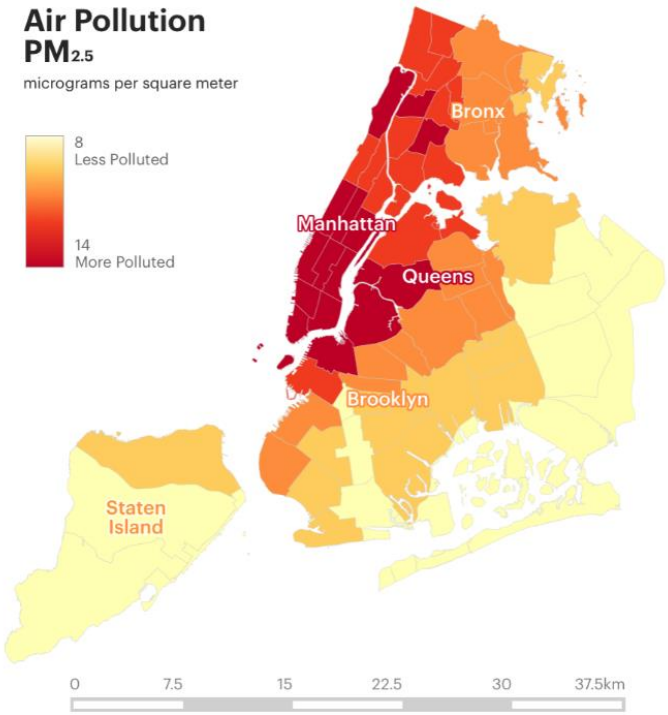


Ilustración 2.1 Concentración de PM2.5 en Nueva York (Nyhan, y otros, 2016)

El exceso de ozono en el aire puede causar problemas respiratorios, provocar asma, reducir la función pulmonar y originar enfermedades pulmonares. (OMS, 2021)

En estudios epidemiológicos se ha revelado que los síntomas de bronquitis en niños asmáticos aumentan en relación con la exposición prolonga al dióxido de nitrógeno (NO_2). (OMS, 2021)

El dióxido de azufre puede afectar al sistema respiratorio y las funciones pulmonares, y causa irritación ocular. (OMS, 2021)

Los seres humanos en todo el mundo están expuestos constantemente a estos componentes quedando expuestos a las consecuencias del impacto de estos componentes en su salud. Dadas las posibles consecuencias en la salud ambiental de los entornos afectados es necesario plantear metodologías para la monitorización de los agentes contaminantes. Las metodologías desarrolladas pueden contribuir al desarrollo de herramientas de gestión ambiental local y regional que faciliten las estrategias de mejora para la salud ambiental.

Este proyecto se centrará en generar un modelo predictivo que ayude a la monitorización y seguimiento de la calidad del aire en la ciudad de Nueva York, en especial en la zona de Manhattan, dados los datos de contaminación registrados en

esta parte de la ciudad. La monitorización y control de la calidad del aire es fundamental para poder tomar las medidas necesarias y plantear medidas de gestión que eviten poner en riesgo a la población local.

La OMS (2021) señala que controlar la contaminación del aire exterior está más allá del control de las personas. En este caso, se requiere que las ciudades tomen medidas al respecto y también que existan instancias normativas nacionales e internacionales en sectores tales como el transporte, gestión de residuos energéticos, construcción y agricultura.

De esta forma, la OMS establece distintas políticas estableciendo, por ejemplo, para el sector de transporte:

> "Adopción de métodos limpios de generación de electricidad; priorización del transporte urbano rápido, las sendas peatonales y de bicicletas en las ciudades, y el transporte interurbano de cargas y pasajeros por ferrocarril; utilización de vehículos pesados de motor diésel más limpios y vehículos y combustibles de bajas emisiones, especialmente combustibles con bajo contenido de azufre." (OMS, 2021)

Además, la ciudad de Nueva York estableció en su plan de sostenibilidad y su hoja de ruta para reducir las emisiones de gases de efecto invernadero, un conjunto de estrategias y medidas para mejorar la calidad del aire que incluyen: (NYC, 2021)

- Transición a vehículos ligeros y pesados más eficientes y menos contaminantes.
- Reducir el uso de vehículos motorizados mediante el cambio a modos de transporte más sostenibles.
- Crear redes de transporte de mercancías más eficientes y ampliar los programas de renovación y sustitución de camiones.
- Reducir la quema de combustibles fósiles en los edificios.

La ciudad de Nueva York ha dispuesto durante los últimos años dispositivos que permiten monitorizar la calidad del aire en la ciudad, proporcionando también a los ciudadanos el acceso a estos datos. Gracias a estas herramientas e información, se podrá crear y desarrollar modelos predictivos que permitan hacer un seguimiento del índice de la calidad del aire en la ciudad.

2.2 Motivación

En un planeta expuesto al constante aumento de la población humana, así como la aceleración de los procesos de consumo, es importante estar preparados ante las consecuencias en la salud humana de tales niveles de desarrollo y alteración del sistema natural.

La respiración es un proceso inerte en los seres vivos. La mayoría de los seres humanos dan por sentado que una acción tan básica y necesaria como respirar, solo implica proporcionar al cuerpo humano el oxígeno que necesita para vivir, pero cuando el aire que se respira está cargado de componentes contaminantes en altas concentraciones, éstos pueden causar efectos verdaderamente nocivos para la salud, como enfermedades cardiovasculares, problemas respiratorios, bronquitis, asma, entre otros.

La OMS indica que, para afrontar la calidad del aire en el mundo, es necesario monitorizar y hacer un seguimiento de sus componentes, para que de esta forma las entidades correspondientes puedan establecer planes de acción que se adapten a la situación de la zona que se esté evaluando y así reducir los niveles de contaminación.

De esta forma, surge la motivación de este proyecto. Resulta interesante plantear como, aprovechando las tecnologías disponibles, se pueden generar a través de la Inteligencia Artificial modelos predictivos que ayuden a monitorizar y hacer seguimiento de los índices de la calidad del aire en una zona predeterminada. Además, no solo se toma en consideración los valores referentes a dichos índices, sino que además se considera la influencia de factores externos como lo son el clima y el tráfico.

2.3 Objetivos

El objetivo general del trabajo será desarrollar modelos predictivos de la contaminación en la ciudad de Nueva York, haciendo uso de las distintas bases de datos y APIs abiertas relacionadas con el trabajo y finalmente representar los resultados en una página web sencilla.

Los objetivos específicos del trabajo serán:

- Realizar un estudio a las diferentes fuentes de información disponibles.
- Identificar las diferentes tecnologías de big data e Inteligencia Artificial que sean convenientes para el estudio a realizar.
- Efectuar la ingestión y fusión de datos dentro de una arquitectura big data.
- Desarrollar el modelo predictivo de la contaminación en la ciudad de Nueva York.
- Crear página web para mostrar los resultados del análisis.
- Evaluar los resultados obtenidos.

Capítulo 3
Competencia específica y aportaciones del trabajo

3.1 Competencia específica

La competencia específica relacionada directamente con el trabajo desarrollado es:

> "Código CP07. Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos." (Escuela de Ingeniería Informática de la ULPGC, 2021)

Durante este proyecto se realizó un estudio sobre las distintas formas de Inteligencia Artificial disponibles para generar modelos predictivos con series de tiempo.

Por otro lado, se trabajó con grandes volúmenes de datos provenientes de diferentes fuentes y se realizó la unión de estos datos en una única estructura que pudiese ser utilizada para entrenar los modelos predictivos propuestos.

Finalmente, se diseñó y se creó una página web, en la que se reflejasen los resultados del trabajo realizado.

3.2 Aportaciones del trabajo

El siguiente trabajo tiene como finalidad aportar al entorno social una herramienta que permita informar a los ciudadanos sobre el estado de la calidad del aire en un futuro próximo o inmediato, específicamente en Manhattan, ya que esta es la zona de la ciudad que más afectada se ve por los agentes contaminantes en el aire. Esto permitirá también la posibilidad de monitorizar y visualizar los índices de contaminación por parte de las entidades correspondientes.

Desde un punto de vista tecnológico, se pretenden generar modelos predictivos para la calidad del aire utilizando distintas herramientas tecnológicas que hagan uso de Inteligencia Artificial (IA).

Debido a que el modelo de predicción para la calidad del aire se genera utilizando datos del tráfico y el clima, la comunidad de científicos que acceda a estos datos podrá ver con claridad qué tan acertado es generar modelos predictivos tomando en cuenta estos factores, para ayudarles a entender la relación que existe entre ellos.

Capítulo 4
Desarrollo

4.1 Metodología aplicada

Inicialmente se estudiaron todas las posibles bases de datos relacionadas con el proyecto y sus APIs[1]. Para esto, se compararon distintas fuentes de información para la obtención de datos referentes al tráfico, la calidad del aire y el clima en la ciudad de Nueva York.

Una vez estudiadas las fuentes de datos disponibles de acceso libre y gratuito, se procedió a realizar la ingestión de los datos históricos utilizando el lenguaje de programación Python en Visual Studio Code. Estos datos fueron almacenados en archivos de tipo CSV[2]. Posteriormente, de la misma forma en la que se hizo la recolección de los datos históricos, se añadió el código necesario para realizar la ingestión de los datos en tiempo real.

A continuación, se procedió a entrenar distintos modelos de predicción con los datos históricos obtenidos, utilizando la plataforma de datos abiertos TensorFlow en conjunto con la biblioteca de redes neuronales Keras de código abierto escrita en Python, que permite desarrollar modelos de predicción complejos. El siguiente paso fue seleccionar el modelo de predicción que proporcionase los mejores resultados, utilizando el error medio absoluto como referencia para hacer dicha selección.

---

[1] **API**: "La interfaz de programación de aplicaciones, conocida también por la sigla **API**, en inglés, application programming interface, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizada por otro software como una capa de abstracción." (Colaboradores de Wikipedia, 2021d)

[2] **CSV**: "Las siglas CSV vienen del inglés "Comma Separated Values" y significan valores separados por comas. Dicho esto, un archivo CSV es cualquier archivo de texto en el cual los caracteres están separados por comas, haciendo una especie de tabla en filas y columnas." (López, 2020)

Finalmente, para la visualización de los datos se desarrolló una página web utilizando el framework Flask escrito en Python, integrando la página web con los datos obtenidos en tiempo real y con el modelo seleccionado.

4.2 Fases de desarrollo del trabajo

El plan de trabajo se desarrolló en 4 fases principales, cada una de ellas divididas en tareas en concreto:

1. Estudio previo y análisis:
   1.1. Estudio de las diferentes fuentes de información disponibles, sus APIs y estructuras de datos.
   1.2. Estudio de las tecnologías de big data más convenientes para la integración de los datos y las inteligencias artificiales más apropiadas para el modelado.
2. Diseño, desarrollo e implementación:
   2.1. Realizar la ingestión y fusión de datos dentro de una arquitectura big data.
   2.2. Desarrollo del modelo predictivo de la contaminación.
   2.3. Desarrollo de la página web en la que se muestran los resultados del modelo.
3. Evaluación / Validación / Prueba:
   3.1. Evaluación de los resultados obtenidos.
4. Documentación / Presentación:
   4.1. Realización de la memoria.
   4.2. Realización de la presentación.

4.3 Estudio previo y análisis

La primera fase del proyecto constó de un estudio previo y un análisis a las distintas fuentes de datos disponibles para recolectar información acerca del tráfico, la calidad del aire y el clima en la ciudad de Nueva York.

Una vez identificados los datos y la estructura de éstos, se procedió a analizar e indagar acerca de las diferentes tecnologías big data que se podrían utilizar para recoger, procesar y analizar los datos.

Finalmente, se estudiaron las distintas opciones que ofrece la inteligencia artificial para generar modelos predictivos que se adaptasen al proyecto.

4.3.1 Fuentes de información

En el momento de seleccionar las APIs, se tomaron en cuenta distintos aspectos, entre los que se encuentra:

- Acceso a los datos de forma libre o gratuito.
- Fuente de los datos fiable.
- Acceso a histórico de datos.

4.3.1.1 Datos de tráfico

La ciudad de Nueva York cuenta con la web "New York City Open Data (NYC OpenData)", que permite acceder de forma libre a datos publicados por agencias de la ciudad.

La web de NYC OpenData, el Centro de Gestión del Tráfico (TMC, por sus siglas en inglés) del Departamento de Transporte de la Ciudad de Nueva York (NYCDOT, por sus siglas en inglés) pone a disposición de los usuarios datos recogidos en tiempo real acerca de la velocidad del tráfico en la ciudad, manteniendo un mapa de los detectores de velocidad del tráfico en toda la ciudad. (The City of New York, 2021).

Los datos son recogidos mediante un circuito de cámaras de televisión cerrado, proporcionados por agencias de transporte y seguridad pública de Nueva York administradas por la Gestión de Tráfico y Sistema de Transporte Inteligente (TRANSCOM, por sus siglas en inglés), permitiendo rastrear las condiciones del tráfico en vivo en lugares claves de la ciudad. (The City of New York, 2021)

Al hacer consultas a la API, se puede indicar la fecha desde la que se desea consultar los datos, permitiendo de esta forma acceder no solo a los datos en tiempo real, sino también a datos más antiguos.

Teniendo en cuenta que la fuente datos es de acceso libre, proviene de una fuente de datos fiable (NYCDOT) y que permite acceder a datos históricos, se seleccionó esta fuente para la recolección de los datos del tráfico.

4.3.1.2 Datos de calidad del aire

Para obtener datos referentes a la calidad del aire en la ciudad de Nueva York se encontraron múltiples API.

- "AirNow API": es un programa de la Agencia de Protección Ambiental de los Estados Unidos, en inglés, U.S. Environmental Protection Agency (EPA). Permite a sus usuarios acceso libre a los datos, recibe predicciones en tiempo real de más de 2000 estaciones de monitoreo y recopila pronósticos para más de 300 ciudades en los Estados Unidos, Canadá y México. Los datos provienen de agencias gubernamentales locales, estatales tribales, provinciales y federales. Permite acceder a datos históricos y en tiempo real. (AirNow, 2021)
- "Air Quality Historical Data Platform": proporciona datos históricos y en tiempo real con acceso ilimitado y gratuito. Recibe datos de más de 100 países. Los datos que ofrecen información acerca de la ciudad de Nueva York provienen de la API de AirNow. (The World Air Quality Project, 2021)
- "BreezoMeter": proporciona datos en tiempo real pronósticos y datos históricos. Ofrece una prueba gratis de 14 días. Obtienen los datos de distintas fuentes: datos de tráfico en tiempo real, información de sensores gubernamentales oficiales, red de sensores locales de bajo costo, datos satelitales, pronóstico del tiempo e información meteorológica, informes sobre incendios activos, cobertura terrestre y más, algoritmos sofisticados, múltiples modelos y técnicas de aprendizaje automático. (BreezoMeter, 2021)

Para recolectar los datos de la calidad del aire se seleccionó como fuente la API de AirNow, la cual es totalmente gratuita y proporciona tanto datos históricos como datos en tiempo real. Además, provienen de fuentes oficiales, lo que hace que los datos sean fiables.

La opción de "Air Quality Historical Data Platform" se descartó debido a que no proporciona datos históricos y utiliza como fuente la API de AirNow, que es la

que se utilizará en este proyecto. La API de "BreezoMeter" se descartó por ser de pago.

4.3.1.3 Datos del clima

Entre las APIs disponibles para obtener datos meteorológicos las evaluadas para este proyecto fueron:

- "National Weather Service": permite a los desarrolladores acceder a pronósticos, alertas y observaciones críticas, junto con otros datos meteorológicos. La API está destinada a ser de datos abiertos y de uso gratuito para cualquier propósito. Es un servicio público ofrecido por el gobierno de los Estados Unidos. (National Weather Service, s.f.)
- "Visual Crossing Weather": se centra en brindar datos meteorológicos de alta calidad y bajo costo, permitiendo hacer un número determinado de peticiones gratis para una misma cuenta al mes o al día. Permite consultar datos históricos, datos en tiempo real y predicciones, los datos son de origen gubernamental asegurando la fiabilidad y la calidad de estos. (Visual Crossing Corporation, 2020)
- "OpenWeather": ofrece pronósticos, alertas, histórico de clima y clima actual. Permite el acceso a los datos de forma libre y paga, los datos históricos no están incluidos en la versión libre. Los datos se actualizan con frecuencia en función de los modelos meteorológicos globales y locales, satélites, radares y una amplia red de estaciones meteorológicas. (OpenWeather, 2021)

Para la obtención de los datos del clima se descartó la opción de OpenWeather, debido a que para acceder a los datos históricos debe pagarse una suscripción. La opción de National Weather Service, a pesar de ser una fuente oficial del gobierno de los Estados Unidos, fue descartada porque no permite acceder a los datos históricos.

Finalmente, se seleccionó la API de Visual Crossing Weather, ya que ofrece datos históricos y en tiempo real. Además, la API obtiene los datos de fuentes gubernamentales, por lo que son fiables y gracias a que ofrece un número de peticiones gratis al mes, se pueden acceder a los datos sin problema.

4.3.2 Tecnologías big data

El término "big data" se utiliza para hacer referencia a grandes volúmenes de datos digitales de naturaleza estructurada y no estructurada que pueden provenir de diferentes fuentes. Esta no es una tecnología específica, sino que está vinculada a otras tecnologías relacionadas con la información digital. (AuraQuantic, 2020)

Aunque el acceso y almacenamiento de grandes cantidades de información para la analítica ha existido durante mucho tiempo, no es hasta la década del 2000 cuando el término de big data cobró impulso, gracias al analista de la industria Doug Laney, quien articuló la definición actual de grandes datos como las tres V: Volumen, Variedad y Velocidad. (SAS Institute Inc., 2021). Posteriormente, este número fue aumentado hasta tener cinco "V".

"La verdadera importancia del big data no gira en torno a la cantidad de datos que se tiene, sino en el uso que se le da." (SAS Institute Inc., 2021).

Los tipos de datos pueden ser:

- Datos estructurados: forman parte de una estructura predefinida. Son fáciles de catalogar, y pueden ser utilizados para análisis y predicciones fiables. Por ejemplo: una hoja de Excel o una base de datos SQL. (AuraQuantic, 2020)
- Datos no estructurados: son aquellos datos que no tienen ni forman parte de una estructura definida. Contienen información valiosa, que no está bien catalogada y estructurada. Su uso resulta complicado a la hora de crear informes y realizar análisis. Por ejemplo: el cuerpo de un email o una base de datos NoSQL. (AuraQuantic, 2020)

Big data cumple con 5 características representativas conocidas como las 5 "V":

Ilustración 4.1 Las 5 "V" del big data (AuraQuantic, 2020)

- Volumen: puede soportar y procesar grandes volúmenes de datos.
- Velocidad: es capaz de procesar grandes volúmenes de datos que se generan con gran rapidez.
- Variedad: los datos proporcionados pueden ser provenientes de distintas fuentes.
- Valor: al tener gran variedad de datos, se deben valorar aquellos que son verdaderamente importantes y relevantes.
- Veracidad: los datos proporcionados deben ser provenientes de fuentes fiables.

Para poder procesar, analizar y almacenar grandes cantidades de datos, las tecnologías big data disponen de una gran cantidad de herramientas de código abierto que permiten la explotación de su software en todos sus procesos. (IIC, 2019)

A continuación, se mostrarán las herramientas de código abierto disponibles que se estudiaron para este proyecto:

1. Hadoop: se considera el framework estándar para el almacenamiento de grandes volúmenes de datos, y se puede utilizar para procesarlos y analizarlos. Dispone de un sistema de archivos distribuido en cada

nodo del clúster y se basa en el proceso de MapReduce[3] de dos fases. (IIC, 2019)

2. Apache Spark: es un motor de procesamiento de datos de código abierto muy rápido. Se puede considerar como la evolución del paradigma MapReduce nativo de Hadoop. Permite ejecutar tareas de procesamiento de datos hasta 100 veces más rápido que su predecesor. Se pueden programar aplicaciones usando lenguajes como Java, Scala, Python o R. (IIC, 2019)
3. Lenguaje R: este es un lenguaje de programación para el cálculo estadístico y la creación gráficos. Es el lenguaje más utilizado por estadistas y profesionales interesados en la minería de datos, la investigación bioinformática y las matemáticas financieras. (IIC, 2019)
4. Python: es un lenguaje de programación avanzado muy fácil de usar. Es una herramienta muy eficiente para big data, ya que cuenta con una gran cantidad de librerías para trabajar con este tipo de datos. (IIC, 2019)
5. Jupyter: es una herramienta que permite ejecutar código por celdas, para comprobar los resultados en cada paso del experimento. Facilita el trabajo exploratorio de análisis de los datos, así como de entrenamiento de modelos de aprendizaje automático. Los notebooks de Jupyter pueden escribirse en diversos lenguajes como R o Julia, y el más utilizado y popular Python. (Barbero, 2020)
6. Keras: es una librería que permite crear modelos de redes neuronales profundas con facilidad usando el lenguaje de programación Python. Es capaz de ejecutarse en el framework TensorFlow y se utiliza en la rama de aprendizaje profundo de la Inteligencia Artificial. (Barbero, 2020)

[3] **MapReduce**: es un framework que proporciona un sistema de procesamiento de datos paralelo y distribuido. (Olmedo, 2021)

Una vez analizadas y vistas las tecnologías de big data más comunes y de código abierto, se seleccionó Python como tecnología para la obtención, análisis y procesamiento de los datos.

A pesar de que existen otras herramientas como Hadoop y Apache Spark que son rápidas y muy populares en cuanto a big data se refiere, Python es un lenguaje de programación que cuenta con múltiples librerías y bibliotecas para big data y una gran comunidad que le da soporte.

Python es uno de los lenguajes de programación más utilizados, debido a que permite crear códigos con gran legibilidad, es fácil de comprender y de implementar. La flexibilidad que ofrece es una gran ventaja que permitirá desarrollar cada una de las fases de este proyecto manteniendo un lenguaje de programación uniforme durante el desarrollo de este.

Para la obtención de los datos desde las APIs, Python ofrece una gran variedad de librerías que permiten obtener información de éstas. Además, cuenta con la librería Pandas, que es útil para guardar los datos en forma de tablas, es rápida e ideal para unir datos provenientes de distintas fuentes y permite manipularlos de forma sencilla.

