Trabajo Fin de Máster
Máster Universitario de Lógica, Computación e Inteligencia Artificial

Técnicas de Deep Learning para la predicción de los niveles de contaminación

Autor: Juan Cañete Rodríguez

Director: Miguel Ángel Gutiérrez Naranjo

Codirector: Miguel Cárdenas Montes

Dpto. Lenguajes y Ciencias de la Computación

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Máster
Máster Universitario de Lógica, Computación e Inteligencia Artificial

Técnicas de Deep Learning para la predicción de los niveles de contaminación

Autor:

Juan Cañete Rodríguez

Director:

Miguel Ángel Gutiérrez Naranjo

Profesor Titular

Codirector:

Miguel Cárdenas Montes

Dpto. Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Resumen

Machine learning, Deep Learning o Time Series Forecasting son expresiones cada vez más frecuentes en nuestra vida diaria. Además, los datos que se generan diariamente por usuarios y empresas se están incrementando continuamente. El objetivo de este trabajo fin de máster es realizar un estudio combinando varias técnicas de inteligencia artificial para predecir los niveles de contaminación de la ciudad de Madrid. En este trabajo se ha usado *Random Forest* y una red neuronal recurrente. Además de estas técnicas, se ha usado la técnica *principal component analysis* con el objetivo de seleccionar las características más importantes de los datos para así minimizar el error en los datos de pronóstico de los niveles de contaminación. Los contaminantes pronosticados son Monóxido de Carbono (CO), Dióxido de Nitrógeno (NO_2) y Ozono (O_3). Con el propósito de obtener pronósticos de calidad, se utilizarán las siguientes variables meteorológicas: temperatura máxima, la dirección del viento o la velocidad media del viento. El objetivo es la obtención del pronóstico de los niveles de contaminación para seis días.

Índice

Resumen **I**

Resumen I

Notación V

1 Introducción **1**

1.1 Estructura de la memoria 3

2 Objetivos **5**

2.1 Objetivos intrínsecos en el trabajo 7

3 Metodología **9**

3.1 Métodos de pronóstico 9

3.1.1 Nowcasting 11

3.2 Métodos de análisis 11

3.2.1 Descomposición Estacional y de Tendencias con Loess 11

3.2.2 Test de rangos con signo de Wilcoxon 14

3.3 Métodos de aprendizaje automático 17

3.3.1 Análisis de Componentes Principales 17

3.3.2 Árboles de decisión: Random Forest 20

3.3.3 Long Short-Term Memory 24

3.3.4 Lenguajes de programación y Frameworks usados 27

4 Preprocesamiento de Series Temporales 29

4.1 Preprocesado de los datos obtenidos de la serie temporal de AEMET 32

4.2 Integración de series temporales 34

4.3 Extracción de características 34

4.4 Correlación entre variables 36

5 Resultados 39

5.1 Evaluación del rendimiento 40

5.2 Resultados con Descomposición STL 41

5.3 Resultados con Random Forest 44

5.4 Resultados con Random Forest y STL 46

5.5 Resultados con Redes Neuronales Recurrentes LSTM 47

5.6 Resultados con LSTM y STL 49

5.7 Comparativa universal de metodologías 51

6 Conclusiones 53

6.1 Trabajo futuro 54

Índice de Figuras 57

Índice de Tablas 59

Bibliografía 61

Notación

$\|\mathbf{v}\|$	Norma del vector $\mathbf{v}$
$\langle \mathbf{v}, \mathbf{w} \rangle$	Producto escalar de los vectores $\mathbf{v}$ y $\mathbf{w}$
$\operatorname{arc\,tg}$	Función arco tangente
$\tanh$	Función tangente hiperbólica
$\sin^y x$	Función seno de x elevado a y
$\cos^y x$	Función coseno de x elevado a y
Sa	Función sampling
sgn	Función signo
$\frac{\partial y}{\partial x}$	Derivada parcial de y respecto a x
MSE	Mean square error
	Mean absolute error
$\|\mathbf{A}\|$	Cardinal, número de elementos del conjunto $\mathbf{A}$
$\mathbf{x}_i, i = 1,2,\ldots,n$	Elementos i, de 1 a n, del vector $\mathbf{x}$
$\mathrm{d}x$	Diferencial de x
$\leqslant$	Menor o igual
$\geqslant$	Mayor o igual
σ_x^2	Varianza de x

1 Introducción

En la actualidad, la población de las ciudades está aumentando progresivamente y con ello, se incrementan los problemas en estas, bien sea por necesidades de abastecimiento energético, colapso del tráfico o la contaminación provocada por diversos factores. Es por ello que los habitantes que residen en grandes ciudades [32, 10] con niveles altos de contaminación pueden tener serios problemas de salud, especialmente, los relacionados con la contaminación. Esta es la principal razón por la que ayuntamientos de dichas ciudades estén activando protocolos de control de tráfico para evitar enfermedades derivadas de la contaminación.

Para comenzar a definir las técnicas usadas en este trabajo, debemos saber que una serie temporal es un conjunto de observaciones o valores ordenados cronológicamente. Además, estos valores pueden estar o no equitativamente almacenados y no son independientes entre sí, es decir, el análisis se debe realizar teniendo en cuenta el orden temporal. Una serie temporal puede ser de tipo discreta o continua, dependiendo de los valores de esta. Por otro lado, si los valores se pueden pronosticar exactamente, se dice que la serie es determinista, por el contrario, si ese pronóstico es parcial y esos futuros valores tienen una distribución de probabilidad se dice que la serie es estocástica.

Las series temporales se clasifican en estacionarias y no estacionarias. Cuando la media y la varianza de los datos son constantes a lo largo del tiempo, se dice que la serie es estacionaria. Este tipo de series no contiene aumentos o disminuciones sistemáticos de sus valores. Por otro lado, en las series no estacionarias la media y la varianza de los valores no son constantes. En las series de tipo no estacionarias se añade una nueva definición, la cual sería la estimación de la tendencia de la serie, donde además, existen dos grupos:

tendencia evolutiva y tendencia determinista.

Una vez introducido el concepto de serie temporal, en este trabajo de fin de máster nos centraremos en pronosticar los niveles de contaminación de la ciudad de Madrid. Para ello, se han usado las series temporales con los datos de los índices de contaminación que se recogen de las estaciones repartidas en Madrid, valores proporcionados por el Ayuntamiento de Madrid. Además de estos datos, se han obtenido series temporales con los datos meteorológicos que están disponibles en la web de OpenData de la Agencia Estatal de Meteorología (AEMET). Los datos meteorológicos y de niveles de contaminación pertenecen a valores diarios que están comprendidos entre el 1 de Enero de 2010 y el 31 de Diciembre de 2018.

El análisis de las series temporales obtenidas tratará de cubrir los **objetivos** de pronóstico realizando los siguientes pasos:

1. En una primera aproximación, las series temporales obtenidas se deben estudiar por separado ya que contienen valores de concepto diferente, por un lado los niveles de contaminación y por otro los valores meteorológicos. Así pues, la representación de gráficos nos ayudará a comprender estas series y ver qué tendencia tienen a lo largo del tiempo. Un paso importante a realizar en este momento es realizar un preprocesado de las series temporales, en otras palabras, si para uno o varios instantes de nuestra serie temporal el dato es nulo podemos rellenar este dato realizando un procedimiento común para todos los datos nulos de la serie. Esto sería, o bien añadir el mismo valor que en el instante anterior o la media de sus valores previos y posteriores. De esta manera, nos aseguramos que nuestra serie temporal tiene valores para todos los instantes de tiempo.

2. En este paso, realizaremos la parte más importante de nuestro estudio. Analizaremos las relaciones que existen entre nuestras dos series temporales de distintas fuentes. Veremos así qué valores meteorológicos afectan a los niveles de contaminación. Como tenemos varias estaciones meteorológicas disponibles, se escogerá la estación en la que exista menos *gaps* de datos, es decir, donde haya menos espacios de tiempo sin datos. De esta manera, el preprocesado de los datos será más sencillo y podremos realizar una buena correlación entre las observaciones meteorológicas y los datos de niveles de contaminación.

3. En el último paso, realizaremos el modelado de las series estudiadas, escogidas las propiedades meteorológicas que más afectan a los niveles de contaminación para así obtener el mejor resultado en nuestro pronóstico. En este paso realizaremos numerosas pruebas con distintas técnicas de aprendizaje automático. Además, se analizarán los resultados y extraeremos algunas conclusiones relativas a los métodos utilizados, de esta manera, se podrá concluir qué modelo es el mejor y el más eficiente para nuestro estudio.

1.1 Estructura de la memoria

Hecha una introducción del estudio que se va a realizar en este trabajo fin de máster, se especificará a continuación cómo se organizará el resto de la memoria:

- En el capítulo 2 se presentan los objetivos a alcanzar en el proyecto. De manera general, se explicarán los objetivos concretos para resolver el problema principal y sus **metodologías** para conseguirlo.
- En el capítulo 3 se hará una descripción teórica detallada de todos las tecnologías usadas en el trabajo de fin de máster. En este capítulo se responderá a las preguntas de cómo funciona la tecnología usada, para qué sirve y por qué se han usado.
- En el capítulo 4 se explica en detalle todo el proceso llevado a cabo para realizar la manipulación, **preprocesado** y visualización de los datos que hemos obtenido de las fuentes de información comentadas previamente. Además, se detalla la integración de la serie temporal de los niveles de contaminación y de las variables meteorológicas, y su posterior extracción de características mediante la técnica PCA.
- Por otro lado, en el capítulo 5 se presentarán las métricas que se usan y los **resultados** obtenidos después de haber realizado diseñado los modelos de las técnicas de aprendizaje automático y profundo. Además, se realiza una comparación de todas las técnicas usadas en este estudio basándose en los resultados de las métricas usadas.
- Finalmente, el capítulo 6 se describen las conclusiones del trabajo fin de máster, detallando qué **objetivos** se han conseguido y cuál será el camino a seguir en un futuro para mejorar los resultados obtenidos.

2 Objetivos

Hasta el momento, se ha hecho una breve introducción de los conceptos que se estudiarán en este trabajo de fin de máster. A continuación se presentan los tres principales objetivos que se pretenden alcanzar:

- **Objetivo educativo.** Como primer objetivo, se busca asentar los conceptos de aprendizaje automático y profundo que se han estudiado a lo largo del curso del máster. De esta manera con los conceptos asimilados, se podrán resolver problemas en futuros desarrollos.
- **Objetivo científico.** Se resolverá un problema utilizando las últimas técnicas de aprendizaje automático y profundo teniendo en cuenta las variables meteorológicas. Además, se aplicarán conceptos como el *nowcasting* para realizar el pronóstico de los niveles de contaminación.
- **Objetivo ciudadano.** Con el pronóstico de los niveles de contaminación se añadiría calidad de vida a los ciudadanos de las grandes ciudades.

Además, se pretende explicar las herramientas usadas en este proyecto, la bases de datos en la que hemos almacenado los datos descargados, las librerías para descargar dichos datos de la fuente de información de AEMET y las técnicas de aprendizaje automático implementadas para llegar a los objetivos marcados para este proyecto. Otro punto importante a tener en cuenta es que en este proyecto se han usado las técnicas más innovadoras que existen actualmente para, de esta manera, obtener los mejores resultados y aprovechar al máximo el hardware que permite que nuestro software sea más eficaz al obtener los resultados.

Los objetivos marcados en este trabajo no son triviales y, como se ha comentado previamente, se han utilizado las tecnologías más novedosas. Por lo tanto, como lo más innovador en el ámbito del aprendizaje automático está escrito o bien en el lenguaje de programación *Python*[29] o bien en R, para desarrollar este proyecto se ha elegido *Python*. Además, tenemos numerosas librerías y *frameworks* muy eficientes que permiten resolver problemas de una manera más sencilla sin la necesidad de escribir una gran cantidad de código. Uno de los *frameworks* elegidos para desarrollar este proyecto ha sido *Scikit-learn* [17], escrito en *Python* y con una extensa librería de técnicas de aprendizaje automático. Otro *framework* muy potente usado en este proyecto, ha sido *Keras*. Este último *framework* nos permite modelar las técnicas de aprendizaje automático más potentes, además de usar el hardware de *NVIDIA*, el cual permite paralelizar el cálculo y por tanto realizar un entrenamiento más rápido. Otra característica importante de este *framework* es que también está escrito en Python y es capaz de ejecutar *Tensorflow* [13], *CNTK* [24] o *Theano* [31]. Además de esto, *Keras*[5] se desarrolló con el objetivo de hacer experimentos rápidamente, razón de peso en la elección de este *framework* para desarrollar nuestro trabajo.

Lo comentado en esta sección hasta ahora ha sido en referencia a las librerías usadas hasta ahora para implementar técnicas de aprendizaje automático. Para completar los objetivos de este trabajo se han utilizado otras técnicas automáticas de descarga de datos para aumentar nuestra información y tener así unos datos más consistentes. Todos estos datos se guardan en forma de serie temporal en una base de datos NoSQL llamada MongoDB[9] para ser consultados posteriormente. Por otro lado, lo modelos de computación y técnicas usadas en este proyecto se pueden clasificar de la siguiente manera:

- Los métodos usados en Python para analizar las series temporales han sido; en primer lugar, el método de Descomposición Estacional y de Tendencias con Loess (STL Decomposition) [6] desarrollado por Cleveland y; por otro lado, se han hecho pruebas no paramétricas para comparar el rango medio y determinar si existe diferencia mediante el test de los rangos con signo de Wilcoxon (Wilcoxon signed rank test) [33].

- Las técnicas de aprendizaje automático empleadas en este trabajo han sido; por un lado, la técnica de **análisis de componentes principales** (Principal Components Analysis, PCA, por su acrónimo en inglés) [19] para describir el conjunto de datos

usando la librería de Python *scikit-learn*; por otro lado, la técnica para el pronóstico de valores con **Random Forest** en regresión [23] usado también la librería de Python *scikit-learn*; y por último, otra técnica que usa Redes Neuronales Recurrentes (Recurrent Neural Networks, RNN, por su acrónimo en inglés) con **Long Short-Term Memory** (LSTM) [14], implementada usando el framework *Keras*.

2.1 Objetivos intrínsecos en el trabajo

Las técnicas de aprendizaje automático están avanzando a pasos muy acelerados en el campo de la investigación y desarrollo en estos últimos años y para anticiparse a eventos que ocurren en la vida diaria, como por ejemplo, la detección prematura de células cancerígenas en una biopsia [34], la regulación del tráfico en una ciudad [2] o incluso evitar un ataque informático [16] en una empresa mediante el análisis de las peticiones a los equipos informáticos.

El impacto que está teniendo en la nuestra vida diaria el aprendizaje automático es tal que existen plataformas donde se realizan competiciones para que investigadores y desarrolladores implementen el mejor algoritmo para resolver cualquier problema de la vida cotidiana. Un ejemplo a destacar fue la competición que lanzó el CERN (*European Laboratory for Particle Physics*) con el objetivo de seleccionar las partículas interesantes que el Gran Colisionador de Hadrones había detectado mediante un algoritmo que implementaba aprendizaje automático y así ahorrar tiempo de computación en analizar esas partículas que no contenían datos relevantes.

Para realizar todo lo comentado hasta aquí, se ha pasado por varias fases que eran necesarias para comenzar el proyecto. En primer lugar, se necesitaba conocer el origen del problema y analizar qué posibles soluciones se podrían dar, además de conocer cuántos datos se disponía y qué posibilidades había de recolectar más información para dar con unos resultados concluyentes. Posteriormente, era necesario la investigación de las técnicas de aprendizaje automático que mejor se adaptaban a nuestras soluciones, para ello, era necesario conocer y comprender a fondo el funcionamiento de estas técnicas para elegir la mejor de ellas. Por último, la fase de diseño de la arquitectura del proyecto dónde estará alojada la base de datos, los servicios que recolectan los datos de las fuentes de información

y las técnicas de aprendizaje automático. Todas estás partes de la arquitectura trabajarán en consonancia para conseguir el objetivo principal del proyecto.

Finalmente, aunque no se han descrito específicamente, con los objetivos que se pretenden alcanzar en la realización de este trabajo de fin de máster, hay otros objetivos intrínsecos que se han ido superando a lo largo de este. Estos objetivos han ayudado a mejorar notablemente el conocimiento de las técnicas investigadas y aplicadas, ya que en la labor de investigación se han encontrado las técnicas de aprendizaje automático más novedosas que existen hoy día. Además, para la extracción de los datos y el enriquecimiento del conocimiento en los datos extraídos, se han aplicado técnicas de *Data Mining*, así como la aplicación de los métodos disponibles en el framework *Keras* usado, de la librería *scikit-learn* y del lenguaje de programación Python.

3 Metodología

3.1 Métodos de pronóstico

Actualmente vivimos en un momento de la historia donde los datos son más importantes que nunca. En 2018 se alcanzó los 33 zettabytes [26], equivalente a 33 millones de terabytes, y estos datos son usados para diferentes ámbitos. Una de las finalidades es la de crear pronósticos que ayuden a adelantar acontecimientos importantes. Estos pronósticos juegan un papel muy importante cada día, por ejemplo en la bolsa de valores, donde pronosticar qué va a ocurrir para ganar en cada operación es valioso para las empresas. Otro pronóstico importante en nuestra vida diaria es el meteorológico. El pronóstico meteorológico se usa diariamente y es una parte imprescindible en nuestro día a día, especialmente en aeropuertos donde estos pronósticos ayudan a decidir el despegue o aterrizaje de millones de aviones alrededor del mundo. Por último, cabe destacar los sistemas de aprendizaje automático para la ciberseguridad con el objetivo principal de prevenir ciberataques realizando pronósticos con datos históricos de ataques previos.

