Modelado de la productividad: predicción a través del análisis de series temporales

Trabajo Fin de Grado

Convocatoria: 8/4/2024

Alumno/a: Javier Coque Fernández

Tutor/a: Mar Angulo Martínez

Grado: Matemáticas computacionales

Agradecimientos

Me gustaría agradecer a mis padres porque qué menos.

Tabla de contenidos

Lista de Figuras

Lista de Tablas

Abstracto

La productividad es un tópico de especial interés y crucial tanto para empresas como para gobiernos. Por otra parte, una de las formas más comunes, interesantes y potentes de estudiar los datos es realizar un análisis de la evolución de los mismos en el tiempo. Este trabajo estudia la productividad media desde el año 1960 hasta 2021 (ambos incluidos). Adicionalmente, se realizan predicciones a futuro de 2 años. Para el análisis de esta serie temporal y su predicción, se han empleado tres técnicas diferentes: ARIMA, LSTM y Prophet siendo LSTM el que peor resultados ofrece y ARIMA los que mejor.

Palabras clave: productividad, series temporales, predicción.

Abstract

Productivity is a topic of special interest and crucial for both companies and governments. On the other hand, one of the most common, interesting and powerful ways to study data is to analyze its evolution over time. This work studies the average productivity from 1960 to 2021 (both included). Additionally, future predictions of 2 years are made. For the analysis of this time series and its prediction, three different techniques have been used: ARIMA, LSTM and Prophet, with LSTM being the one that offers the worst results and ARIMA the best.

Key words: productivity, time series, forecasting.

Glosario

AR	Auto-regressive
MA	Moving Average
ARMA	Autoregressive Moving Average
ARIMA	Auto-regressive Integrated Moving
RRNN	Redes Neuronales
RNN	Recurrent Neuronal Network
LSTM	Long Short-Term Memory (red neuronal)
ADF	Augmented Dickey-Fuller
KPSS	Kwiatkowski-Phillips-Schmidt-Shin
ACF	Autocorrelation Function
PACF	Partial Autocorrelation Function
H_0	Hipótesis nula
H_A	Hipótesis alternativa

Notación

ec.	Ecuación.
$\mathbb{N}$	Números naturales.
$\mathbb{R}$	Números reales.
$\mathbb{C}$	Números complejos.
$v \in \mathbb{R}^X$	Vector v de números reales de X dimensiones.
v^{T}	Vector traspuesto.
$\mid$	Tal que (en ecuaciones matemáticas).
$\sim$	Se aproxima (en ecuaciones matemáticas).
$\in$	Pertenece (en ecuaciones matemáticas).
$\forall$	Para todo (en ecuaciones matemáticas).
$\setminus$	Excluyente (en ecuaciones matemáticas).
i.e.	Procede del latín *id est* que traducido al español es *esto es*.
(a, b)	Intervalo abierto. Incluye números de a hasta b sin incluir a estos.
$[a, b]$	Intervalo cerrado. Incluye números de a hasta b incluyendo a estos.
$\{a, b\}$	Intervalo que únicamente comprende números a y b.
$[a, b, c]$	Intervalo que únicamente comprende números a, b y c.
palabra$^{\mathrm{AX}}$	En la sección de apéndice número X se puede encontrar más información.

1 Introducción

El trabajo es un concepto que existe desde tiempos inmemoriales. El interés del ser humano en ser más productivo y más eficiente en su trabajo viene prácticamente desde que el hombre es hombre. Fue hace más de 8.000 años cuando pasó de tener una economía basada en la caza y recolección, a otra mucho más eficiente; fundamentalmente basada en la agricultura y ganadería. Esto dio lugar a una nueva etapa en la historia de la humanidad: el Neolítico.

De todas maneras, no hace falta remontarse tan atrás en el tiempo para que la mejora de la productividad en la vida de las personas marcase nuevos hitos en la historia de la humanidad. La Primera Revolución Industrial (s. XVIII) fue marcada por una etapa de desarrollo tecnológico y descubrimiento de nuevos métodos de producción que permitían al ser humano ser capaz de producir mayor cantidad de bienes, de una calidad mayor, en un menor tiempo (propia definición de productividad). A la Primera Revolución Industrial le sigue, un siglo más tarde, la Segunda Revolución Industrial. Esta superó las innovaciones productivas de la primera. Al igual que su predecesora, fue una etapa de gran crecimiento económico y social debido a los avances tecnológicos y nuevas implementaciones de producción en las fábricas para mejorar la productividad de estas. El ejemplo más claro de esto último es el desarrollo en cadena de Henry Ford, quien revolucionó el sector automovilístico.

En la actualidad, la productividad sigue estando a la orden del día en múltiples ámbitos. Un ejemplo de ello es el campo del *software* —área de especial interés e importancia en este trabajo—, en el cual, durante los últimos años, se han ido desarrollando nuevas metodologías de trabajo para aumentar la productividad. Las metodologías que más popularidad han ganado, en tiempos recientes, son las metodologías *agile* (ágiles), las cuales sustituyen a las anteriores; conocidas como metodologías pesadas. Por nombrar un par de metodologías ágiles más importantes, destacan Scrum y Kanban.

1.1 Justificación y contexto

La productividad, hoy en día, es un tema que cada vez va ganando mayor más popularidad; cada vez se pueden hacer más cosas en menos tiempo. Normalmente, cuando uno piensa en productividad, lo lleva al ámbito individual, en donde el individuo quiere ser cada vez más eficiente con su tiempo y sacarle el mayor provecho posible al día. Es por ello que no es de asombro ver cómo libros que tratan sobre la productividad como *Deep Work* del Dr. Cal Newport son tan vendidos; o cómo los *podcasts* más escuchados del famoso Dr. Andrew Huberman tratan sobre este mismo tema.

Pero la productividad no es de solo interés individual. Las empresas actuales cada vez se involucran más en cómo exprimir el máximo potencial de sus trabajadores y que sean más productivos y eficientes en su puesto de trabajo. Es un tema de tanto interés, que no es difícil encontrar casos de directivos obsesionados con que sus empleados sean lo más productivos posibles. Un ejemplo de ello es Marissa Mayer, quien en 2013, al convertirse CEO de Yahoo, restringió el trabajo telepresencial; obligando de esta manera a los trabajadores a ir a las oficinas para poder controlar que estaban *siendo productivos*. Otro ejemplo de esto es la empresa de Amazon, la cual ordenó construir *The Spheres* hace relativamente no mucho. Esto se trata de tres invernaderos creados exclusivamente para fomentar un ambiente de trabajo eficiente, creatividad y productividad entre los trabajadores. De la misma manera, otras empresas como Google o Microsoft también disponen de campus al aire libre pensados para la interacción entre empleados, cambio de ideas, y mejora de la productividad.

Una capa por encima del interés individual y empresarial de la productividad, está el interés gubernamental. Para un gobierno es fundamental que las empresas de su país sean lo más productivas posibles. Esto es de especial interés para esta autoridad por varios motivos. El primero y más claro, es que cuanto más productivas sean las empresas de un país, más riqueza generarán con todo lo que eso conlleva: crecimiento económico, mayor PIB (Producto Interior Bruto), generación de empleo, etc. Además, esto tiene efectos a largo plazo destacables, pues los gobiernos de los países tienen muy en cuenta las opiniones económicas y otras cuestiones de aquellos países con mayor productividad. Esto, proporciona al país un *status* internacional de vital importancia.

1.2 Planteamiento del problema

Para estudiar una variable, la mejor forma de hacerlo es a través de las matemáticas; concretamente la estadística. Esta es la rama de las matemáticas que se ocupa de recopilar, organizar, analizar e interpretar estos datos. Su objetivo principal es extraer conclusiones significativas acerca de ellos.

El análisis de estos datos se puede realizar a lo largo del tiempo, dando lugar a un nuevo y particular problema de modelado estadístico e inferencia. Esto se debe a la propia naturaleza de estos datos, pues, al estar medidos a lo largo del tiempo, guardan (normalmente) una dependencia entre sí. Esto es lo que da lugar a las **series temporales** y su particular análisis; pues es justamente esta dependencia entre los datos lo que vuelve a las series temporales tan peculiares respecto a otro tipo de datos que carecen de esta característica.