Por otro lado, Jupyter permite escribir código con el lenguaje de Python, y se utilizará para crear, visualizar y documentar de una forma práctica, visual y rápida los modelos de predicción que se llevarán a cabo.

Es relevante destacar que Python cuenta con frameworks como Tensorflow y librerías como Keras, utilizadas por la Inteligencia Artificial en el aprendizaje automático y el aprendizaje profundo, las cuales se utilizarán para generar los distintos modelos predictivos que se propondrán a lo largo este proyecto.

Además, Python cuenta con el framework Flask para hacer páginas web, que será el utilizado para crear la web en la que se mostrarán los resultados obtenidos. De esta forma, se utilizará Python desde la obtención de los datos hasta la visualización de dichos resultados.

4.3.3 Inteligencias Artificiales

La Inteligencia Artificial se inició antes de la década de los 80. Consistía en establecer un conjunto de reglas que les decía a las computadoras lo qué debían hacer, cómo y en qué momento. (Lazcano, 2020)

> "La **Inteligencia Artificial** (IA) es, en informática, la inteligencia expresada por máquinas, sus procesadores y sus softwares, que serían los análogos al cuerpo, el cerebro y la mente, respectivamente, a diferencia de la inteligencia natural demostrada por humanos y ciertos animales con cerebros complejos. En ciencias de la computación, una máquina «inteligente» ideal es un agente flexible que percibe su entorno y lleva a cabo acciones que maximicen sus posibilidades de éxito en algún objetivo o tarea." (Colaboradores de Wikipedia, 2021g)

Posteriormente, surge el aprendizaje automático (en inglés Machine Learning / ML), que empieza a aportar a las computadoras una verdadera capacidad de aprendizaje, que se adaptaba mucho mejor con el concepto de "Inteligencia Artificial" que las técnicas que se utilizaban en sus inicios. (Lazcano, 2020)

> "**Machine Learning** es una disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente. Aprender en este contexto quiere decir identificar patrones complejos en millones de datos. La máquina que realmente aprende es un algoritmo que revisa los datos y es capaz de predecir comportamientos futuros. Automáticamente, también en este contexto, implica que estos sistemas se mejoran de forma autónoma con el tiempo, sin intervención humana." (González, 2021)

En este sentido, Lazcano (2020) indica que la IA no es más que la capacidad que tiene un ordenador de mostrar un comportamiento "inteligente". Mientras que el Machine Learning es una disciplina de la IA que consiste en crear y mejorar dicho comportamiento.

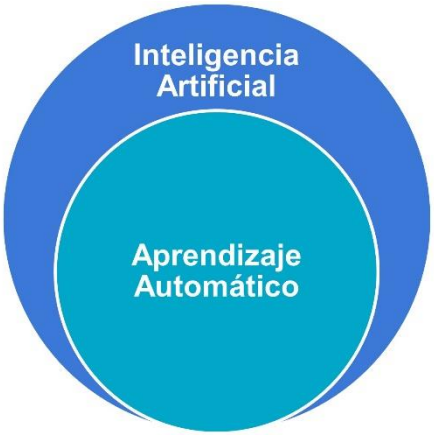


Ilustración 4.2 IA y ML

El ML como método de creación de la IA:

> "Se especializa en técnicas estadísticas para la programación de algoritmos capaces de aprender a realizar tareas. Así, el machine learning se encarga del "aprendizaje", el cual es la base para lograr los distintos campos de aplicación de la Inteligencia Artificial, como lo son el Computer Vision para que las computadoras sean capaces de reconocer las imágenes, o el Natural Language Processing para que puedan comprender el lenguaje humano." (Lazcano, 2020)

En función de lo explicado anteriormente, Lazcano (2020) indica que, para desarrollar el aprendizaje automático en la IA, se puede involucrar en el ML el desarrollo de Redes Neuronales Artificiales (RNA).

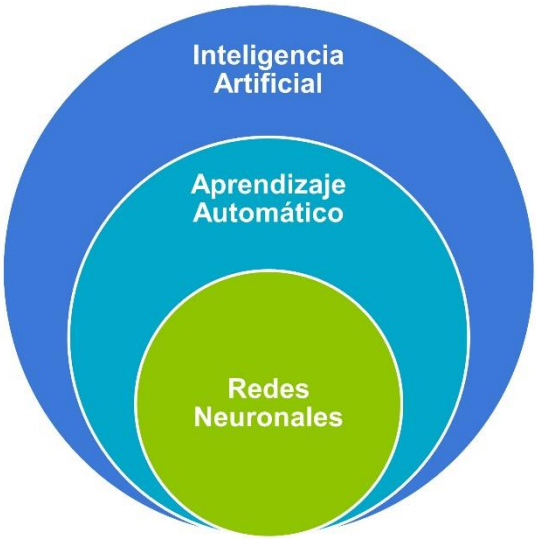


Ilustración 4.3 IA, ML y RNA

> "Las **redes neuronales artificiales** (también conocidas como sistemas conexionistas) son un modelo computacional que fue evolucionando a partir de

diversas aportaciones científicas que están registradas en la historia. Consiste en un conjunto de unidades, llamadas neuronas artificiales, conectadas entre sí para transmitirse señales. La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo unos valores de salida." (Colaboradores de Wikipedia, 2021e)

Cuando las redes neuronales se usan a gran profundidad, surge el concepto de aprendizaje profundo o en inglés Deep Learning (DL), pasando del aprendizaje automático al profundo, consiguiendo de esta forma que los sistemas basados en IA pasen de aprender conceptos a comprender contextos y entornos complejos. (Lazcano, 2020)

"El **Deep Learning** lleva a cabo el proceso de Machine Learning usando una red neuronal artificial que se compone de un número de niveles jerárquicos. En el nivel inicial de la jerarquía la red aprende algo simple y luego envía esta información al siguiente nivel. El siguiente nivel toma esta información sencilla, la combina, compone una información algo un poco más compleja, y se lo pasa al tercer nivel, y así sucesivamente." (Moreno, 2016)

Analizando en mayor profundidad el concepto de ML, se puede decir que, dado un conjunto de datos, éste se encarga de representar su estructura y generalizar su comportamiento aprendiendo a partir de estos. (Márquez, 2018)

En ML, cuando se hace referencia a aprender, como indica Márquez (2018): "no se trata de memorizar y recolectar datos. Se trata de crear un modelo a partir de la información suministrada para poder generar conclusiones sobre ejemplares nunca vistos".

Este proyecto se centrará en hacer predicciones en tiempo real, es decir, se analizará el tiempo, el tráfico y la calidad del aire en el momento y se predecirá cómo cambiará la calidad del aire en el futuro. Este tipo de enfoque se conoce como modelos de series de tiempo.

"Una serie de tiempo es una secuencia de datos u observaciones, medidos en determinados momentos y ordenados cronológicamente. Visualmente, es una curva que evoluciona en el tiempo.

Una serie de tiempo es un conjunto de observaciones sobre los valores que toma una variable (cuantitativa) a través del tiempo. Por tanto, una serie de tiempo es una forma estructurada de representar datos." (PRICING - Revenue Management, 2021)

Hacer predicciones con series de tiempo consiste en extender los valores históricos de la serie al futuro. De esta forma, dado un conjunto de valores históricos, se pueden predecir los valores futuros de forma cronológica; un ejemplo práctico sería: dadas 24 horas en el pasado, se predicen las próximas 12horas. Los rangos de tiempo dependerán de la configuración del modelo.

Una vez definida la estructura big data sobre la que se trabajará, los conceptos de IA y el tipo de modelo que se pretende realizar, se plantea realizar los modelos predictivos utilizando redes neuronales.

Las redes neuronales trabajan muy bien con series de tiempo. Gracias a las funcionalidades de la librería Keras, se podrán probar estructuras de redes neuronales y así poder seleccionar el modelo que mejores resultados obtenga.

Se crearán modelos de predicción de tipo: lineal, denso, red neuronal convolucional y de red neuronal recurrente.

A continuación, se explicará con mayor detalle el funcionamiento de las redes neuronales artificiales.

4.3.3.1 Redes neuronales artificiales

Las redes neuronales artificiales están basadas en redes neuronales biológicas. Una neurona está compuesta por dendritas, el soma y el axón. En conjunto funcionan de la siguiente forma: las dendritas captan los impulsos nerviosos que emiten otras neuronas, estos impulsos se procesan en el soma y se transmite a través del axón que emite un impulso nervioso hacia las neuronas contiguas. (García-Olalla Olivera, 2019)

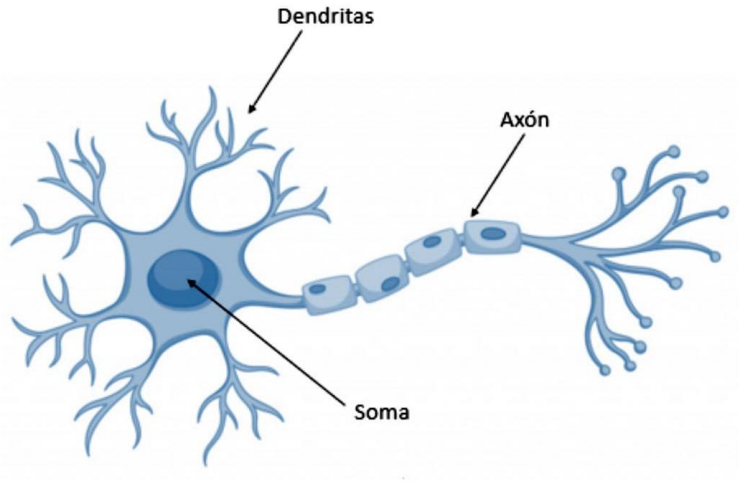


Ilustración 4.4 Neurona biológica (García-Olalla Olivera, 2019)

De esta misma forma, una neurona artificial o perceptrón se representa del siguiente modo:

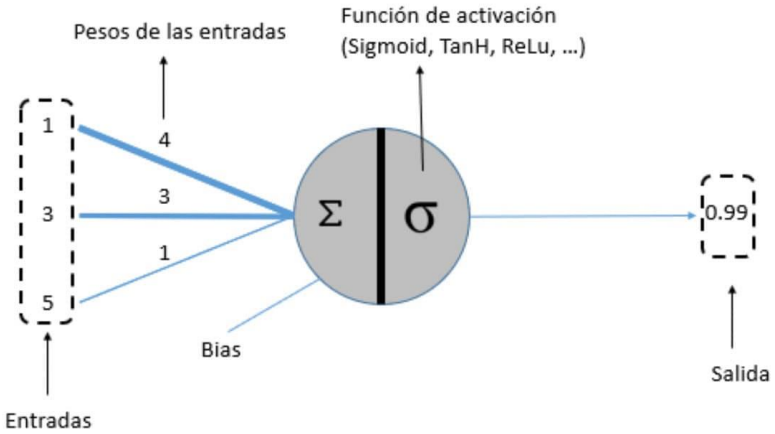


Ilustración 4.5 Neurona artificial (García-Olalla Olivera, 2019)

> "En el caso de las neuronas artificiales, la suma de las entradas multiplicadas por sus pesos asociados determina el "impulso nervioso" que recibe la neurona. Este valor se procesa en el interior de la célula mediante una función de activación que devuelve un valor que se envía como salida de la neurona." (García-Olalla Olivera, 2019)

El cerebro humano está compuesto de neuronas conectadas entre sí. Con las redes neuronales artificiales sucede algo similar. Éstas se agrupan en capas y estas capas se conectan entre sí. Esta estructura se compone por una capa de entrada,

ninguna o muchas capas ocultas y una capa de salida. Un ejemplo claro sería el siguiente:

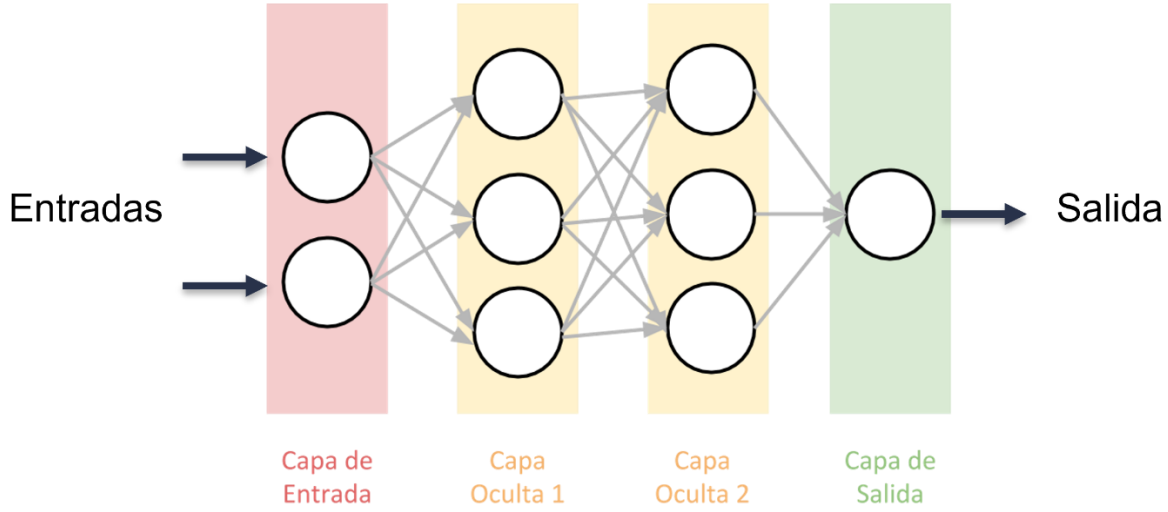


Ilustración 4.6 Esquema de capas en una Red Neuronal (Bagnato, 2017)

Siguiendo el ejemplo de la Ilustración 4.6 una red neuronal artificial funcionaría de la siguiente forma:

La capa de entrada recibe el conjunto de los datos que alimentaran la red neuronal. En este ejemplo, la capa de entrada cuenta con dos neuronas. Las neuronas de la capa de entrada proporcionan los datos que utilizará la capa oculta 1 que tiene 3 neuronas, estos datos se desconocen. De forma sucesiva, la capa oculta 1 proporciona los datos de la capa oculta 2 que también tiene 3 neuronas, y ésta a su vez proporciona los datos a la capa de salida que tiene 1 neurona. Finalmente, la capa salida proporciona el resultado de la red.

De esta forma, solo se conocen los datos proporcionados a la capa de entrada y los datos generados por la capa de salida. Todo lo que sucede dentro de las capas ocultas se desconoce. Esto sucede porque las capas ocultas de una red neuronal contienen unidades no observables.

La cantidad total de capas otorga "profundidad" al modelo, por eso se usa el término de "Aprendizaje Profundo". Cada una de las conexiones de la red neuronal se asocia a un peso, este peso es el que dictamina la importancia que tendrá esa relación en la neurona al multiplicarse por el valor de entrada. Inicialmente, estos

pesos se asignan aleatoriamente, pero a medida que se va iterando estos pesos se van ajustando solos. (Bagnato, 2017)

Imitando el comportamiento de las neuronas biológicas, cada neurona artificial cuenta con una función de activación. (Bagnato, 2017)

"En las redes neuronales artificiales, la **función de activación** de un nodo define la salida de ese nodo dada una entrada o un conjunto de entradas." (Colaboradores de Wikipedia, 2021b)

La **función de activación** determina si la suma de los valores recibidos (multiplicados previamente por el peso de la conexión) supera el umbral que hace que la neurona se active y dispare un valor hacia la siguiente capa conectada. (Bagnato, 2017)

Finalmente, se llega a la capa final con la predicción una vez que todas las capas han finalizado sus cómputos. Este proceso es conocido como **propagación hacia adelante**.

Las funciones de activación más utilizadas en problemas de redes neuronales son las siguientes (véase Ilustración 4.7):

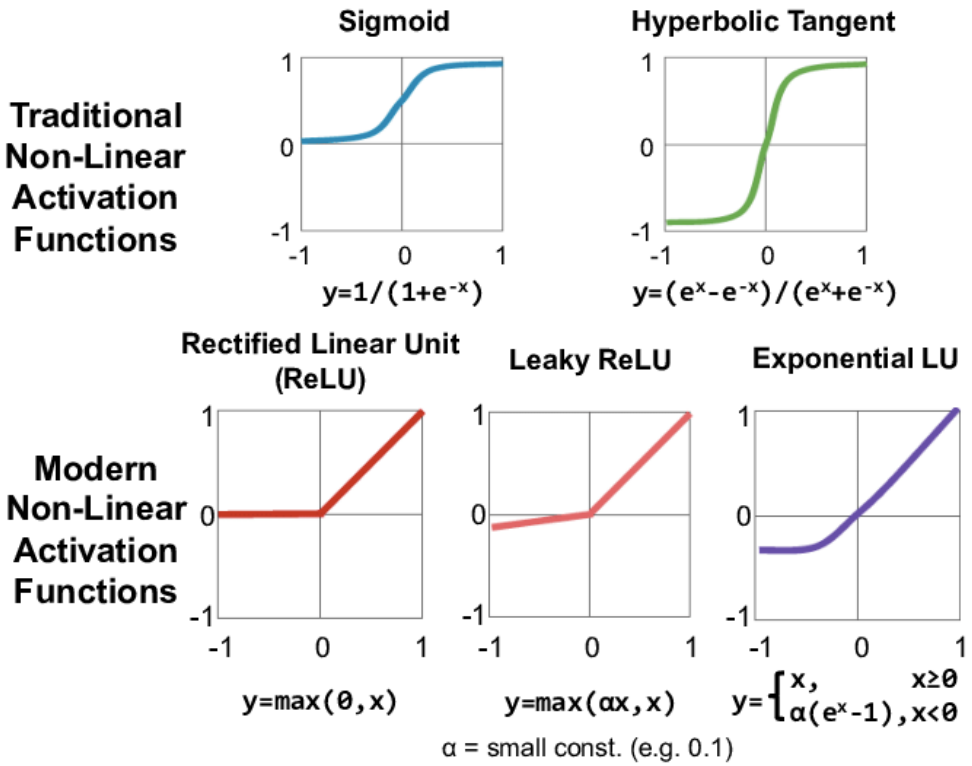


Ilustración 4.7 Funciones de activación

- Función Lineal:

  "Actúa como función identidad trasladando a su salida el resultado obtenido en la primera parte del procesamiento de la neurona (la suma ponderada de entradas, por ejemplo)." (Gavilán, 2020)

- Función Rectified Linear Unit ('ReLU'):

  "El valor de la función de activación es igual a su entrada mientras ésta sea mayor que cero (en ese caso sí actúa como lineal) pero, en caso contrario, el valor de activación es nulo. Su labor, en el fondo, no es más que eliminar valores negativos." (Gavilán, 2020)

- Función Sigmoidea (logística):

  "Se trata de una función continua y derivable (importante en los algoritmos de aprendizaje), con un rango entre 0 y 1 y generalmente cerca de esos extremos (0 ó 1). Esto la hace muy adecuada para problemas de clasificación." (Gavilán, 2020)

- Función Tangente Hiperbólica:

  "Una función con una forma parecida a la sigmoidea pero que, en este caso, proporciona valores entre 1 y -1, es decir, admite valores negativos en la salida." (Gavilán, 2020)

- Función 'softmax' (función exponencial normalizada):

  "Es una función de activación un poco particular, puesto que la salida no sólo depende de las entradas de la neurona sino también de la salida de las otras neuronas de la misma capa. Cuando se aplica una función de activación, la suma de las activaciones de la capa debe ser 1. Se utiliza en problemas de clasificación en múltiples categorías y etiquetas, en las que cada neurona de salida representa una categoría." (Gavilán, 2020)

Dado que los modelos que se llevarán a cabo durante el desarrollo de este proyecto corresponden a problemas de series temporales, para la creación de éstos se utilizará la función 'ReLU', ya que esta función da muy buenos resultados y es de las más recomendadas en problemas de este tipo. Por otro lado, las funciones 'Sigmoidea', 'Tangente Hiperbólica' y 'softmax' no son las más indicadas para el tipo de datos y de modelo que se pretende realizar en este proyecto, ya que estas funciones dan mejores resultados para problemas de clasificación, Por tanto, fueron descartadas.

Una vez explicado el funcionamiento básico de una red neuronal, se pueden explicar términos un poco más específicos como: qué es la **propagación hacia atrás**, el concepto de **pérdida** y **épocas** de la red.

Retomando la explicación anterior (ejemplo de la Ilustración 4.6 Esquema de capas en una Red Neuronal , cada neurona de una red neuronal toma los valores de entrada multiplicados por una ponderación para representar la fortaleza de esa conexión. (La Vigne, 2019)

> "La propagación hacia atrás detecta las ponderaciones correctas que se deben aplicar a los nodos de una red neuronal mediante la comparación de las salidas actuales de la red con los resultados correctos o deseados. La diferencia entre el resultado deseado y el resultado actual se calcula mediante la función de pérdida o costo. En otras palabras, la función de pérdida nos indica el grado de precisión que tiene nuestra red neuronal al realizar predicciones para una entrada determinada." (La Vigne, 2019)

La **propagación hacia atrás** surge cuando el algoritmo retrocede y ajusta las ponderaciones y los sesgos después de calcular una respuesta. Cuanto menor sea la pérdida para una red, más precisa será. Cada ciclo de corrección de propagación hacia atrás y hacia adelante para reducir la pérdida se denomina época. (La Vigne, 2019)

Se puede decir que la propagación hacia atrás consiste en determinar las mejores ponderaciones y sesgos de entrada para obtener un resultado más preciso o "minimizar la pérdida". (La Vigne, 2019)

El objetivo del aprendizaje automático y el aprendizaje profundo es reducir la diferencia entre el resultado previsto y el resultado real. Esto se denomina **función de coste** o **función de pérdida**, estas son funciones convexas. (ICHI.PRO, 2021)

Se debe procurar minimizar la función de pérdida y encontrar el valor optimizado para los pesos. Para esto, se debe asegurar que el algoritmo generalice bien. Esto ayudará a hacer una mejor predicción para los datos que el modelo no utilizó durante el entreno. La ejecución de múltiples iteraciones con diferentes pesos ayuda a encontrar el costo mínimo. (ICHI.PRO, 2021)

Al momento de entrenar los modelos de predicción, se le indicará un optimizador y la función de pérdida que se utilizará. El optimizador actualizará los parámetros de peso para minimizar la función de pérdida. La función de pérdida actuará como una guía, indicando al optimizador si se está moviendo en la dirección correcta para encontrar el mínimo global.