Esto hace que las técnicas que se usan en el pronóstico estén avanzando a pasos agigantados para que cada vez sea más eficiente realizar los cálculos y que el error del pronóstico sea mínimo. Por esta razón, existen numerosos frameworks que implementan las técnicas de aprendizaje automático que usan el hardware de manera que se pueda paralelizar los procesos de entrenamiento de una manera muy sencilla para el programador y así centrarse en lo realmente importante en un estudio. Sin embargo, aunque se avance en estas técnicas de pronóstico a nivel de software, hay fases que son inapelables a la hora de realizar un

estudio de este ámbito y obtener así el mejor resultado. A continuación, se muestran las principales fases que un estudio debe seguir:

- Lo primero que se debe hacer es analizar el problema que se va a resolver para así decidir qué datos necesitamos y qué variables son las más importantes. Este paso es crucial puesto que es el que va a decidir toda la investigación a futuro. Si los datos que hemos decidido obtener no son los correctos o las variables no son lo suficientemente determinantes en nuestro estudio, los resultados que obtendremos no serán los deseados y por tanto, el pronóstico no será lo suficientemente bueno.

- Una vez analizado el problema, procederemos a investigar qué fuentes de datos son las más convenientes para obtener la información necesaria. En este caso, hay que estudiar qué facilidades tenemos para obtener los datos ya que el proceso de obtención de estos se puede complicar si, por ejemplo, los datos no los proporciona una *interfaz de programación de aplicaciones* (Application Programming Interface, API, de su acrónimo en inglés) o la fuente de información es inestable, es decir, que se pierda la conexión continuamente al intentar acceder a los datos.

- Cuando ya tengamos todos los datos a nuestra disposición, tendremos que hacer un preprocesado para que estos tengan el formato correcto y, además, rellenar los huecos en los que no haya datos. Para rellenar los datos, se seguirá el mismo patrón para toda la serie temporal para así tener unos datos congruentes. Una vez rellenos los huecos, podremos representar los datos gráficamente para así, estudiar su tendencia y patrones que tengan los datos.

- Por último, una vez realizado todo el trabajo referente a recolección de datos y preprocesado, tendremos que realizar un estudio para decidir qué técnica de aprendizaje automático es la mejor para realizar el pronóstico que necesitamos. Este último paso será explicado en detalle en la próxima sección puesto que hay varias técnicas a emplear para realizar un pronóstico adecuado.

Como ya se ha comentado, estas cuatro fases descritas son importantísimas e insustituibles en la realización de un proyecto donde se requiera un pronóstico de cualquier tipo. De esta forma, se conseguirá que el acierto en el pronóstico esté dentro de un límite estable

y aceptable. Este límite en la precisión del pronóstico suele ser mayor del 80 % para que dicho pronóstico sea confiable.

3.1.1 Nowcasting

El *nowcasting* es el término que se le atribuye al pronóstico en un periodo desde las 2 horas hasta las 6 horas de las condiciones meteorológicas. Por lo tanto, esta predicción detallada se realiza mediante una extrapolación de los parámetros conocidos de los datos meteorológicos, pudiendo mejorar el pronóstico de algunas características, como por ejemplo, la fuerza de una tormenta o la velocidad del viento.

En este trabajo, se utilizarán los datos meteorológicos descargados de la ciudad de Segovia, debido a la proximidad a Madrid, para mejorar los pronósticos meteorológicos de esta y poder, de esta manera, mejorar los resultados de pronóstico de los datos de contaminación.

3.2 Métodos de análisis

En esta sección se describirá en profundidad los métodos de análisis que se han empleado en este trabajo fin de máster para estudiar la serie temporal compuesta por los datos obtenidos de la web de *OpenData* de *AEMET* y los datos de niveles contaminación proporcionados por el Ayuntamiento de Madrid.

3.2.1 Descomposición Estacional y de Tendencias con Loess

La Descomposición Estacional y de Tendencia (STL) [6] es un método robusto y versátil para realizar descomposiciones de series temporales, donde Loess es un método para estimar relaciones no lineales. Este método fue desarrollado por Cleveland, McRae y Terpenning (1990) con el objetivo de descomponer una serie temporal y mostrar las distintas partes de esta para así comprender mejor su comportamiento. Las partes en las que este método descompone una serie temporal son las siguientes:

- **Componente Estacional:** Patrones que se repiten en un periodo de tiempo fijo. Por ejemplo, una página web podría recibir más visitas durante los fines de semana, entonces, esto produciría datos con una estacionalidad de siete días.

- **Tendencia:** Refleja el incremento o el decremento del valor de la serie temporal a largo plazo de una manera suavizada. Podría ser una tendencia lineal o no.

- **Componente de Aleatoriedad:** Este componente también la suelen llamar "Componente de Ruido", "Componente Irregular" o "Componente Restante", y contiene la parte residual de la serie temporal original después de obtener la componente estacional y la tendencia.

En la figura 3.1 se muestra una ilustración de la descomposición de una serie temporal mediante Descomposición STL.

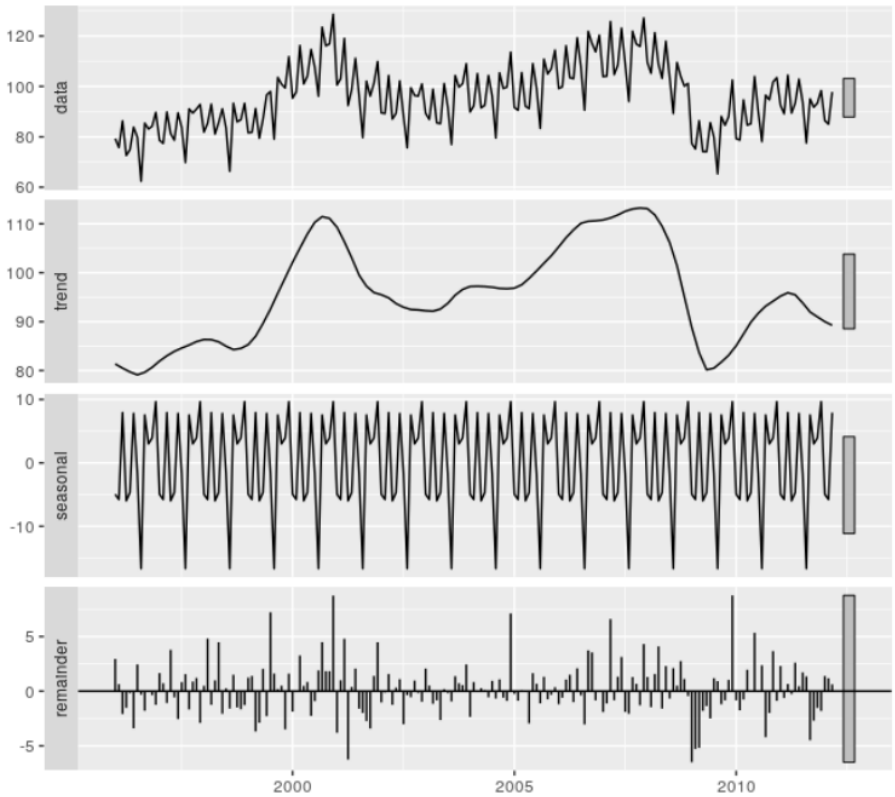

Figura 3.1 Descomposición Estacional y de Tendencia con Loess de una serie temporal. Esta figura muestra cuatro subfiguras en las que se muestran los datos reales, la tendencia, la componente estacional y el componente de aleatoriedad..

Este método de descomposición muestra varias ventajas en comparación con el método de Extracción de Estacionalidad en ARIMA [7] (Seasonal Extraction in ARIMA Time Series, SEATS, en su acrónimo en inglés) o con el método X11 [8]. Estas ventajas del método de descomposición STL son:

- A diferencia con SEATS y X11, STL puede manejar cualquier tiempo de estacionalidad, no solo datos trimestrales o mensuales.
- El componente estacional puede cambiar a lo algo del tiempo y este ratio de cambio puede ser controlado por el usuario.
- La uniformidad del ciclo de tendencia puede ser controlado también por el usuario.
- Puede ser robusto para los valores atípicos, por lo tanto, las observaciones inusualmente ocasionales no afectarían a lo estimado en el ciclo de la tendencia y tampoco a la componente estacional. Sin embargo, si afectaría a la componente de aleatoriedad.

Por otro lado, STL tiene algunas desventajas, como por ejemplo, no maneja las variaciones del calendario automáticamente y, además, sólo provee facilidades para las descomposiciones aditivas.

Para lograr una descomposición correcta, es importante elegir correctamente entre los dos modelos que existen, aditivo y multiplicativo. Para este trabajo fin de máster, se ha optado por la descomposición de la serie temporal *aditiva*, esto es, que las diversas componentes se relacionan por agregación, por lo que la serie original resulta ser la suma de sus integrantes. La descomposición aditiva tendría la siguiente forma:

$$y_t = E_t + T_t + A_t$$

donde y_t es la serie temporal, E_t es la componente estacional, T_t es la tendencia y por último, A_t es el componente de aleatoriedad.

A continuación mostraremos los pasos más importantes en la descomposición de una serie temporal usando el método que se describe en esta sección:

1. En primer lugar, tendremos que detectar la tendencia subyacente, para ello, se suaviza la serie temporal usando la Media Móvil Central (Centred Moving Average, CMA, en su acrónimo en inglés). Es muy importante en la descomposición, usar el mismo tamaño de rango de estacionalidad en toda la serie temporal, además, es muy importante conocer el periodo de estacionalidad: semanal, mensual, etc. En caso de no conocer la estacionalidad, deberíamos de usar la transformación de Fourier.

2. A continuación, se elimina la tendencia calculada previamente de la serie temporal para, de esta manera, obtener como resultado una nueva serie temporal donde se expone la estacionalidad. El concepto estacionalidad de una serie temporal es la variación periódica y predecible. Este periodo puede ser menor o igual a un año y está intrínseco en la serie temporal.

3. De la serie temporal previamente calculada, es decir, la estacionalidad, es trivial calcular la **media de estacionalidad**. Para ello, se suma toda la estacionalidad y se divide por el periodo de estacionalidad.

4. Por último, de las series previamente calculadas podemos extraer la componente de aleatoriedad, restando la componente estacional y la tendencia a la serie original.

La descomposición STL se usa a menudo para eliminar el efecto estacional de una serie temporal y, así poder tener una visión más clara de la tendencia de esta. Además, si lo que se necesita es realizar un pronóstico, a menudo, se usa la descomposición STL, más concretamente la tendencia, para realizar dicho pronóstico.

3.2.2 Test de rangos con signo de Wilcoxon

El test de rangos con signo de Wilcoxon [33] es un test no paramétrico equivalente al T-test dependiente [21]. Como el test de rangos con signo de Wilcoxon no da por sentado la normalidad en los datos, se puede usar cuando esta suposición se ha incumplido y el uso del T-test dependiente no es apropiado. Este test de Wilcoxon se usa para comparar dos conjuntos de datos que tengan el mismo origen pero distinto rango de tiempo. Este tipo de comparaciones pueden ocurrir cuando se desea investigar cualquier cambio en los valores en un instante de tiempo con respecto a otro o cuando estos valores están sujetos a más de una condición.

Un ejemplo para comprender el uso del test de rangos con signo de Wilcoxon podría ser si se quisiera comparar si existe diferencia entre los fumadores de cigarrillos diarios antes y después de ocho semanas de terapia para dejar de fumar. Otro ejemplo donde se podría usar este test, es para comprender si hay diferencia en el tiempo reacción de un individuo que está expuesto a diferentes tipos de iluminación. Por último, mencionar que en este trabajo de fin de máster se usa este test de Wilcoxon para comparar los niveles de contaminación entre

el rango de meses octubre y marzo para una serie temporal y, entre marzo y septiembre para la otra serie temporal y así comprender si existen diferentes niveles de contaminación.

Para que un test de rango con signos de Wilcoxon nos dé unos resultados satisfactorios, debemos conocer unos puntos clave. Estos puntos clave nos hará decidir cuando elegir para analizar nuestros datos este test. Parte del proceso la engloba el comprobar que los datos pueden ser analizados mediante este test de Wilcoxon. Esto es necesario realizarlo porque únicamente es apropiado realizar este test si los datos cumplen las siguiente tres suposiciones para que el test de Wilcoxon dé un resultado válido:

1. Los primero que debe cumplir nuestros datos es que la variable dependiente debería medirse en el ordinal o nivel continuo. Por ejemplo, los cinco puntos de niveles de escalar que explican el nivel de satisfacción de un cliente al comprar un producto, van desde "No mucho" a "Sí, mucho", incluyendo de esta manera la escala de Likert [1] en esta variable ordinal. Por otro lado, un ejemplo de variables continuas sería el peso de una persona, el nivel de inteligencia o la puntuación en un examen.

2. Lo siguiente que se debe cumplir es que la variable independiente deberían ser de dos categorías, "grupos relacionados" o "pares complementarios". La primera categoría, "Grupos relacionados", indica que los mismos sujetos están presentes en ambos grupos, esto es debido a que cada sujeto se ha medido en dos ocasiones en la misma variable dependiente. Por otro lado, la categoría "pares complementarios" evalúa la hipótesis de que la puntuación para dos variables se obtuvieron de una misma distribución, es decir, calcula la diferencia entre cada conjunto de pares complementarios y después sigue el mismo procedimiento del test de rango de signo para comparar el ejemplo la mediana.

3. La distribución de las diferencias entre los dos grupos relacionados deben de ser de forma simétrica, en otra palabras, si la distribución de la diferencias tiene una forma simétrica, se puede analizar el estudio realizando el test de rangos con signo de Wilcoxon.

Las condiciones enumeradas anteriormente añaden un tiempo extra en el análisis de las series temporales pero por el contrario, si hemos realizado estas comprobaciones

previas antes de realizar el test de Wilcoxon, evitaremos sorpresas cuando obtengamos los resultados finales de este test.

El proceso del test de rango con signo de Wilcoxon sigue los siguientes pasos, tomando N como el tamaño del ejemplo, por ejemplo, el número de pares de una serie temporal, por lo tanto, el número de elementos será de $2N$, siendo $i = 1,...,N$ con $x_{1,i}$ y $x_{2,i}$ los indicadores de la medida:

1. Para $i = 1,...,N$, se calcula $|x_{2,i} - x_{1,i}|$ y $sgn(x_{2,i} - x_{1,i})$ donde sgn es la función signo.
2. Excluir pares con $|x_{2,i} - x_{1,i}| = 0$, de esta manera se reduce el tamaño el ejemplo.
3. Ordenar los N_r pares restantes de menor a mayor por la diferenciación del valor absoluto, $|x_{2,i} - x_{1,i}|$
4. Clasificar los pares, comenzando por el más pequeño cómo 1, siendo R_i la clasificación dada.
5. Calcular el test estadístico W

$$W = \sum_{i=1}^{N_r} [sgn(x_{2,i} - x_{1,i}) * R_i]$$

   la suma de los rangos por signo.

6. W sigue una distribución específica, teniendo esta distribución un valor esperado a cero y la varianza de:

$$\sigma = \frac{N_r(N_r+1)(2N_r+1)}{24}$$

   De esta manera, W se puede comparar a un valor crítico desde una referencia. El test en los dos sentidos consiste en rechazar H_0 si $|X| < W_{critical,N_r}$, siendo H_0 la diferencia entre los pares que siguen una distribución simétrica alrededor de cero.

7. Como el valor de N_r incrementa, la distribución de W converge a una distribución normal. Así que, para $N_r \geq 20$, una evaluación Z se puede calcular como:

$$z = \frac{W}{\sigma_W},$$

   donde

$$\sigma_W = \sqrt{\frac{N_r(N_r+1)(2N_r+1)}{24}}$$

Para representar un test doble, H_0 es rechaza si $z_{critical} < |z|$. Por otro lado, un test simple se puede representar o bien, con la distribución exacta o con una aproximación de la distribución.

8. Para $N_r < 20$ el test original se aplica la estadística *Tukey* de Siegel [22], siendo esto, el menor valor de dos sumas de rango por signo dadas.

3.3 Métodos de aprendizaje automático

En esta sección se explicarán las técnicas de aprendizaje automático empleadas en este trabajo, técnicas que son la base de los objetivos a alcanzar en este proyecto y donde se ha invertido la mayor parte del tiempo.

3.3.1 Análisis de Componentes Principales

La elección de las variables más relevantes en una serie temporal es tan importante como elegir la técnica de aprendizaje automático que más se ajuste a los objetivos que necesitamos alcanzar puesto que esta elección afecta directamente en el resultado final. Es por eso que se ha utilizado la técnica de análisis de componentes principales (PCA, en su acrónimo en inglés) para realizar la mejor elección en las variables de la serie temporal de este trabajo.