El interés de estudiar una variable a lo largo del tiempo es su tema muy actual y que se sigue estudiando

a día de hoy. Las series temporales son una forma muy atractiva de almacenar y estudiar los datos; y cada vez más común. Estas aparecen en múltiples campos de gran importancia e interés de estudio como puede ser la economía: en el mercado bursátil. También se hallan en el ámbito social para estudiar el crecimiento de población de un país o ciudad, así como en el campo de la medicina para ser capaz de monitorizar el ritmo cardíaco de un paciente; o en meteorología para el pronóstico del tiempo. Es por ello que conviene conocerlas un poco más de cerca y saber cómo analizarlas y qué beneficios se pueden extraer de su estudio.

Son por estas razones, por las que se ha decidido estudiar la variable de este trabajo, la productividad, a lo largo del tiempo. Pues no solo permite estudiar la productividad pasada, sino que también, con la ayuda de ciertos algoritmos que se explicarán más adelante, las series temporales permiten poder realizar predicciones de la variable estudiada a futuro, lo cual es algo muy característico del campo de *Big Data*: un campo actualmente en auge y muy demandado en la actualidad.

1.3 Objetivos del trabajo

Este trabajo, al abarcar un campo tan amplio y con tantas posibilidades de desarrollo como son las series temporales, es necesario acotar y definir muy bien los objetivos que se atienden a él. Para este trabajo en particular, los objetivos que se han propuesto son:

- Analizar la productividad media europea a lo largo del tiempo.
- Emplear varios algoritmos matemáticos parar realizar una predicción a futuro de la variable objetivo.
- Predecir la productividad media en países europeos a corto y medio plazo.
- Realizar una comparación entre modelos, basándose en una métrica de evaluación, y determinar cuál ha sido mejor y por qué.
- Ver hasta qué punto los distintos modelos obtenidos ofrecen resultados parecidos o no y por qué.
- Plantear futuras líneas de trabajo posibles que puedan utilizar los resultados y conclusiones de este trabajo.

2 Estado de la cuestión

Uno de los primeros algoritmos específicos para la modelización de series temporales, fue el suavizado exponencial simple (Holt, 1957), introducido a en la década de los $50'$ del siglo pasado. Este método, para realizar predicciones futuras, computa una media ponderada sobre los datos observados, dando mayor peso a los datos más recientes. Muchas extensiones entonces, surgieron a raíz de este algoritmo y se empezaron a usar durante los años siguientes debido a los buenos resultados que eran capaces de obtener, con respecto a métodos anteriores ya existentes, como era el de media móvil (este último, simplemente promediaba los valores pasados para pronosticar valores futuros). Una de las extensiones que cabe mencionar, es la que realizó Peter R. Winters, dando lugar al método Holt-Winters (Winters, 1960) o, también conocido como, suavizado exponencial triple.

Estos algoritmos eran principalmente los únicos empleados para modelizar series temporales hasta la publicación de los métodos ARIMA en el año 1970 (Box et al., 2015). Los autores de este libro, George Box y Gwilym Jenkins, son considerados los precursores del análisis de las series temporales por su aportación, tan estadística como novedosa, a este campo. Su estudio ha sentado las bases para el análisis y el pronóstico de series temporales hasta el día de hoy —pese a tener varios años—. Este método, debido a su fuerte componente estadística, supondrá la parte principal de explicación y desarrollo de este trabajo.

2.1 Marco teórico del trabajo

Antes de entrar en el desarrollo del trabajo, hay ciertos conceptos teóricos que uno debe conocer de antemano. Estos conceptos son los que se muestran a continuación.

Posteriormente, se explicarán los tres algoritmos aplicados en este trabajo: ARIMA, LSTM y Prophet. Se han elegido estos algoritmos, no solo por su buen rendimiento, sino que se ha considerado de especial interés evaluarlos y compararlos, pues, pertenecen a épocas totalmente distintas. ARIMA aparece en 1970, LSTM en 1997 y Prophet en 2017.

2.1.1 Conceptos

2.1.1.1 Procesos estocásticos y series temporales

Un proceso estocástico es el modelo matemático utilizado para describir datos recopilados a lo largo del tiempo. Es decir, se trata de un conjunto de variables aleatorias[A1] (muestras) $X_1, X_2, \cdots, X_n$

obtenidas en n instantes de tiempo diferentes —bajo las mismas condiciones—.

La realización de estas muestras da lugar a un vector de valores observados; uno por cada variable aleatoria $x_1, x_2, \cdots, x_n$.

Una **serie temporal** es, por tanto, la consecuencia de un proceso estocástico. Se trata de un conjunto de valores observados –i.e. el vector $x_1, x_2, \cdots, x_n$— ordenados cronológicamente a lo largo del tiempo.

Hay varios tipos de serie temporal:

- **univariante o multivariante** dependiendo de si es estudia una o varias variables a lo largo del tiempo.
- **discreta** o **continua** si los datos se recogen de manera puntual o continua.
- **regular** o **irregular** dependiendo de si la frecuencia con la que se recogen los datos es equidistante o no.

Este trabajo se enfocará en series temporales univariantes, discretas y regulares. A estas series temporales se las referirá con la letra Y_t; mientras que y_t se empleará para valores observados de la serie temporal en un instante t de tiempo determinado.

2.1.1.2 Periodo y Lag

La frecuencia con la que se recopilan estos datos se le denomina **periodo** y puede ser diario, mensual, trimestral, anual, etc. Se designa con la letra t al periodo presente u observado —dependerá del contexto—. Es decir, si el periodo es diario y t es el tiempo presente, $t-1$ es ayer y $t+1$ el día de mañana. En este trabajo se empleará la letra T para designar el número total de periodos (observaciones) de la serie temporal.

El concepto de ***lag*** (retraso) se suele designar con la letra k y está muy ligado con el de periodo. Se emplea para denotar una diferencia de k periodos entre dos observaciones de la serie temporal.

De esta manera, si el periodo son días, t el periodo presente y $k = 2$:

$$y_{t-k} = y_{t-2} \tag{2.1.1}$$

Luego, en la ec. 2.1.1, y_{t-2} hace referencia a antes de ayer.

El *lag* k también puede hacer referencia a periodos futuros. En ese caso se denotará como y_{t+k}. Aunque, salvo que se especifique lo contrario, el *lag* hará referencia a valores pasados.

Asimismo, el rango de valores que puede tomar k varía según el dato observado[1].

2.1.1.3 Componentes de una serie temporal

Una serie temporal tiene principalmente 4 componentes: tendencia, estacionalidad, ciclo y residuo.

Tendencia

La **tendencia** se puede interpretar como el comportamiento o evolución de una serie temporal a lo largo del tiempo. La figura 2.1 muestra un ejemplo de una serie temporal con tendencia ascendente.

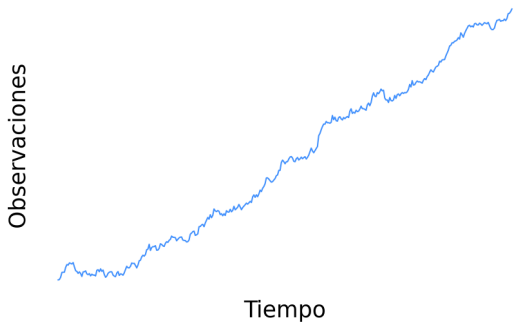

Figura 2.1: Ejemplo de serie temporal con tendencia ascendente.

Estacionalidad

La **estacionalidad** es uno o varios patrones sistemáticos que se repiten en la serie temporal con cierta periodicidad y durante un periodo de tiempo constante. Véase la figura 2.2.