Además de estos conceptos, se debe tener en cuenta que las redes neuronales cuentan con dos fases en la modelización.

1- Fase de entrenamiento:

Durante esta fase se le proporciona a la red un conjunto de datos de entrenamiento. Estos datos serán los que se utilizarán para determinar los pesos que definen el modelo de la red neuronal. Los pesos se calculan de forma iterativa, procurando reducir el error cometido entre la salida obtenida por la red neuronal y la salida deseada.

2- Fase de Prueba:

La fase de prueba se utiliza para verificar que el modelo no se ajuste demasiado a los datos, perdiendo la habilidad de generalizar su aprendizaje. Esto se conoce como sobreajuste (overfitting). Además, verifica que efectivamente la red haya completado el aprendizaje. En caso contrario, se tendría un caso de subajuste (underfitting), lo que indicaría que el modelo es muy simple y no consigue obtener resultados.

Por otro lado, las redes neuronales pueden ser de aprendizaje supervisado y no supervisado:

> "El **aprendizaje supervisado** es una técnica para deducir una función a partir de datos de entrenamiento. Los datos de entrenamiento consisten en pares de objetos (normalmente vectores): una componente del par son los datos de entrada y el otro, los resultados deseados. La salida de la función puede ser un valor numérico (como en los problemas de regresión) o una etiqueta de clase (como en los de clasificación).
>
> El objetivo del aprendizaje supervisado es el de crear una función capaz de predecir el valor correspondiente a cualquier objeto de entrada válida después de haber visto una serie de ejemplos, los datos de entrenamiento. Para ello, tiene que

generalizar a partir de los datos presentados a las situaciones no vistas previamente." (Colaboradores de Wikipedia, 2021a)

"**Aprendizaje no supervisado** es un método de Aprendizaje Automático donde un modelo se ajusta a las observaciones. Se distingue del Aprendizaje supervisado por el hecho de que no hay un conocimiento a priori. En el aprendizaje no supervisado, un conjunto de datos de objetos de entrada es tratado. Así, el aprendizaje no supervisado típicamente trata los objetos de entrada como un conjunto de variables aleatorias, siendo construido un modelo de densidad para el conjunto de datos." (Colaboradores de Wikipedia, 2020)

El tipo de red utilizado para este proyecto será de aprendizaje supervisado, ya que para generar los modelos se indicará un conjunto de datos de entreno para los cuales la función de salida generará un valor numérico.

4.4 Herramientas empleadas

A continuación, se listarán y se definirán todas las herramientas empleadas durante el proyecto y las librerías más relevantes utilizadas.

1. Python:

Python es un lenguaje de programación de código abierto multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje muy simple y fácil de entender. Su filosofía hace hincapié en la legibilidad de su código. (Colaboradores de Wikipedia, 2021f)

Python permite plasmar ideas complejas en pocas líneas de código, cosa que no es posible en otros lenguajes. Gracias a que es un lenguaje de escritura rápido, escalable, robusto y de código abierto, se convierte en un aliado perfecto para la Inteligencia Artificial. (Soloaga, 2019)

Además, cuenta con bibliotecas como Keras y Tensorflow, que permiten resolver problemas usando aprendizaje automático con gran facilidad.

2. Visual Studio Code:

Es un editor de codificación gratuito, compatible con muchos lenguajes entre ellos los más populares: Python, Java, C++, HTML, JavaScript, entre otros. Permite instalar plugins[4] de gran utilidad que ayudan al programador en su labor.

Incluye un control de fuente integrado para guardar el trabajo a lo largo del tiempo y no perder el progreso. Dispone de una vista gráfica en paralelo para comparar versiones del código de diferentes momentos. Se integra con gran facilidad con aplicaciones para el control de versiones como GitHub. (Microsoft, 2016)

3. GitHub:

> "Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git." (Colaboradores de Wikipedia, 2021c)

Permite comparar el código en sus distintas versiones, restaurar versiones antiguas y fusionar cambios en distintas versiones. Es de uso gratuito, los repositorios pueden ser públicos o privados. Además, se puede dar acceso a otros usuarios para editar y cargar contenido a los repositorios.

4. CSV:

Un archivo CSV (valores separados por comas) es un tipo especial de archivo que puede crear o editar en Excel. En lugar de almacenar la información en columnas, los archivos CSV almacenan datos separados por comas. Cuando el texto y los números se guardan en este tipo de archivo, es fácil moverlos de un programa a otro. (Microsoft, 2021)

Es uno de los formatos de datos más habituales usados con Python. Existen muchas librerías en Python que permiten manipular este tipo de archivos. Entre las más polares se encuentra la librería Pandas.

[4] **Plugins:** "son pequeños programas complementarios que amplían las funciones de aplicaciones web y programas de escritorio." (1&1 IONOS España S.L.U, 2020)

5. Pandas:

> "Pandas es una herramienta de manipulación y análisis de datos de código abierto rápida, potente, flexible y fácil de usar, construida sobre el lenguaje de programación Python." (Pandas, 2021)

Entre sus aspectos más destacados se incluye la funcionalidad de series de tiempo, con generación de rango de fechas y conversión numérica de frecuencia, cambio de fecha y retraso, entre otros. Está altamente optimizado para el rendimiento. Permite remodelación flexible y rotación de conjuntos de datos. (Pandas, 2021)

6. Jupyter:

> "JupyterLab es una interfaz de usuario basada en web para Project Jupytery está estrechamente integrada en Adobe Experience Platform. Proporciona un entorno de desarrollo interactivo para que los científicos de datos trabajen con Jupyter Notebooks, código y datos." (Adobe, 2021)

7. Tensorflow

> "TensorFlow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que les permite a los investigadores innovar con el aprendizaje automático y, a los desarrolladores, compilar e implementar con facilidad aplicaciones con tecnología de aprendizaje automático." (Abadi, y otros, TensorFlow, 2021)

8. Keras:

> "Keras es una biblioteca de Redes Neuronales de Código Abierto escrita en Python. Es capaz de ejecutarse sobre TensorFlow, Microsoft Cognitive Toolkit o Theano." (Keras, 2021a)

> "Sigue las mejores prácticas para reducir la carga cognitiva: ofrece API consistentes y simples, minimiza la cantidad de acciones del usuario necesarias para casos de uso comunes y proporciona mensajes de error claros y procesables. También cuenta con una extensa documentación y guías para desarrolladores." (Keras, 2021b)

9. Flask:

Es un framework minimalista para crear aplicaciones web escrito en Python. Diseñado para que la puesta en marcha sea rápida y sencilla, con la capacidad de escalar a aplicaciones complejas. (Flask - The Pallets Projects, 2021)

4.5 Recolección de datos

Una vez conocidas las fuentes que proporcionaran los datos para el tráfico, el clima, la calidad del aire, y la tecnología big data que se utilizará (en este caso Python), se procedió a hacer la recolección de los datos.

Los tipos de datos recolectados de las distintas API son de tipo estructurado, lo que quiere decir que tienen forma y sentido. Los datos provienen de distintas fuentes de datos, teniendo en común una única columna con los datos referentes a la fecha y hora a los que corresponden.

Para realizar la recolección de los datos históricos para cada una de las fuentes de datos, se escribió un método para solicitar los datos a la API en un rango de fecha dado y se hicieron los ajustes necesarios a los datos para que estos fuesen más fáciles de manipular.

Una vez creados los métodos para recolectar los datos históricos de las API, se procedió a crear un método principal hacer las llamadas a las API. Este método consistió en realizar la recolección de los datos dada una fecha y un número de meses, es decir, si se hace una solicitud con fecha 2021/01/01 a las 00:00:00 y con número de meses igual a 5, se solicitaría a las API datos desde enero hasta mayo.

De esta forma, se consiguió obtener los datos históricos para el rango de fecha deseado. En este caso, los datos que se recolectaron fueron desde el 01 de julio de 2019 hasta el 30 de junio de 2021, siendo estos los datos que se utilizaron para entrenar los modelos de predicción.

Sin embargo, fue necesario crear un método adicional que ayudase al método principal a unificar todos estos datos en un único documento. Durante la recolección de los datos se utilizó la librería de Pandas, que permite manipular los datos en tablas y tener columnas en formatos de series de tiempo, siendo de gran utilidad para unificar

los datos en un único documento CSV. Se utilizó como condición para unir la información de las distintas fuentes la fecha y la hora a la que correspondían.

Posteriormente, para realizar la recolección de los datos en tiempo real se utilizaron métodos muy similares a los de la recolección de los datos históricos, pero añadiendo algunas condiciones adicionales necesarias para asegurar que funcionasen de forma correcta. Se modificaron los métodos de las API para que, en lugar de consultar datos para un rango de fecha, lo hicieran a partir de una fecha y una hora dada.

Para la recolección de los datos en tiempo real se creó una clase principal que controlase la lógica en la que se hacen las peticiones a las API y la forma en la que se unen en un mismo archivo en tiempo real.

La lógica establecida fue la siguiente:

1. Se crea el documento de tipo CSV donde se guardarán los datos.
2. Se guarda en una variable la hora actual en Nueva York.
3. Se ejecutan simultáneamente 3 rutinas que controlaran las llamadas a cada una de las API respectivamente (una rutina para la API de tráfico, una para la API del clima y otra para la API de la calidad del aire).
4. Cuando se genera un dato nuevo, cada una de las rutinas del paso anterior llama a un método de escritura. Este método de escritura es el encargado de escribir los datos en el documento CSV creado en el paso uno.

Cuando se quiere ejecutar de forma simultánea distintas rutinas o acceder a un mismo recurso (en este caso método), es necesario utilizar hilos. Esta es una técnica que permite que una aplicación ejecute simultáneamente varias operaciones

Técnicamente, el funcionamiento de esta clase principal funciona de la siguiente forma:

Una vez creado el documento donde se guardarán los datos y la variable con la hora en la que se ejecuta el método principal, se crearon tres hilos, uno para cada una de las rutinas encargadas de controlar las llamadas a las API.

La rutina del tráfico consulta los datos a partir de la variable de hora establecida inicialmente y, una vez se han recogido los datos del tráfico, se ejecuta un hilo que llama al método que se encarga de unir los datos en el archivo donde se guardan. Posteriormente, para las siguientes iteraciones se consulta el último registro de fecha y hora registrado en el documento de unión y se solicitan los datos a la API a partir de esa fecha. Una vez se detecta que se han registrado valores para la consulta, se ejecuta con un hilo el método de unión. Este proceso se repite constantemente de forma cíclica.

Las rutinas que controlan las llamadas a las API del clima y la calidad del aire funcionan de la misma forma. Dada una fecha y una hora se consultan a la API los datos correspondientes a esa hora en particular y, una vez registrados los datos, se ejecuta un hilo para acceder al método de unión de los datos. Una vez se registran los nuevos datos, se consultan a la API los datos de la siguiente hora y se repite todo el proceso descrito anteriormente de forma cíclica.

Para poder registrar datos provenientes de las API de la calidad del aire y del clima es necesario que ya se hayan registrado todos los datos del tráfico para la hora de los datos que se desean registrar. Por lo que, mientras no se completen los datos de tráfico para una hora, no se registran otros valores distintos a los del tráfico, dejando a las rutinas del clima y la calidad del aire en espera hasta que puedan guardar sus datos.

En el siguiente esquema (véase Ilustración 4.8) se hace una pequeña demostración de la forma en que la que se registran los datos.

Ilustración 4.8 Esquema de la unión de los datos de las distintas APIs en tiempo real

En el **paso 1,** se han registrado todos los valores del tráfico para la 1 de la mañana y para las 2 de la mañana aún no se han registrado valores nuevos. Por otro lado, se cuenta con los valores para la 1 de la mañana de la calidad del aire y del clima.

En el **paso 2,** se solicita registrar los valores a la 1 de la mañana para la calidad del aire y el clima. Como se cumple la condición en la que se establece que, para registrar datos distintos del tráfico es necesario que los datos correspondientes al tráfico ya estén guardados para la hora de los nuevos datos a registrar, se añaden de forma exitosa.

En el **paso 3,** las rutinas de calidad de aire y de clima actualizan sus datos a las 2 de la mañana, pero cuando solicitan registrarlos en el documento con todos los datos, no se cumple la condición ya que aún faltan datos para el tráfico para las 2 de la mañana, por lo que se queda en espera hasta que se registren los datos del tráfico para esa hora.

A continuación, se explicará con detalle los datos recolectados para cada una de las API consultadas. Tanto para los datos históricos como para los datos en tiempo real se usaron las mismas fuentes de información, por lo que los datos proporcionados en ambos casos tienen el mismo tipo de contenido y formato.

4.5.1 Recolección de datos del tráfico

Esta API guarda para la fecha, hora, minutos y segundos en los que un sensor registra un vehículo, la velocidad, el tiempo de viaje, la calle, coordenadas de ubicación, entre otros. Cada fila representa la velocidad promedia en la que los vehículos han viajado desde un punto a otro. Un sensor al comienzo de una calle detecta un vehículo, y otro sensor ubicado al final lo vuelve a detectar. En función del tiempo que tardó en volver a ser registrado y la distancia entre los sensores se calcula la velocidad promedia del vehículo.

La frecuencia con la que se cargan los datos es de varias veces al día y varía desde unos pocos minutos hasta varias horas. Normalmente, durante la madrugada es cuando hay menos tráfico, y los datos tienden a tardar unas cuantas horas en actualizarse, mientras que durante el día normalmente no se tarda más de una hora en registrar nuevos datos.

Los tipos de datos recolectados de la API son los siguientes (véase tabla 4.1):

Tabla 4.1 Datos API de tráfico

Dato	Tipo de dato	Significado
Fecha para tráfico (datetime_traffic)	Marca de tiempo	Fecha, hora, minutos y segundos en los que se registra el valor.
Velocidad (speed)	Texto	Velocidad media del vehículo (Km/h)
Tiempo de viaje (travel_time)	Texto	Tiempo que tarda en pasar por la calle (segundos)
Nombre de la calle (link_name)	Texto	Nombre de la calle en la que se hace el registro

Adicionalmente, se añaden dos columnas adicionales antes de guardar los datos. La primera sería una columna que indica el día de la semana (weekday) y la segunda una columna de fecha y hora (datetime). Debido a que los datos para la calidad del aire y el clima vienen por hora y las predicciones también se realizaran por

hora, es irrelevante mantener los minutos y segundos en los que se registraron los datos. Además, en el momento de unir los datos en un solo archivo, se hace bajo la condición de que coincidan la columna de fecha y hora sin tomar en consideración los segundos. La estructura de los datos recolectados para esta API quedaría así (véase tabla 4.2):

Tabla 4.2 Columnas para datos de tráfico

datetime	datetime_traffic	weekday	speed	travel_time	link_name

4.5.2 Recolección de datos de la calidad del aire

Para realizar consultas a la API de la calidad del aire se le proporciona un cuadro delimitador geográfico con las coordenadas de Manhattan. Para la zona delimitada se indica una lista de parámetros, tipo de datos, el formato (CSV, JSON, KML, XML) y la fecha/hora de inicio y finalización (opcional). Para esta configuración la API devuelve los datos de observación para la zona de monitoreo. Si no se proporciona una fecha de inicio y finalización, devuelve la ultima hora de datos.

Este servicio permite hacer hasta 500 solicitudes en un período de una hora, después de lo cual devolverá un mensaje de error hasta la próxima hora. Esto no implica ningún impedimento ya que no es necesario realizar tantas solicitudes para una misma hora.

Los datos recolectados por esta API para la zona de Manhattan provienen de la estación "CNNY" que pertenece al Departamento de Conservación Ambiental del Estado de Nueva York siendo este una fuente fiable, dicha estación está ubicada a 40.719732º de latitud y a -73.948239º de longitud (véase Ilustración 4.9).

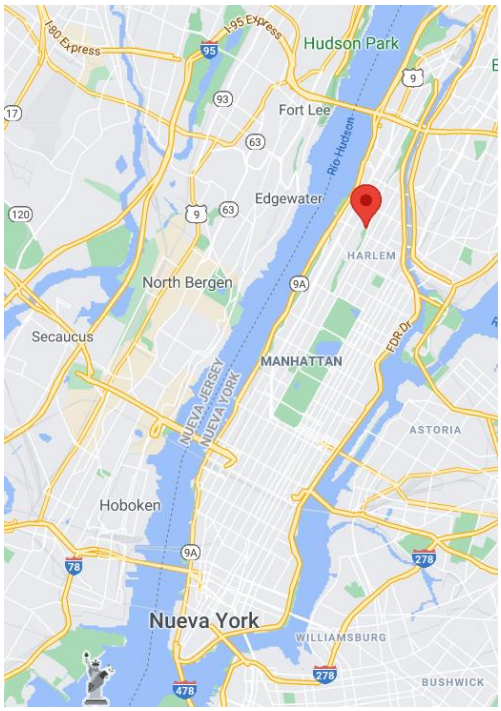


Ilustración 4.9 Estación CNNY (Google, s.f.)

Esta estación detecta las concentraciones de PM2.5 y de ozono (O3) en el aire, por lo que se consultaran a la API los valores del índice de la calidad del aire (AQI) para estos dos componentes. El PM2.5 es, entre todos los contaminantes, el que más afecta esta zona de la ciudad y el causante de mayor número de afectados. Después del PM2.5, el O3 es uno de los contaminantes que peores efectos tiene sobre la salud, por lo que se decidió hacer las predicciones del AQI tomando en cuenta únicamente estos dos componentes.

Por otro lado, aunque se podría consultar la concentración para cada uno de estos componentes, estos valores no son tan fáciles de interpretar como los correspondientes al índice de cada uno de estos.

Se debe recordar que la OMS estableció los índices de la calidad del aire (AQI) para que indistintamente del contaminante se tuviese una escala genérica para todos que indicase que efectos tienen sobre la salud.

El índice de la calidad el aire que proporciona la API lo desarrolló la EPA. Este índice tiene la misma funcionalidad que los índices establecidos por la OMS, el cual es informar sobre la calidad del aire indistintamente del componente. Dicho índice informa sobre niveles de ozono y otros contaminantes comunes del aire en la misma

escala. El índice funciona de tal forma que, mientras más alta es la calificación del AQI, mayor es el impacto en la salud (véase tabla 4.3).

Tabla 4.3 Categoría AQI

Número AQI	Categoría AQI (descriptor)
0 - 50	Bien
51 - 100	Moderado
101 - 150	No saludable para grupos sensibles
151 - 200	Insalubre
201 - 300	Muy insalubre
3001 - 500	Peligroso

La frecuencia de los datos es por hora o por día. Normalmente, cuando se solicitan datos de una hora y pasan más de dos horas sin que la estación registre los datos para esa hora es porque no se observaron los valores. Esto no suele suceder, pero en el caso de que suceda, el método que solicita los datos a la API se encarga de detectar la situación y solicita los datos para la hora anterior. Se toman los valores de la hora anterior, debido a que el AQI no tiende a cambiar de forma drástica de una hora a la otra.

Adicionalmente, para hacer las consultas, la fecha y hora deben proporcionarse en la zona horaria UTC para mantener la uniformidad con los métodos de las otras API que hacen las consultas para la hora en Nueva York. Para esto, simplemente se añadió un método que transforma la hora de la zona de Nueva York a la de UTC. De la misma forma, cuando se recogen los datos en UTC, estos datos son convertidos a la hora equivalente para la ciudad de Nueva York.

Finalmente, los datos obtenidos de la API se corresponden a los valores del AQI para el PM2.5 y el O3 y la fecha y hora (datetime) en la que se registraron los valores, quedando la siguiente estructura (véase tabla 4.4):

Tabla 4.4 Columnas para datos de la calidad del aire

datetime	AQI_PM2.5	AQI_OZONE

4.5.3 Recolección de datos del clima

Al momento de solicitar los datos para la API del clima es necesario indicar la fecha, la localización, las unidades de medición y otras especificaciones técnicas. Los datos proporcionados cuentan con una columna de fecha y hora, y más columnas con elementos meteorológicos para cada hora del día.

Todos los conjuntos de datos son una estructura de tabla simple que está disponible en varios formatos, incluido el texto sin formato delimitado, CSV, JSON y ODATA. Los datos incluyen una sola fila de información de encabezados que indica la información en la columna.

Los datos dentro del sistema se calculan a nivel de datos por hora para una mayor precisión y consistencia de los datos. Las unidades utilizadas son las establecidas por los Estados Unidos. Los valores vacíos (o valores nulos) se utilizan para indicar una ausencia de datos. Esto puede ser por la falta de información meteorológica, por datos desconocidos o porque simplemente las ubicaciones no informan de todos los elementos meteorológicos.

En la siguiente tabla se detallarán cada uno de los elementos meteorológicos para los que la API proporciona información (véase tabla 4.5):

Tabla 4.5 Datos API del clima (Visual Crossing, 2020)

Elemento	Nombre de columna	Unidad	Descripción
Temperatura mínima	Minimum Temperature	Grados Fahrenheit	Temperatura mínima durante un período
Temperatura máxima	Maximum Temperature	Grados Fahrenheit	Temperatura máxima durante un período
Temperatura	Temperature	Grados Fahrenheit	Temperatura media durante un período
Punto de rocío	Dew Point	Grados Fahrenheit	El punto de rocío es la temperatura a partir de la cual se debe enfriar el aire para que el vapor de agua se condense en rocío o escarcha.
Humedad relativa	Relative Humidity	Porcentaje	La humedad relativa es la cantidad de vapor de agua presente en el aire comparada con la cantidad máxima posible para una temperatura dada, expresada como porcentaje medio. Los niveles de comodidad humana se encuentran típicamente entre el 30 y el 70%. Los

			valores superiores al 70% se consideran húmedos. Los valores inferiores al 30% se consideran secos.
Índice de calor	Heat Index	Grados Fahrenheit	El índice de calor es una medida de qué tan caliente se siente, combinando la temperatura real del aire con la humedad relativa. La alta humedad se asocia con incomodidad al hacer que la temperatura aparente sea más alta de lo que se sentiría de otra manera. Los valores del índice de calor solo se calculan cuando la temperatura es superior a 80 ° F (aproximadamente 26,7 ° C) y la humedad relativa es superior al 40%. Se devuelve un valor vacío fuera de estos rangos.
Velocidad del viento	Wind Speed	Millas por hora	La velocidad y la dirección del viento indican la velocidad máxima del viento para la ubicación y el período de tiempo solicitados. La velocidad del viento se mide típicamente a 10 m sobre el suelo en un lugar sin obstrucciones cercanas.
Ráfaga de viento	Wind Gust	Millas por hora	La ráfaga de viento es la medida máxima de la velocidad del viento durante un periodo corto de tiempo (generalmente menos de 20 segundos). Hay que tener en cuenta que una ráfaga de viento requiere que la velocidad del viento medida a corto plazo sea significativamente mayor que la velocidad media del viento. Normalmente, la velocidad del viento debería ser de 10 nudos más (11 mph o 18 km / h). Cuando la ráfaga de viento no cumple con estos criterios, se devuelve un valor nulo / vacío.
Dirección del viento	Wind Direction	Grados desde el norte	La dirección del viento indica la dirección desde donde sopla el viento. Las unidades de la dirección del viento son grados desde el norte. El valor varía de 0 grados (desde el norte) a 90 grados (desde el este), 180 grados (desde el sur), 270 (desde el oeste) hasta 360 grados.
Sensación térmica	Wind Chill	Grados Fahrenheit	La sensación térmica es la medida de qué tan frío se siente, combinando la temperatura real del aire con la velocidad del viento. El viento hace que las temperaturas se sientan más frías que cuando el aire está quieto. Los valores de sensación térmica solo se calculan cuando la temperatura es inferior a 50 ° F (aproximadamente 10 ° C) y la velocidad del viento es superior a 3 mph (5 km / h). Se devuelve un valor vacío fuera de estos rangos.