PCA [18, 19] es una técnica en la que se pretende realizar una reducción de la dimensionalidad de los datos de entrada, transformando un gran conjunto de variables de datos en un conjunto más reducido, el cual aún contiene la información más importante, es decir, este método consigue reducir la dimensión de un conjunto de datos sin perder información.

Normalmente, reducir el número de variables en un conjunto de datos suele penalizar la precisión de estos, aunque dependiendo del conjunto de datos que se esté estudiando, se puede asumir más o menos la penalización de la precisión en los datos a cambio de simplicidad en ellos, puesto que un conjunto de datos más reducido es más fácil de representar y de estudiar. Otra ventaja en la reducción de las variables es que hace más fácil y rápido analizar el conjunto de datos para los algoritmos de aprendizaje automático. Por lo tanto, PCA reduce el número de variables de un conjunto de datos para simplificar su estudio, preservando al máximo la cantidad de información.

A continuación, se mostrará los pasos que la técnica PCA realiza en la determinación de la importancia de las variables de una serie temporal:

1. **Estandarización o Normalización:** Este paso es crítico en la realización de la técnica PCA y es necesario realizar la normalización de los datos antes de comenzar, ya que esta técnica es bastante sensible a la varianza de las variables iniciales. En otras palabras, si hay mucha diferencia entre los rangos de las variables iniciales, esas variables con valores altos dominarán sobre las que tienen un valor bajo, lo que conducirá a resultados sesgados.

   Por lo tanto, los valores de la serie temporal estarán entre cero y uno, matemáticamente, se puede expresar de la siguiente forma:

$$z = \frac{value - mean}{stardard \quad deviation}$$

2. **Cálculo de la matriz de covarianza:** En este paso, las variables del conjunto de datos varían de la media uno respecto a otro, es decir, se intenta buscar una relación entre ellos puesto que, algunas veces las variables están relacionadas las unas con las otras de alguna manera.

   Por consiguiente, como necesitamos buscar esa correlación entre las variables, se calcula la matriz de covarianza. Esta matriz es una matriz simétrica pxp, donde p es el número de dimensiones, que contiene la covarianza asociadas con todas las posibles pares de combinaciones de las variables iniciales. Un ejemplo de un conjunto de datos con tres variables x,y y z, cuya matriz de covarianza sería de dimensión $3x3$:

$$\begin{pmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{pmatrix}$$

   En la matriz representada, podemos observar que es simétrica con respecto a la diagonal, además, el signo en el resultado de estas operaciones de covarianza es importante:

- Si el valor es positivo, las dos variables incrementan o decrementan a la vez, es decir, están correladas.

- Si el valor es negativo, una variable incrementará cuando por el contrario la otra decrementa, es decir, estarán inversamente correladas.

3. **Calcular los vectores y valores propios de la matriz de covarianza para identificar los componentes principales:** Los vectores y autovalores son conceptos de álgebra lineal que necesitan ser calculados para determinar los principales componentes del conjunto de datos.

   Los principales componentes son variables que se construyen como combinaciones lineales de las variables iniciales. Los vectores propios de la matriz de covarianza son direcciones de los ejes donde hay más varianza, es decir, donde hay más información en cada componente principal.

4. **Vector de características:** Como se ha comentado en el paso previo, calcular el autovector y ordenarlo en orden descendente con respecto a los valores propios, nos permite encontrar los componentes principales en orden a su importancia. En este paso, se decide si mantenemos los componentes, lo descartamos o si se le da menos importancia. Si se decide mantener este ultimo vector de importancia, se añadirá a la matriz de vectores, la cual se llama matriz de vectores de características.

5. **Reestructurar los datos con los ejes de principales componentes:** En este último paso, se unirá todos los vectores propios de la matriz de covarianza teniendo en cuenta los ejes de los datos originales. Por lo que, se realizará una multiplicación de la matriz transpuesta del conjunto de datos originales por la transpuesta del vector de características.

$$ConjuntoFinal = VectordeCaracterstica^T * DatosOriginalesNormalizados^T$$

   De esta manera, obtendremos un conjunto de datos final dándole más importancia a las variables que más relevancia tenga en la serie temporal.

3.3.2 Árboles de decisión: Random Forest

Los árboles de decisión [11, 3] son una herramienta de apoyo para la toma de decisiones que usan árboles cómo modelo de decisión y sus posibles consecuencias. Los algoritmos de árboles de decisión son muy potentes cubriendo todas las áreas del problema, donde su estructura se compone de nodos internos que representan "test" sobre atributos concretos, además, cada rama representa el resultado del test y cada hoja del nodo representa una etiqueta de la clase, lo que es lo mismo, la decisión tomada después del cálculo de todos su atributos.

Los términos comunes que se suelen usar cuando de árboles de decisión se habla son los siguientes:

- **Nodo raíz:** Representa todo el conjunto de datos que es dividido en dos o más homogéneos conjuntos.
- **Partición:** Es el proceso de división del nodo en uno o más sub-nodos.
- **Nodo de decisión:** Cuando un sub-nodo se divide en más sub-nodos.
- **Hoja:** Estos son los nodos que no se vuelven a dividir en más sub-nodos. Se dice entonces que este elemento es una hoja del árbol.
- **Podado:** Cuando se eliminan sub-nodos de un árbol de decisión, bien para aumentar el rendimiento del árbol o para hacerlo más sencillo y así mejorar su predicción.
- **Rama/Sub-árbol:** Es una subsección del árbol completo.
- **Nodo Padre y Nodo Hijo:** Un nodo que es dividido en más sub-nodos se le llama nodo padre de los sub-nodos, mientras que los sub-nodos son los nodos hijos.

En los árboles de decisión la variable objetivo suele ser categórica y entre los usos de este tipo de técnica está cálculo de la probabilidad de que un objeto dado pertenezca a una categoría, estos modelos se llama árboles de clasificación. Está técnica también se puede usar para estimar valores de regresión, modelos llamados árboles de regresión.

Antes de describir la técnica de *Random Forest*, se hará una breve explicación de los métodos que se usan en los árboles de decisión:

- **Bootstrapping:** El bootstrapping [28] es una técnica usada para estimar las estadísticas de un conjunto de datos mediante el muestreo de estos con reemplazamiento, es

decir, se selecciona elementos aleatorios y se agrupa en subconjuntos de datos del conjunto de datos original, tal y como se muestra en la figura 3.2. Este algoritmo nos permite entender mucho mejor la media y la desviación típica del conjunto de datos para un mejor estudio.

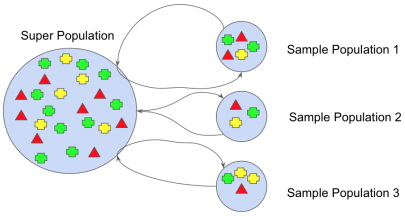

Figura 3.2 Representación del Bootstrapping..

- **Bagging:** El término *bagging*[28] viene de *Bootstrap Aggregation* y es un método simple y muy potente. Este método aplica el procedimiento *Bootstrap* explicado anteriormente a la alta variabilidad de cualquier algoritmo de aprendizaje automático, especialmente en árboles de decisión. Un proceso detallado sería:

  1. Suponemos que hay N observaciones y M características y de esto, se selecciona aleatoriamente una observación con reemplazo, es decir, usando *Bootstrapping*.

  2. Un subconjunto de características se selecciona para crear un modelo con este subconjunto y la observación previamente seleccionada.

  3. Una característica del subconjunto se selecciona para darnos el mejor valor de particionado en los datos de entrenamiento.

  4. Esto se repite para crear muchos modelos, todos ellos entrenados en paralelo.

  5. Por último, se da una predicción basada en la agregación de predicción de todos los modelos.

- **Boosting:** El método *boosting* [28] se refiere a un grupo de algoritmos que utiliza las medias ponderadas para hacer que los resultados sean eficientes y más robustos. A diferencia de *bagging* que el modelo se ejecuta independientemente y al final agrega la predicción a todos los modelos, *boosting* trabaja en todo momento en equipo.

En otras palabras, cada modelo se ejecuta teniendo en cuenta las características del resultado del modelo anterior. Un ejemplo de este método se ilustra en la figura 3.3.

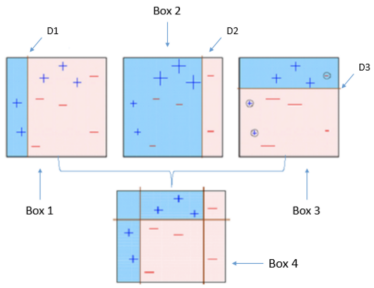

Figura 3.3 Representación del Boosting..

Una vez dada una breve introducción a los árboles de decisión, se describirá la técnica de *Random Forest* [23]. El *Random Forest* es un algoritmo de aprendizaje automático supervisado flexible y fácil de implementar, que incluso sin ajustar sus hiperparámetros, nos puede dar buenos resultados. Además, es uno de los algoritmos más usados debido a su simplicidad y que se pueden usar tanto para modelos de clasificación o de regresión.

Como ya se ha comentado, *Random Forest* es un algoritmo de aprendizaje supervisado en el cual, como su nombre indica, se crea un bosque de árboles aleatoriamente. En otras palabras, el bosque que se construye en un *Random Forest* es una combinación de árboles de decisión entrenados en su mayoría con el método "Bagging". Una de las mayores ventajas es que se puede usar tanto para problemas de clasificación como para problemas de regresión. Un ejemplo de un *Random Forest* se ilustra en la figura 3.4. Otra de las ventajas es que *Random Forest* tiene los hiperparámetros parecidos al de un árbol de decisión o al de un clasificador por *bagging*.

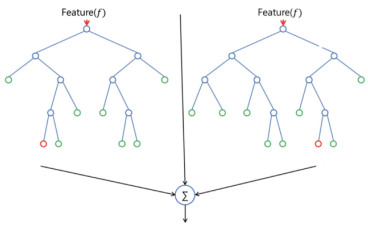

Figura 3.4 Representación de un árbol de decisión con Random Forest..

Durante el crecimiento de los árboles que contiene el *Random Forest*, este añade más aleatoriedad al modelo en lugar de buscar la característica más importante mientras separa un nodo. De esta manera, busca la mejor característica en un subconjunto de características aleatorio. Por lo tanto, se consigue una mayor diversidad en el resultado a diferencia de otros algoritmos comentados anteriormente, que generaliza el resultado.

Formalmente, *Random Forest* consiste en una colección de árboles aleatorios basados en regresión $r_n(x,\theta_m,\mathscr{D}_n)$,$m \geq 1$, donde $\theta_1,\theta_2,...$ son salidas de la variable aleatoria θ. Los árboles aleatorios se combinan de la siguiente forma usando regresión agregada:

$$\bar{r}_n(X,\mathscr{D}_n) = E_\theta[r_n(X,\theta,\mathscr{D}_n)]$$

donde E_θ indica la previsión con respecto al parámetro aleatorio, condicionalmente en X y el conjunto de datos $\mathscr{D}_n$. Además de todo esto, los hiperparámetros más importantes de un *Random Forest* que hacen incrementar el poder de predicción o hacer el modelo más rápido tomando como referencia los parámetros que nos ofrece la librería *scikit-learn* son los siguientes:

- **Incremento del poder predictivo**

  1. **n_estimators:** Parámetro que ajusta el número de árboles que se construirán en el proceso de ejecución del algoritmo. En general, un mayor número de árboles incrementa la precisión y hacen la predicción más estable pero afectará al tiempo de computación.

  2. **max_features:** Este parámetro indica el número máximo de características que *Random Forest* considerará para dividir un nodo.

  3. **min_sample_leaf:** Este parámetro indica el número mínimo de hojas que son necesarias para dividir un nodo interno.

- **Incremento de la velocidad del modelo**

  1. **n_jobs:** Parámetro que configura el número de procesos paralelos con los que se trabajarán.

2. **random_state:** Este parámetro hará que el modelo pueda ser reproducible quitándole aleatoriedad, es decir, el modelo producirá los mismos resultados y así poder identificar posibles errores del modelo.

3. **oob_score:** Este parámetro indica si se usará los ejemplos *out-of-bag* para estimar la generalización de la precisión.

Por último, como ya se ha comentado anteriormente, las ventajas de usar *Random Forest* es que se puede usar tanto para tareas de clasificación o regresión, además de ser fácil de implementar y obtener unos buenos resultados sin necesidad de configurar muchos hiperparámetros. Aunque esta facilidad para entrenar un conjunto de datos nos puede llevar a un sobreajuste, situación que hay que evitarla completamente en la implementación de algoritmos de aprendizaje automático. Otra limitación importante en *Random Forest* es el gran número de árboles que se pueden generar en el algoritmo, de manera que puede conllevar a una ineficiencia en el momento de realizar una predicción en tiempo real.

3.3.3 Long Short-Term Memory

Para explicar *Long Short-Term Memory* (LSTM) [14], primeramente se realizará una breve introducción al concepto de *Redes Neuronales Recurrentes* (RNN, de su acrónimo en inglés) [25].

Las RNN son una clase de redes neuronales artificiales donde sus conexiones entre nodos forman bucles de retroalimentación a lo largo del tiempo, lo cual permite exponer la información durante un número de pasos en el entrenamiento. Las RNN se diseñaron para tomar una serie de entradas sin limite predeterminados de tamaño. Por lo tanto, estas redes neuronales toman decisiones que son influenciadas por datos del pasado en la serie temporal, es decir, de alguna manera recuerdan el pasado. Una ilustración del funcionamiento básico de una una RNN se ilustra en la figura 3.5.

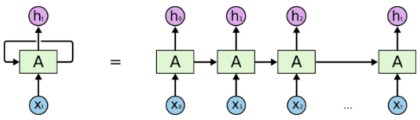

Figura 3.5 Representación de una red neuronal recurrente.

RNN se usan en aprendizaje profundo y son especialmente potentes en los casos de uso donde es muy crítico la predicción de un resultado satisfactorio. Estas pueden procesar datos desde el principio de la serie temporal hasta el final usando bucles de retroalimentación, de esta manera, la entradas son conectadas a la salida de la red neuronal permitiendo un proceso secuencial a lo largo de la serie temporal. Algunos de los casos de uso que en las RNN se usan son las siguientes:

1. **Modelado del lenguaje y generación de texto:** Dada una secuencia de de palabras, se puede predecir con una cierta probabilidad la siguiente palabra. Esto se usa por ejemplo, en los teclados de los teléfonos móviles donde se da un número de palabras que son probables que sean las siguientes en escribirse mientras se escribe un texto.
2. **Traducción de texto:** Para traducir de un lenguaje a otro, se usan RNN para realizar los modelos.
3. **Reconocimiento de voz:** En la predicción de los segmentos fonéticos basados en la entrada de ondas también se usa las RNN.
4. **Mercado bursátil:** Se usan RNN para predecir los siguientes valores que tomará la divisa, en concreto la arquitectura LSTM.

Una vez hecha la introducción de las RNN, podemos decir que LSTM es una conocida arquitectura de RNN usada en aprendizaje profundo [12]. Las LSTM se adaptan a los problemas de aprendizaje en las series temporales y son diseñadas para evitar el problema a la dependencia a largo plazo. En particular, estas superan el *problema de la pérdida por gradiente* de las RNN tradicionales. La idea clave en LSTM es el concepto del *concepto de la celda* de cada neurona, la cual es actualizada usando la correspondiente puerta de entrada, de salida o la puerta del olvido tal y como se ilustra en la figura 3.6.

La arquitectura del LSTM se ha usado en muchas áreas de investigación obteniendo resultados muy interesantes. Algunas de las aplicaciones más recientes donde se han utilizado LSTM ha sido en [30], donde se presentó un video de alta calidad sintetizado de Barack Obama con alta precisión en el movimiento de los labios y los gestos. Otro ejemplo donde se ha usado LSTM, ha sido en el estudio de eventos clínicos en unidades de cuidado intensivo [20].

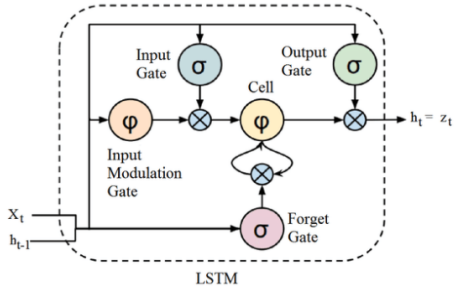

Figura 3.6 Representación de una red neuronal recurrente con LSTM..

Una red LSTM [27] calcula un mapeo de una secuencia de entrada $x = (x_1,...,x_T)$ a una secuencia de salidas $y = (y_1,...,y_T)$ ejecutando la activación de la unidad de la red neuronal usando las siguientes ecuaciones de manera iterativa desde $t = 1$ a T:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \tag{3.1}$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f) \tag{3.2}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \tag{3.3}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \tag{3.4}$$

$$m_t = o_t \odot h(c_t) \tag{3.5}$$

$$y_t = \phi(W_{ym}m_t + b_y) \tag{3.6}$$

donde la W indica el termino de matriz de pesos, (por ejemplo, W_ix es la matriz de pesos desde la puerta de entrada a la salida), W_ic, W_fc, W_oc son las matrices de pesos de las conexiones, la b indica el término del vector de *bias* o *sesgo*, σ es la función logística de sigmoide, *i, f, o* y c son la puerta de entrada, puerta de olvido, puerta de salida y vector de activación de celda respectivamente, siendo todas del mismo tamaño que la celda del vector de activación de salida m, $\odot$ es el producto de los vectores, por último, g y h son las

celdas de entras y las funciones de activación de salida de la celda.