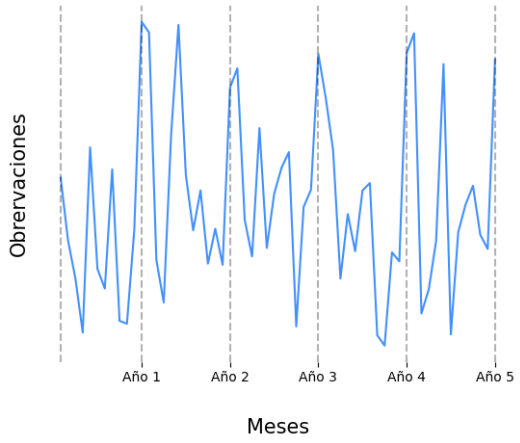

Figura 2.2: Ejemplo de serie temporal con estacionalidad.

[1]En las observaciones inicial y final, i.e. y_0 e y_t, $k \in [0, T]$. Sin embargo, en y_1 e y_{t-1}, $k \in [0, T-1]$.

La serie temporal en la figura 2.2 es estacional porque justo antes del final de cada año —i.e. frecuencia periódica—, se puede ver una subida repentina en las observaciones. Además, la duración de estas subidas es constante.

La figura 2.2 también sirve para aclarar que una serie temporal con estacionalidad no es lo mismo que una serie temporal periódica.

Ciclos

Los **ciclos** ocurren cuando una serie temporal presenta subidas y bajadas repentinas que no son periódicas ni regulares (véase figura 2.3).

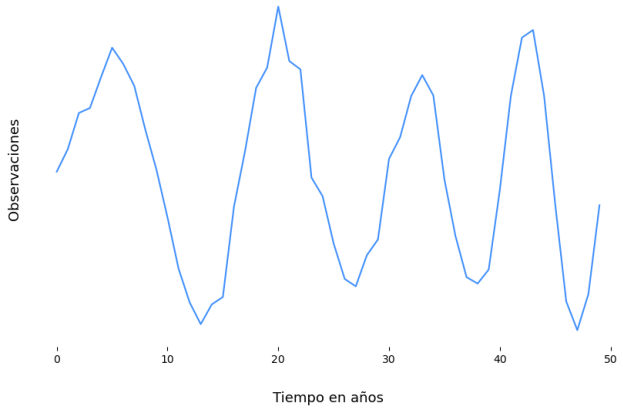

Figura 2.3: Ejemplo del componente cíclico.

No hay que confundir estacionalidad con patrones cíclicos; no tienen nada que ver: el ciclo no tiene duración fija ni periodicidad. Además, es más impredecible y lo normal es que se supere el año entre un ciclo y el siguiente.

Residuo

El residuo[2] es un componente que presentan todas las series temporales. Se trata de variables aleatorias $X_1, X_2, \cdots$ independientes entre sí e idénticamente distribuidas (v.a.i.i.d)[3] (véase la figura 2.4).

[2]Residuo y ruido son lo mismo. Se emplean de manera intercambiable en este trabajo.

[3]Variables Aleatorias Independientes Idénticamente Distribuidas.

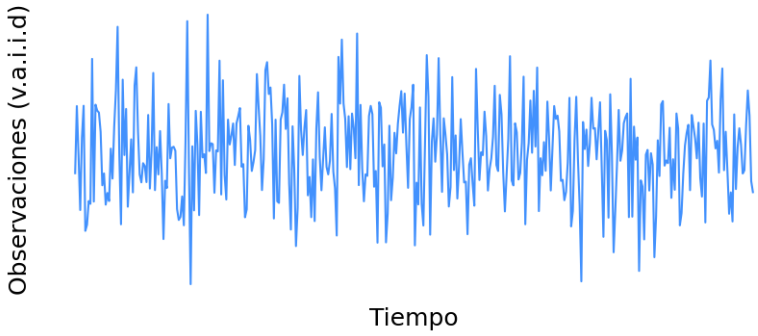

Figura 2.4: Ruido.

En el residuo, están recogidos todos aquellos eventos impredecibles como catástrofes, azar, pandemias, etc.

Sumando estas componentes se obtiene el modelo estructural más básico de una serie temporal: la descomposición aditiva (Harvey & Peters, 1990).

$$Y_t = T_t + S_t + R_t \tag{2.1.2}$$

En donde:

- Y_t es la serie temporal.
- T_t es la tendencia-ciclo.
- S_t es la estacionalidad.
- R_t es el residuo (o ruido).

Alternativamente, está la descomposición multiplicativa:

$$Y_t = T_t \times S_t \times R_t \tag{2.1.3}$$

La distinción entre 2.1.2 y 2.1.3 es un tanto artificial, pues si se hace el logaritmo de la ec. 2.1.3 se obtiene una descomposición aditiva de $log(Y_t)$.

En la figura 2.5 se pueden ver las distintas componentes de una serie temporal de una manera más clara.

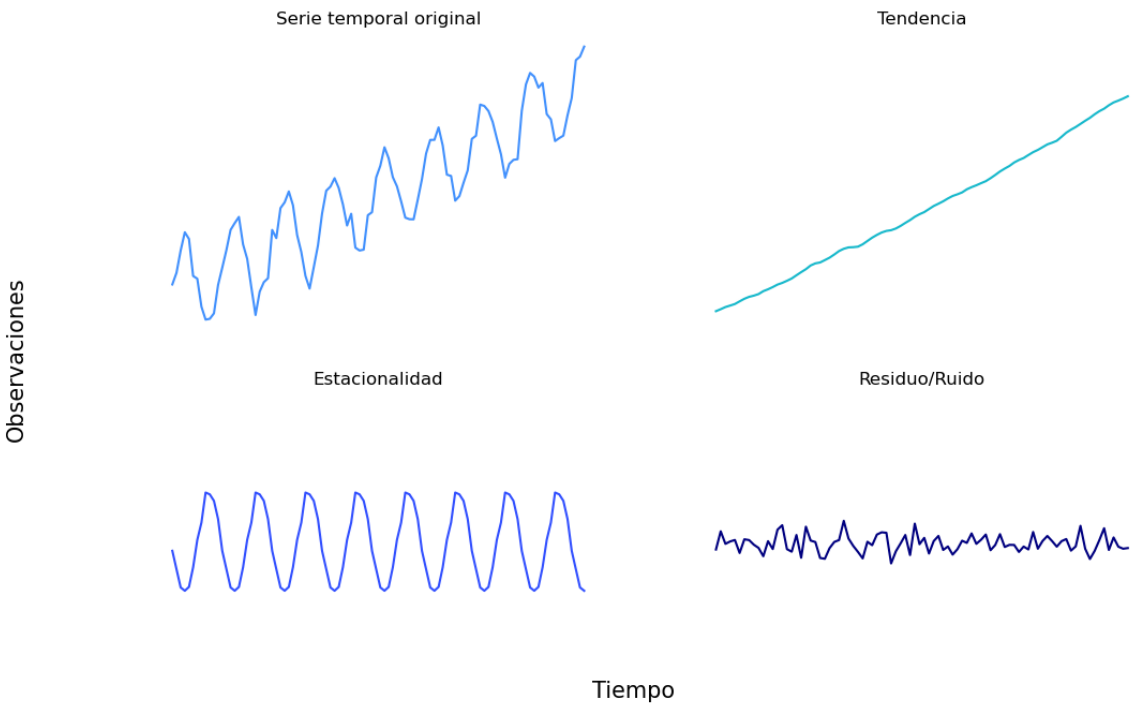

Figura 2.5: Serie temporal descompuesta en tendencia-ciclo, estacionalidad y residuo.

También, cabe mencionar que el modelo estructural aditivo no es el único. Los modelos estructurales tienen la ventaja que se le pueden añadir y quitar componentes según las características de la serie temporal. Como se verá más adelante, el algoritmo Prophet incluye vacaciones/eventos especiales en su modelo estructural.

2.1.1.4 Estacionariedad

Un proceso estocástico se dice que es (débilmente[4]) estacionario si sus propiedades no cambian a lo largo del tiempo (véase figura 2.6). Luego, si un proceso estocástico es estacionario, la serie temporal que este proceso genera también lo será.

[4]Existen dos tipos de estacionariedad: fuerte y débil. En este trabajo, cuando se mencione el concepto de estacionariedad, se estará refiriendo a estacionariedad débil.