Precipitación	Precipitation	Pulgadas	Es la cantidad de precipitación que cayó o se prevé que caiga en el período de tiempo especificado.
Cobertura de precipitación	Precipitation Cover	Porcentaje	Esta es la proporción de tiempo durante el cual se registró precipitación medible durante un período de tiempo, expresada en porcentaje. Por ejemplo, si en un día de 24 horas hay seis horas de lluvia medible, la cobertura de precipitación es del 25% (6/24 * 100).
Profundidad de la nieve	Snow Depth	Pulgadas	Indica la profundidad de la cantidad de nieve en el suelo durante un determinado período de tiempo
Visibilidad	Visibility	Millas	La visibilidad es la distancia que se puede ver a la luz del día. Esta cuenta de fenómenos meteorológicos como neblina, vaho, niebla o humo.
Cobertura de nubes	Cloud Cover	Porcentaje	Porcentaje de nubes que cubren el cielo para la zona
Presión del nivel del mar	Sea Level Pressure	Milibares	La presión atmosférica en un lugar que elimina la reducción de presión debido a la altitud del lugar.
Condiciones	Conditions	-	Condiciones meteorológicas notables informadas en un lugar en particular, como tormentas eléctricas, lluvia, etc.

Los elementos para el índice de calor, ráfaga de viento y sensación térmica son valores que se calculan o registran solo si se cumplen ciertas condiciones, registrándose en caso contrario valores vacíos (nulos), por lo que existe la probabilidad de tener valores nulos para estos elementos.

A los modelos de predicción no se le pueden pasar valores nulos, resultando que, cuando se intenten hacer predicciones en tiempo real o entrenar el modelo, si para una hora dada no existen registros para algunos de estos elementos, el modelo fallaría. Para evitar este tipo de problemas se tomó la decisión de no tener en incluir estas columnas en el conjunto de datos. La estructura final de los datos quedaría de la siguiente forma (véase tabla 4.6):

Tabla 4.6 Columnas para datos del clima

datetime	**Minimum Temperature**	**Maximum Temperature**	**Temperature**	**Dew Point**	**Relative Humidity**	**Wind Speed**
Wind Direction	**Snow Depth**	**Visibility**	**Cloud Cover**	**Sea Level Pressure**		**Conditions**

4.6 Modelos de predicción

Una vez recolectados los datos históricos a utilizar para entrenar los modelos de predicción, se procedió a analizarlos y prepararlos para poder crear los diferentes modelos, y posteriormente seleccionar el que mejores resultados obtuviese.

En esta parte se utilizó la aplicación Jupyter Lab junto a la plataforma Tensorflow y la librería Keras, que permite crear redes neuronales con gran facilidad.

Para este proyecto se realizarán modelos de predicción a una hora, esto quiere decir que, para cada hora del día se hará una predicción de la calidad del aire, alimentando el modelo inicialmente con 24 horas de pasado. Posteriormente, una vez elegido el modelo, se estudiarán los resultados obtenidos si se le aplican distintos rangos de tiempo de pasado Estos rangos varían entre 6 y 48 horas, y se pretende buscar con que rango se generan mejores predicciones (véase sección 4.7 Selección del modelo).

Debido a que la frecuencia de registro de los datos consumidos en tiempo real es, en la mayoría de los casos, de unas 2 horas aproximadas, para generar predicciones que sean útiles y en tiempo futuro, el modelo generará una salida de al menos 3 horas de predicciones.

De la misma forma, al igual que con la cantidad de horas que se alimenta el modelo, se intentará hacer predicciones para más de 3 horas. Dependiendo de la calidad de los resultados, se harán predicciones a 3 horas o más.

4.6.1 Análisis y preparación de los datos históricos[5]

Inicialmente, se procedió a analizar la calidad de los datos históricos recogidos desde julio de 2019 hasta julio de 2021. Para poder utilizar los datos es importante no contar con valores vacíos (NaN o nulos). Esto es debido a que no se pueden pasar valores nulos al momento de entrenar los modelos o de hacer predicciones, y todos los datos pasados a los modelos deben ser numéricos.

[5] En el repositorio de este proyecto se encuentra una copia del notebook de Jupyter que se utilizó para analizar y preparar los datos históricos.

4.6.1.1 Inspeccionar y limpiar

Los datos recolectados incluyen las siguientes columnas:

'datetime', 'datetime_traffic', 'weekday', 'speed', 'travel_time', 'link_name', 'AQI_PM2.5', 'AQI_OZONE', 'Minimum Temperature','Maximum Temperature', 'Temperature', 'Dew Point', 'Relative Humidity', 'Wind Speed', 'Wind Direction' ,'Precipitation', 'Precipitation Cover', 'Snow Depth', 'Visibility','Cloud Cover', 'Sea Level Pressure', 'Conditions'

A continuación, se hizo una primera limpieza de las columnas y se observaron las estadísticas del conjunto de datos.

Observando las columnas, se pueden descartar algunas directamente ya que no aportan valor e información relevante para el modelo predictivo. Las columnas borradas son las siguientes:

- datetime_traffic: Esta columna indica la fecha y la hora en la que se registraron los datos de tráfico. No aporta valor, ya se cuenta con una columna datetime que indica la hora en la que fue recolectada la información.
- link_name: Esta columna indica la calle en la que fueron registrados los datos del tráfico. Debido a que el modelo se realizará para todo Manhattan, se considera irrelevante la calle en la que se registran los datos.

Posteriormente, se observaron las estadísticas del conjunto de datos y se ajustaron algunos valores atípicos[6] que se encontraron. Para la columna 'Precipitation Cover' no se registró ningún dato, por lo que se procedió a eliminarla. Luego, se convirtieron en numéricas aquellas columnas que no eran de tipo categórico, siendo estas las columnas de 'weekday' y 'Conditions'.

Para poder generar el modelo predictivo en función del AQI, se incorporará a los datos una columna 'AQI' que será igual al mayor de los valores de los AQIs recolectados, es decir, AQI = max(AQI_PM2.5, AQI_OZONE). Esta columna 'AQI' representa la calidad del aire. Una vez creada la columna 'AQI', se borraron las

[6] **Valores atípicos:** son valores que son extremadamente grandes o pequeños en comparación con el resto de los valores registrados.

otras dos columnas, ya que estas eran linealmente dependientes de la columna a predecir.

Finalmente, como el modelo que se realizará llevará a cabo predicciones a 1 hora, los datos deben estar agrupados por hora. Para cumplir con esta condición, se utilizó la columna 'datetime' para agrupar los datos, conservando el valor medio por cada hora para el resto de las columnas.

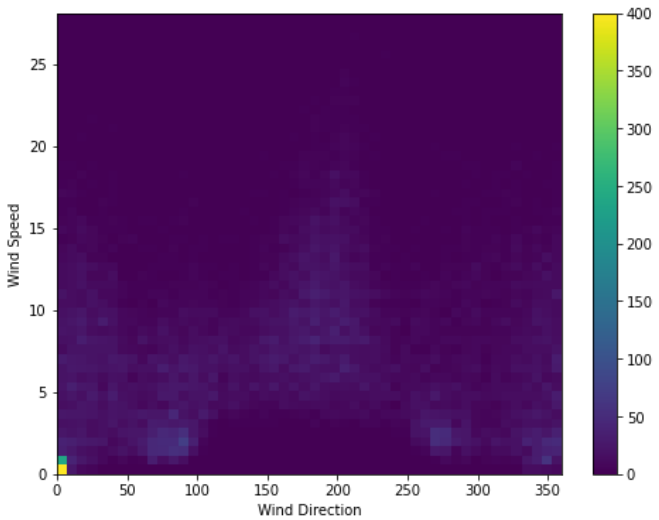
Una vez agrupados los datos, éstos se verifican nuevamente para observar el número de valores nulos. Si siguen existiendo filas con valores nulos, se borrarán las líneas correspondientes a estos valores, debido a que el modelo no admite valores vacíos.

4.6.1.2 Ingeniería de características

Antes de sumergirse en la construcción del modelo, es importante comprender los datos y asegurarse de que los que se pasan al modelo están en el formato indicado.

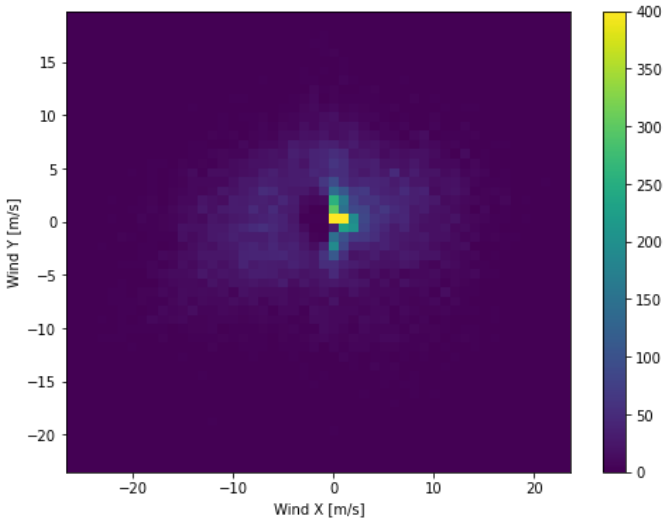
Viento:

La columna 'Wind Direction' da la dirección del viento en unidades de grados. Los ángulos no son buenas entradas para el modelo. Los valores de 360° y 0° deben estar cerca entre sí y envolverse suavemente, por lo que la dirección no debería importar si el viento no sopla. Para el conjunto de datos recolectados, la distribución de los datos del viento se ve así (véase Gráfica 4.1): (Apache License, Version 2.0 , 2021)



Gráfica 4.1 Distribución de los datos de viento

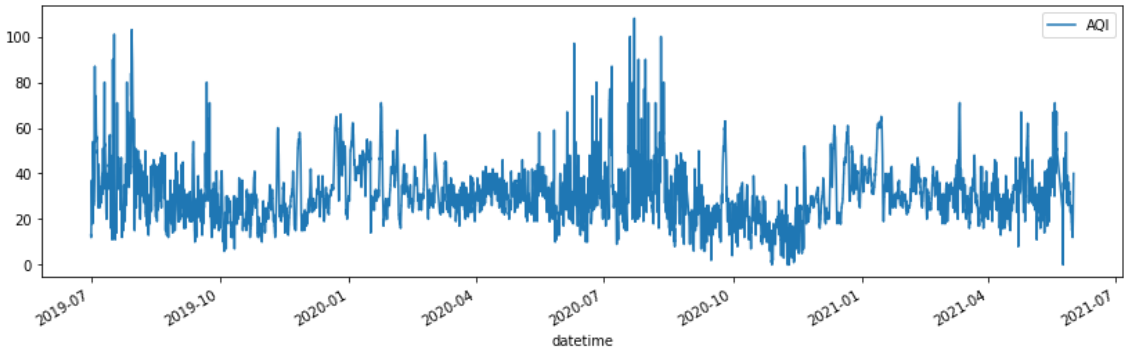
Para el modelo es más fácil interpretar los datos del viento si se transforman en un vector, por lo que se transformó la velocidad y la dirección del viento en un vector, quedando la siguiente distribución:



Gráfica 4.2 Distribución de los datos de viento - vector

Hora:

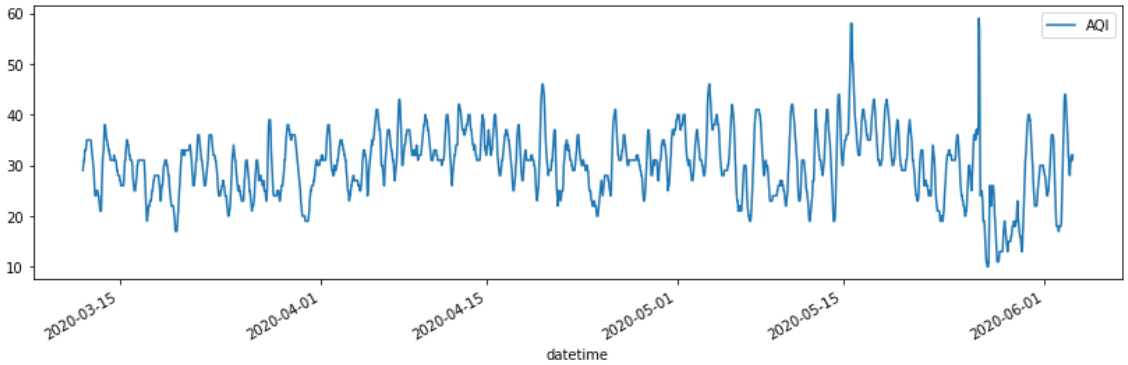
Al igual que con el viento, la columna de 'datetime' es útil, pero no en la forma en la que se tiene, ya que al ser una columna de tipo datetime el modelo no la interpretará como es debido, por lo que se procedió a convertirla en segundos. Una vez convertida en segundos se puede ver la evolución de la característica del 'AQI' a lo largo del tiempo (Apache License, Version 2.0 , 2021).



Gráfica 4.3 Evolución del AQI a lo largo del tiempo

Tomando una muestra un poco más pequeña se puede observar con mayor facilidad el tipo de serie de tiempo al que se corresponden los datos (véase gráfica 4.5). En este caso, la serie de tiempo es estacionaria. Esto es de esperarse debido a

que existe una clara relación entre los datos del clima, del tráfico y de la calidad del aire, por lo que, tomando en cuenta que el clima es un tipo de dato estacionario que se repite para cada año (teniendo 4 estaciones; primavera, verano, otoño e invierno), es normal que los datos para la calidad del aire también tengan un comportamiento similar.



Gráfica 4.4 Evolución del AQI a lo largo de un corto período de tiempo

Para considerar que una serie de tiempo es estacionaria, ésta debe cumplir tres criterios básicos:

1. "La media de la serie no debe ser una función de tiempo; sino que debe ser constante. La siguiente imagen (Ilustración 4.10) muestra una serie que cumple con esta condición y otra que no la cumple." (Lopez Briega, 2016)

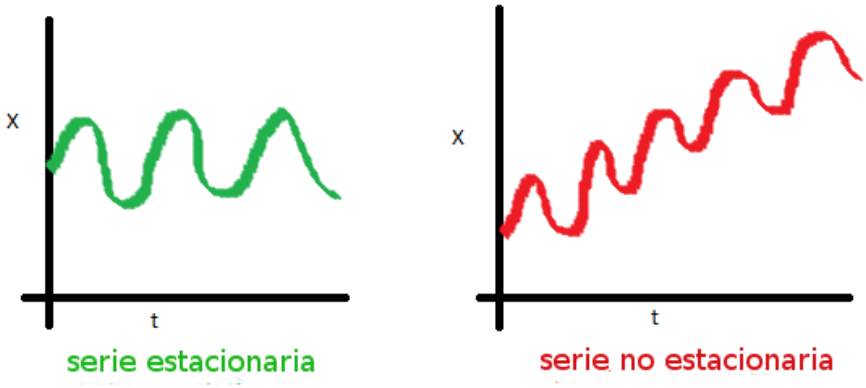


Ilustración 4.10 Media de una serie de tiempo (Lopez Briega, 2016)

2. "La varianza de la serie no debe ser una función del tiempo. El siguiente gráfico (Ilustración 4.11) representa una serie cuya varianza no está

afectada por el tiempo (es estacionaria) y otra que no cumple con esa condición." (Lopez Briega, 2016)

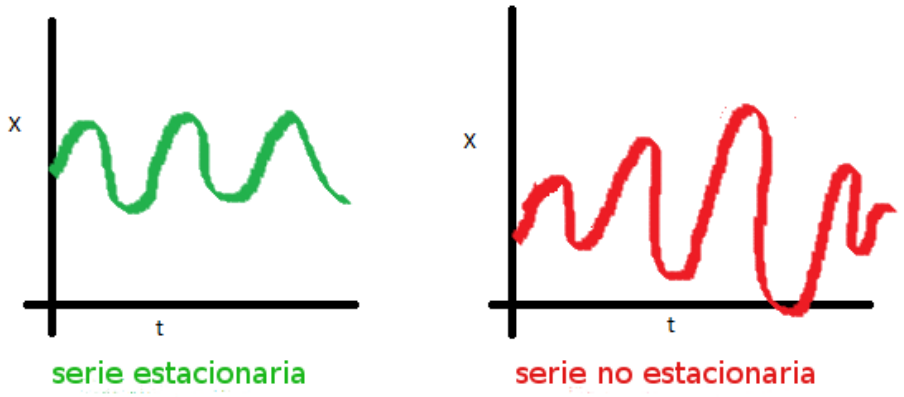


Ilustración 4.11 Varianza de una serie de tiempo (Lopez Briega, 2016)

3. "La covarianza de la serie no debe ser una función del tiempo. En el gráfico de la derecha (Ilustración 4.12), se puede observar que la propagación de la serie se va encogiendo a medida que aumenta el tiempo. Por lo tanto, la covarianza no es constante en el tiempo para la serie roja." (Lopez Briega, 2016)

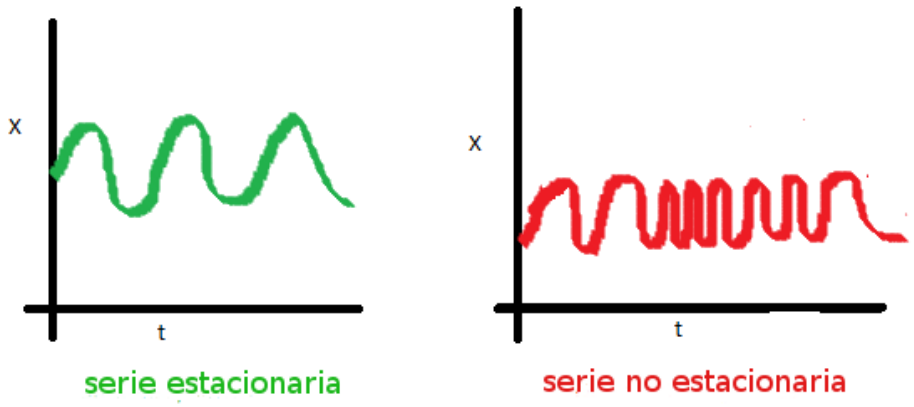


Ilustración 4.12 Covarianza de una serie de tiempo (Lopez Briega, 2016)

Una vez definidos estos criterios, si se observan las gráficas Gráfica 4.3 y Gráfica 4.4, se puede confirmar fácilmente que la naturaleza de los datos se corresponde con series de tiempo estacionarias, teniendo claramente una periodicidad diaria y anual. El tiempo en segundos tampoco es de gran valor para el modelo, pero gracias a que se ha identificado que los datos son periódicos, existe un método simple para convertir los segundos en una señal clara de "Hora del día" y "Hora del año" que es utilizando senos y cosenos. (Apache License, Version 2.0 , 2021)

Para crear los valores para la señal de "Hora del día", se usaron las ecuaciones 4.1 y para los valores de "Hora del año" las ecuaciones 4.2:

$$\text{día\_sin} = \sin\left(\frac{2 \cdot \pi \cdot segundos}{día}\right) \qquad \text{día\_cos} = \cos\left(\frac{2 \cdot \pi \cdot segundos}{día}\right)$$

Ecuación 4.1 Senos y cosenos para hora del día

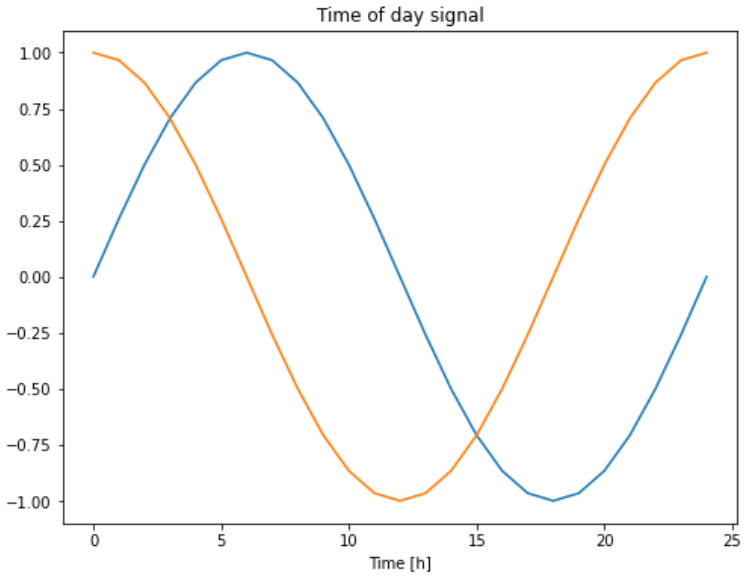
$$\text{año\_sin} = \sin\left(\frac{2 \cdot \pi \cdot segundos}{año}\right) \qquad \text{año\_cos} = \cos\left(\frac{2 \cdot \pi \cdot segundos}{año}\right)$$

Ecuación 4.2 Senos y cosenos para hora del año

siendo:

$$día = 24 \cdot 60 \cdot 60 \qquad año = (365{,}2425) \cdot día$$

Graficando los valores para el seno y el coseno de la hora del día, se vería claramente como se cumple el período para las 24 del día (véase Gráfica 4.5). Lo mismo ocurre para las horas del año.