3.3.4 Lenguajes de programación y Frameworks usados

Tal y como se ha comentado en anteriores secciones, el trabajo de fin de máster se ha desarrollado en el lenguaje de programación *Python*, además, se han usado los frameworks *scikit-learn* y *Keras*, este a su vez, ha usado como backend *Tensorflow*.

Python es un lenguaje de programación de propósito general, orientado a objetos y cuya popularidad ha sido relativamente reciente. La popularidad se debe a que *Python* apuesta por la simplicidad, versatilidad y rapidez a la hora de desarrollar. Además de todo esto, *Python* es muy potente, con la posibilidad de ser ejecutado tanto en distribuciones Linux, Windows y MacOs X; y es de código abierto, es decir, puede ser modificado y redistribuído libremente bajo una determinada licencia. Otra de sus mayores características, es la gran cantidad de librerías que existen para *Python* y que además, sean fácilmente integrables en un proyecto.

El framework usado en el desarrollo de este trabajo para implementar las técnicas de PCA explicada en el apartado 3.3.1 y *Random Forest* explicada en el apartado 3.3.2 es *scikit-learn*, y por otro lado, el framework que se ha usado para implementar las redes neuronales *LSTM* explicada en el apartado 3.3.3 es *Keras*.

Scikit-learn es una librería de *Python* que proporciona muchos algoritmo de aprendizaje supervisados y no supervisados. Esta desarrollado sobre algunas de las tecnologías más famosas que existen en la amplia biblioteca de librerías de *Python* como por ejemplo *Numpy*, *Pandas* y *Matplotlib*. La funcionalidades que incluye *scikit-learn* son: regresión, clasificación, clustering, selección de modelos y preprocesado.

Keras [5] es una API de alto nivel para desarrollar técnicas de redes neuronales, está implementada en *Python* y es capaz de ejecutarse en los principales *backends* que existe actualmente como *TensorFlow*, *CNTK* o *Theano*. Este *framework* fue desarrollado centrándose en la idea de poder realizar experimentaciones rápidas, es decir, que el desarrollador sea capaz de transformar las ideas del modelo pensado en resultados lo más rápido posible. *Keras* es una capa intermedia entre el desarrollador y la compleja librería librería *TensorFlow*. *Keras* nos permite realizar un prototipo rápidamente, soporta tanto redes neuronales convolucionales, redes neuronales recurrentes, como una combinación de

estos dos tipos. Además, es posible ejecutar ejecutar el código usando la *CPU* o la *GPU*, facilitando de esta manera la paralelización del cálculo de los modelos.

4 Preprocesamiento de Series Temporales

Hasta ahora, se ha hecho una introducción sobre todas las técnicas que se van a usar para realizar el modelo y el pronóstico de la serie temporal que se analiza en este trabajo fin de máster. El siguiente paso es realizar una introducción del proceso llevado a cabo para realizar el preprocesado de los datos con el objetivo de entender a qué nos enfrentamos y cuáles son las características principales de estos datos y, de esta manera, poder usarlos en las técnicas descritas anteriormente.

En secciones anteriores se ha comentado que se dispone de dos conjuntos de datos en los que se basa este trabajo. El primer conjunto de datos es el proporcionado por el Ayuntamiento de Madrid, que contiene los niveles de contaminación sobre los índices de Monóxido de Carbono (CO), Dióxido de Nitrógeno (NO_2) y Ozono (O_3) que se han detectado en las estaciones meteorológicas de Madrid. El segundo conjunto datos se ha descargado de OpenData de AEMET usando la API que esta plataforma ofrece, como ya se ha explicado previamente en secciones anteriores. Los dos conjuntos de medidas que tenemos disponibles se han tomado desde el 1 de Enero de 2010 hasta el 31 de Diciembre de 2018 con una frecuencia diaria, por lo tanto, tenemos un total de 3286 medidas por cada índice de contaminación y por cada estación o lo que es lo mismo, un total de ocho años completos de datos. Cada registro de los datos proporcionados por el Ayuntamiento de Madrid contiene tres niveles de contaminación, es decir, el CO, NO_2 y O_3, y por otro lado, cada registro del conjunto de datos de AEMET contiene veinte registros de datos y estos

son los siguientes:

- **fecha**
- **indicativo**, es el identificador único de la estación meteorológica.
- **nombre**
- **provincia**
- **altitud**, es la altitud a la que se encuentra la estación meteorológica.
- **tmed**, temperatura media del día de la muestra.
- **prec**, riesgo de precipitación del día.
- **tmin**, temperatura mínima.
- **horatmin**, hora en la que se registró la temperatura mínima.
- **tmax**, temperatura máxima.
- **horatmax**, hora en la que se registró la temperatura máxima.
- **dir**, dirección del viento.
- **velmedia**, velocidad media del día.
- **racha**, racha de viento máxima del día.
- **horaracha**, hora en la que se registró la racha máxima.
- **sol**, intensidad de radiación ultravioleta proveniente del Sol.
- **presMax**, presión atmosférica máxima del día.
- **horaPresMax**, hora en la que se registró la presión atmosférica máxima.
- **presMin**, presión atmosférica mínima del día.
- **horaPresMin**, hora en la que se registró la presión atmosférica mínima.

De todos estos registros, se seleccionarán los más relevantes para el pronóstico de la serie temporal utilizando la técnica PCA anteriormente descrita, aunque de esto se hablará en más detalle en la próxima sección.

Con todo lo dicho, a continuación se muestran las gráficas que representan las series temporales sin ningún tipo de preprocesamiento. En la figura 4.1 se muestra los datos de

Ozono, en la figura 4.2 el dióxido de nitrógeno y, por último, en la figura 4.3 el monóxido de carbono.

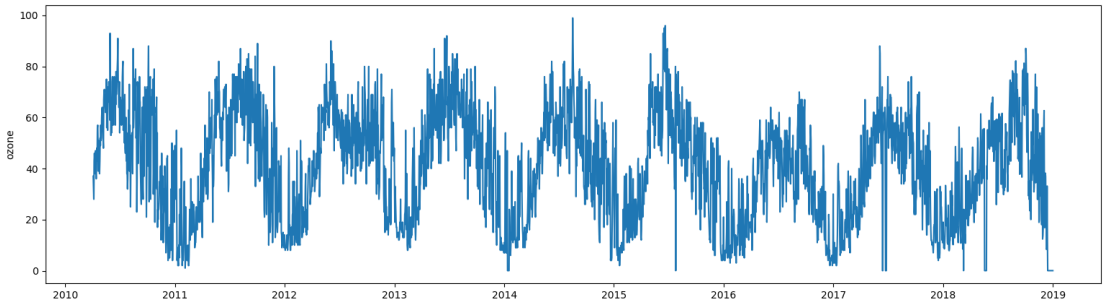

Figura 4.1 Gráfica con datos en bruto sobre el ozono.

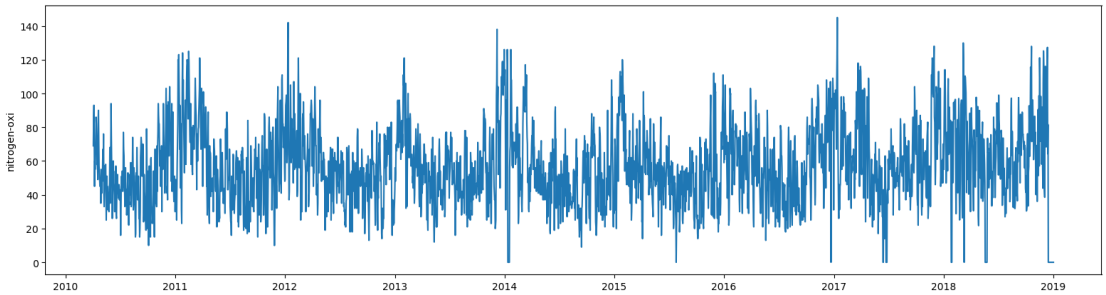

Figura 4.2 Gráfica con datos en bruto sobre el dióxido de nitrógeno.

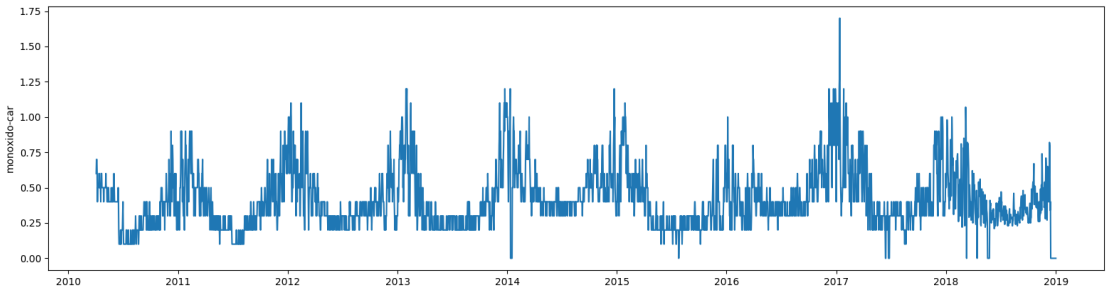

Figura 4.3 Gráfica con datos en bruto sobre el monóxido de carbono.

Por otro lado, se muestran las gráficas que muestran la representación de todos los registros de la serie temporal de AEMET. Tal y como se puede observar en las gráficas, los datos son de gran calidad ya que las fuentes de las que se han obtenido son fiables y no contienen ningún espacio de tiempo en el que no existan datos. A pesar de que los datos no

hay espacios de tiempo sin registros, hay que puntualizar que en los datos descargados de AEMET sí que hay registros vacíos datos en algunos puntos de la serie temporal, como por ejemplo, en el registro que indica la dirección del tiempo o la intensidad de radiación ultravioleta del sol tal y como se puede observar en la figura 4.4.

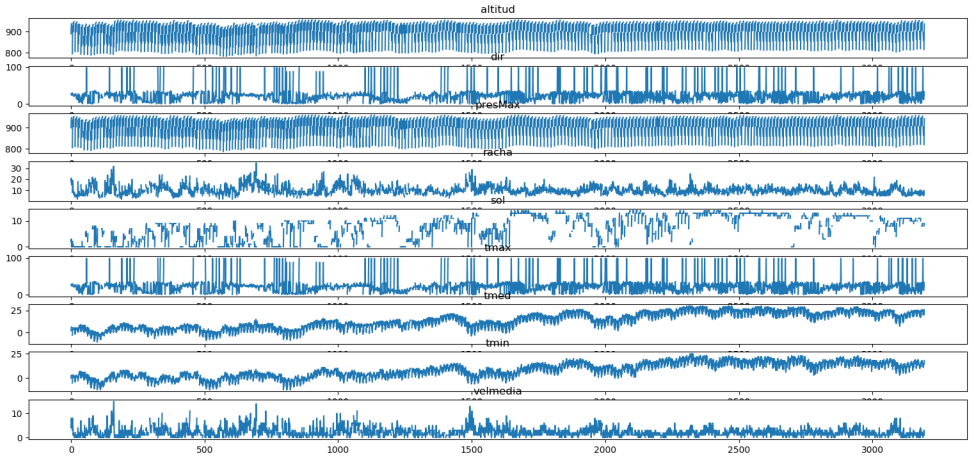

Figura 4.4 Gráfica en la que se muestra los componentes de los que se compone la serie temporal descargada de la fuente AEMET..

4.1 Preprocesado de los datos obtenidos de la serie temporal de AEMET

Anteriormente, se ha explicado la procedencia de los datos, que características tienen y cuáles son sus defectos después de haber sido extraídos de la fuente de información. En esta sección, procederemos a explicar qué procesos se han realizado para mitigar los posibles fallos que tengan los datos extraídos, que como ya se ha comentado anteriormente, los datos de AEMET contenían propiedades vacías.

Para realizar el preprocesado de esta serie temporal y que toda la serie contenga datos continuos, se debe aplicar una técnica común a todas las instancias. Para rellenar los huecos vacíos teníamos varias opciones y estas se explican a continuación:

- Se podría eliminar la instancia que contenga algún hueco vacío para así evitar realizar un preprocesado que conlleva tiempo de implementación y posteriormente un tiempo

de computación, es otras palabras, sería eliminar una fila que tiene datos que resume la climatología de un día completo en una determinada zona de Madrid. Esto no es recomendable en nuestro caso porque nuestro conjunto de datos es bastante limitado y podemos perder muchos datos valiosos. Otra razón por la que no se realizará esta técnica es que el conjunto de datos en el que hay datos ausentes es el obtenido de AEMET y en esta serie temporal realizaremos la técnica PCA, la cual, obtendrá las variables más relevantes de este conjunto de datos y puede que entre ellas no estén las variables que tienen huecos vacíos.

- Podemos rellenar los huecos usando la **media** del conjunto de datos o bien de los valores que más se acerquen en la serie temporal, es decir, tomar los valores más cercanos al instante en el que está el hueco y realizar la media.
- Se puede calcular la **mediana** de toda la serie o como se ha explicado en el punto anterior, tomar los valores más cercanos de la serie temporal para calcular la mediana.
- Por último, se puede calcular la **moda**, que como en los casos anteriores pueden ser la moda de toda la serie temporal o bien de los valores más cercanos al valor ausente.

La decisión de qué regla usar y cómo los calculamos es decisión del desarrollador que está realizando el preprocesado, además, se debe decidir al principio de este preprocesado y no cambiar la regla a lo largo del preprocesado porque esto, puede llevar a que los resultados finales no sean los esperados.

En este estudio, se ha decidido realizar la **mediana** de los valores cercanos a los valores ausentes de la serie temporal, ya que la mediana nos dará una medida más robusta de la observación ausente. Tal y como se ha explicado anteriormente, se rellenarán los datos vacíos en la serie temporal obtenida de AEMET, realizando la media de los valores que son cercanos a los valores ausentes, concretamente, se tomará cinco valores anteriores al valor ausente y cinco valores posteriores al valor ausente. De esta manera, tendremos valores suficientes para rellenar el valor vacío con otro que se acerca a la realidad.

Hasta este punto, se ha hablado de los datos ausentes numéricos en el preprocesado, pero hay otro punto importante en esta serie temporal de AEMET, y es que los registros que contienen horas o fechas son de tipo *string*. Para estos registros que especifican un instante determinado de tiempo y que están en formato **string**, se deben convertir a tipo **datetime**.

Por lo tanto, para estos valores también se realizará un preprocesado para que estos registros (horatmax, horatmin, horaPresMax, ...) tengan valores numéricos en el formato *timestamp*.

4.2 Integración de series temporales

Una vez que tenemos los datos preprocesados de la serie temporal de valores meteorológicos, podemos realizar el siguiente paso. Este paso consiste en integrar las variables más importantes de la serie temporal de valores meteorológicos con la serie temporal de niveles de contaminación.

Previamente, se ha comentado que disponemos de 3286 registros por cada nivel de contaminación y por cada estación, entonces, si disponemos de trece estaciones meteorológicas con datos públicos en Madrid, tenemos 42718 observaciones disponibles para analizar. Las estaciones meteorológicas de Madrid en los datos de AEMET son: Aranjuez, Buitrago del Lozoya, Colmenar Viejo, Getafe, Madrid Aeropuerto, Ciudad Universitaria, Cuatro Vientos, Retiro, Puerto Alto del León, Puerto de Navacerrada, Robledo de Chavela, Somosierra y Torrejón de Ardoz.

Para realizar el estudio, se ha decidido elegir la estación meteorológica del aeropuerto de Madrid ya que es la serie temporal más completa. En la figura 4.5 se muestra el gráfico con todos los componentes de la serie temporal de AEMET junto con los datos de contaminación proporcionados por el Ayuntamiento de Madrid.

4.3 Extracción de características

Una vez que tenemos las observaciones de contaminación y el conjunto de datos de AEMET integrados, podremos realizar un análisis más profundo de esta serie temporal. De esta manera, podremos estudiar cuáles son las variables dependientes y, en caso de que no se necesiten algunas de ellas, reducir la dimensionalidad de la serie temporal.

Para realizar lo comentado anteriormente, utilizaremos la técnica PCA explicada en la sección 3.3.1 usando la librería *scikit-learn*. Una vez aplicado el algoritmo PCA a nuestra serie temporal, podemos observar el porcentaje de importancia que cada variable representa en la serie temporal tal y como se puede observar en la figura 4.6.