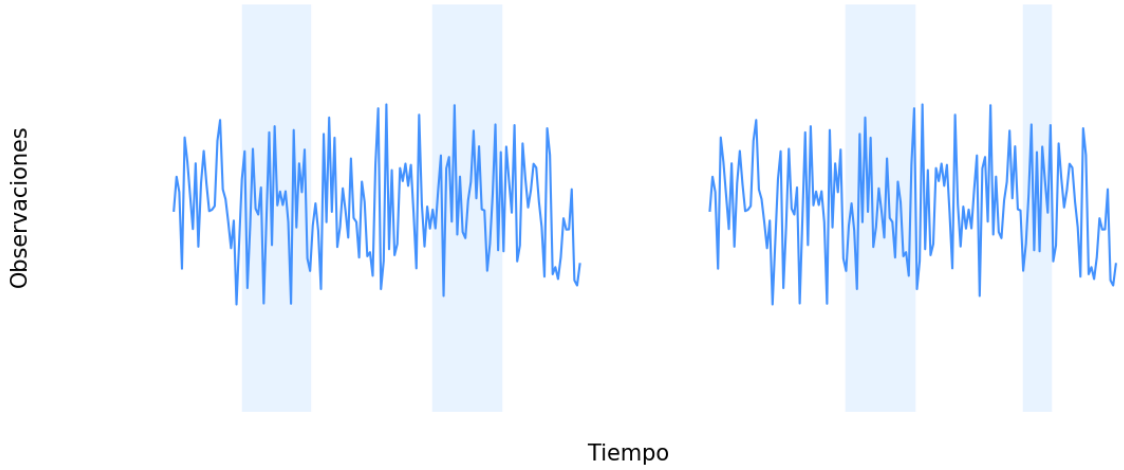

Figura 2.6: Ejemplo serie estacionaria: no importa en qué instante se observe la serie temporal ni en qué ventana temporal, sus propiedades no cambian.

Que las propiedades de una serie temporal no cambien a lo largo del tiempo implica:

1. La media (esperanza[A2]) de la serie temporal es constante.

$$E[Y_t] = E[y_k] = \mu \qquad \forall k \in [0, T]$$

2. La varianza[A3] de la serie temporal es constante.

$$\mathrm{Var}[Y_t] = \mathrm{Var}[y_k] = \sigma^2 \qquad \forall k \in [0, T]$$

   A esto se le conoce como **homocedasticidad**. De lo contrario, si la varianza no es constante, se dice que la serie temporal presenta **heterocedasticidad**.

3. La covarianza (o correlación) entre $y_a,\ y_b \mid a, b \in [0, T] = 0$ con $a \neq b$. Es decir:

$$\mathrm{Cov}(y_0, y_k) = \mathrm{Cov}(y_1, y_{1+k}) = \cdots = \mathrm{Cov}(y_t, y_{t-k}) = 0 \qquad \forall k \in [0, T]$$

El punto 3. da lugar a la definición de *autocovarianza* en *lag* k como:

$$\gamma_k = \mathrm{Cov}(y_t, y_{t-k}) = E[(y_t - E[y_t])](y_{t-k} - E[y_{t-k}])] = E((y_t - \mu)(y_{t-k} - \mu)) \tag{2.1.4}$$

Luego:

$$\gamma_0 = E(y_t - \mu)^2 = \mathrm{Var}(y_t) = \sigma^2$$

Por tanto, la *autocorrelación* en *lag* k se puede definir como:

$$\rho_k = \frac{\mathrm{Cov}(y_t, y_{t-k})}{\sqrt{\mathrm{Var}(y_t) \cdot \mathrm{Var}(y_{t-k})}} = \frac{\mathrm{Cov}(y_t, y_{t-k})}{\sqrt{\mathrm{Var}(y_t) \cdot \mathrm{Var}(y_t)}} = \frac{\mathrm{Cov}(y_t, y_{t-k})}{\mathrm{Var}(y_t)} = \frac{\gamma_k}{\gamma_0} \tag{2.1.5}$$

Entonces, una serie temporal con tendencia —ascendente o descendente— no va a ser estacionaria porque su media no va a ser igual en cualquier intervalo de tiempo. A su vez, una serie temporal con estacionalidad tampoco va a ser estacionaria; pues la covarianza entre dos puntos consecutivos, en el periodo estacional, no es igual a cero.

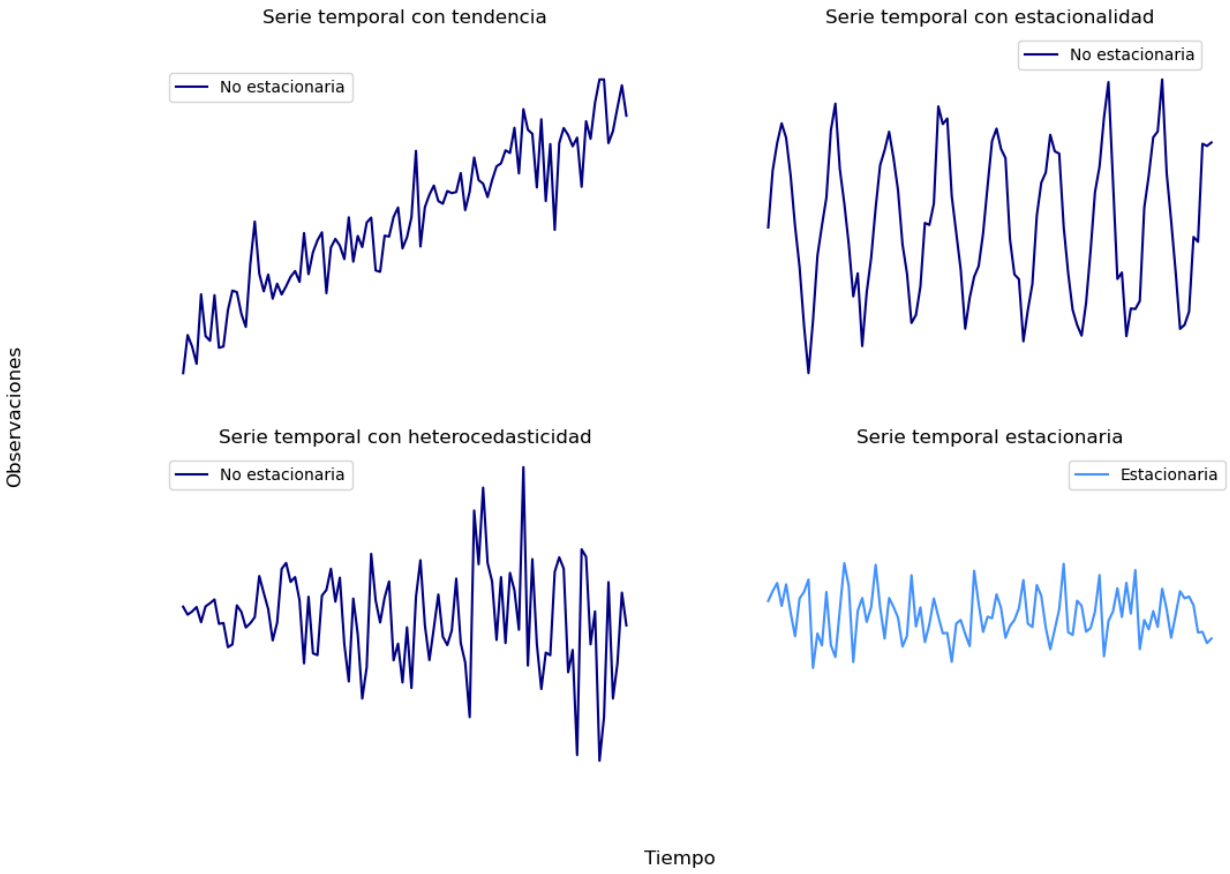

Figura 2.7: Ejemplos series temporales no estacionarias (azul oscuro) y serie temporal estacionaria (azul claro).