Gráfica 4.5 Hora del día en senos y cosenos

Finalmente, llegados a este punto se han transformado todos los datos a un formato numérico, se han borrado las filas con valores vacíos, se han agrupado los datos por fecha/hora y se han modificado algunas columnas para que aporten mayor utilidad al modelo, por lo que el conjunto de datos ya está preparado para empezar a entrenar los modelos.

4.6.2 Creación de modelos[7]

Una vez analizados y preparados los datos se procedió a dividirlos en datos de entreno, validación y prueba (test), dividiéndolos en 70%, 20% y 10% respectivamente, de forma correlativa. Debido a que las predicciones se realizarán en función del tiempo, éstos no deben tomarse de forma aleatoria. Esto es por dos razones:

1. Por un lado, se garantiza que aún sea posible dividir los datos en ventanas de muestras consecutivas. (Apache License, Version 2.0 , 2021)
2. En segundo lugar, se garantiza que los resultados de la validación y prueba sean más realistas y se evalúen en función de los datos recopilados después de que el modelo haya sido entrenado. (Apache License, Version 2.0 , 2021)

Para crear los modelos, se utilizó el método "fit" de la librería de Keras. Normalmente, este método funciona de la siguiente forma: se le pasa como argumento un conjunto de datos con los valores de entreno y éste lo divide en datos para el entreno y datos para la validación. Estos datos de validación son los que utiliza el método fit de Keras para evaluar la pérdida y la métrica indicada del modelo al final de cada época. Es decir, los datos de validación son los datos que utiliza el método fit de Keras para evaluar el modelo en cuestión.

Explicando con un poco más de detalle el funcionamiento interno del método fit, éste hace lo siguiente: entrena el modelo con los datos de entreno durante un número determinado de épocas. Al final de cada época, el modelo hace predicciones para los datos de validación. En ningún momento el modelo en sí accede a los datos enteros de la validación, únicamente accede a los datos que necesita para generar las predicciones. Posteriormente, el método fit (que sí tiene total control sobre los datos), toma la predicción que generó el modelo y el valor esperado para así generar las estadísticas correspondientes y evaluar cada época.

Cuando se hablen de datos de validación a lo largo de este documento, no deben confundirse con los datos de test o prueba, pues no son lo mismo. En este

[7] En el repositorio de este proyecto se encuentra el Jupyter notebook con las clases, métodos y líneas de código utilizados en este apartado.

caso, los datos de validación son los que utiliza el método fit para evaluar el modelo y generar las estadísticas para este de forma interna, y los datos de test serán los datos que se utilizarán fuera de este método cuando se quiera evaluar el modelo final.

En lugar dividir los datos en entreno y test, que es lo habitual al momento de generar modelos predictivos y dejar que el método fit seleccione el conjunto de datos que va a usar para validar los modelos por defecto, se decidió, hacer la selección de los datos de validación personalmente (véase Ilustración 4.13). De esta forma, se asegura conocer con total exactitud los datos que utiliza Keras para evaluar el modelo.

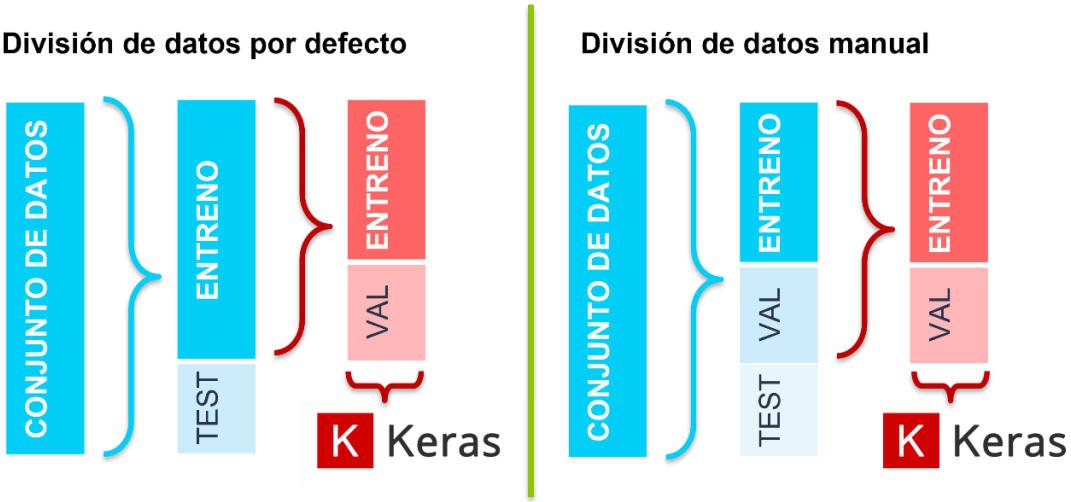


Ilustración 4.13 Funcionamiento datos de entreno en Keras

Una vez divididos los datos, se procedió a normalizarlos. Los modelos darán mejores resultados con los datos escalados.

"Normalizar datos es una técnica que se aplica a un conjunto de datos para reducir su redundancia. El objetivo principal de esta técnica es asociar formas similares a los mismos datos en una única forma de datos." (R. PowerData, 2017)

Para este proyecto, se utilizó un promedio simple. Además, se debe tener en cuenta que la media y la desviación estándar solo deben calcularse utilizando los datos de entrenamiento para que los modelos no tengan acceso a los valores en los conjuntos de validación y prueba (ver ecuación 4.3). (Apache License, Version 2.0 , 2021)

$$valor\ normalizado = \frac{valor\ sin\ normalizar - media\ de\ los\ datos\ de\ entreno}{desviación\ estandar\ de\ los\ datos\ de\ entreno}$$

Ecuación 4.3 Formula para normalizar los datos

A continuación, se creó un modelo de base, uno lineal, uno denso, uno de redes neuronales convolucionales y uno de redes neuronales recurrentes de varios pasos de tiempo, es decir, que generaran predicciones para más de una hora.

Inicialmente, a los modelos se les dio una entrada de 24 horas en el pasado y se configuraron para que hagan predicciones a las siguientes 3 horas. Más adelante en este proyecto, se analizarán los resultados para estos modelos, seleccionándose el mejor y probándose con distintas entradas de tiempo y número de predicciones, como ya se mencionó anteriormente (véase sección 4.7 Selección del modelo).

Antes de empezar a crear los modelos, se definieron algunas clases y métodos que fueron de gran ayuda al momento de entrenarlos.

Para conseguir pasar las entradas al modelo con las horas deseadas y las horas de predicción que se quieren obtener, se le pasaron los datos en forma de ventanas de muestras consecutivas de datos (véase Ilustración 4.14). Estas ventanas permiten indicar el número de pasos de tiempo de entrada y de salida, el tiempo de compensación entre ellos y las características que se utilizan para alimentar el modelo, en este caso, el 'AQI'. (Apache License, Version 2.0 , 2021)

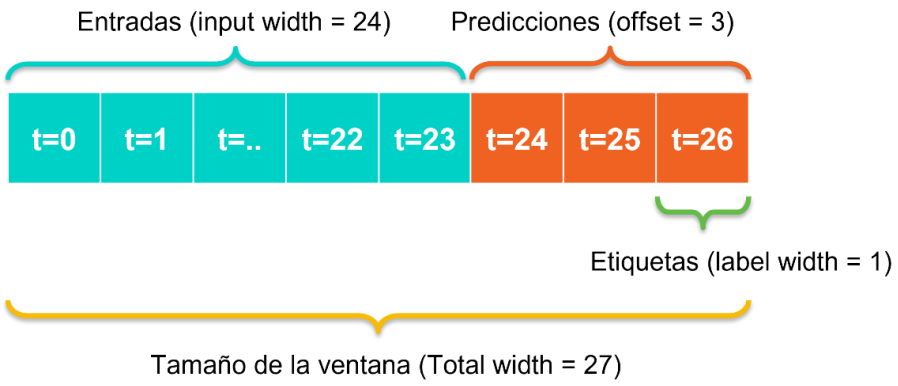


Ilustración 4.14 Ventana de datos

En función de esto, se creó una clase 'WindowGenerator', que incluye en su método '__init__' toda la lógica necesaria para crear los índices de entrada y etiqueta (salidas). (Apache License, Version 2.0 , 2021)

Dada una lista de entradas consecutivas, se creó el método split_window que convierte los datos en una ventana de entradas y una ventana de etiquetas (véase Ilustración 4.15) (Apache License, Version 2.0 , 2021)

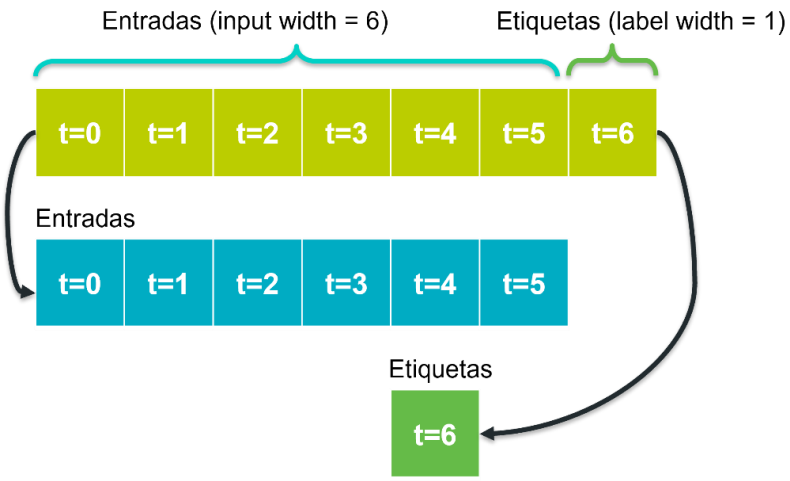


Ilustración 4.15 Método split_window

Luego, se creó el método make_dataset que toma una serie de tiempo DataFrame y lo convierte en un tf.data.Dataset de (entrada, salida) usando la función 'preprocessing.timeseries_dataset_from_array'. (Apache License, Version 2.0 , 2021)

Finalmente, se creó el método compile_and_fit que se utilizó para compilar y entrenar los distintos modelos que se evaluaron.

Algoritmo 4.1 Método compile_and_fit (Apache License, Version 2.0 , 2021)

```
MAX_EPOCHS = 20

def compile_and_fit(model, window, patience=2):
    # The patience parameter is the amount of epochs to check
    for improvement
    early_stopping = tf.keras.callbacks.EarlyStopping(
                                            monitor='val_loss',
                                             patience=patience,
                                                    mode='min')

    model.compile(loss=tf.losses.MeanAbsoluteError(),
                        optimizer=tf.optimizers.Adam(),
                    metrics=[tf.metrics.MeanAbsoluteError()])

    history = model.fit(window.train, epochs=MAX_EPOCHS,
                              validation_data=window.val,
                              callbacks=[early_stopping])

return history
```

A continuación, se explicará con detalle cada una de las partes del método compile_and_fit. Este método utiliza la librería de Keras conjunto con Tensorflow para establecer el 'early_stoping', compilar y entrenar el modelo.

Los modelos se entrenaron utilizando una devolución de llamada EarlyStopping (variable early_stopping), que comprueba la condición de entrenamiento para cada época. Si transcurre una cantidad determinada de épocas sin mostrar mejoría, entonces detiene automáticamente el entrenamiento.

Tabla 4.7 Método compile_and_fit( ) (Abadi, y otros, API Tensorflow Core v2.5.0 Python, 2021)

Método	Argumento	Descripción
tf.keras.callbacks.EarlyStopping	monitor='val_loss'	Se le asigna el conjunto de datos a observar. En este caso, como se explicó anteriormente, los datos de validación son los que se utiliza el método fit para evaluar el modelo.
	patience= 2	Número de épocas sin mejora después de las cuales se detendrá el entrenamiento.
	mode='min'	En el modo 'min' el entrenamiento se detendrá cuando la cantidad monitoreada haya dejado de disminuir.
tf.keras.Model.compile	loss=tf.losses.MeanAbsoluteError()	Nombre de la función de perdida que se utilizará. Se utiliza el error absoluto medio que calcula la media de la diferencia absoluta entre las etiquetas y las predicciones.
	optimizer=tf.optimizers.Adam()	Optimizador que se utiliza. Se usa el optimizador de Adam que implementa el algoritmo de Adam.
	metrics=[tf.metrics.MeanAbsoluteError()])	Indica la métrica que el modelo evaluará durante el entrenamiento y las pruebas con los datos de validación.
tf.keras.Model.fit	x=window.train	Indica los datos de entreno.
	epochs=MAX_EPOCHS	El número máximo de épocas permitido.
	validation_data=window.val,	Datos que se utilizaran para la validación del modelo
	callbacks=[early_stopping]	Lista de devoluciones de llamada para aplicar durante el entrenamiento. En este caso tras cada época hace una llamada EarlyStopping que le indicará si debe detener el entreno.

Más adelante se indicó con el método 'compile' el tipo de función de perdida, optimizador y métricas para el modelo y con el método fit se entrenó el modelo. El optimizador, el tipo de perdida y las métricas a utilizar no se eligieron de forma aleatoria. Cada uno de estos valores se seleccionó por un motivo.

El optimizador que se eligió fue el Adam (Adaptative Moment Estimation, en español Estimación Adaptativa de Momentos). Este es uno de los optimizadores más utilizados en la actualidad y, tras probar otros optimizadores se comprobó que este era el que mejor se adaptaba al conjunto de datos.

El tipo de perdida que se seleccionó es el error medio absoluto, ya que es el tipo de función de pérdida que se utiliza en modelos de series de tiempo. Lo mismo se puede extrapolar para la métrica seleccionada.

Se registró la precisión de entrenamiento y la validación en el objeto "history" para poder acceder a las estadísticas generadas por el método fit cuando entrenaba y evaluaba el modelo.

A continuación, se explicarán los distintos modelos creados durante este proyecto. Se debe tener en cuenta que estos modelos utilizaran una ventana de 24 entradas, 3 predicciones y 3 etiquetas, para generar 3 horas de predicciones a partir de 24 horas de pasado.

4.6.2.1 Modelo base

El primer modelo que se creó fue un modelo base que copia el último valor de entrada y lo replica para las salidas. Esta es una lógica aceptable debido a que el AQI normalmente cambia de forma lenta durante el tiempo.

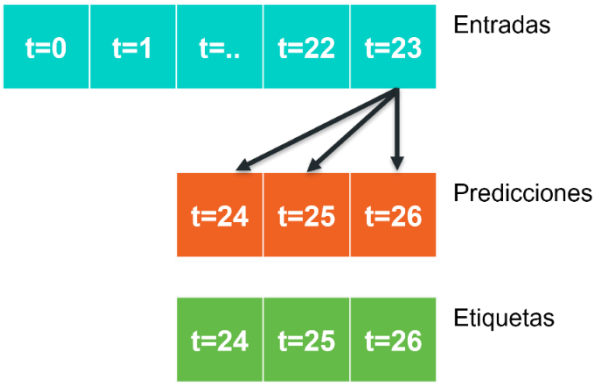


Ilustración 4.16 Ventana modelo base

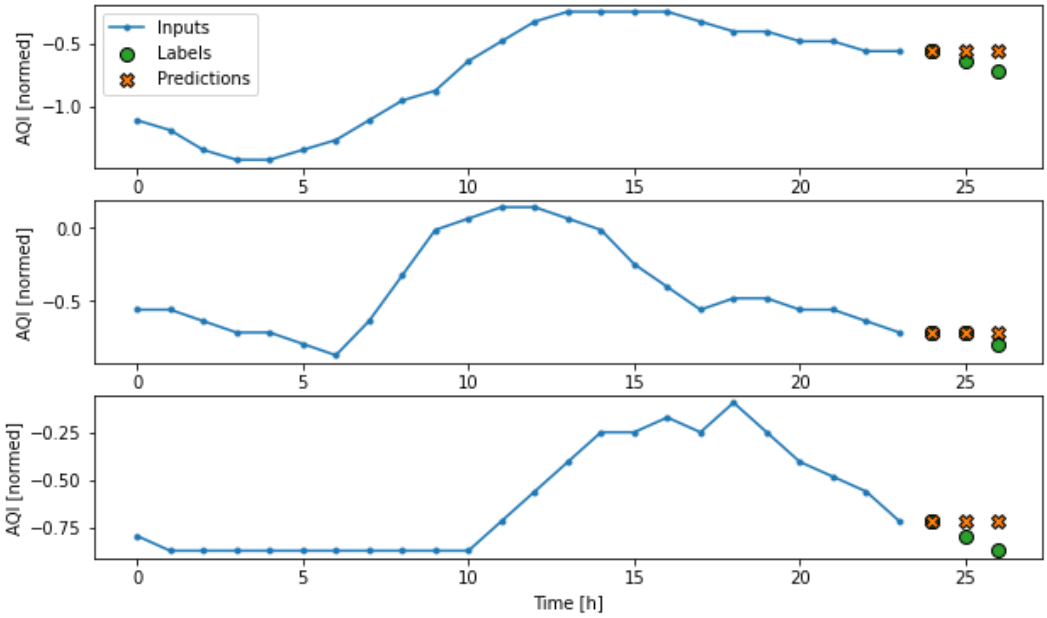
Algoritmo 4.2 Modelo base (Apache License, Version 2.0 , 2021)

```
class MultiStepLastBaseline(tf.keras.Model):
  def call(self, inputs):
    return tf.tile(inputs[:, -1:, :], [1, OUT_STEPS, 1])
last_baseline = MultiStepLastBaseline()
last_baseline.compile(loss=tf.losses.MeanSquaredError(),
                      metrics=[tf.metrics.MeanAbsoluteError()])
```

Básicamente, el algoritmo 4.2 crea una clase que, cuando se llama, retorna el último de los valores registrados en los datos de entrada y lo compila generando estadísticas para los siguientes 3 pasos de tiempo (véase Ilustración 4.16)

Se obtuvo una pérdida de 0.1922 y un error medio absoluto de 0.2185, siendo estos resultados medianamente aceptables.

Creando tres ventanas de datos distintas se graficaron las predicciones del modelo correspondientes a cada ventana contra los verdaderos valores. Este modelo haría predicciones de la siguiente forma (ver Ilustración 4.16): dadas 24 horas de entrada predice las siguientes 3 horas. Se debe tener en cuenta que en estas gráficas los valores para el AQI están normalizados (véase Gráfica 4.6).



Gráfica 4.6 Predicciones modelo base

4.6.2.2 Modelo lineal

Un modelo lineal simple funciona mejor que cualquier modelo base, pero tiene poca potencia. Esto se debe a que tendrá que hacer varias predicciones de tiempo basado en el último paso de éste (véase Ilustración 4.17), por lo que solo captura una porción de baja dimensión del comportamiento, probablemente basándose en la hora del día y la época del año. (Apache License, Version 2.0 , 2021)

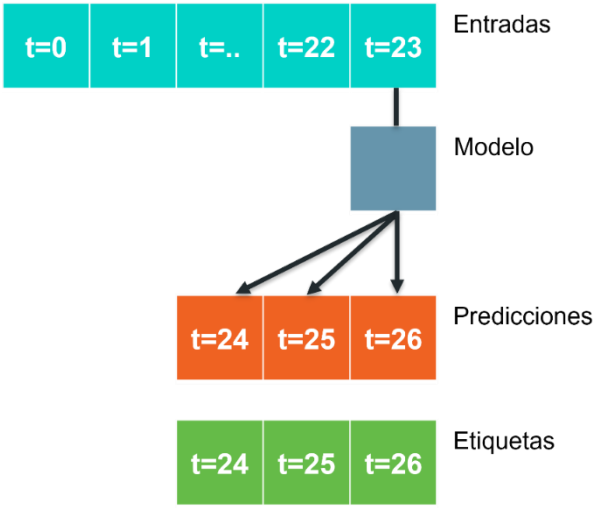


Ilustración 4.17 Ventana modelo lineal

Algoritmo 4.3 Modelo lineal (Apache License, Version 2.0 , 2021)

```
multi_linear_model = tf.keras.Sequential([
    # Take the last time-step.
    # Shape [batch, time, features] => [batch, 1, features]
    tf.keras.layers.Lambda(lambda x: x[:, -1:, :]),

    # Shape => [batch, 1, out_steps*features]
    tf.keras.layers.Dense(OUT_STEPS*num_features,
                          kernel_initializer=tf.initializers.zeros()),

    # Shape => [batch, out_steps, features]
      tf.keras.layers.Reshape([OUT_STEPS, num_features])
])
```

Para este y el resto de los modelos se utiliza tf.keras.Sequential, que permite agrupar en un modelo una pila lineal de capas.

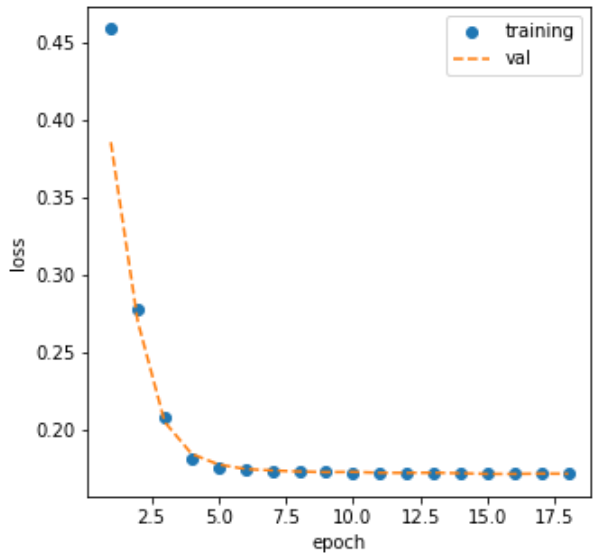
La capa de entrada es la capa Lambda, que sirve para envolver expresiones arbitrarias. Básicamente consiste en tomar el último valor del conjunto de datos de entrada, que será a partir del cual se harán las predicciones (véase Ilustración 4.17).

La segunda capa (en este caso es la única capa oculta), es una capa de tipo Dense, que es una capa de red neuronal regular densamente conectada. A esta se le indica un número de neuronas igual a los pasos de tiempo que se quieren predecir multiplicado por las columnas (características) del conjunto de datos. Esto no se hace por nada en particular, solo sirve para tener una referencia. Lo verdaderamente importante es evaluar si con este número de neuronas el modelo, en lugar de generalizar, particulariza los resultados (overfitting). Esto ocurriría con un número excesivo de neuronas. La otra posibilidad es que el modelo no resuelva el problema (underfitting), lo que indicaría que el número de neuronas es demasiado pequeño. El argumento kernel_initializer sirve para inicializar las salidas de la capa a ceros.