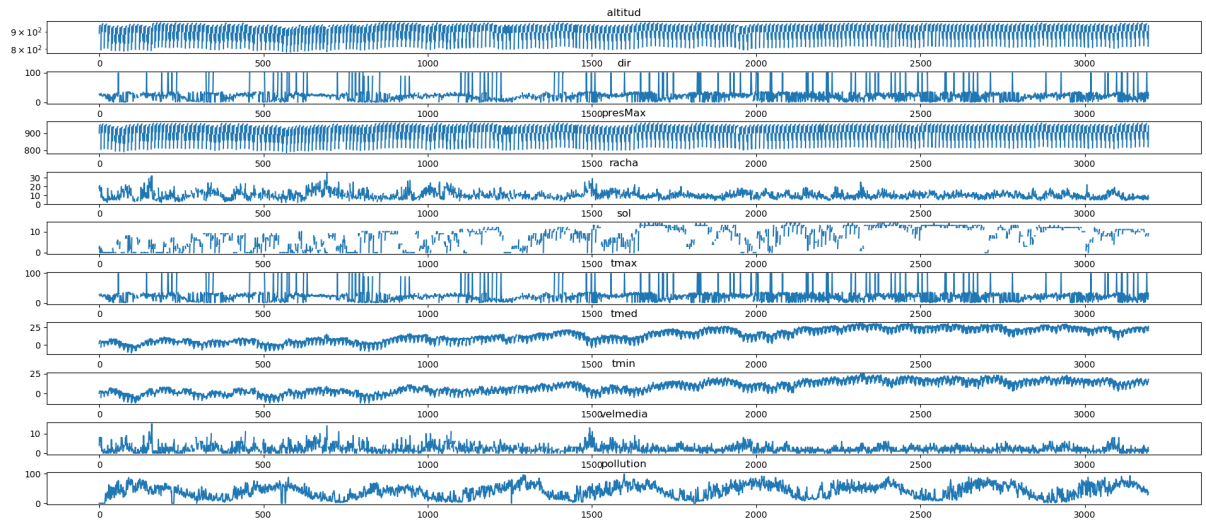

Figura 4.5 Gráfica en la que se muestra los componentes de los que se compone la serie temporal descargada de la fuente AEMET integrados con los datos de contaminación..

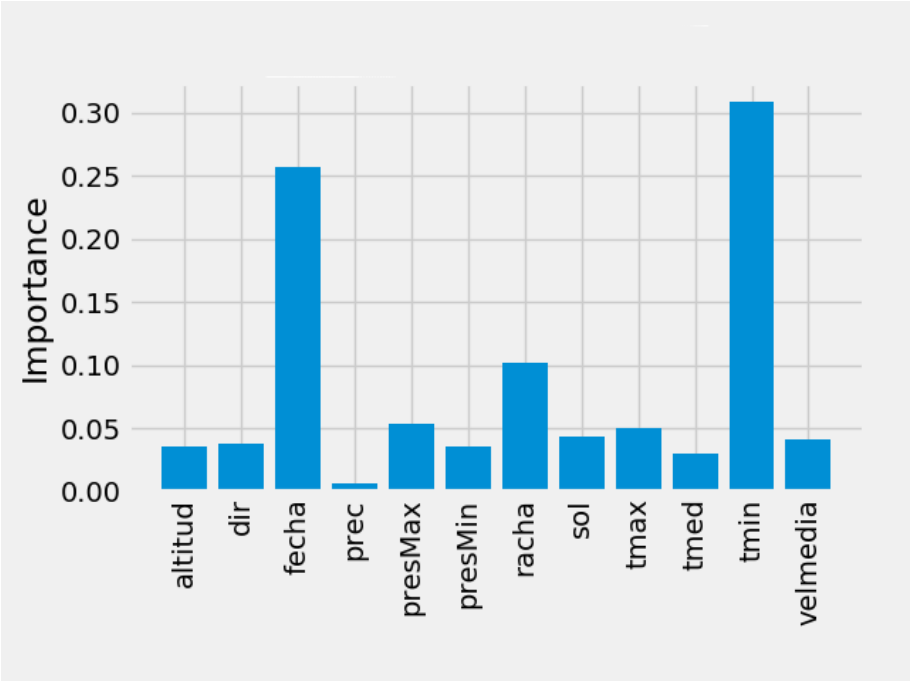

Figura 4.6 Gráfica de barras con el porcentaje de importancia de cada variable..

Por ello, analizando el resultado del algoritmo PCA, y con el objetivo de reducir la dimensionalidad de nuestra serie temporal, se eligen las siguiente variables: dir, fecha, racha, presMax, tmax, tmin y velmedia. Con lo cual, nuestra serie temporal gráficamente sería tal y como se ilustra en la figura 4.7.

En definitiva, en esta sección se ha explicada cómo hemos conseguido reducir la

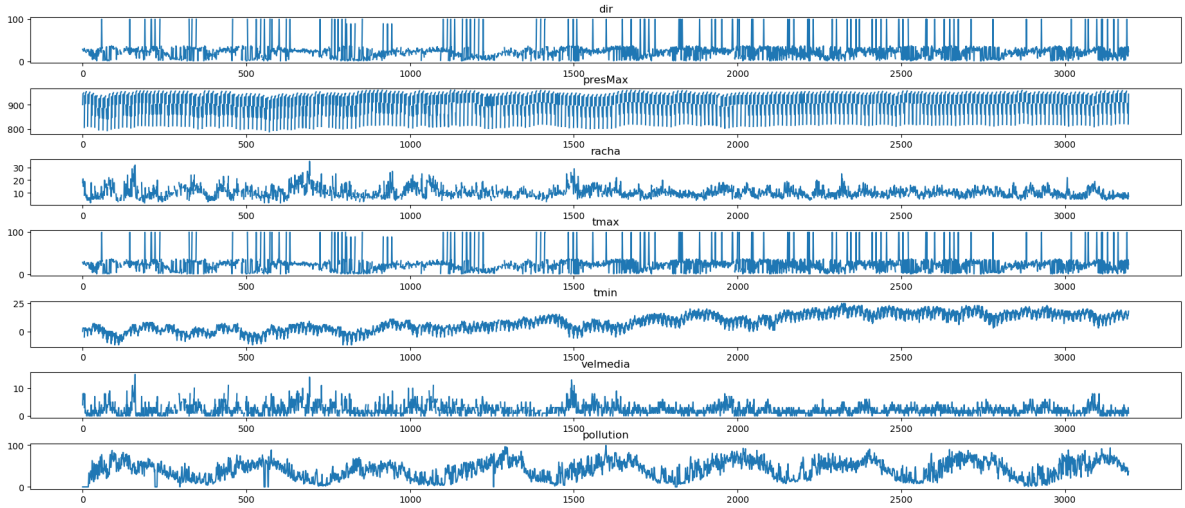

Figura 4.7 Gráfica en la que se muestra los componentes principales de la serie temporal con las características de meteorología y de contaminación..

dimensionalidad de la serie temporal. Las características eliminadas en nuestra serie no afectan al resultado final pero, sin embargo, sí que aumenta el tiempo de computación ya que nuestro modelo tendría que procesar más variables. Por lo tanto, ahora tendremos un modelo más simple y ligero, que se traduce en una reducción en el tiempo de computación.

4.4 Correlación entre variables

Dado que tenemos información de varios años, vamos a realizar agrupaciones de los datos de contaminación por meses y obtendremos las medianas de estos. De esta manera, clarificar el contenido de la serie temporal en los niveles de contaminación. La elección de obtener la mediana de los datos radica en que esta medida es más robusta, además, es la medida que se ha usado para añadir a los valores ausentes, como ya se ha explicado en la sección 4.1.

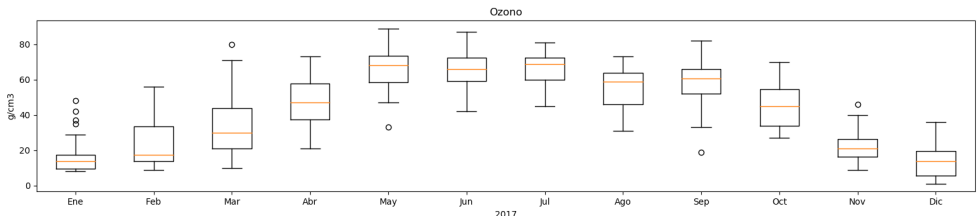

Figura 4.8 Boxplot en la que se muestra las medianas mensuales de los niveles del Ozono..

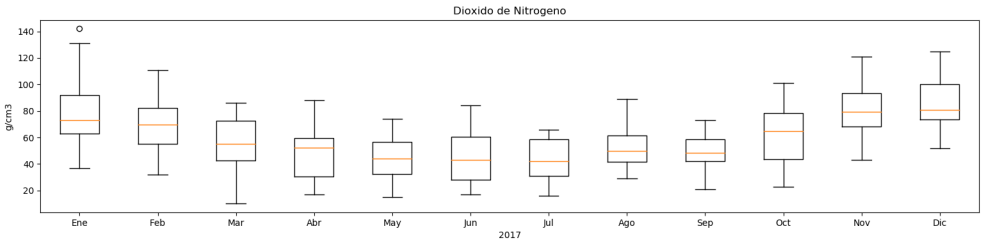

Figura 4.9 Boxplot en la que se muestra las medianas mensuales de los niveles del dióxido de nitrógeno.

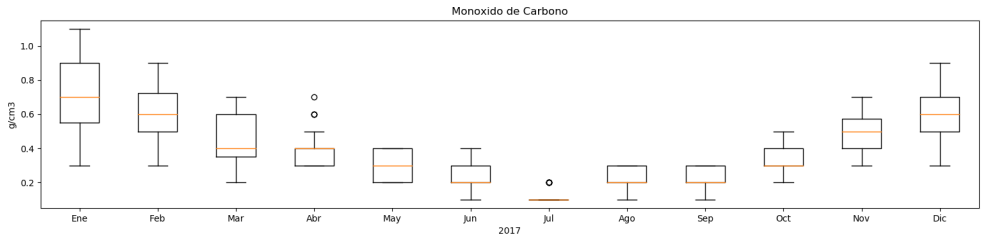

Figura 4.10 Boxplot en la que se muestra las medianas mensuales de los niveles del monóxido de carbono.

En las figuras 4.8, 4.9 y 4.10 se muestran las medianas (líneas naranjas) de la serie temporal que representan los niveles de contaminación del O_3, NO_2 y CO, respectivamente, para el año 2017 para facilitar el estudio de estas. En ellas se puede observar que los comportamientos de los niveles de contaminación difiere el uno del otro. Además, en la mayoría de los casos existe la presencia de valores atípicos (*outliers*, de su apelativo en inglés) en algunos de los meses y también distribuciones asimétricas.

Por otro lado, podemos observar que los niveles del NO_2 y del CO en verano son más bajos que en invierno. Sin embargo, los niveles de O_3 son mucho más altos en verano que en invierno.

Como tenemos datos de temperatura, se ilustrará a continuación una figura en la que se muestra la temperatura máxima junto con la serie temporal de los niveles de contaminación. De esta manera, añadiremos más solidez a la hipótesis que se ha expuesto anteriormente.

Una vez que se ha representado la correlación entre la temperatura máxima y las observaciones de los gases contaminantes del O_3, NO_2 y CO, respectivamente, en las figuras 4.11, 4.12 y 4.13, podemos postular por hipótesis nula que las altas temperaturas medioambientales pueden afectar directamente a los gases contaminantes.

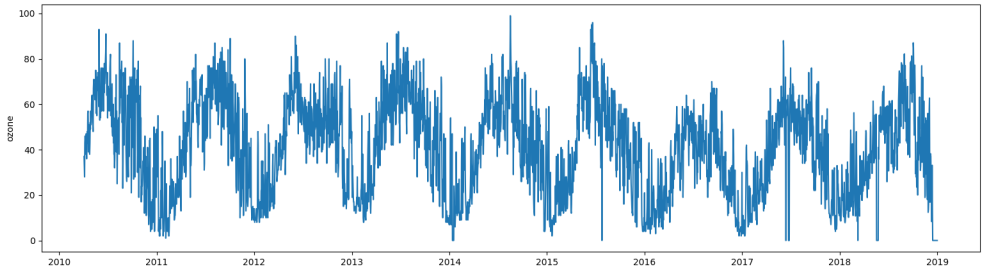

Figura 4.11 Correlación en la que se muestra las medianas mensuales de los niveles del Ozono junto a la temperatura máxima..

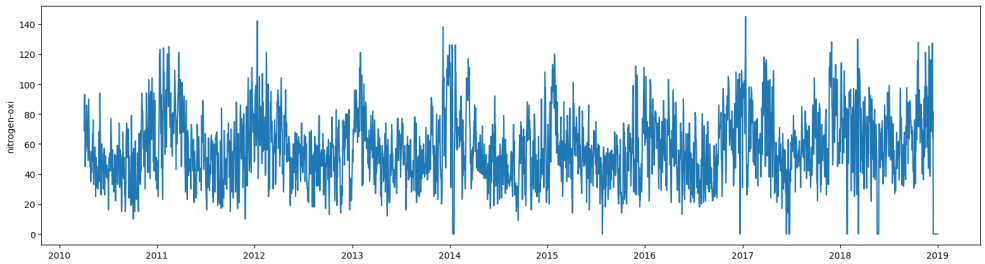

Figura 4.12 Correlación en la que se muestra las medianas mensuales de los niveles del dióxido de nitrógeno junto a la temperatura máxima.

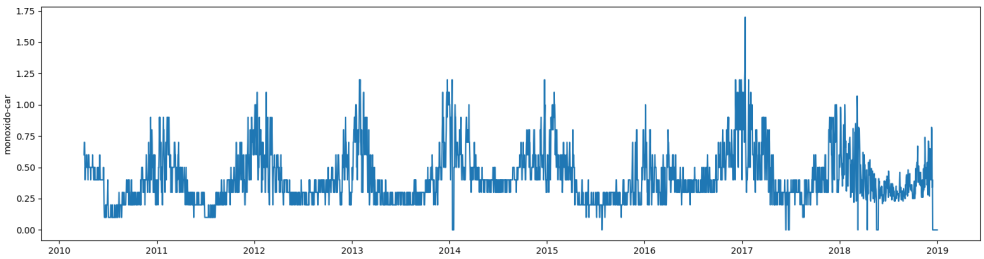

Figura 4.13 Correlación en la que se muestra las medianas mensuales de los niveles del monóxido de carbono junto a la temperatura máxima.

5 Resultados

En este capítulo se expondrán los resultados obtenidos del trabajo realizado en este estudio. Por lo tanto, se marca como objetivo mostrar todos los resultados que se obtienen de las diferentes técnicas de aprendizaje automático. De esta manera, se podrá comparar estos resultados obtenidos y realizar una valoración final.

Como ya se ha comentado anteriormente en la sección 3, para obtener los resultados se ha realizado los siguientes pasos:

- En primer lugar, se han analizado los datos para entender mejor el problema y poder llegar así a una solución concluyente.
- Después de analizar los datos, se ha investigado las fuentes de datos que nos pueden ofrecer información relevante para ayudar a encontrar la solución. En estos datos, se encuentran el pronóstico meteorológico de la ciudad de Madrid y Segovia, descargados de la fuente *AEMET*, para complementarlos con los datos de contaminación que nos ha ofrecido del Ayuntamiento de Madrid.
- En el siguiente paso, se ha realizado un preprocesado de los datos para que estos tengan un formato correcto y no haya huecos sin información en estos.
- Por último, se ha realizado un estudio con diferentes técnicas de aprendizaje automático y así decidir qué técnica es la mejor para llegar a obtener el mejor pronóstico.

Para realizar el entrenamiento de los modelos de las diferentes técnicas de aprendizaje automático usadas, los datos disponibles han sido divididos en dos grupos para poder así pronosticar los niveles de contaminación de Madrid usando las siguientes funciones de pérdida (*loss function*, en su acrónimo en inglés):

- En el primer grupo, el conjunto de datos se denomina conjunto de entrenamiento. Este conjunto es el 85 % del conjunto de datos total disponible y está compuesto por todas las medidas de nivel de contaminación y variables meteorológicas. En el caso de la técnica de *random forest* este porcentaje de datos se tomarán aleatoriamente de la serie temporal, mientras que para la técnica LSTM, las observaciones estarán comprendidas entre el 1 de Enero de 2010 y el 15 de Julio de 2017.

- El segundo grupo, el conjunto de datos se denomina conjunto de test y está compuesto por los datos restantes, es decir, el 15 % del conjunto total. Este conjunto de test contiene los valores de niveles de contaminación y variables meteorológicas, que al igual que en el conjunto de entrenamiento, para *random forest* se tomarán valores aleatorios del 15 % restante del conjunto total y para la técnica LSTM, los valores estarán comprendidos entre el 15 de Julio de 2017 y el 31 de Diciembre de 2018. La finalidad de este grupo es poder evaluar la validez del modelo creado usando el Error Cuadrático Medio y el Error Absoluto Medio.

5.1 Evaluación del rendimiento

Para poder continuar, se necesita exponer las métricas que nos permiten evaluar el resultado obtenido de los modelos utilizados en este trabajo y explicados anteriormente en las secciones 3.3.2, 3.3.1 y 3.3.3.

De forma que, las métricas usadas en este estudio son Error Cuadrático Medio (Mean Squared Error, MSE, de su acrónimo en inglés) y el Error Absoluto Medio (Mean Absolute Error, MAE, de su acrónimo en inglés) [4]. Las fórmulas de cada una de las métricas son:

- La fórmula de la métrica MSE es:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  donde $(y_i - \hat{y}_i)^2$ se denomina residuo e indica la diferencia entre el valor real y_i y el valor que pronostica el modelo $\hat{y}_i$, y por último, n es el número de observaciones que se pronostican en la serie temporal.