En la figura 2.7 las dos gráficas de arriba no son estacionarias —por lo que ha comentado en el párrafo anterior—. Asimismo, la gráfica de abajo a la izquierda, tampoco es estacionaria porque su varianza no es constante a lo largo del tiempo —i.e. presenta heterocedasticidad—. Se puede apreciar cómo

los picos van adquiriendo mayor amplitud con el paso del tiempo. Sin embargo, la gráfica de abajo a la derecha sí es estacionaria: la tendencia es nula, pues, la media es constante; no hay ningún patrón claro; luego tampoco hay estacionalidad. Y, por último, la varianza se mantiene constante a lo largo del tiempo, por lo que esta serie temporal presenta homocedasticidad.

2.1.1.5 ACF & PACF

En los ejemplos del apartado anterior, ha estado muy claro a simple vista si una serie temporal era estacionaria o no. Puede darse el caso en el que esto no sea tan evidente. Véase la figura 2.8.

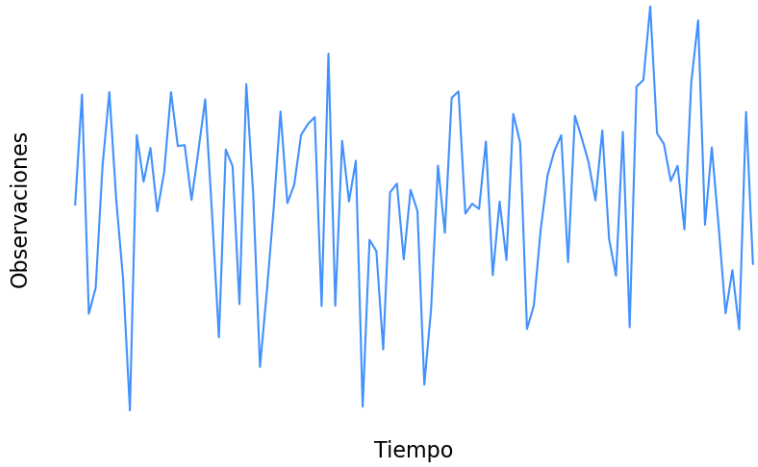

Figura 2.8: Ejemplo de serie temporal con dudosa estacionariedad.

Para ello, existen dos técnicas que ayudan a determinar si una serie temporal es (o no) estacionaria. En este apartado se va a ver una de estas técnicas.

ACF (*Autocorrelation Function*) y PACF (*Partial Autocorrelation Function*) son los gráficos que ayudan —entre otras cosas— a determinar la estacionariedad de una serie temporal (Hyndman & Athanasopoulos, 2018). Estos gráficos aportan información acerca de la dependencia del tiempo presente (y_t) con sus valores pasados (y_{t-k})[5]. A esta dependencia es a lo que se denomina como **autocorrelación** —definida matemáticamente en el apartado anterior, ec. 2.1.5—.

Se muestra un ejemplo de ambos gráficos. Véase figura 2.9.

[5]Otra manera de entender esto podría ser: grado de influencia de observaciones pasadas con la presente.

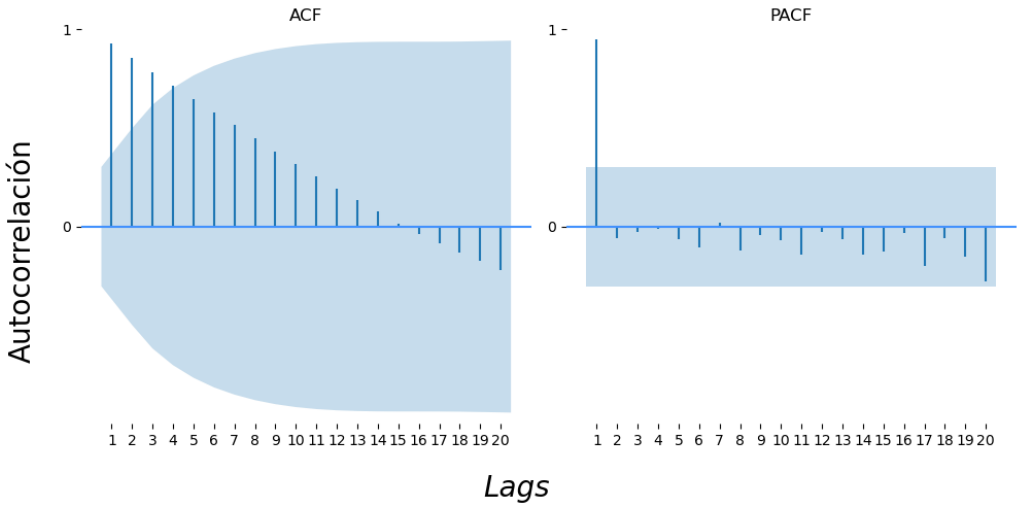

Figura 2.9: Ejemplo de un gráfico ACF y su PACF asociado.

En ambos gráficos, el eje x representa los lags k con $k \in [1, ..., 20]$[6]. Y el eje y es la autocorrelación de dicho *lag* (y_{t-k}) con el valor presente (y_t).

La diferencia entre ACF y PACF reside en que si y_t está correlado con y_{t-1}, lo más probable es que y_{t-1} esté correlado con y_{t-2}. Luego la correlación entre y_t e y_{t-2} puede estar influenciada por y_{t-1}. PACF, a diferencia de ACF, muestra la correlación *pura* entre y_t e y_{t-2}. En general, PACF muestra la correlación entre y_t e y_{t-k} sin la influencia de valores intermedios[7].

La autocorrelación entre un valor pasado y_{t-k} y el valor presente y_t se considera alta, si la línea vertical en dicho *lag* k sobrepasa la zona azul. Teniendo en cuenta esto, es posible determinar si una serie temporal es estacionaria con estos gráficos si estos muestran que todas las líneas verticales —i.e. $\forall k \in [1, 20]$— caen dentro de la zona azul[8].

[6]Por convención, solo se dibuja hasta el lag 20. El lag $k = 0$ siempre va a ser 1, pues, es la autocorrelación del valor y_t consigo mismo. Por ende, este valor no se tiene en cuenta en el estudio y algunas gráficas, como 2.9, no lo muestran.

[7]Se deja un ejemplo en el *apéndice*[A4] de autocorrelación parcial que puede ayudar a aclarar el concepto.

[8]Más concretamente, la zona azul se establece en $\pm\frac{1.96}{T}$. El valor ± 1.96 indica un intervalo de confianza del 95% en una distribución gaussiana[A5]. Esto, se traduce en que para que se considere que una serie temporal sea estacionaria a partir de ACF y PACF, hay un margen de que el 5% de las líneas verticales sobrepasen la zona azul.

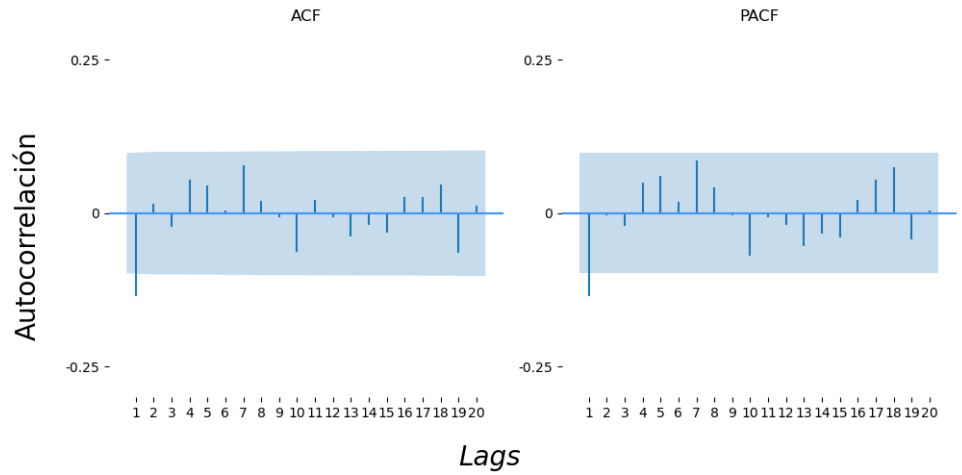

Figura 2.10: Ejemplo de gráficas ACF y PACF en una serie temporal estacionaria.