La capa de salida es una capa de tipo Reshape, que sirve para devolver en las dimensiones indicadas las predicciones, es decir, devolverá los 3 pasos de tiempo y las características para cada uno.

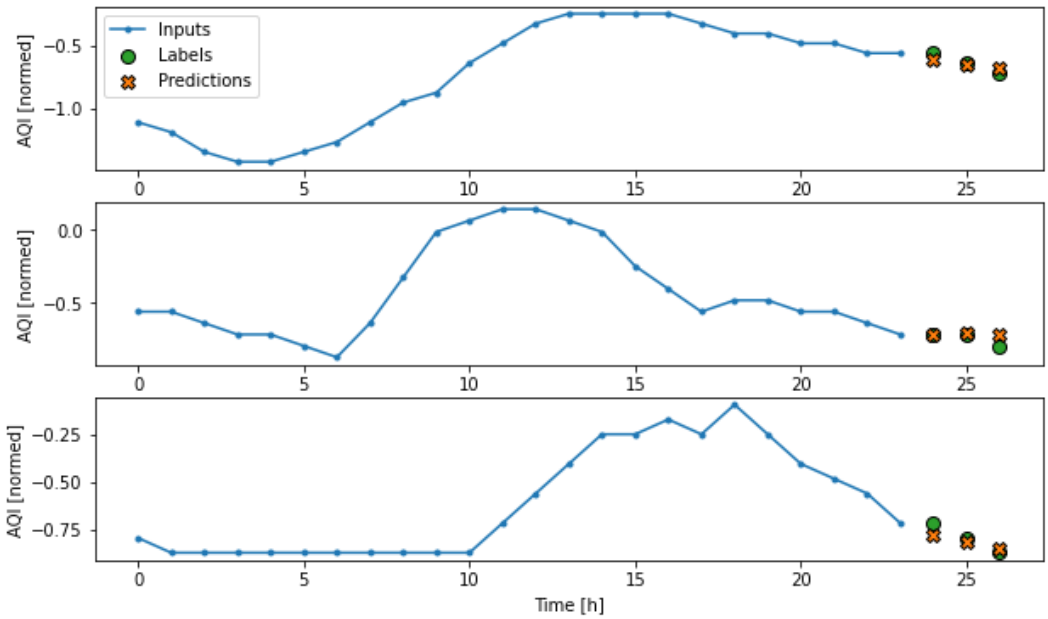
Observando los resultados obtenidos la perdida fue de 0.1714 y el error medio absoluto de 0.1714 estos resultados fueron considerablemente inferiores al modelo anterior. A diferencia del modelo de base, el modelo lineal da mejores resultados, teniendo una perdida y un error medio absoluto considerablemente más bajo.

Graficando la pérdida (loss) para cada una de las épocas (epoch) en la que se entrenó el modelo para los datos de validación y de entreno, se puede observar que no existe Overfitting ni Underfitting (véase Gráfica 4.7). Este tipo de gráfica será utilizada para este modelo y los siguientes, y se usa el conjunto de estadísticas que devuelve el método fit. Estas estadísticas incluyen el valor de perdida para los datos de entreno y de perdida para los datos validación que son los utilizados por el método para probar el modelo tras cada época.



Gráfica 4.7 Modelo lineal: perdida - época

De igual manera que como se hizo para el modelo de base, se grafican 3 ventanas (se utilizarán las mismas ventanas para todos los modelos) en las que se puede observar un ejemplo visual de los resultados del modelo en cuestión (véase Gráfica 4.8)



Gráfica 4.8 Predicciones modelo lineal

4.6.2.3 Modelo denso

Posteriormente, se realizó un modelo similar al lineal, pero en esta ocasión se agregó entre la entrada y la salida del modelo una capa 'layers.Dense' para darle más

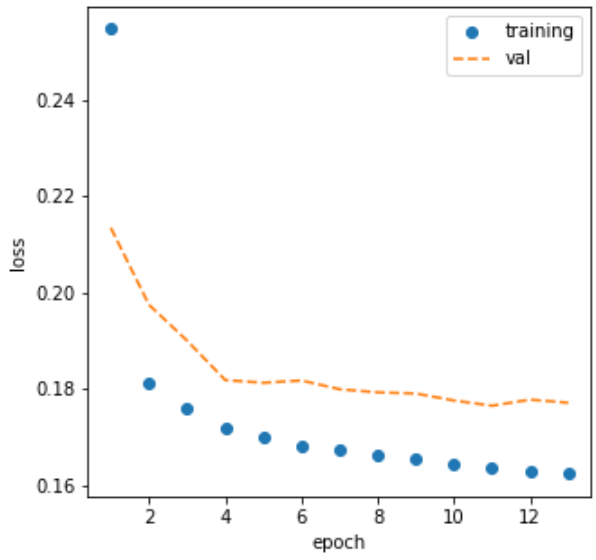
potencia al modelo. El modelo se basa en un solo paso de tiempo de entrada, igual que el modelo lineal (véase Ilustración 4.17).

Algoritmo 4.4 Modelo denso (Apache License, Version 2.0 , 2021)

```
multi_dense_model = tf.keras.Sequential([
    # Take the last time step.
    # Shape [batch, time, features] => [batch, 1, features]
    tf.keras.layers.Lambda(lambda x: x[:, -1:, :]),

    # Shape => [batch, 1, dense_units]
    tf.keras.layers.Dense(256, activation='relu'),

    # Shape => [batch, out_steps*features]
    tf.keras.layers.Dense(OUT_STEPS*num_features,
                  kernel_initializer=tf.initializers.zeros()),

    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([OUT_STEPS, num_features])
])
```

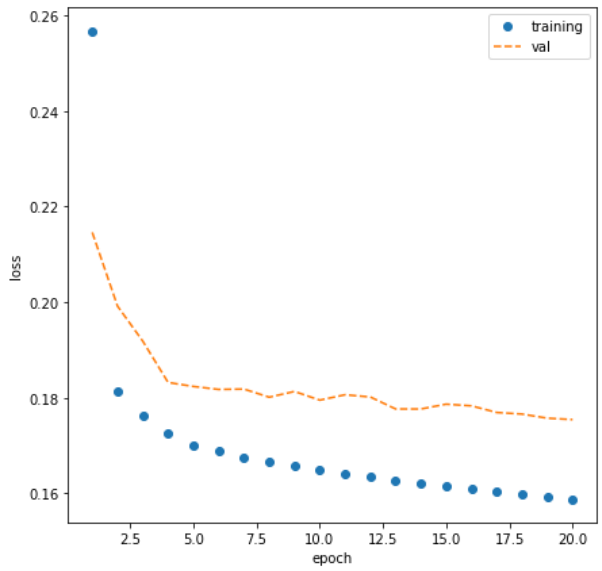
Además de la capa layers.Dense, se añade la capa tf.keras.layers.Dense(256, activation='relu'), de esta forma, se tendrían dos capas ocultas. La capa nueva cuenta con 256 neuronas y una función de activación de tipo 'relu', que transforma los valores introducidos de tal forma que anula los valores negativos y deja los positivos tal y como entran.

Los resultados obtenidos fueron 0.1771 para la pérdida y 0.1771 para el error medio absoluto, siendo estos similares a los del modelo lineal. Observando la gráfica de perdida - época (véase Gráfica 4.9) se observa, en primer lugar, que el modelo iteró durante menos épocas de lo que lo hizo para el modelo lineal, por lo que se deduce que consiguió el punto mínimo de perdida mucho más rápido.



Gráfica 4.9 Modelo denso: perdida - época

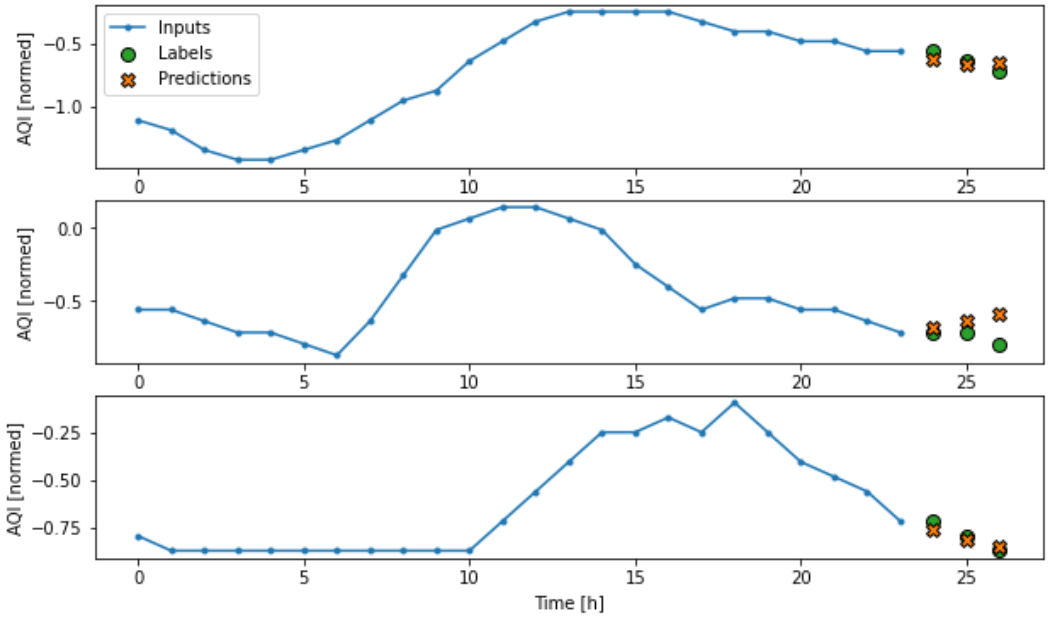
Al añadir la nueva capa con más neuronas se puede observar que, para los datos de validación hay un poco de overfitting. Es decir, al momento de hacer predicciones el modelo no generaliza tan bien como lo hacía para el modelo lineal. Para ver esto con más claridad, se ha entrado el modelo para un mayor número de épocas. (véase Gráfica 4.10)



Gráfica 4.10 Modelo denso: perdida - época (más iteraciones)

Se puede observar en la Gráfica 4.10 como a partir de la época 5 aproximadamente para los datos de validación, el modelo deja de generalizar y apenas se nota mejoría en las predicciones al terminar el entreno.

Siguiendo la línea de trabajo, se graficaron algunas predicciones hechas por el modelo con la misma ventana de datos (véase Gráfica 4.11).



Gráfica 4.11 Predicciones modelo denso

4.6.2.4 Modelo red neuronal convolucional (CNN)

"Un modelo convolucional hace predicciones basadas en un historial de ancho fijo, lo que puede conducir a un mejor rendimiento que el modelo denso, ya que se puede ver cómo cambian las cosas con el tiempo". (Apache License, Version 2.0 , 2021)

En este caso, el historial de ancho fijo será de 3 pasos de tiempo.

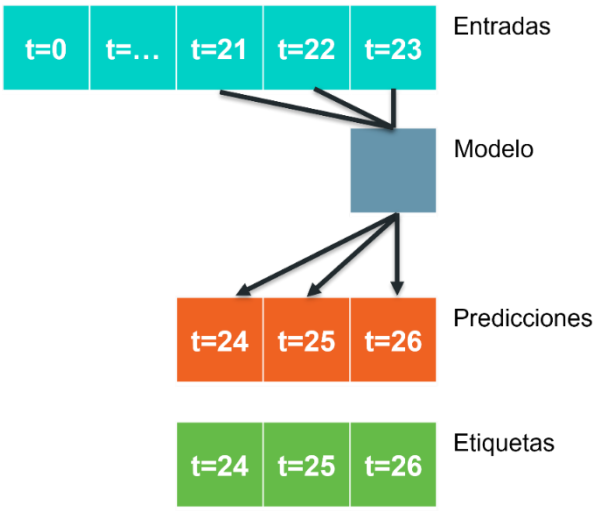


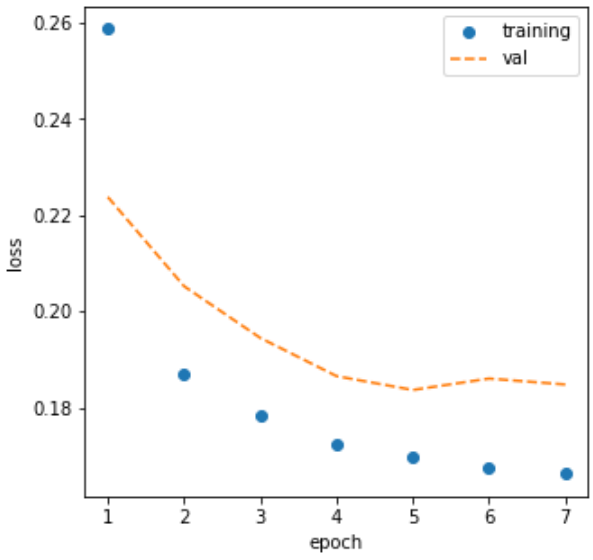
Ilustración 4.18 Ventana modelo CNN

Algoritmo 4.5 Modelo CNN (Apache License, Version 2.0 , 2021)

```
CONV_WIDTH = 3

multi_conv_model = tf.keras.Sequential([
    # Shape[batch,time,features]=>[batch, CONV_WIDTH,features]
    tf.keras.layers.Lambda(lambda x: x[:, -CONV_WIDTH:, :]),

    # Shape => [batch, 1, conv_units]
    tf.keras.layers.Conv1D(256, activation='relu',
                           kernel_size=(CONV_WIDTH)),

    # Shape => [batch, 1,  out_steps*features]
    tf.keras.layers.Dense(OUT_STEPS*num_features,
                  kernel_initializer=tf.initializers.zeros()),

    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([OUT_STEPS, num_features])
])
```

Este modelo es muy similar al modelo denso, pero la primera capa (capa de entrada) en esta ocasión, en lugar tomar el ultimo valor, utiliza los valores definidos en el ancho fijo (véase Ilustración 4.18). La segunda capa, en lugar de tratarse de una capa de tipo Dense es una capa de tipo Conv1D, a la que se le indica el mismo número de neuronas y la misma función de activación, y se le añade el kernel_size que es el ancho fijo que se definió previamente. Las otras dos capas funcionan de la misma forma que en los modelos anteriores.

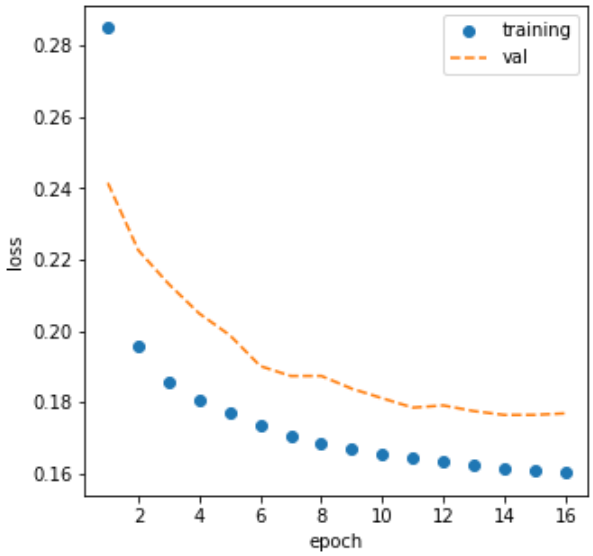
Observando los resultados para la pérdida (0.1848) y el error medio absoluto (0.1848), no existe una gran mejora entre este modelo y el denso. Si se observa la gráfica de pérdida, ésta es bastante similar a la del modelo anterior (véase Gráfica 4.12).



Gráfica 4.12 Modelo CNN: perdida - época

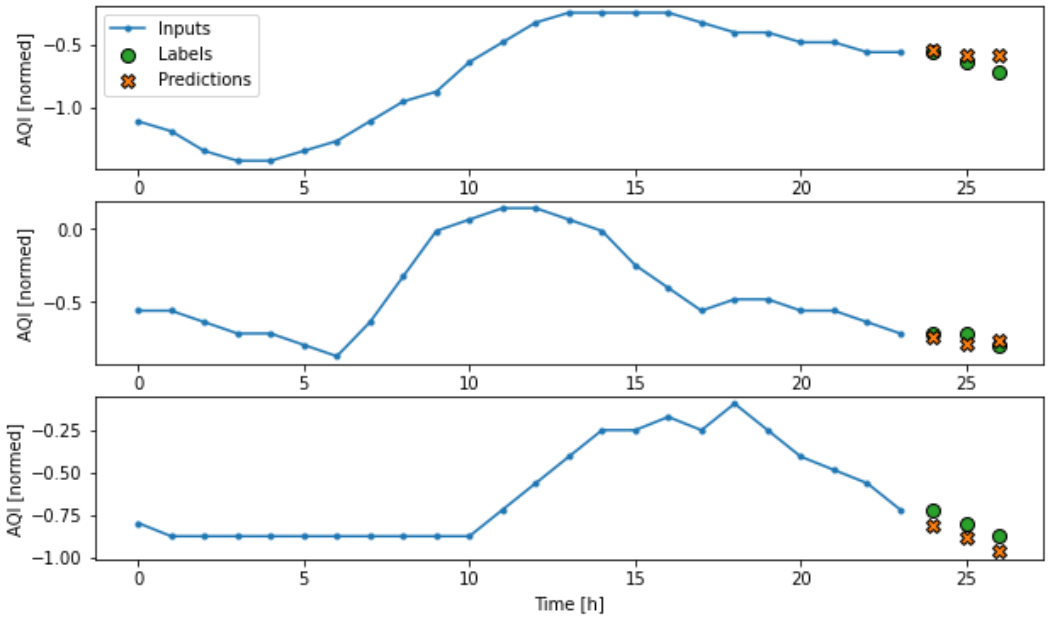
Tras analizar la Gráfica 4.12, se observa que los datos de validación tienen un poco de overfitting, por lo que reduciendo el número de neuronas se podrían mejorar los resultados. Pero, se debe tener precaución de no reducir en extremo el número de neuronas, ya que podría darse que el modelo simplemente no aprendiera (underfitting).

Ejecutando el mismo modelo, pero esta vez con la mitad de las neuronas en la capa convolucional, se obtienen mejores resultados y se reduce el overfitting (véase Gráfica 4.13). Obteniendo una pérdida de 0.1769 y un error medio absoluto de 0.1769.



Gráfica 4.13 Modelo CNN: perdida - época (con menos neuronas)

Al igual que como se hizo para los apartados anteriores, se incluye las gráficas donde se puede visualizar un ejemplo de las predicciones hechas por el modelo que mejor resultados obtuvo (véase Gráfica 4.14).



Gráfica 4.14 Resultados modelo CNN

4.6.2.5 Modelo de red neuronal recurrente (RNN)

Un modelo recurrente puede aprender a usar un largo historial de entradas, si es relevante para las predicciones que hace el modelo. Aquí, el modelo acumulará el estado interno durante las horas indicadas en input_width, antes de realizar una única predicción para las próximas horas que se indicaron en OUT_STEPS. (Apache License, Version 2.0 , 2021)

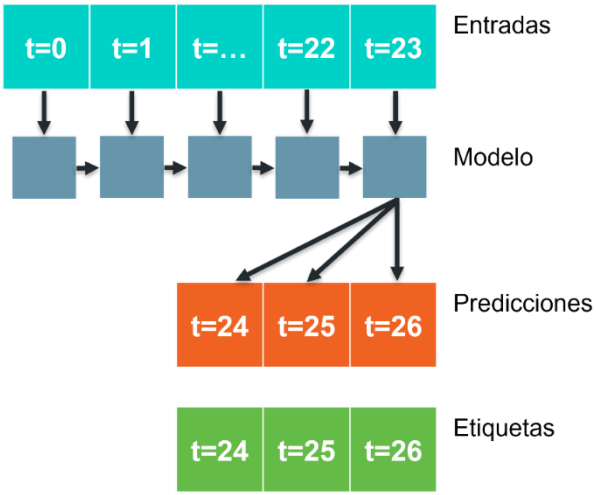


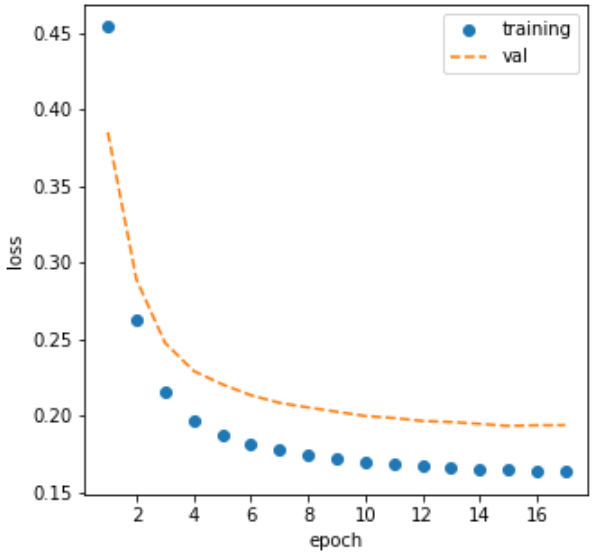
Ilustración 4.19 Ventana modelo RNN

Algoritmo 4.6 Modelo RNN (Apache License, Version 2.0 , 2021)

```
multi_lstm_model = tf.keras.Sequential([
    # Shape [batch, time, features] => [batch, lstm_units]
    # Adding more `lstm_units` just overfits more quickly.
    tf.keras.layers.LSTM(32, return_sequences=False),

    # Shape => [batch, out_steps*features]
    tf.keras.layers.Dense(OUT_STEPS*num_features,
            kernel_initializer=tf.initializers.zeros()),

    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([OUT_STEPS, num_features])
])
```

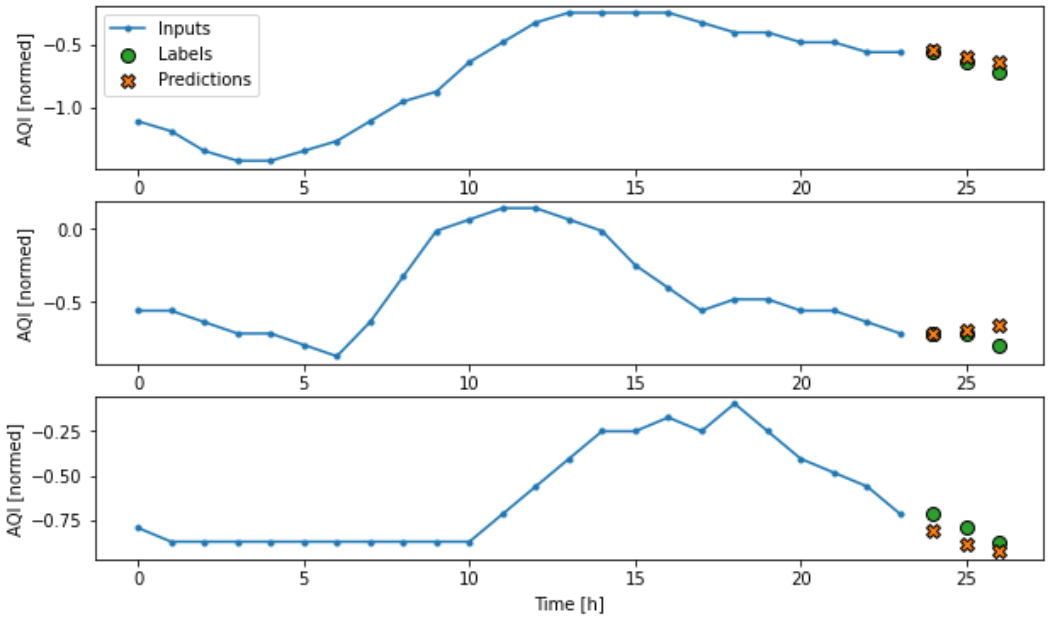
La capa de entrada de este algoritmo es una capa de tipo LSTM. Como este modelo solo necesita producir una salida en el último paso de tiempo, (véase Ilustración 4.19), se configura return_sequences=False y se le asigna un total de 64 neuronas. Las siguientes dos capas del algoritmo funcionan de la misma forma que en el modelo lineal.