- La fórmula de la métrica MAE es:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

donde $|y_i - \hat{y}_i|$ denota el valor absoluto del error entre la observación real y_i y la observación que pronostica el modelo $\hat{y}_i$, y por último, al igual que en MAE, n es el número de observaciones que se pronostican en la serie temporal estudiada.

Ya hemos comentado que estas métricas nos permiten evaluar nuestro modelo de pronóstico. En el caso de las métricas que se usan en este estudio, tanto para MSE como para MAE, son usadas para medir la inconsistencia del modelo entre el valor pronosticado $\hat{y}$ y la etiqueta actual y. Los valores que devuelven estas técnicas de evaluación no son negativos y miden la robustez del modelo, además, cuanto menor sea el valor, más robusto será el modelo. En otras palabras, unos valores más bajos de MSE y MAE, nos indicarán que la serie temporal pronosticada es más análoga a la serie temporal observada.

5.2 Resultados con Descomposición STL

Teniendo en cuenta lo previamente analizado de los niveles de contaminación y en las figuras 4.8,4.9 y 4.10 podemos observar el ciclo anual de nuestras series temporales. Aplicando la descomposición STL se pueden observar fácilmente las tendencia tal y como se muestra en las figuras 5.1,5.2 y 5.3.

Como podemos observar en las gráfica que se ilustran en las figuras 5.1 y 5.2 la tendencia del O_3 y NO_2, respectivamente, tiene una regularidad bastante marcada con algunas excepciones. Sin embargo, la tendencia del CO no tiene una regularidad, teniendo periodos irregulares. Esto añade dificultad al modelo que hace el pronóstico.

Para tener una visión más clara de la descomposición STL, se ilustrará a continuación en las figuras 5.4, 5.5 y 5.6 las gráficas que muestran la descomposición de la serie temporal para el periodo de la serie comprendido entre 1 de Enero de 2017 y 31 de Diciembre 2017.

En estas figuras 5.4, 5.5 y 5.6, podemos comparar el comportamiento entre estos gases y observar que la tendencia del O_3 es contraria con respecto al comportamiento de los gases NO_2 y CO. Como se puede observar, durante los meses de verano, los niveles de contaminación del O_3 aumentan mientras que, los niveles de NO_2 y CO.

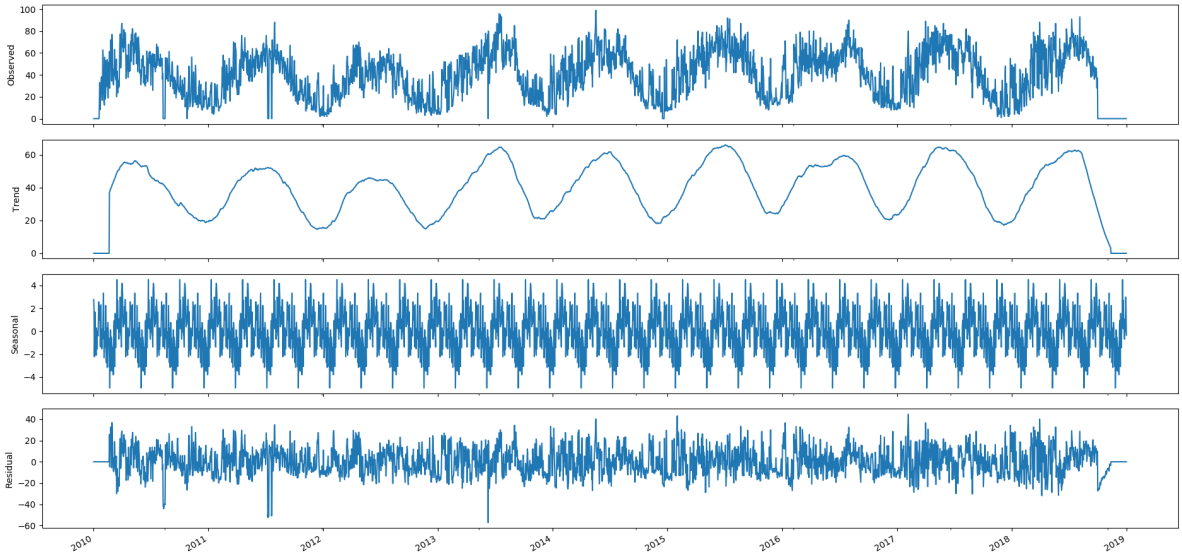

Figura 5.1 Descomposición Estacional y de Tendencias con Loess de la serie temporal del O_3.

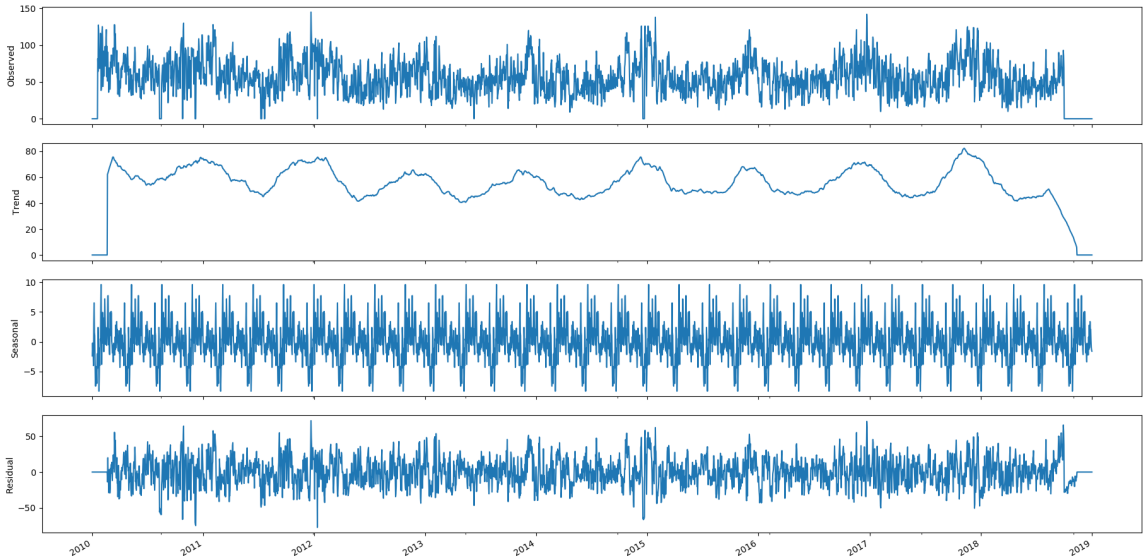

Figura 5.2 Descomposición Estacional y de Tendencias con Loess de la serie temporal del NO_2.

Tanto la tendencia como la componente estacional, serán las series temporales de entrada usando la técnica de aprendizaje automático Random Forest, explicada en la sección 3.3.2, y la técnica de LSTM de aprendizaje profundo, explicada en la sección 3.3.3. La técnica de descomposición STL suele usarse principalmente para estudiar las series temporales y explorar cambios a lo largo del tiempo, también se puede usar para realizar modelos de pronóstico simples [15] y, de esta manera, ayudar aún más si cabe a comprender el

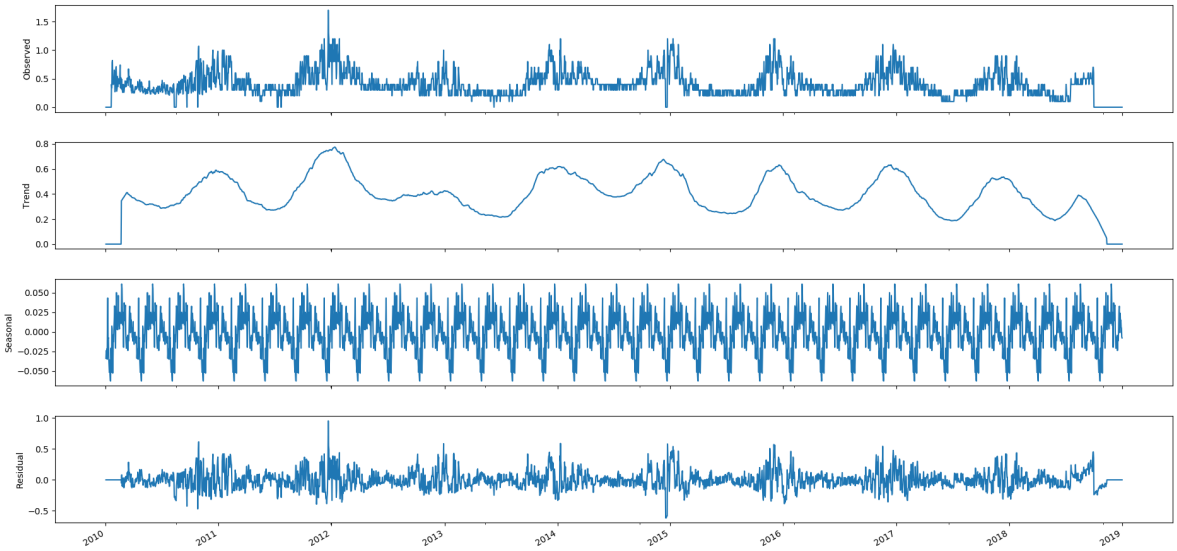

Figura 5.3 Descomposición Estacional y de Tendencias con Loess de la serie temporal del CO.

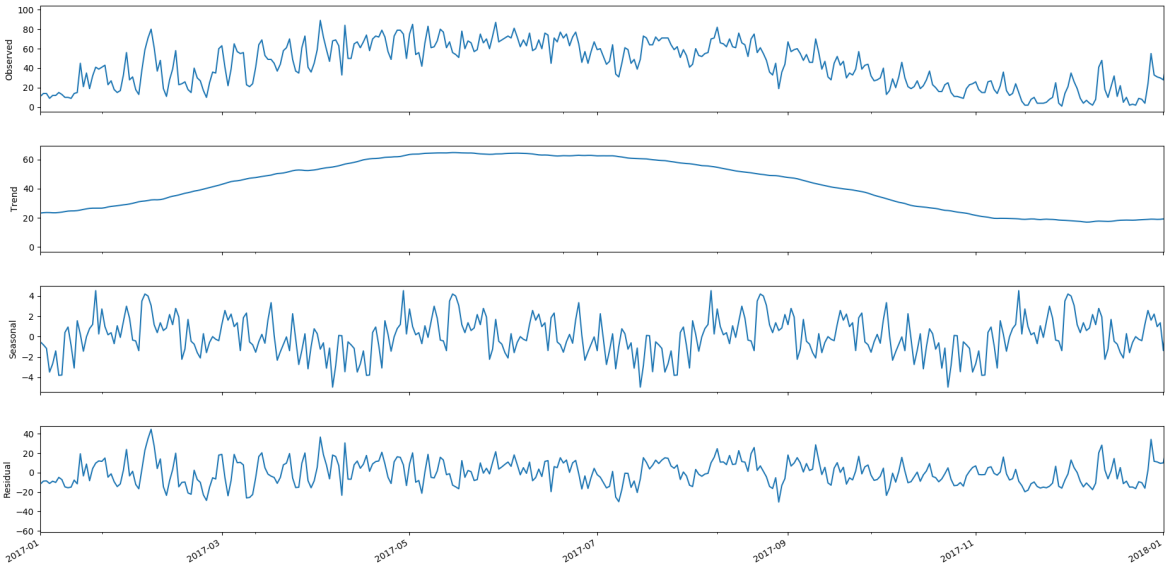

Figura 5.4 Descomposición Estacional y de Tendencias con Loess de la serie temporal del O_3 para el periodo de la serie comprendido entre 1 de Enero de 2017 y 31 de Diciembre 2017.

comportamiento de la serie temporal.

El resultado de usar estas técnicas sobre las series temporales que han sido descompuestas mediante el método presentado en esta sección, se explicará en las siguientes secciones. Además, se realizará la comparación con la serie temporal sin descomponerla con el método STL.

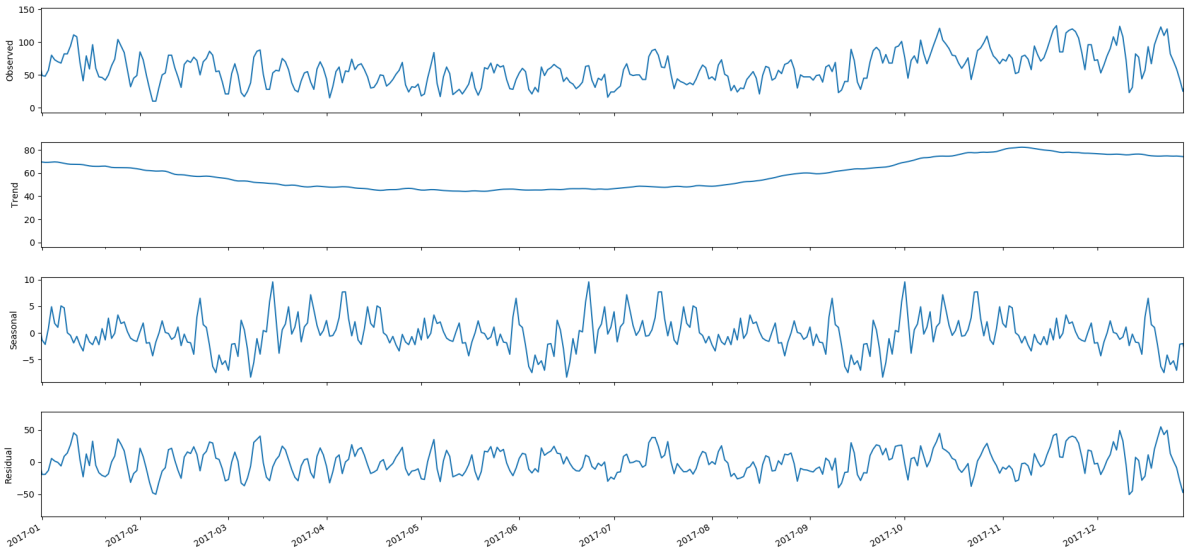

Figura 5.5 Descomposición Estacional y de Tendencias con Loess de la serie temporal del NO_2 para el periodo de la serie comprendido entre 1 de Enero de 2017 y 31 de Diciembre 2017.

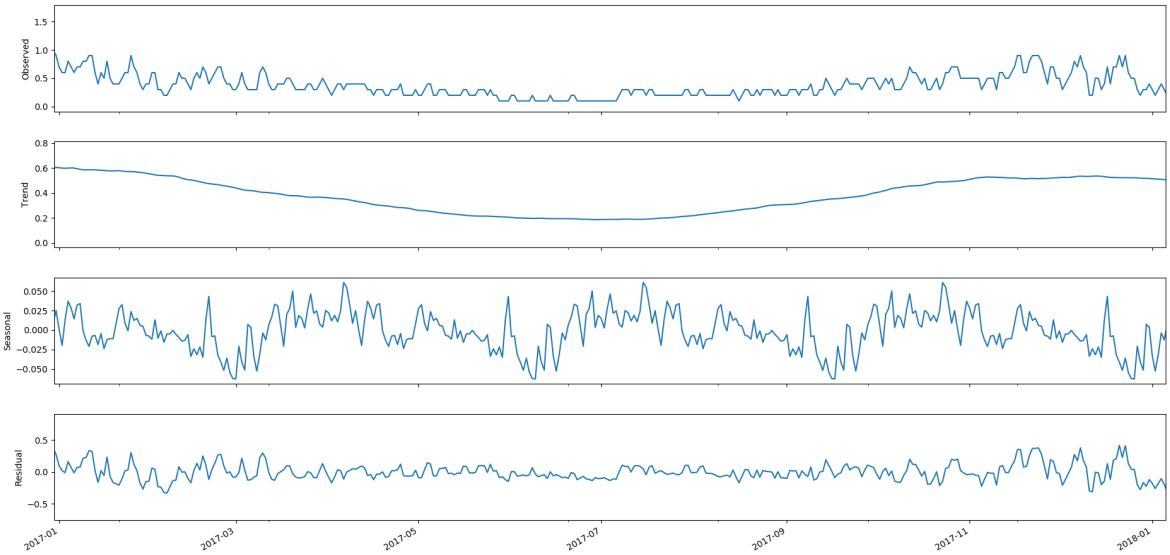

Figura 5.6 Descomposición Estacional y de Tendencias con Loess de la serie temporal del CO para el periodo de la serie comprendido entre 1 de Enero de 2017 y 31 de Diciembre 2017.

5.3 Resultados con Random Forest

Como ya se comentó anteriormente en la sección 3.3.2, utilizaremos la técnica de aprendizaje automática *random forest*, técnica basada en árboles de decisión, para implementar un modelo que realice el pronóstico de los niveles de contaminación. Este tipo de árboles

de decisión son rápidos de implementar, eficientes en grandes bases de datos y, además, se obtiene con ellos unos resultados óptimos en el pronóstico. El árbol de decisión que se implementará en este trabajo de fin de máster sera de regresión, ya que los niveles de contaminación a pronosticar son continuos.

Basándonos en la figura 4.7, que muestra la serie temporal compuesta por las variables meteorológicas (*dir, fecha, racha, presMax, tmax, tmin* y *velmedia*) como entrada y como resultado el nivel de contaminación de O_3, NO_2 y CO.Se realizará un entrenamiento por cada elemento de contaminación. Por lo tanto, tendremos como resultado tres modelos separados que realizarán el pronóstico de un nivel de contaminación concreto.