En la figura 2.10 se muestra cómo alrededor del 5% de las líneas horizontales sobrepasa los límites. Por tanto, se puede pensar que la serie temporal es estacionaria.

2.1.1.6 Contrastes de hipótesis: ADF & KPSS

Hay otra prueba, adicional a los gráficos ACF y PACF, para contrastar la estacionariedad de una serie temporal. Esta prueba, más definitiva que la anterior, son los contrastes de hipótesis[A6].

Hay varios contrastes de hipótesis que se pueden aplicar; los dos más comunes son el ADF (Dickey & Fuller, 1979) y KPSS (Kwiatkowski et al., 1992). Ambos contrastes, lo que tratan de determinar es la existencia de raíces unitarias[A7] para confirmar si la serie temporal es estacionaria o no.

Su comparación se puede ver en la tabla 4.

	ADF	KPSS
H_0	La serie temporal no es estacionaria	La serie temporal es estacionaria
H_A	La serie temporal es estacionaria	La serie temporal no es estacionaria

Tabla 4: Tabla comparativa entre los contrastes de hipótesis ADF y KPSS.

2.1.2 Familia ARIMA

La familia ARIMA —al igual que LSTM y Prophet— son algoritmos, no modelos[A10]. El algoritmo ARIMA en concreto, se trata de una combinación de los algoritmos *Autoregressive* (AR) y *Moving Average* (MA)[9] que serán explicados a lo largo de esta sección.

La razón por la que en apartados anteriores se ha hecho tanto hincapié en series temporales estacionarias es por la *I* en medio de AR**I**MA: viene de *Integrated*. Esto quiere decir que ARIMA solo es un algoritmo válido para **series temporales estacionarias**. Esto, aunque de primeras pueda parecer que ARIMA quede restringido a un tipo de series temporales —las estacionarias—, no es así. Existen una serie de transformaciones para convertir cualquier serie temporal no estacionaria en una que sí lo sea.

2.1.2.1 Transformaciones aplicadas a series temporales no estacionarias

Diferenciar respecto a la tendencia

Este método, también conocido como método de Box-Jenkins (Box et al., 2015), es muy eficaz para eliminar la tendencia de una serie temporal. Consiste en restar al valor de la observación y_t el valor de la de y_{t-1} $\forall t \in [0, T]$.

$$y'_t = y_t - y_{t-1} \tag{2.1.6}$$

A esto se le conoce como diferencia de **primer orden**; pero puede darse el caso en el que, después de haber aplicado esta transformación, la serie temporal siga presentando tendencia —i.e. media no constante—. En ese caso, se procede a hacer una diferencia de **segundo orden**.

$$\begin{aligned} y''_t &= y'_t - y'_{t-1} \\ &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ &= y_t - 2y_{t-1} + y_{t-2}. \end{aligned}$$

Nunca suele ser necesario ir más allá del segundo orden (Hyndman & Athanasopoulos, 2021).

[9] La familia ARIMA no solo abarca los algoritmos AR, MA y ARIMA; sino que también ARMA y SARIMA. El primero es la combinación de AR y MA sin implicar que la serie temporal sea estacionaria. La *S* de **S**ARIMA viene de *Seasonality* (estacionalidad) para series temporales que presentan esta característica. Los detalles de estos dos últimos algoritmos van más allá del alcance de este trabajo.

Véase la figura 2.11 en la que se aplica una diferencia de primer orden a la serie temporal de arriba a la izquierda de la figura 2.7.

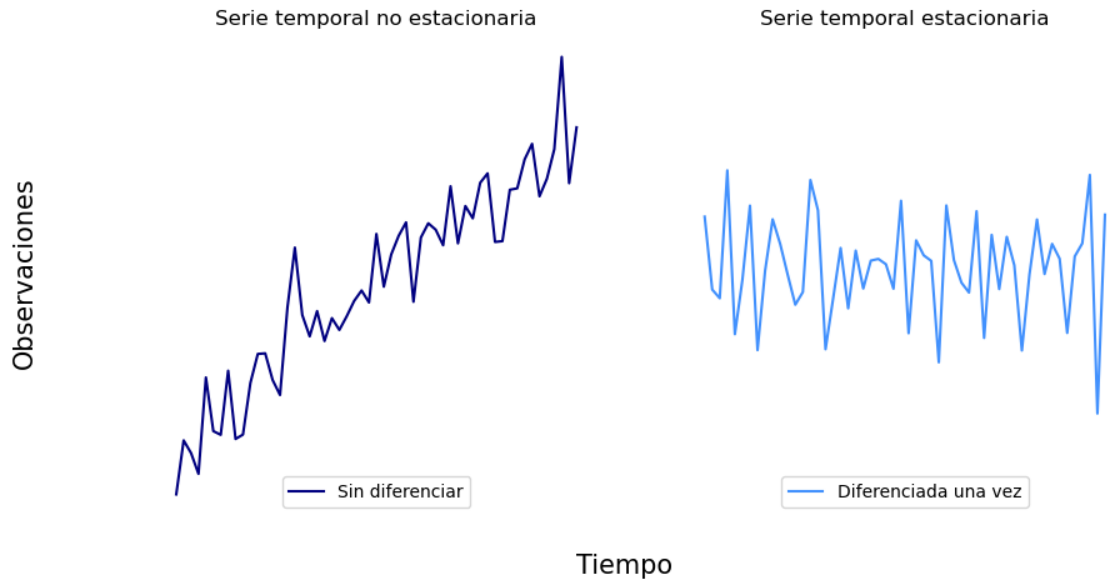

Figura 2.11: Ejemplo serie temporal sin diferenciar, junto a su diferencia de primer orden.

La serie temporal de la izquierda claramente no es estacionaria porque presenta una clara tendencia ascendente. Sin embargo, si se le aplica una diferencia de primer orden y posteriormente se pasa el test ADF, el p-valor de este test es de 2.5×10^{-7}. Luego, se **rechaza** la hipótesis nula de que la serie temporal no es estacionaria.

Diferenciar respecto a la estacionalidad

El método de diferencia para la tendencia de Box-Jenkins recientemente visto, también se puede aplicar para eliminar el componente de estacionalidad de una serie temporal. La estacionalidad, al ser periódica, se produce cada m *lags*. Luego:

$$y'_t = y_t - y_{t-m}$$

Eliminar heterocedasticidad

Si la varianza de la serie temporal no es constante a lo largo del tiempo, entonces habrá que aplicar a esta la transformación logarítmica —también conocida como Box-Cox (Box & Cox, 1964)—. Esto es:

$$Y'_t = log(Y_t) \tag{2.1.7}$$

Véase un ejemplo de esta transformación en la figura 2.12.

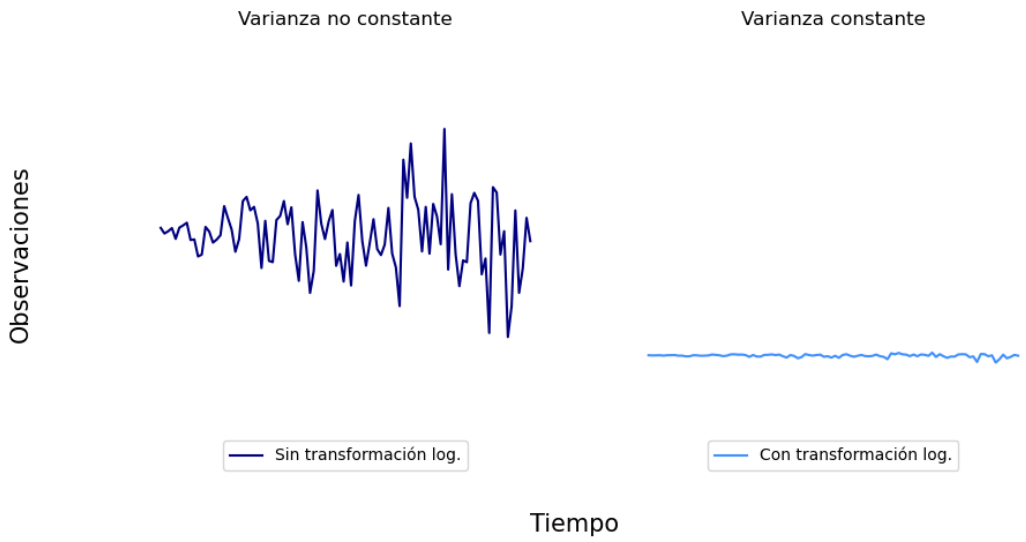

Figura 2.12: Ejemplo de la transformación de Box-Cox.