Observando los resultados obtenidos, este tuvo una pérdida de 0.1938 y un error medio absoluto de 0.1938. Si se observa el comportamiento de los datos en la gráfica de perdida (véase Gráfica 4.15), se deduce que el modelo no tiene ni overfitting ni underfitting, es decir, el modelo generaliza bien las predicciones.



Gráfica 4.15 Modelo RNN: perdida - época

Se puede observar en la Gráfica 4.16 algunos ejemplos de predicciones obtenidas por el modelo.

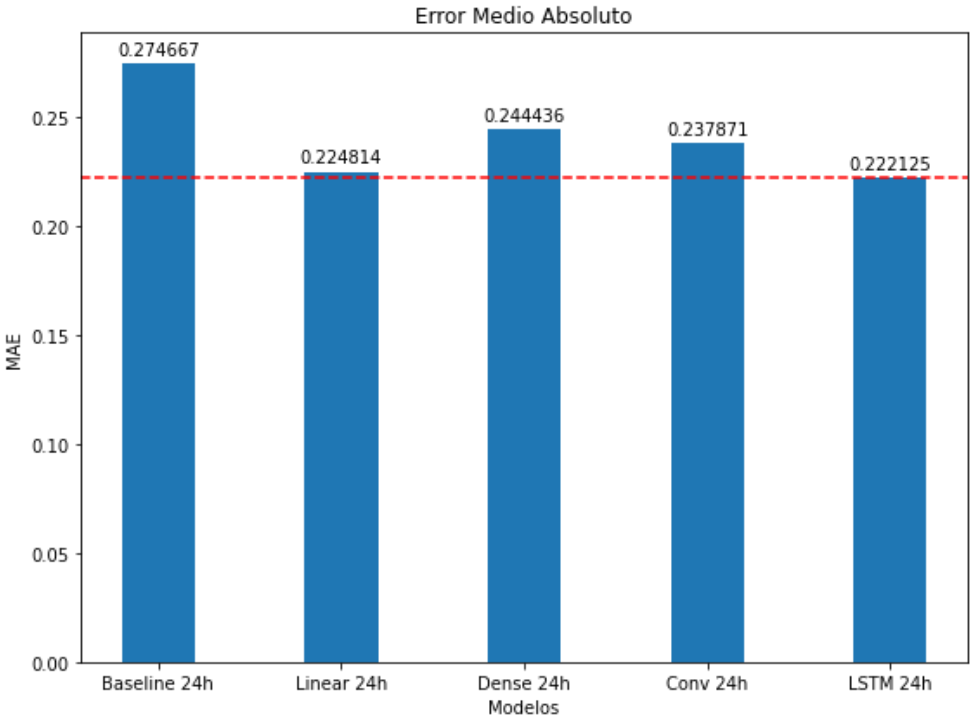


Gráfica 4.16 Resultados modelo RNN

4.7 Selección del modelo

Una vez creados los distintos modelos de predicción, se procedió a evaluar y comparar con los datos de prueba el error medio absoluto de cada uno. Para ello, a medida que se iban creando los modelos, se evaluaban con los datos de prueba y se guardaban en un archivo los resultados obtenidos.

Observando la Gráfica 4.17, se pueden ver los resultados obtenidos para cada uno de los modelos. En el apartado anterior, se aseguró que los modelos que se guardaron no tuviesen sobreajuste o subajuste, conservándose así, para el modelo convolucional, el modelo probado con menos neuronas, ya que fue éste el que dio mejores resultados. Además, gracias a esto, se tiene la certeza de que todos los modelos estudiados a continuación generalizan sus predicciones (no existe ni sobreajuste ni subajuste). Por tanto, la selección de los modelos en este apartado se centrará en observar solamente el error medio absoluto.



Gráfica 4.17 Error Medio Absoluto de los modelos

Dados los 5 cinco tipos de modelos desarrollados en este proyecto, se procede a analizarlos para seleccionar el modelo definitivo que se utilizará para hacer las predicciones.

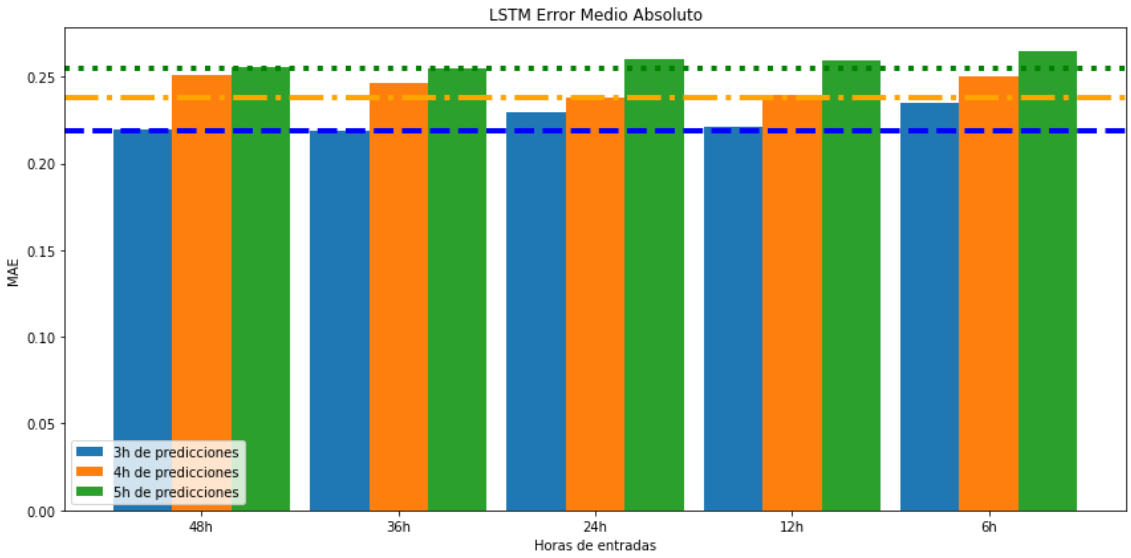
En primer lugar, se descarta automáticamente el modelo de base. Este modelo se hizo a modo demostrativo de la evolución del AQI a lo largo del tiempo y se basa en copiar la última entrada del conjunto de datos, por lo que es de esperarse que en comparación con los otros modelos no obtuviese tan buenos resultados.

Observando el modelo denso y el convolucional, aunque estos obtuvieron muy buenos resultados con los datos de entreno y validación, al evaluarlo con el conjunto de datos que se reservó para realizar las pruebas, éstos mostraron un error ligeramente mayor que el modelo lineal y el RNN.

Entre el modelo lineal y el LSTM se puede ver que ambos obtuvieron resultados similares. Aun así, finalmente se decidió tomar el modelo LSTM, porque sumado a que da mejores resultados, es un modelo más complejo que no solo utiliza la última entrada del conjunto de datos (como sucede con el modelo lineal), sino que toma en cuenta todas las entradas, actualizando el modelo con cada una de éstas. Esto

lo convierte en un modelo mucho más potente que un modelo lineal (véase Ilustración 4.19).

Una vez seleccionado el modelo LSTM, se probó usando distintas ventanas, con entradas que varían desde 48 hasta 6 horas y con salidas de 3, 4 y 5 horas. Posteriormente, se graficaron los resultados obtenidos para cada una de las ventanas probadas (véase Gráfica 4.18).



Gráfica 4.18 MAE del modelo RNN con distintas ventanas de tiempo

Analizando la Gráfica 4.18 donde se ve reflejado el error medio absoluto para las distintas ventanas probadas, se puede observar con gran claridad que los mejores resultados se obtienen con predicciones de 3 horas. Esto es de esperarse debido a que, mientras menos predicciones se hagan, menos probabilidades tiene el modelo de fallar. Por lo que, analizando los resultados para las ventanas con predicciones de 3 horas, gracias a la línea horizontal azul que marca el borde en el que se encuentra el menor de los errores para estas horas de predicción, se deduce que con 48, 36 y 12 horas de entradas se obtienen los mejores resultados, siendo la diferencia entre estos mínima. Entre estos tres valores de entrada, se seleccionó el de 12 horas, motivado en que, a mayor el número de valores de entrada, más lento es el proceso de entreno del modelo. Por tanto, se seleccionó entonces, por cuestiones de optimización, el modelo que devolvía mejores resultados con un menor número de horas.

De esta forma, se selecciona para hacer las predicciones oficiales que se mostrarán en la web el modelo de red neuronal recurrente con 12 horas de entradas y 3 horas de predicciones.

4.8 Integración de las partes y página web

Llegados a este punto, se cuenta con: los métodos necesarios para recolectar los datos históricos y en tiempo real, el análisis de los datos que se usaron para entrenar los modelos, un conjunto de modelos de predicción y el modelo de predicción de los que se obtuvo los mejores resultados. Cada una de estas partes se hicieron de forma independiente y consecutiva.

Finalmente, para poder crear una página web funcional donde se muestren las predicciones en tiempo real del modelo seleccionado, se necesitó hacer algunas modificaciones a la clase encargada de la recolección de los datos en tiempo real. Posteriormente, se creó una clase que hiciera un procesamiento de los datos recolectados, para que sean compatibles con el tipo de datos que espera el modelo. Luego, se creó una clase que crease la ventana de datos que se le proporcionará al modelo y otra en donde se generasen las predicciones correspondientes y se reentrenase el modelo a medida que se recogen datos nuevos.

El objetivo es mostrar en la página web las predicciones en tiempo real a medida que se van recolectando los datos desde las APIs (véase Ilustración 4.20).

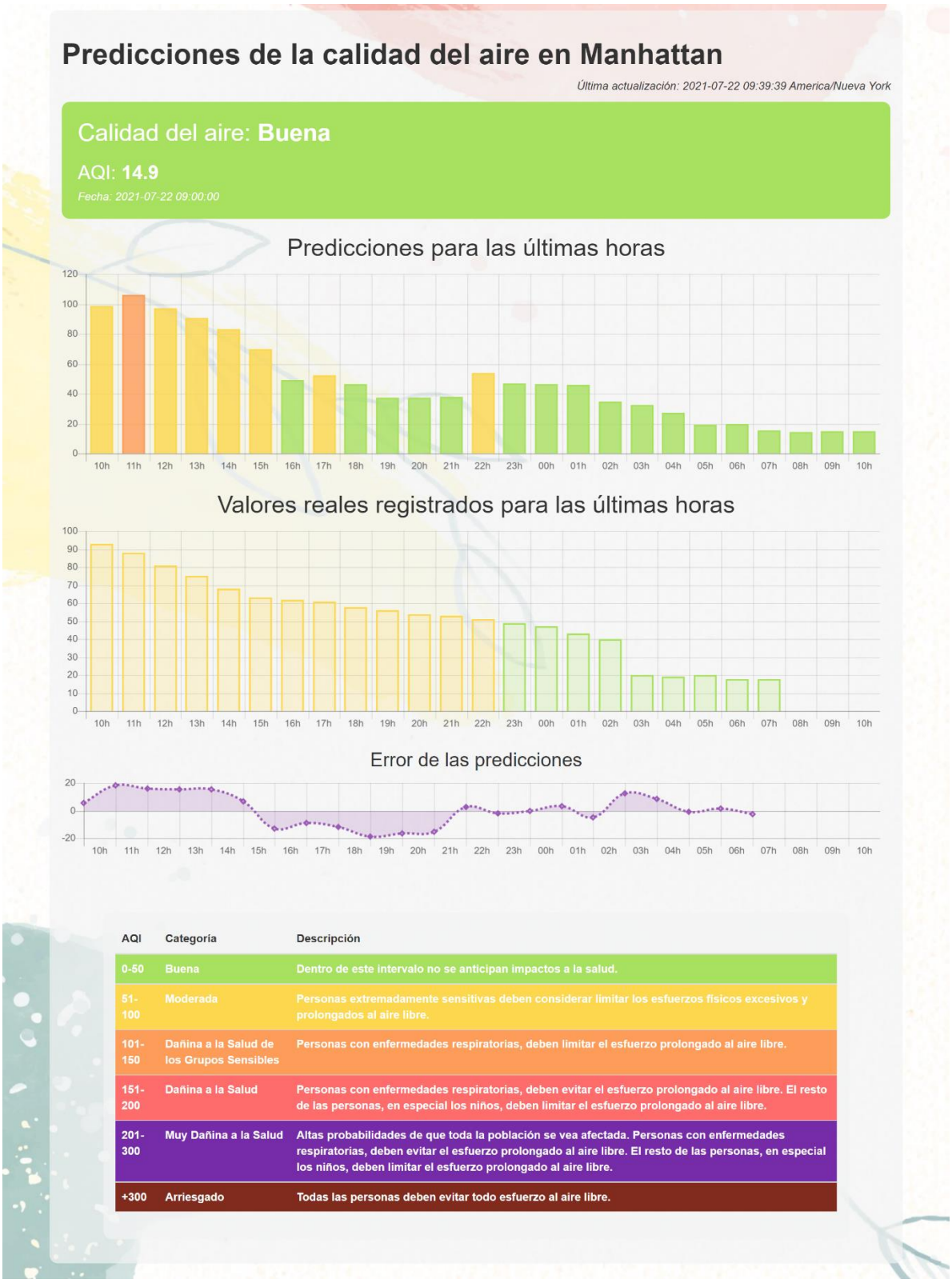


Ilustración 4.20 Página web: Inicio

Para alcanzar esto, se modificó y añadió código adicional a la clase encargada de recoger los datos en tiempo real, para que se detectase el momento en el que todos los datos para una hora estuviesen completos y llamase a la clase que hace el

preprocesamiento de los datos. Una vez guardados los datos para la hora consultada en el formato indicado, se hace una llamada a la clase encargada de hacer las predicciones y reentrena el modelo, indicándole la fecha y hora a partir de la cual hará las predicciones.

Adicionalmente, se tuvo que modificar esta clase para que, en lugar de consultar los datos a partir de la hora en la que se ejecuta, se recolectasen los datos unas horas atrás. Esto se debe a que, para que el modelo pueda hacer las predicciones, es necesario que se le proporcionen 12 horas de entradas y de la forma en la que se consultaban los datos, se tendría que esperar al menos 12 horas para poder empezar a hacer las primeras predicciones. Además, teniendo en cuenta que la frecuencia a la que se actualizan los datos no es constante y puede variar desde minutos a horas, es importante contar con unas horas adicionales de datos recolectados para asegurar que se realicen predicciones desde el momento en el que se ejecuta el programa. A continuación, se va a presentar un ejemplo para aclarar este problema:

Se ejecuta por primera vez el programa que integra la recolección de los datos en tiempo real, el preprocesamiento, el modelo y la página web. En lugar de consultar los datos para la hora actual se consultan a partir de las últimas 12 horas (número de entradas que necesita el modelo). Se debería esperar que el modelo empiece a generar las primeras predicciones, pero debido a que la frecuencia de los datos varía, aún pueden faltar 3 horas de información para completar las 12 horas necesarias, por lo que se tendría que esperar hasta que se completasen los datos para estas primeras 12 horas.

Siguiendo el hilo del ejemplo expuesto, si se cuenta con un modelo que hace predicciones a 3 horas, aumentar el número de horas hacia atrás en la consulta a las APIs, es decir, consultando los datos con las 12 horas de entradas que necesita el modelo para empezar a realizar predicciones más estas 3 horas, se estaría consultando a las APIs las últimas 15 horas. Continuando con el ejemplo, aún faltarían las últimas 3 horas, pero debido a que ahora se están consultando 15 horas de pasado, ya se contaría con 12 horas registradas, por lo que el modelo podría empezar a hacer predicciones de las 2 horas anteriores (que todavía no se han recogido de las APIs) y de la hora actual.

Sin embargo, si la frecuencia de los datos en el momento en el que se ejecuta el programa por primera vez fuese mayor, es decir, de más de 3 horas, se tendría que esperar obligatoriamente a que se registrasen nuevos datos para alcanzar a recolectar los datos de entrada que necesita el modelo. Este problema se escapa del alcance de este proyecto, debido a que la frecuencia de los datos del tráfico varía durante el día y la fuente oficial que proporciona estos datos es la encargada de establecer dicha frecuencia.

Las otras dos clases creadas en este apartado son clases bastante sencillas. La clase de procesamiento de los datos se encarga del preprocesamiento de los datos recolectados, modificando las columnas para que sean compatibles con las entradas del modelo. Además, normaliza los datos y, una vez el modelo hace sus predicciones, se encarga de desnormalizar dichas predicciones para que sean más fáciles de entender.

La clase encargada de generar la ventana de datos que se le proporcionara al modelo es similar a la utilizada para entrenar los distintos modelos que se hicieron en este proyecto. La principal función de esta ventana es proporcionar al modelo las ultimas entradas registradas en el conjunto de datos, ya sea para entrenarlo o para hacer predicciones.

Por otro lado, la clase creada para entrenar y hacer las predicciones funciona de la siguiente manera: una vez se registran datos nuevos, con ayuda de la clase que genera las ventanas de datos, se entrena el modelo (conservando los pesos del modelo en todo momento). Además, genera las nuevas predicciones, llama a la clase de procesamiento para desnormalizar las predicciones y las guarda en un archivo CSV. Finalmente, para generar los datos en la página web se lee el contenido del archivo CSV con las predicciones y se refresca constantemente para mostrar siempre los resultados actualizados.

Adicionalmente, se añadió a la página web dos apartados más. En el primer apartado se puede visualizar una tabla con las predicciones de las últimas 24 horas y los valores reales (véase Ilustración 4.21). El otro apartado es de estadísticas, y permite observar con mayor detalle la diferencia entre los valores para el AQI generados por el modelo y los verdaderos valores registrados.

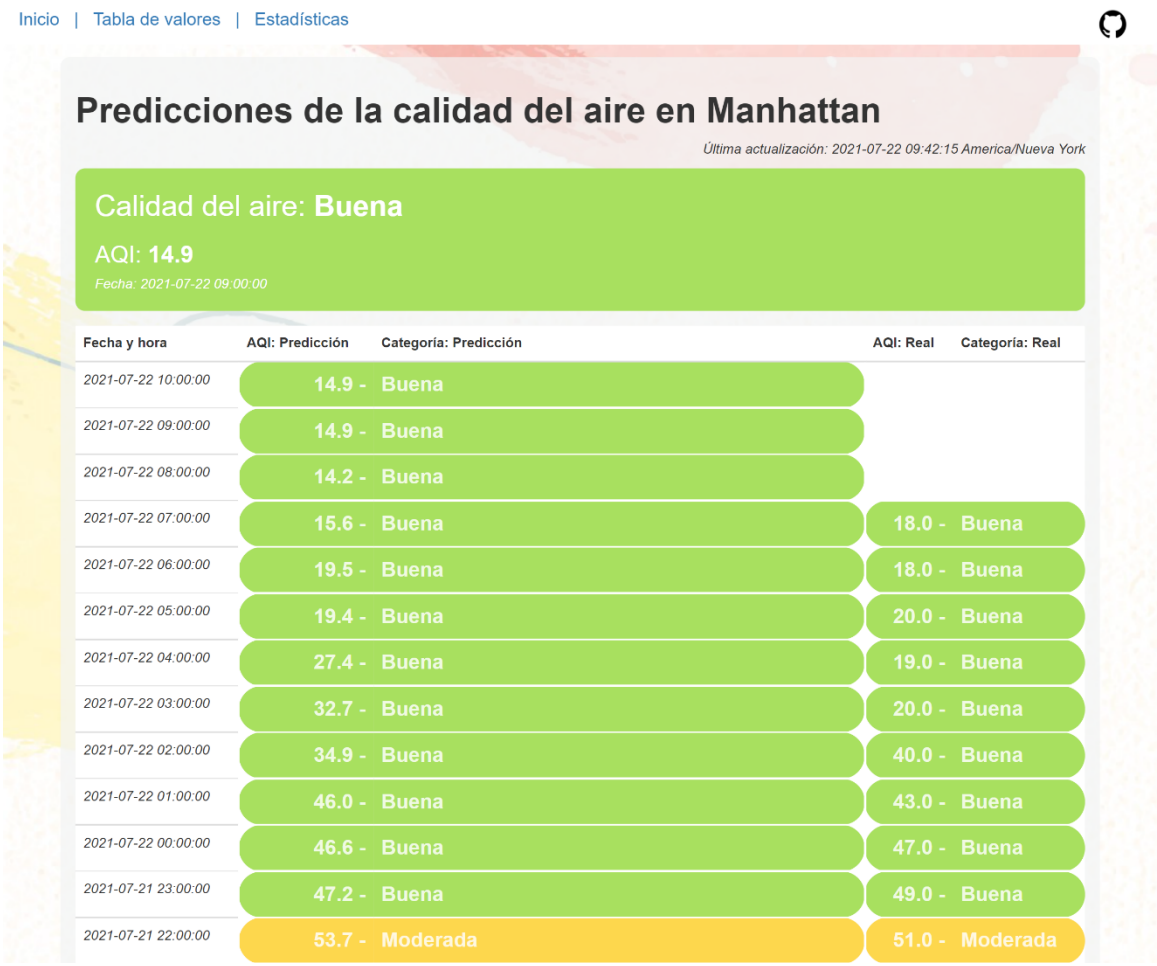


Ilustración 4.21 Página web: Tabla de valores

Para el apartado de estadísticas se añadieron distintos tipos de gráficas. La gráfica principal muestra la última predicción generada por el modelo para cada hora (véase Ilustración 4.22), mientras que las otras muestran las predicciones generadas por el modelo a un paso de tiempo, dos pasos de tiempo, tres pasos de tiempo y todos los pasos de tiempo (véase Ilustración 4.23). Estas gráficas muestran las predicciones que hace el modelo dadas 12 horas históricas para la siguiente hora (un paso de tiempo), las siguientes 2 horas (dos pasos de tiempo), las siguientes 3 horas (tres pasos de tiempo) y para todos los pasos de tiempo. Esto servirá para evaluar qué tan bien genera las predicciones para cada paso de tiempo. Es más probable que el modelo haga una mejor predicción para la hora siguiente a partir de la que se hacen las predicciones que pasadas tres horas.