Por lo tanto, tendremos una configuración de *Random Forest* con 2000 árboles de decisión que se ajustarán lo máximo posible al conjunto de entrenamiento dado. Con esta configuración, se mejorará el pronóstico y se evitará el sobreajuste.

Las figuras 5.7, 5.8 y 5.9 muestran la serie temporal de O_3, NO_2 y CO, respectivamente, con la representación de los pronósticos (puntos rojos) realizados por el modelo. Como se explicó al comienzo del capítulo, el conjunto de test que se ha usado contenía valores escogidos aleatoriamente del 25 % del total del conjunto de datos y la predicción que ha realizado se encuentra dentro del *intervalo de confianza*. Después de realizar 25 ejecuciones independientes sobre el conjunto de entrenamiento (*epochs*, de su apelativo en inglés), los valores MSE son:133.90 ± 4.0087 para el O_3, 308.69 ± 8.05042 para el NO_2 y 0.11354 ± 0.016116 para el CO. Por otro lado, los valores de MAE son: 8.12 ± 0.04 para el O_3, 12.56 ± 0.3 para el NO_2 y 0.08 ± 0.01 para el CO.

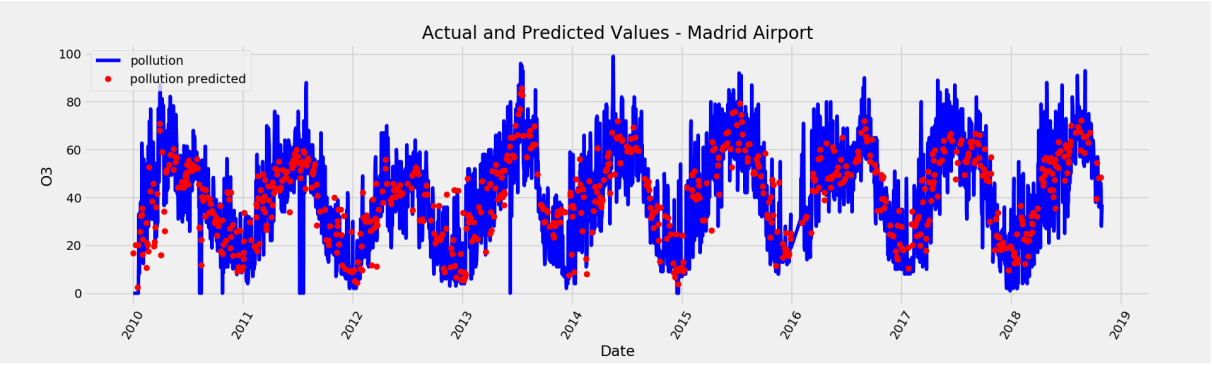

Figura 5.7 Predicción de niveles de contaminación con Random Forest para el Ozono.

Los valores de MSE y MAE son previsiblemente elevados debido a que la predicción

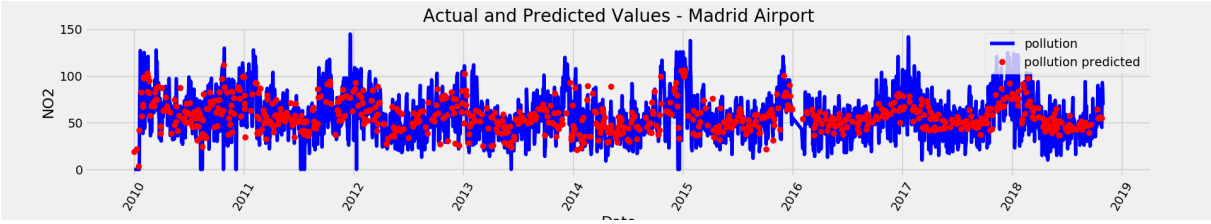

Figura 5.8 Predicción de niveles de contaminación con Random Forest para el Dióxido de Nitrógeno.

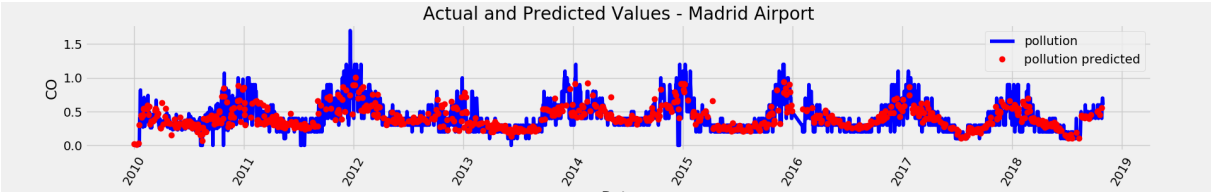

Figura 5.9 Predicción de niveles de contaminación con Random Forest para el Monóxido de Carbono.

para las observaciones con picos altos de niveles de contaminación no son triviales de predecir. Estos casos especiales donde las observaciones de niveles de contaminación recogidas son repentinamente más altas que en el instante inmediatamente anterior, son casos a estudiar en más profundidad.

5.4 Resultados con Random Forest y STL

En esta sección se presentan los resultados aplicando el modelo de *random forest* con la descomposición STL, tal y como se mencionó en la sección anterior 5.2. En este caso, las predicciones que se han obtenido han sido utilizando una de las series temporales que la descomposición STL devuelve, es decir, la serie temporal de la tendencia. Los resultados para la MSE son: 6.2616 ± 4.0087 para el O_3, 0.6944 ± 0.035042 para el NO_2 y 0.000135 ± 0.0000676 para el CO. Por otro lado, los valores de MAE son: 0.74 ± 0.014 para el O_3, 0.41 ± 0.0073 para el NO_2 y $0.00000268 \pm 0.00000041$ para el CO.

Como se puede apreciar en las figuras 5.10, 5.11 y 5.12 los resultados no son óptimos con respecto a los valores reales aunque los valores de MSE y MAE hayan sido mejores. Esto es debido a que los valores pronosticados se han ajustado a la tendencia de la serie temporal perfectamente. Sin embargo, esto no se ajusta a la realidad de los niveles de contaminación, ya que las observaciones tomadas registran en ocasiones valores muy por

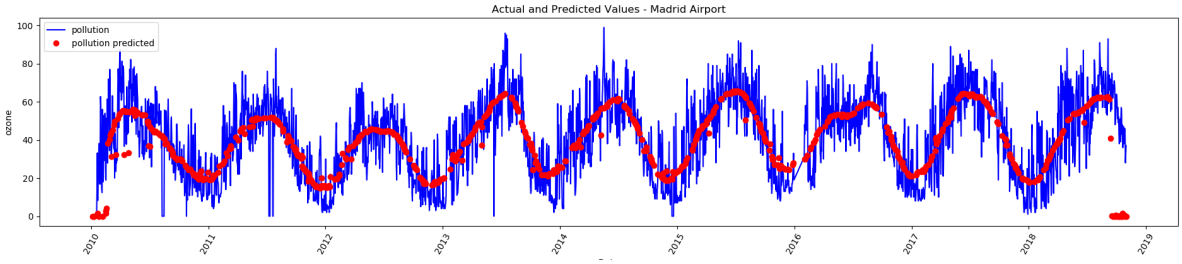

Figura 5.10 Predicción de niveles de contaminación con respecto a la tendencia usando Random Forest para el Ozono.

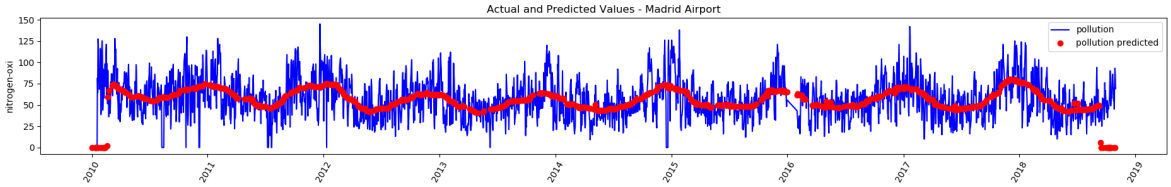

Figura 5.11 Predicción de niveles de contaminación con respecto a la tendencia usando Random Forest para el Dióxido de Nitrógeno.

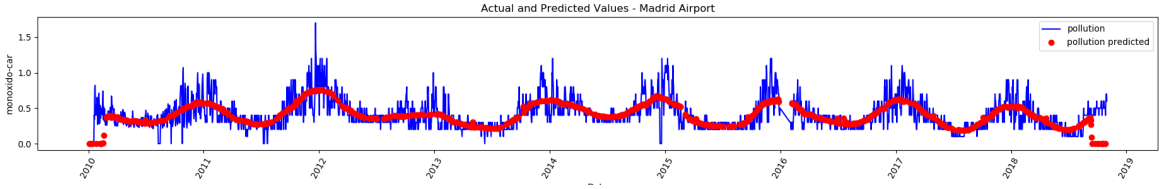

Figura 5.12 Predicción de niveles de contaminación con respecto a la tendencia usando Random Forest para el Monóxido de Carbono.

encima de la tendencia de la serie temporal.

5.5 Resultados con Redes Neuronales Recurrentes LSTM

En esta sección aplicaremos las redes neuronales recurrentes para resolver el problema que en este trabajo se plantea usando *Keras*. Se usará para ello la técnica de aprendizaje profundo de LSTM, la cual es capaz de reutilizar los estados internos para usarlos como entrada secuencial en el procesamiento, como ya se explicó en la sección anterior 3.3.3.

Apoyándonos en la figura 4.7, que se ilustra la serie temporal compuesta por las variables meteorológicas (*dir, fecha, racha, presMax, tmax, tmin* y *velmedia*) como entrada y como resultado el nivel de contaminación de O_3, NO_2 y CO. Se realizará un modelo distinto para cada elemento de contaminación, ya que los niveles de contaminación tienen tendencias y comportamiento distintos.

Para trabajar con esta técnica de red neuronal recurrente, es necesario normalizar los datos de entrada. Esto es debido a que las redes neuronales recurrentes son muy sensibles a los datos que no son cercanos a los valores que las funciones de activación emplean. Se normalizaran los datos utilizando la función *MinMaxScaler* de la librería *scikit-learn*. Esta función utiliza la fórmula:

$$MinMaxScaler(ts) = \frac{x_i - min(ts)}{max(ts) - min(ts)}$$

donde ts es la serie temporal que normalizaremos, x_i es el valor a normalizar, $min(ts)$ el valor mínimo de la serie temporal y $max(ts)$ el valor máximo de la serie temporal.

Una vez normalizada la serie temporal, obtendremos valores que estarán comprendidos entre 0 y 1. De esta manera, tendremos la serie temporal preparada para utilizarla en la red neuronal recurrente.

El modelo que se diseña para esta red neuronal recurrente, con el objetivo de minimizar los valores de MSE y MAE, con una capa oculta que contiene 50 neuronas y una neurona en la capa de salida que devolverá la predicción del nivel de contaminación, con la función de activación *tanh*. La capa de entrada estará compuesta un paso y siete características (*dir, fecha, racha, presMax, tmax, tmin* y *velmedia*). Además, se usará MAE como función de pérdida y la versión *Adam* del descento por el gradiente estocástico.

Por otro lado, el model se entrenará 25 veces con un tamaño de agrupación (*batch*, en su apelativo en inglés) de 90. Hay que tener en cuenta que el estado interno de la red neuronal recurrente LSTM en *Keras* se inicializa al final de cada *batch*. Por esa razón, el estado interno de esta red neuronal está en función del número de días que contiene la serie temporal.

La evolución de la función de pérdida MAE, se ilustra en la figura 5.13, cuyo valor de salida para el conjunto de entrenamiento tiende a 0.16 de tasa de error y para el conjunto de test tiende a 0.19.

Los valores de pronóstico que devuelve la red neuronal recurrente se muestran en las figuras 5.14,5.15 y 5.16 para los niveles de contaminación O_3, NO_2 y CO, respectivamente.

En estas ilustraciones, podemos observar que el 25 % de los datos de test están comprendidos entre las fechas 15 de Julio de 2017 y el 31 de Diciembre de 2018. En ellas, podemos observar que los valores de pronóstico han mejorado con respecto a los niveles de contaminación

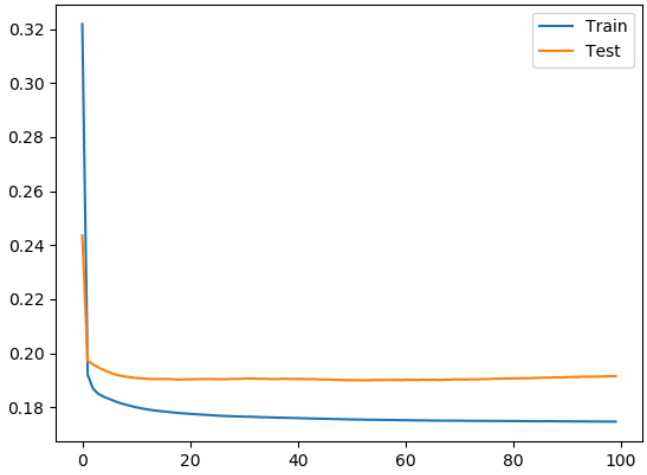

Figura 5.13 Evolución de la función de pérdida MAE para la salida de la red neuronal recurrente LSTM..

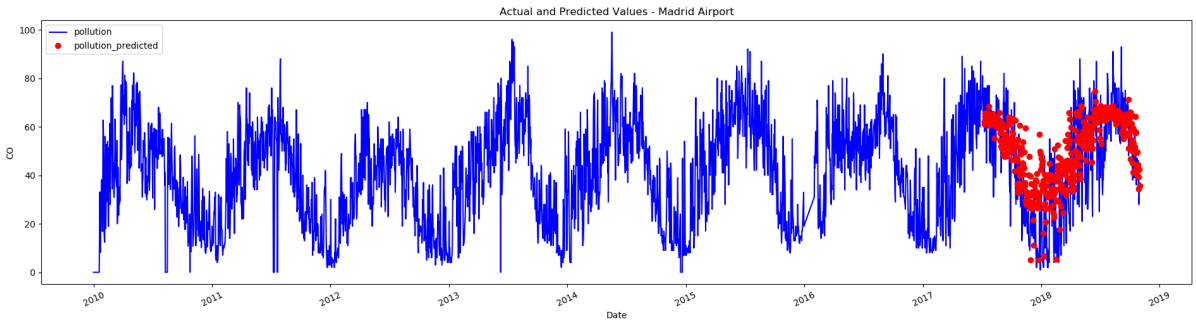

Figura 5.14 Predicción de niveles de contaminación usando LSTM para el Ozono.

de O_3 y CO pero han empeorado considerablemente en el caso de NO_2. Los resultados de la MAE son: 6.37 ± 0.4 para O_3, 11.37 ± 0.6 para NO_2 y 17.14 ± 0.3 para CO. No obstante, podemos observar que la ilustraciones presentadas en esta sección tienen algo en común y es que, para las observaciones con niveles de contaminación altos repentinos, la red neuronal recurrente tampoco es capaz de predecir su valor al igual que ocurría con los modelos de *random forest* creados.

5.6 Resultados con LSTM y STL

Al igual que en la sección anterior 5.4, vamos a presentar los resultados aplicando el modelo creado en la sección anterior 5.5. Entonces, se realizarán las predicciones usando una de

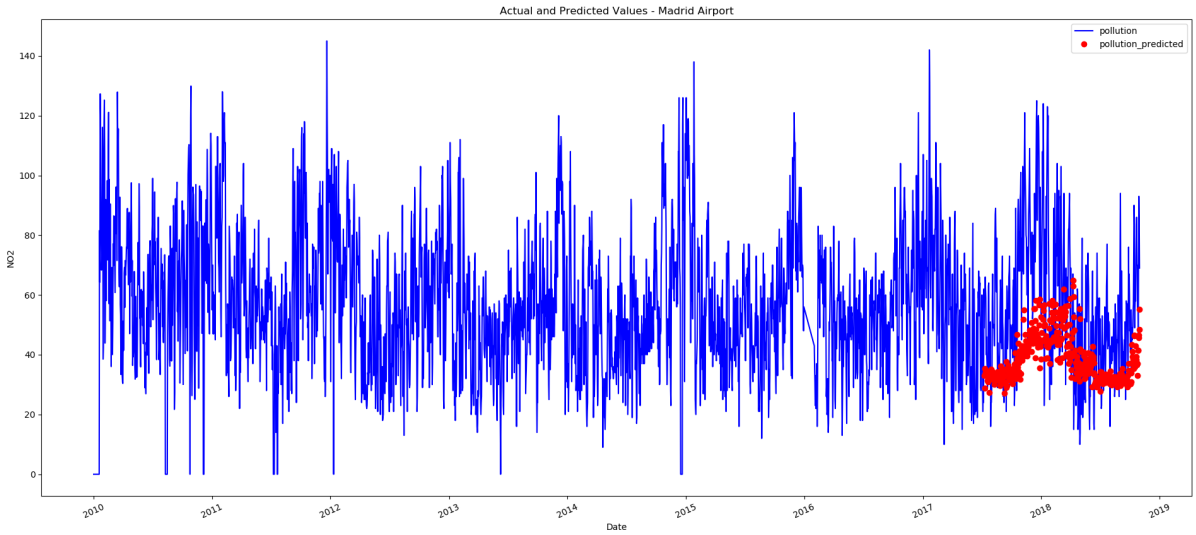

Figura 5.15 Predicción de niveles de contaminación usando LSTM para el Dióxido de Nitrógeno.