En la figura 2.12 se puede ver cómo, tras una transformación logarítmica, se pasa de una serie temporal que presenta heterocedasticidad (izquierda), a una serie temporal que presenta homocedasticidad (derecha).

Cabe remarcar que esta transformación solo es válida cuando la serie temporal solo toma valores positivos[10]—pues no existe el logaritmo de un número negativo—. Sin embargo, muchas de las series temporales cumplen con esta condición (precios, población, ventas, etc.).

2.1.2.2 Descomposición de Wold

La Descomposición de Wold es uno de los teoremas fundamentales de las series temporales (Mills, 2019). En él están basados los modelos ARIMA. Este teorema (a grandes rasgos) prueba que cualquier serie temporal estacionaria, puramente no determinista[11] $y_t - \mu$ puede ser representada como una suma de variables aleatorias independientes —i.e. ruido— asociadas a unos determinados pesos. A esta combinación lineal también se la conoce como **filtro lineal**.

[10] Hay otras transformaciones más recientes para eliminar la heterocedasticidad de una serie temporal que no tienen este problema (Mills, 2019, pág 16).

[11] Puramente no determinista se refiere a que se elimina de la serie temporal cualquier comportamiento que pueda ser predicho a partir de valores pasados. Por eso la serie temporal debe ser estacionaria y se sustrae su media μ.

$$y_t - \mu = \varepsilon_t + \psi_1\varepsilon_{t-1} + \psi_2\varepsilon_{t-2} + \cdots = \sum_{j=0}^{\infty} \psi_j\varepsilon_{t-j}, \qquad \psi_0 = 1 \tag{2.1.8}$$

Nota

La demostración de este teorema va más allá del contenido de este trabajo. Esta se puede encontrar en (Brockwell and Davis, 1991, pág 188).

En la ec. 2.1.8, ε_t, ε_{t-1}, ε_{t-2}, ... son variables aleatorias sin correlación entre sí (ruido). Es decir:

$$E(\varepsilon_t) = 0, \qquad \mathrm{Var}(\varepsilon_t) = \sigma^2, \qquad \mathrm{Cov}(\varepsilon_t, \varepsilon_{t-k}) = 0 \quad \forall k \in [1, T] \tag{2.1.9}$$

Estos pesos, en caso de ser infinitos, se asume que son sumables. De esta manera:

$$\sum_{j=0}^{\infty} |\psi_j| < \infty \tag{2.1.10}$$

en cuyo caso se dice que el sumatorio *converge*. A esto se le conoce como **condición de estacionariedad** y será esencial en los procesos autoregresivos (AR).

La ec. 2.1.8 es muy importante, pues de ahí salen los dos procesos de los que se compone ARIMA:

- **AR**: *Autoregressive* (autoregresivo)
- **MA**: *Moving Average* (media móvil)

Estos dos procesos generan modelos. Se pueden usar por separado o en conjunto con ARMA o ARIMA.

2.1.2.3 Procesos autoregresivos (AR)

Este algoritmo, parte del supuesto de que las observaciones pasadas y_{t-k} son buenos estimadores de la observación presente y_t. Luego, este algoritmo tratará de determinar qué observaciones pasadas son las que más influyen en el valor presente. Es decir, habrá que establecer qué datos pasados presentan una **alta correlación** con el dato presente.

Si a la ec. 2.1.8 se le asignan los pesos específicos $\psi_j = \phi^j$, y sin pérdida de la generalidad se asume que $\mu = 0$, se obtiene lo siguiente:

$$\begin{aligned} y_t &= \varepsilon_t + \phi\varepsilon_{t-1} + \phi^2\varepsilon_{t-2} + \cdots \\ y_t &= \varepsilon_t + \phi \underbrace{\left(\varepsilon_{t-1} + \phi\varepsilon_{t-2} + \phi^2\varepsilon_{t-3} + \cdots\right)}_{y_{t-1}} \\ y_t &= \phi y_{t-1} + \varepsilon_t \end{aligned} \tag{2.1.11}$$

Un proceso AR tiene un parámetro, p. Este parámetro determina qué valores pasados influyen a la hora de estimar el valor presente. El valor de p hace referencia al número de ϕ de la fórmula de AR. Por ejemplo, la ec. 2.1.11 es un proceso AR(1).

En general, un proceso AR(p) se puede definir de la siguiente manera:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} + \cdots + \phi_p y_{t-p} + \varepsilon_t \tag{2.1.12}$$

Para determinar un valor aproximado de p, se pueden emplear los gráficos ACF y PACF explicados en la sección 2.1.1.5.

ACF & PACF en modelos AR

El ACF indica si el algoritmo AR(p) es apropiado para modelar la serie temporal (Mills, 2019). En otras palabras, si el valor p va a ser > 0. Esto ocurrirá (seguramente) si el ACF presenta 2 tipos de patrones:

- Se aprecia un decremento exponencial a medida que el *lag* k se va haciendo más grande.
- Se aprecia un patrón sinusoidal(con forma de seno) decreciente.

Véase la siguiente figura donde se puede ver esto recientemente explicado de una manera mucho más gráfica.

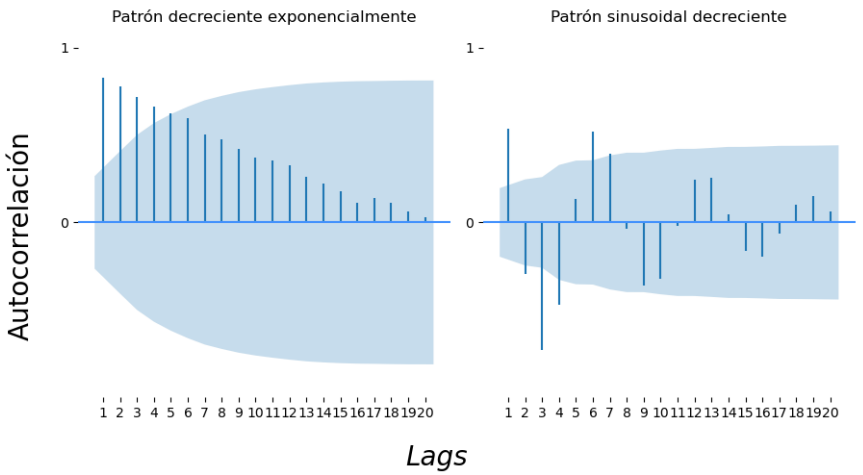

Figura 2.13: Ejemplo patrón decreciente exponencialmente y patrón sinusoidal.

En realidad, que ambos patrones decaigan con *lags* mayores tiene lógica, pues indica que cuanto más atrás en el tiempo se observe, menor influencia va a tener en el presente[12].

Nota

Esto está demostrado en el *apéndice*[A10].

Ahora, el PACF, como ya se ha dicho con anterioridad, muestra la autocorrelación *pura* entre la observación presente con las pasadas. Este gráfico, por tanto, indica un valor aproximado a p, siendo este valor los *lags* asociados a las líneas verticales que sobresalen de la zona azul. El PACF de la figura 2.13 indica los siguientes posibles valores óptimos de p: $1, 2, 3, 4$ y 6 (estos son valores aproximados).

Caben a destacar dos detalles del proceso AR(p)[13]:

- No se debe interpretar la constante c de la fórmula como la media de la serie temporal.
- No es necesariamente estacionario.

2.1.2.4 Media Móvil (MA)

La Media Móvil, más conocida como *Moving Average* (MA) en inglés, se trata de la representación finita de 2.1.8 (Descomposición de Wold).