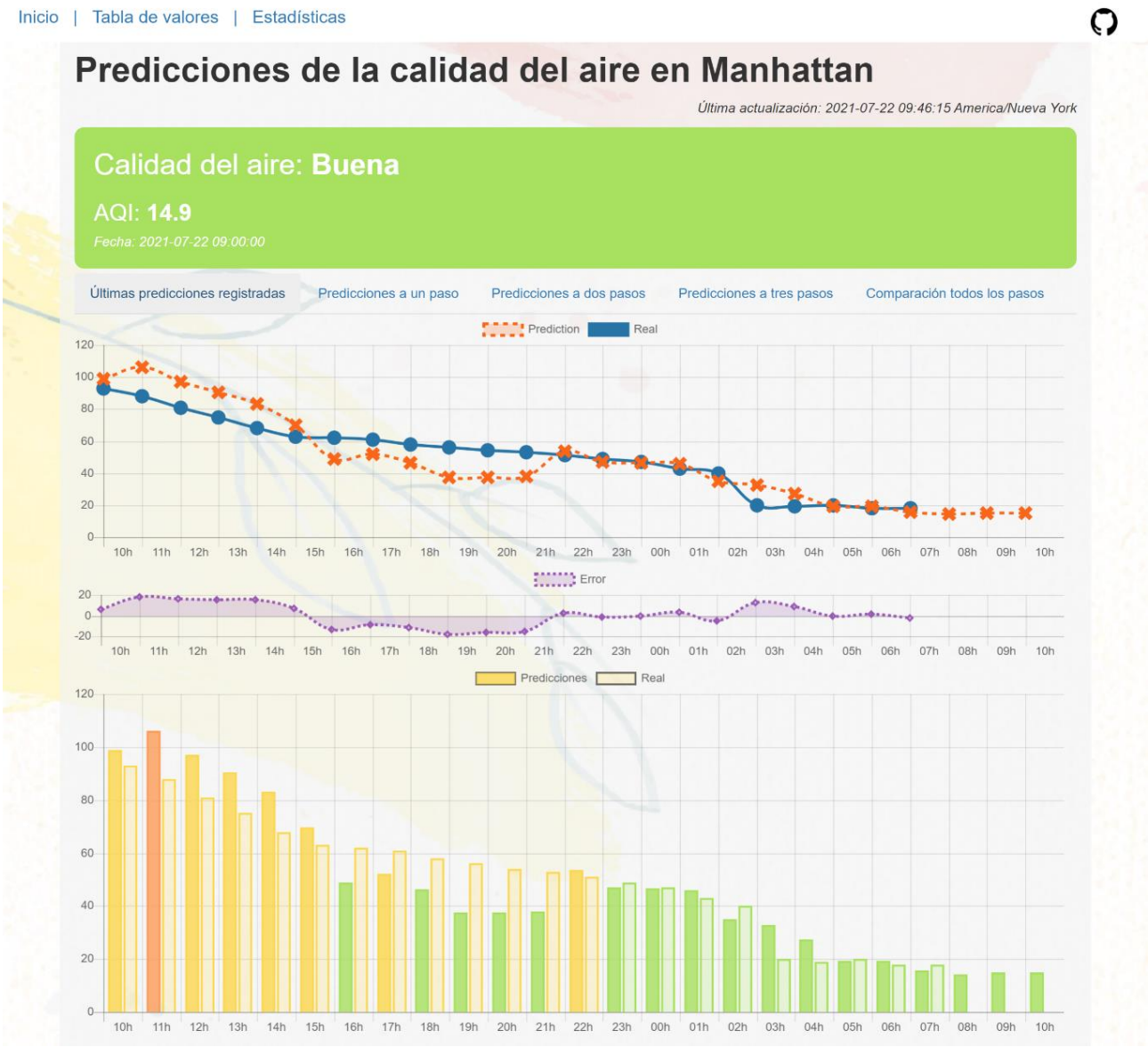


Ilustración 4.22 Página web: Estadísticas "Últimas predicciones registradas"

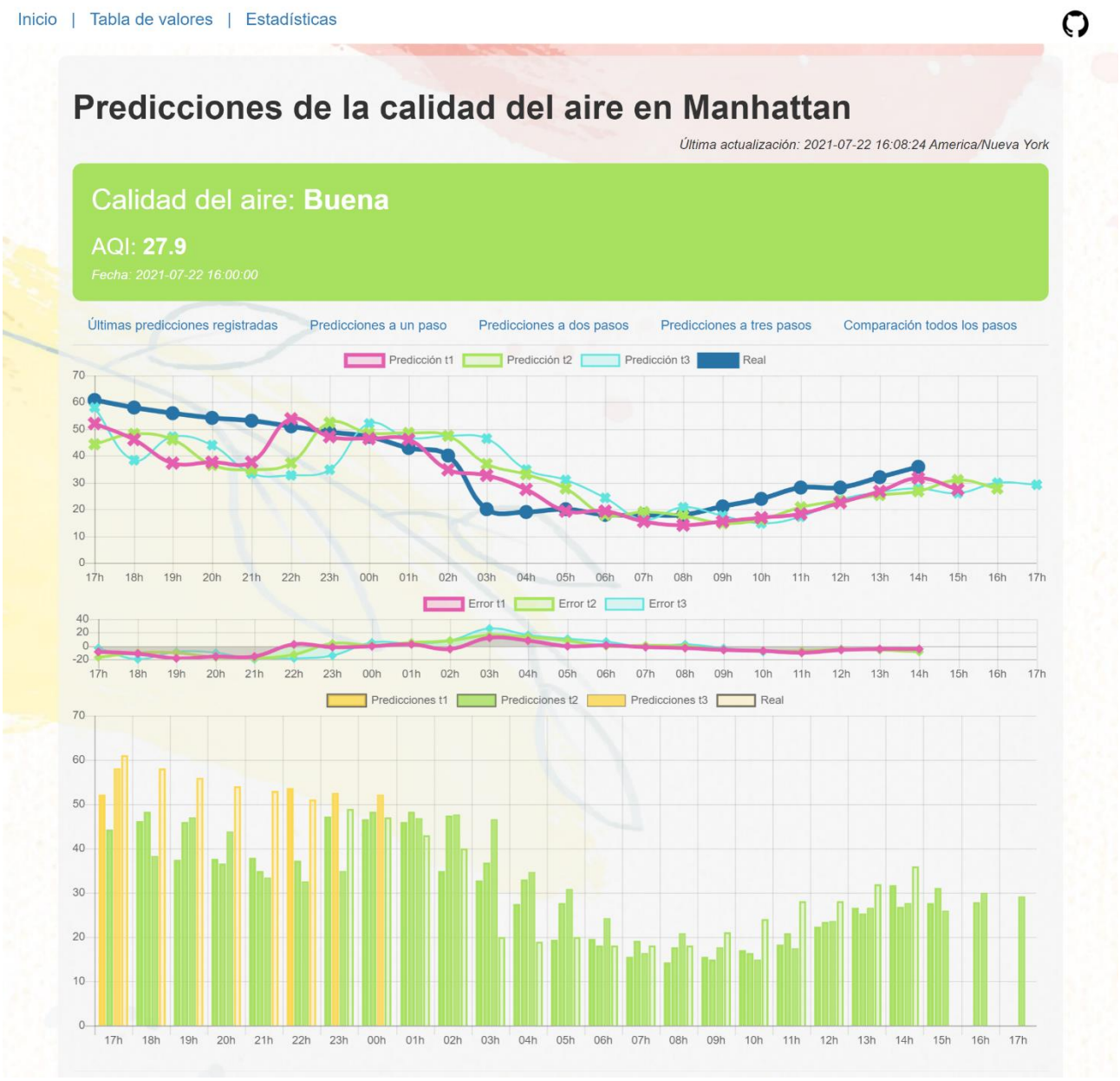


Ilustración 4.23 Página web: Estadísticas "Comparación todos los pasos"

4.9 Ajuste a la planificación y a los objetivos

La planificación inicialmente prevista y los objetivos no fueron modificados durante el desarrollo del proyecto.

- ✓ En primer lugar, se hizo el análisis previo y el estudio de las herramientas necesarias para llevar a cabo el proyecto.
- ✓ En segundo lugar, se realizó la recolección de los datos, se analizó cada uno de éstos y se adaptaron para que tuviesen el formato indicado al momento de hacer los modelos predictivos.
- ✓ En tercer lugar, se plantearon distintos modelos de predicción y se evaluaron para seleccionar el más apropiado para este proyecto.
- ✓ En cuarto lugar, se desarrolló una página web sencilla en la cual se reflejan no solo las predicciones en tiempo real realizadas por el modelo, sino que además cuenta con un apartado que muestra las estadísticas de éste.

Capítulo 5
Conclusiones y trabajo futuro

Durante el desarrollo de este trabajo se comentó la importancia de la calidad del aire y de sus efectos en la salud. Además, se observó que Manhattan es la zona de la ciudad de Nueva York que más afectada se ve por los altos índices de contaminantes en el aire.

Se estudiaron los conceptos de Big Data e Inteligencia Artificial. Este último se explicó en mayor medida, pasando del aprendizaje automático al aprendizaje profundo y definiendo la estructura y conceptos de importancia relacionados con las redes neuronales artificiales.

Para poder generar los modelos predictivos que se llevaron a cabo, se estudiaron distintas fuentes de datos que proporcionasen información acerca del tráfico, la calidad del aire y el clima en la ciudad de Nueva York. Una vez identificadas estas fuentes, se estudiaron los datos proporcionados y se adaptaron para utilizarlos.

Se analizaron los distintos modelos para predecir la calidad del aire en Manhattan (lineal, denso, red neuronal convolucional y red neural recurrente) y se seleccionó el que se consideró más apropiado para generar las predicciones.

Finalmente, se creó una página web en la que se mostrasen las predicciones en tiempo real y se añadió un apartado de estadísticas que permitiese observar y analizar los resultados del modelo utilizado en tiempo real.

Como mejoras para este proyecto se propone:

- Ejecutar el proyecto en tiempo real desde un servidor.
- Hostear la página web para que cualquiera pueda acceder a ella.

Capítulo 6

Bibliografía

Escuela de Ingeniería Informática de la ULPGC. (01 de 07 de 2021). *Objetivos y Competencias del GII*. Obtenido de Universidad de Las Palmas de Gran Canaria: https://www.eii.ulpgc.es/tb_university_ex/?q=objetivos-y-competencias-del-gii

1&1 IONOS España S.L.U. (09 de 10 de 2020). *¿Qué es un plugin y para qué se usa?* Obtenido de IONOS Digitalguide: https://www.ionos.es/digitalguide/servidores/know-how/que-es-un-plugin/

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., & Otros. (08 de 07 de 2021). *API Tensorflow Core v2.5.0 Python*. Obtenido de Tensorflow API: https://www.tensorflow.org/api_docs/python/tf/all_symbols

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., & Otros. (19 de 05 de 2021). *TensorFlow*. Obtenido de https://www.tensorflow.org/

Adobe. (2021). *Información general sobre la interfaz de usuario de JupyterLab | Adobe Experience Platform*. Obtenido de Adobe Experience League: https://experienceleague.adobe.com/docs/experience-platform/data-science-workspace/jupyterlab/overview.html?lang=es

AirNow. (11 de 07 de 2021). *AirNow API*. Obtenido de AirNow API Developer Tools: https://docs.airnowapi.org/

Apache License, Version 2.0 . (17 de 06 de 2021). *Time series forecasting | TensorFlow Core*. Obtenido de TensorFlow: https://www.tensorflow.org/tutorials/structured_data/time_series

AuraQuantic. (22 de 12 de 2020). *Qué es Big Data*. Obtenido de AuraPortal: https://www.auraquantic.com/es/que-es-big-data/

Bagnato, J. (14 de 11 de 2017). *Aprendizaje Profundo: una Guía rápida*. Recuperado el 12 de 07 de 2021, de Aprende Machine Learning: https://www.aprendemachinelearning.com/aprendizaje-profundo-una-guia-rapida/

Barbero, Á. (24 de 09 de 2020). *8 herramientas para proyectos de Big Data e Inteligencia Artificial*. Obtenido de Instituto de Ingeniería del Conocimiento: https://www.iic.uam.es/innovacion/herramientas-big-data-inteligencia-artificial/

BreezoMeter. (2021). *BreezoMeter*. Recuperado el 12 de 07 de 2021, de https://www.breezometer.com/

Colaboradores de Wikipedia. (28 de 07 de 2020). *Aprendizaje no supervisado*. Obtenido de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/Aprendizaje_no_supervisado

Colaboradores de Wikipedia. (15 de 02 de 2021a). *Aprendizaje supervisado*. Obtenido de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/Aprendizaje_supervisado

Colaboradores de Wikipedia. (04 de 06 de 2021b). *Activation function*. Obtenido de Wikipedia, la enciclopedia libre: https://en.wikipedia.org/wiki/Activation_function

Colaboradores de Wikipedia. (20 de 06 de 2021c). *GitHub*. Obtenido de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/GitHub

Colaboradores de Wikipedia. (28 de 06 de 2021d). *Interfaz de programación de aplicaciones*. Obtenido de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

Colaboradores de Wikipedia. (03 de 07 de 2021e). *Red neuronal artificial*. Obtenido de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/Red_neuronal_artificial#cite_note-1

Colaboradores de Wikipedia. (03 de 07 de 2021f). *Python*. Obtenido de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/Python

Colaboradores de Wikipedia. (04 de 07 de 2021g). *Inteligencia artificial*. Obtenido de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/Inteligencia_artificial#cite_note-1

Flask - The Pallets Projects. (2021). *Flask*. Obtenido de https://palletsprojects.com/p/flask/

García-Olalla Olivera, O. (16 de 10 de 2019). *Redes Neuronales artificiales: Qué son y cómo se entrenan*. Obtenido de Xeridia: https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-i

Gavilán, I. G. (20 de 05 de 2020). *Catálogo de componentes de redes neuronales (II): funciones de activación*. Obtenido de Ignacio G.R. Gavilán: https://ignaciogavilan.com/catalogo-de-componentes-de-redes-neuronales-ii-funciones-de-activacion/

González, B. (06 de 07 de 2021). *¿Qué es Machine Learning?* Obtenido de Cleverdata: https://cleverdata.io/que-es-machine-learning-big-data/

Google. (s.f.). *Estación CNNY*. Recuperado el 11 de 07 de 2021, de https://www.google.com/maps/place/40%C2%B049'11.0%22N+73%C2%B056'53.7%22W/@40.7960485,-73.9910868,11.48z/data=!4m5!3m4!1s0x0:0x0!8m2!3d40.819732!4d-73.948239

ICHI.PRO. (2021). *Descripción general de diferentes optimizadores para redes neuronales*. Obtenido de ICHI.PRO: https://ichi.pro/es/descripcion-general-de-diferentes-optimizadores-para-redes-neuronales-247308806799555

IIC. (20 de 08 de 2019). *7 Herramientas Big Data para tu empresa*. Obtenido de Instituto de Ingeniería del Conocimiento: https://www.iic.uam.es/innovacion/herramientas-big-data-para-empresa/

Keras. (2021a). *Backend utilities*. Obtenido de Keras API reference: https://keras.io/api/utils/backend_utils/

Keras. (2021b). *Keras*. Recuperado el 12 de 07 de 2021, de https://keras.io/

La Vigne, F. (02 de 05 de 2019). *¿Cómo aprenden las redes neuronales?* Obtenido de Microsoft Docs: https://docs.microsoft.com/es-es/archive/msdn-magazine/2019/april/artificially-intelligent-how-do-neural-networks-learn

Lazcano, R. (06 de 01 de 2020). *Diferencia entre inteligencia artificial, machine learning y deep learning*. Obtenido de Enzyme advising group blog: https://blog.enzymeadvisinggroup.com/inteligencia-artificial-machine-learning

Lopez Briega, R. E. (26 de 09 de 2016). *Series de tiempo con Python*. Obtenido de Raul E. Lopez Briega | Matemáticas, análisis de datos y python: https://relopezbriega.github.io/blog/2016/09/26/series-de-tiempo-con-python/

López, P. (29 de 06 de 2020). *¿Qué es un archivo CSV y para qué sirve?* Obtenido de Geeknetic: https://www.geeknetic.es/Archivo-CSV/que-es-y-para-que-sirve

Márquez, V. (29 de 10 de 2018). *¿Qué es exactamente Machine Learning?* Obtenido de Medium: https://medium.com/latinxinai/qu%C3%A9-es-exactamente-machine-learning-77441201a65b

Microsoft. (14 de 04 de 2016). *Get Started with Visual Studio Code*. Recuperado el 12 de 07 de 2021, de Visual studio code: https://code.visualstudio.com/learn

Microsoft. (2021). *Crear o editar archivos .csv para importarlos a Outlook*. Obtenido de Soporte de microsoft: https://support.microsoft.com/es-es/office/crear-o-editar-archivos-csv-para-importarlos-a-outlook-4518d70d-8fe9-46ad-94fa-1494247193c7

Moreno, C. G. (11 de 22 de 2016). *¿Qué es el Deep Learning y para qué sirve?* Obtenido de Indra company: https://www.indracompany.com/es/blogneo/deep-learning-sirve

Moscoso, P. M. (21 de 08 de 2010). *Estudio revela que la ciudad de Nueva York presenta una contaminación atmosférica elevada*. Obtenido de Natura medio ambiental:

https://www.natura-medioambiental.com/estudio-revela-que-la-ciudad-de-nueva-york-presenta-una-contaminacion-atmosferica-elevada/

National Weather Service. (s.f.). *API Web Service*. Recuperado el 11 de 07 de 2021, de National Weather Service: https://www.weather.gov/documentation/services-web-api

NYC. (01 de 07 de 2021). *NYC Environmental Health*. Obtenido de New York City: https://nyccas.cityofnewyork.us/nyccas2021/web/report

Nyhan, M., Grauwin, S., Britter, R., Misstear, B., McNabola, A., Laden, F., . . . Ratti, C. (2016). "Exposure Track"—The Impact of Mobile-Device-Based Mobility Patterns on Quantifying Population Exposure to Air Pollution. *Environmental Science & Technology*, 9671-9681.

Olmedo, Y. (20 de 08 de 2021). *¿Qué es MapReduce?* Obtenido de SolidQ: https://blogs.solidq.com/es/business-analytics/que-es-mapreduce/

OMM. (2009). *El tiempo el clima y el aire que respiramos*. Ginebra: Organización Meteorológica Mundial.

OMS. (01 de 07 de 2021). *Calidad del aire y salud*. Obtenido de Organización Mundial de la Salud: https://www.who.int/es/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health

OpenWeather. (2021). *OpenWeather API*. Recuperado el 11 de 07 de 2021, de https://openweathermap.org/

Pandas. (2021). *pandas - Python Data Analysis Library*. Obtenido de pandas: https://pandas.pydata.org/

Patiño Restrepo, J. F., Celis Rodríguez, É., & Díaz Cortés, J. (2015). *Gases sanguíneos, fisiología de la respiración e insuficiencia respiratoria aguda*. Médica Panamericana.

PRICING - Revenue Management. (12 de 04 de 2021). *Series de Tiempo*. Recuperado el 12 de 07 de 2021, de Pricing: https://www.pricing.cl/conocimiento/series-de-tiempo/

R. PowerData. (30 de 01 de 2017). *Preprocesar y normalizar datos*. Obtenido de PowerData: https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/preprocesar-y-normalizar-datos-4-pasos-para-limpiar-y-mejorar-datos

Real Academia Española. (01 de 07 de 2021). *Respirar*. Obtenido de RAE: https://dle.rae.es/respirar

SAS Institute Inc. (01 de 07 de 2021). *Big Data: Qué es y por qué importa*. Obtenido de SAS: https://www.sas.com/es_es/insights/big-data/what-is-big-data.html

Soloaga, A. (14 de 03 de 2019). *Python, los 5 usos más importantes de este lenguaje de programación*. Obtenido de El Blog de Akademus: https://www.akademus.es/blog/programacion/principales-usos-python/

The City of New York. (11 de 07 de 2021). *NYC DOT - Motorists & Parking - Real-Time Traffic Cameras*. Obtenido de New York City: https://www1.nyc.gov/html/dot/html/motorist/atis.shtml

The World Air Quality Project. (20 de 06 de 2021). *https://aqicn.org/here/es/*. Recuperado el 12 de 07 de 2021, de https://aqicn.org/here/es/

Visual Crossing. (23 de 03 de 2020). *Visual Crossing Weather Data Documentation*. Recuperado el 01 de 07 de 2021, de Visual Crossing: https://www.visualcrossing.com/resources/documentation/weather-data/weather-data-documentation/

Visual Crossing Corporation. (2020). *Visual crossing API*. Recuperado el 11 de 07 de 2021, de Visual crossing: https://www.visualcrossing.com/