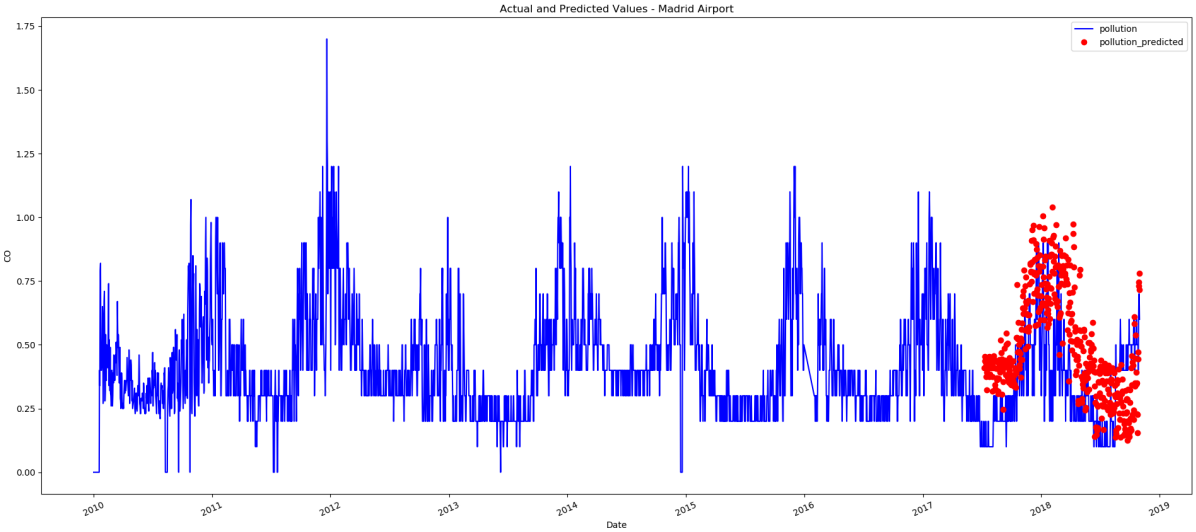

Figura 5.16 Predicción de niveles de contaminación usando LSTM para el Monóxido de Carbono.

las series temporales que la descomposición STL devuelve, para este experimento, la serie temporal de la tendencia.

Una vez realizado el entrenamiento de la red neuronal recurrente con el modelo creado en la sección 5.5, los resultados para MAE son: 1.75 ± 0.02202 para el O_3, 2.61 ± 0.05205 para el NO_2 y 0.758 ± 0.001549 para el CO. Al igual que en el caso de la sección 5.4, la

serie tiene unos valores de MAE que tiende a cero aunque el valor de error que se calcula es con respecto a la serie temporal de la tendencia y no con respecto a los valores reales de niveles de contaminación.

5.7 Comparativa universal de metodologías

Como última sección en este capítulo, se presentará una comparativa universal de los resultados obtenidos realizando las metodologías expuestas en secciones anteriores. Para ello, vamos a realizar el *test de rangos con signo de Wilcoxon* explicado en la sección 3.2.2 con el interés de:

- Medir la eficiencia de las metodologías de aprendizaje automático y profundo expuestas en este trabajo.
- Establecer qué técnica de las planteadas en este trabajo son las mejores para continuar con el trabajo de pronóstico de los niveles de contaminación de Madrid.

A continuación se muestran las tablas 5.1, 5.2 y 5.3 que muestran la comparativa de las medidas de MSE y MAE de las metodologías usadas en este trabajo. Como se puede observar en dichas tablas, las técnicas basadas en aprendizaje automático han obtenido unos valores más bajos para MSE y MAE. Aunque para los niveles de contaminación del *CO*, podemos observar que con la técnica de aprendizaje automático LSTM se obtiene mejores resultados que con Random Forest.

Por otro lado, un punto a tener en cuenta es que las técnicas de aprendizaje automático también han obtenido unos valores más bajos cuando se han usado sobre la descomposición STL con respecto a la serie temporal no descompuesta. Como ya hemos visto en una sección anterior, al eliminar la componente de aleatoriedad y aplicar la técnica únicamente sobre la tendencia de la serie temporal, los resultados de MSE y MAE son más óptimos.

Esta comparativa general de las técnicas de aprendizaje automático y profundo usadas en este trabajo nos da una visión general de los resultados obtenidos. No obstante, el resultado más relevante que se deduce al observar estas tablas es que para cada serie temporal de nivel de contaminación se necesita crear un modelo distinto en el caso de las técnicas de aprendizaje profundo. En el caso contrario, *Random Forest* se adapta perfectamente y es capaz de realizar un mejor pronóstico, ya que la técnica de *Random Forest* implementa en

Tabla 5.1 Mean Squared Error de Random Forest y Random Forest con descomposición STL para 25 ejecuciones independientes de la serie temporal.

Serie Temporal	RF	RF+STL
O_3	66.0 ± 0.0087	55.68 ± 1.0037
NO_2	91.0 ± 0.05042	78.21 ± 2.022
CO	156.0 ± 0.86116	95.7 ± 1.8506

Tabla 5.2 Mean Squared Error de LSTM y LSTM con descomposición STL para 25 ejecuciones independientes de la serie temporal.

Serie Temporal	LSTM	LSTM+STL
O_3	105.0 ± 0.1157	78.0 ± 5.02202
NO_2	101.0 ± 0.05042	84.0 ± 2.05205
CO	77.0 ± 0.02013	45.0 ± 3.101549

Tabla 5.3 Mean Absolute Error de Random Forest y LSTM con descomposición STL para 25 ejecuciones independientes de la serie temporal.

Time Series	RF	LSTM
CO	0.119 ± 0.01	7.14 ± 0.3
NO_2	12.56 ± 0.3	11.37 ± 0.6
O_3	12.303 ± 0.04	14.37 ± 0.4

cada subnodo del árbol un árbol de decisión, abordando todos los casos posibles para cada serie temporal.

6 Conclusiones

En el trabajo final de máster que se presenta, se ha llevado a cabo la obtención de datos, tanto haciendo uso de la API de AEMET como los obtenidos del Ayuntamiento de Madrid, se ha hecho el preprocesado de estos y, por último, creado un modelo para realizar el pronóstico de la serie temporal estudiada. Como ya se comentó al comienzo de este documento, los altos niveles de contaminación en las grandes ciudades es preocupante y eso hace que se creen protocolos de control de tráfico para evitar sobrepasar dichos niveles. Es por ello, que se necesita realizar un pronóstico de los niveles de contaminación que nos ayude a anticiparnos para realizar una decisión correcta en la aplicación de los protocolos.

Para este trabajo de fin de máster, se han obtenido datos climatológicos de las estaciones meteorológicas disponibles en la fuente de información de AEMET OpenData a través de la API que ofrece. Además de estos datos, el Ayuntamiento de Madrid ha proporcionado los niveles de contaminación recogidos en la ciudad. Estas dos series temporales de las fuentes de información de las que se han recogido los datos, están comprendidas entre el 1 de Enero de 2010 y el 31 de Diciembre de 2018, haciendo un total de ocho años de datos con una frecuencia diaria. Estos datos han sido preprocesados para que las variables tengan el formato correcto para el posterior análisis que se realiza de ellos.

Debido que los datos meteorológicos estaban publicados en una fuente de información, se realizó los métodos necesarios para hacer la obtención de estos y almacenarlos en la base de datos interna para disponer de ellos los más sencillo posible. Tal y como se comentó en el capítulo 2, los datos se han almacenado en una base de datos NoSQL llamada *MongoDB*.

Con el objetivo de realizar el análisis de los datos, se han utilizado técnicas de aprendizaje automático y profundo con el fin de realizar comparativas entre ellas y, de esta manera,

comprender el funcionamiento de dichas técnicas en profundidad. Las técnicas usadas han sido: la *descomposición estacional y de tendencias con Loess*, el test de rangos con signo de *Wilcoxon*, análisis de componentes principales, *Random Forest* y, por último, Redes Neuronales Recurrentes *Long Short-Term Memory*. Con esto, hacemos una demostración de que la técnica de *Random Forest* se ha adaptado mejor a las observaciones de los niveles de contaminación aunque en los picos de niveles de contaminación, tanto para *Random Forest* como para *LSTM*, las predicciones no han sido lo suficientemente óptimas. Estos casos especiales de niveles de contaminación son caso de estudio y razón para continuar el trabajo en solventar este problema planteado en este trabajo.

Por otro lado, se ha utilizado el lenguaje de programación *Python*, además de utilizar la librería *scikit-learn* y el *framework Keras* bajo el *backend TensorFlow*. Estas herramientas nos han ayudado a realizar, tanto el preprocesado de los datos como el modelado de las técnicas de aprendizaje automático debido a que disponen de una gran variedad de métodos implementados para realizar todo lo comentado.

Además del código adjunto a la memoria, se han realizado varios *scripts* necesarios para realizar mantenimiento de la base de datos y copias de seguridad de esta para evitar una pérdida de datos.

Finalmente, se puede concluir que se han cumplido los objetivos propuestos para este estudio, es decir, se ha creado una serie temporal integrando los datos meteorológicos de AEMET y los proporcionados por el Ayuntamiento de Madrid, se han creado unos modelos de aprendizaje automático y profundo y, por último, se ha ampliado el conocimiento sobre la aplicación de estas técnicas en un problema real.

6.1 Trabajo futuro

Como se ha podido intuir a lo largo de este trabajo de fin de máster, aún queda mucho por hacer para obtener un pronóstico óptimo con el fin de que se pueda aplicar dicho pronóstico en una aplicación real. Es por ello que, se continuará con este estudio en una futura tesis que comenzará en los próximos meses y en esta se profundizará en todos los detalles para que finalmente el pronóstico de esta serie temporal sea lo más ajustada a la realidad posible.

Por su parte, se seguirá contando con el apoyo del Centro de Investigación de Energéticas Medioambientales (CIEMAT). Además se seguirá recolectando datos, tanto los proporcionados

por el Ayuntamiento de Madrid como los que se obtienen de la fuente de información que ofrece la Agencia Estatal de Meteorología (AEMET) para poder realizar así un estudio en profundidad de las tendencias y relaciones de las condiciones meteorológicas con los niveles de contaminación.

Índice de Figuras

3.1 Descomposición Estacional y de Tendencia con Loess STL 12

3.2 Bootstrapping 21

3.3 Boosting 22

3.4 Random Forest 22

3.5 Red Neuronal Recurrente 24

3.6 LSTM 26

4.1 Gráfica Ozono 31

4.2 Gráfica Dióxido de Nitrógeno 31

4.3 Gráfica Monóxido de Carbono 31

4.4 Gráfica datos meteorológicos 32

4.5 Gráfica datos meteorológicos y contaminación 35

4.6 Análisis de componentes principales 35

4.7 Gráfica de características 36

4.8 Boxplot de las medianas mensuales ozono 36

4.9 Boxplot de las medianas mensuales dióxido de nitrógeno 37

4.10 Boxplot de las medianas mensuales monóxido de carbono 37

4.11 Correlación de las medianas mensuales de ozono y temperatura 38

4.12 Correlación de las medianas mensuales de dióxido de nitrógeno y temperatura 38

4.13 Correlación de las medianas mensuales de monóxido de carbono y temperatura 38

5.1 Descomposición STL Ozono 42

5.2 Descomposición STL Dióxido de Nitrógeno 42
5.3 Descomposición STL Monóxido de Carbono 43
5.4 Descomposición STL Ozono anual 43
5.5 Descomposición STL Dióxido de Nitrógeno anual 44
5.6 Descomposición STL Monóxido de Carbono anual 44
5.7 Predicción Random Forest Ozono 45
5.8 Predicción Random Forest Dióxido de Nitrógeno 46
5.9 Predicción Random Forest Monóxido de Carbono 46
5.10 Predicción tendencia Random Forest Ozono 47
5.11 Predicción tendencia Random Forest Dióxido de Nitrógeno 47
5.12 Predicción tendencia Random Forest Monóxido de Carbono 47
5.13 Función de Pérdida MAE para LSTM 49
5.14 Predicción LSTM Ozono 49
5.15 Predicción LSTM Dióxido de Nitrógeno 50
5.16 Predicción LSTM Monóxido de Carbono 50

Índice de Tablas

5.1 Mean Squared Error de Random Forest 52
5.2 Mean Squared Error de LSTM 52
5.3 Mean Absolute Error de RF y LSTM 52

Bibliografía

[1] Gerald Albaum, *The likert scale revisited*, Market Research Society. Journal. **39** (1997), no. 2, 1–21.

[2] P. Bedi, N. Mediratta, S. Dhand, R. Sharma, and A. Singhal, *Avoiding traffic jam using ant colony optimization - a novel approach*, International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007), vol. 1, Dec 2007, pp. 61–67.

[3] Rajesh S. Brid, *Decision tree - a simple way to visualize a decision*, May 2019.

[4] Isaac Changhau, *Loss functions in neural network*, May 2018.

[5] François Chollet et al., *Keras*, *https://keras.io*, 2015.

[6] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning, *Stl: A seasonal-trend decomposition*, Journal of official statistics **6** (1990), no. 1, 3–73.

[7] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, *Arima models to predict next-day electricity prices*, IEEE Transactions on Power Systems **18** (2003), no. 3, 1014–1020.

[8] Estela Bee Dagum and Kim Chiu, *Derivation and properties of the x11arima and census x11 linear filters*, Journal of Official Statistics **12** (1996), no. 4, 329.

[9] MongoDB Databae, *Mongodb*, May 2019.

[10] Murray M. Finkelstein, Michael Jerrett, and Malcolm R. Sears, *Traffic Air Pollution and Mortality Rate Advancement Periods*, American Journal of Epidemiology **160** (2004), no. 2, 173–177.

[11] Mark A Friedl and Carla E Brodley, *Decision tree classification of land cover from remotely sensed data*, Remote sensing of environment **61** (1997), no. 3, 399–409.

[12] John Cristian Borges Gamboa, *Deep learning for time-series analysis*, CoRR **abs/1701.01887** (2017).

[13] Google, *Tensorflow*, May 2018.

[14] Sepp Hochreiter and Jürgen Schmidhuber, *Long short-term memory*, Neural Computation **9** (1997), no. 8, 1735–1780.

[15] Rob J. Hyndman and George Anthanasopoulos, *Forecasting: principles and practice - forecasting with decomposition*, May 2018.

[16] N. B. Idris and B. Shanmugam, *Artificial intelligence techniques applied to intrusion detection*, 2005 Annual IEEE India Conference - Indicon, Dec 2005, pp. 52–55.

[17] INRIA, *scikit-learn*, May 2018.

[18] Zakaria Jaadi, *A step by step explanation of principal component analysis*, May 2019.

[19] Ian Jolliffe, *Principal component analysis*, Springer, 2011.

[20] Deepak A Kaji, John R Zech, Jun S Kim, Samuel K Cho, Neha S Dangayach, Anthony B Costa, and Eric K Oermann, *An attention based deep learning model of clinical events in the intensive care unit*, Plos One **14** (2019), no. 2, e0211057.

[21] Tae Kyun Kim, *T test as a parametric statistic*, Korean journal of anesthesiology **68** (2015), no. 6, 540.

[22] Jerome Klotz et al., *Nonparametric tests for scale*, The Annals of Mathematical Statistics **33** (1962), no. 2, 498–512.

[23] Andy Liaw, Matthew Wiener, et al., *Classification and regression by randomforest*, R news **2** (2002), no. 3, 18–22.

[24] Microsoft, *The microsoft cognitive toolkit*, May 2018.

[25] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, *Recurrent neural network based language model*, Eleventh annual conference of the international speech communication association, 2010.

[26] Guadalupe Moreno, *A la espera de un big bang de datos*, May 2019.

[27] Haşim Sak, Andrew Senior, and Françoise Beaufays, *Long short-term memory recurrent neural network architectures for large scale acoustic modeling*, Fifteenth annual conference of the international speech communication association, 2014.

[28] Jinde Shubham, *Ensemble learning - bagging and boosting*, May 2018.

[29] Python Software, *Python*, May 2018.

[30] Supasorn Suwajanakorn, Steven M. Seitz, and Ira Kemelmacher-Shlizerman, *Synthesizing obama: learning lip sync from audio*, ACM Trans. Graph. **36** (2017), no. 4, 95:1–95:13.

[31] Theano Development Team, *Theano*, May 2018.

[32] Heather E. Volk, Fred Lurmann, Bryan Penfold, Irva Hertz-Picciotto, and Rob McConnell, *Traffic-Related Air Pollution, Particulate Matter, and AutismAir Pollution, Particulate Matter, and Autism*, JAMA Psychiatry **70** (2013), no. 1, 71–77.

[33] RF Woolson, *Wilcoxon signed-rank test*, Wiley encyclopedia of clinical trials (2007), 1–3.

[34] Yuzheng Wu, Maryellen L Giger, Kunio Doi, Carl J Vyborny, Robert A Schmidt, and Charles E Metz, *Artificial neural networks in mammography: application to decision making in the diagnosis of breast cancer.*, Radiology **187** (1993), no. 1, 81–87.