[12]Hay una excepción a esto dicho, y es cuando hay estacionalidad cada m periodos. Entonces, se podrá observar en el ACF que cada m *lags*, la autocorrelación sube.

[13]Se deja en el *apéndice*[A10] las demostraciones de estos dos puntos.

MA, al igual que AR, tiene un parámetro: denominado con la letra q. Este parámetro también indica el número de pesos finitos que se tienen en cuenta para la creación del modelo.

$$y_t = \mu + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots \theta_q\varepsilon_{t-q} \tag{2.1.13}$$

Este proceso, al ser finito, sus pesos son sumables y por tanto, se cumple la condición de estacionariedad 2.1.10. En otras palabras, todos los procesos MA son estacionarios[14] (a diferencia de los procesos AR).

La interpretación de MA es distinta a la de AR. MA se podría ver como un algoritmo que para estimar el valor de una variable en un instante t de tiempo, tiene en cuenta errores del pasado en torno a una media (media de la serie temporal). En concreto, será el parámetro q quien determine el número de errores pasados que influyen en el valor presente.

Para determinar el parámetro q de un proceso MA, es posible recurrir también a los gráficos ACF y PACF para obtener una aproximación de cuál puede ser el valor óptimo de este.

ACF & PACF en modelos MA

Hay una curiosa relación entre los ACF y PACF de procesos AR(p) y procesos MA(q): son completamente opuestos (Mills, 2019, pág 43).

Para saber si un proceso MA es viable para realizar predicciones en una serie temporal —o lo que es lo mismo si el parámetro q puede ser mayor a cero— hay que fijarse en el **PACF**. Si presenta un patrón **decreciente exponencialmente** o **sinusoidal**, entonces seguramente el parámetro q será mayor a 0.

Por el contrario, la última línea vertical en sobresalir de la zona azul en el ACF indicará una aproximación del valor óptimo de q —esto es igual que en el PACF de un proceso AR—.

2.1.2.5 ARIMA

ARIMA combina los dos procesos recientemente vistos. Luego, su representación es la siguiente:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots \theta_q\varepsilon_{t-q} \tag{2.1.14}$$

El algoritmo ARIMA tiene la particularidad de que **sí** se impone que los parámetros $\phi_1, \phi_2, \cdots, \phi_p$ de

[14] Se deja en el *apéndice*[A10] esta demostración.

AR(p) deban cumplir las condiciones de estacionariedad.

ARIMA(p, d, q) tiene 3 parámetros: p, q ya se han mencionado. La novedad de ARIMA es el segundo parámetro: la d. Esta d hace referencias al número de diferencias —respecto a la tendencia 2.1.2.1— que hay que aplicar a una serie temporal para que esta sea estacionaria. En ningún caso suele ser necesario aplicarla más de dos veces (Hyndman & Athanasopoulos, 2018). Luego, se puede considerar que el valor de d está restringido a valores $[0, 1, 2]$.

Nota

Cabe remarcar que un proceso ARIMA puede tener p ó q = 0; dando lugar a un proceso MA(q) o AR(p) respectivamente.

El ACF y PACF también juegan un papel importante a la hora de saber si ambos parámetros p y q pueden mayores que cero. Su interpretación es la misma que para los procesos AR y MA respectivamente. Esto se resume muy bien en la siguiente tabla (Adejumo et al., 2018).

	ACF	PACF
AR(p)	Decae exponencialmente (de manera sinusoidal o no)	Lag significativo hasta cierto valor k
MA(q)	Lag significativo hasta cierto valor k	Decae exponencialmente (de manera sinusoidal o no)
ARIMA(p, d, q) con $p, q > 0$	Decae exponencialmente (de manera sinusoidal o no)	Decae exponencialmente (de manera sinusoidal o no)

Tabla 5: Tabla comparativa entre los modelos AR(p), MA(q) y ARIMA(p, d, q) y las respectivas características de sus ACF y PACF correspondientes.

2.1.2.6 Estimación de parámetros

En el apartado anterior, se ha visto una manera muy rápida, visual y **aproximada** cómo estimar los valores óptimos p y q con los gráficos ACF y PACF. Luego, hay que probar distintos modelos ARIMA en torno a estos valores p y q. En esta sección se vera un método para determinar si un modelo ARIMA(p, d, q) es mejor (o peor) que otro ARIMA(p', d', q').

Asimismo, tampoco se ha hablado de cómo estimar los parámetros c y $\phi_1, \phi_2, \cdots, \phi_p$ en un modelo AR(p) ni de los parámetros $\theta_1, \theta_2, \cdots, \theta_q$ en un modelo MA(q). Esto también será cubierto en esta sección.

Método de máxima verosimilitud

Este método es el que se emplea para estimar los parámetros c y $\phi_1, \phi_2, \cdots, \phi_p$ en un modelo AR(p) y de los parámetros $\theta_1, \theta_2, \cdots, \theta_q$ en un modelo MA(q).

Para explicar el MMV (Método de Máxima Verosimilitud) (Myung, 2003), primero hay que entender qué es la verosimilitud.

La verosimilitud consiste en otorgar una credibilidad a un parámetro desconocido. Esta verosimilitud se calcula mediante la función de verosimilitud (L), la cual hace uso de probabilidad condicionada.

Sea Y_t la serie temporal y δ un parámetro[15] que se trata de estimar (c, ϕ, θ).

$$L(Y_t, \delta) = f(y_t, y_{t-1}, y_{t-2}, \cdots, y_{t-T}/\delta) \tag{2.1.15}$$

La ec. 2.1.15 se puede leer como: *¿cuál es la probabilidad de que la serie temporal sea Y_t si el parámetro desconocido δ toma un determinado valor?*

- Luego L, al ser una probabilidad, está acotado a valores $0 \leq L \leq 1$.

El MMV consiste en averiguar el valor del parámetro δ que maximice L. Es decir, el valor del parámetro δ más creíble con el que se hubiera obtenido la serie temporal Y_t. A este valor (o valores) δ que maximiza $L(Y_t, \delta)$ se le conoce como EMV (Estimador Máximo Verosímil). Se le suele denotar con la letra L directamente.

AIC & BIC

El Criterio de Información de Akaike (AIC por sus siglas en inglés) (Bozdogan, 1987) es muy útil para comparar modelos ARIMA con distintos valores de p, d ó q y determinar cuál es mejor. Su fórmula es:

[15] En la explicación se menciona parámetro en singular; pero de la misma manera se puede extender a parámetros $\delta_1, \delta_2, \cdots$ como sería este caso.

$$\text{AIC} = 2(p + q + k + 1) - 2 \cdot log(L) \tag{2.1.16}$$

En donde:

- L es el estimador máximo verosímil.
- $k = 1$ si $c \neq 0$ y $k = 0$ si $c = 0$.

Hay una versión mejorada del AIC: AICc. Sin embargo, este criterio no se empleará en el desarrollo.

Por último, otro criterio que tiene mejores propiedades teóricas (Mills, 2019) que el AIC, es el Criterio de Información Bayesiano (BIC por sus siglas en inglés) (Schwarz, 1978).

$$\text{BIC} = \text{AIC} + [log(T) - 2] \cdot (p + q + k + 1) \tag{2.1.17}$$

Los 3 criterios (AICc también) se basan en que cuanto más bajo sea el valor obtenido, mejor es el modelo. Estos criterios penalizan valores altos de p, d y q, pues, es lo que está sumando en las fórmulas. Esto quiere decir que debe haber un equilibrio entre eficacia y complejidad en un modelo ARIMA[A11].

En este trabajo, se van a emplear los criterios AIC y BIC a la hora de elegir el mejor modelo. Como cada uno es más fiable que otro en unas determinadas condiciones (Vrieze, 2012), una práctica común es tener en cuenta ambos a la hora de elegir el mejor modelo. En el *apéndice*[A12] se explica cómo funcionan.

2.1.2.7 Test para la validación del modelo

Debido a que ARIMA tiene una base estadística, su validación, pese a que puede hacerse separando los datos en *train* y en *test* —como es común hacer en aprendizaje automático—, es más común el enfoque estadístico que, por ejemplo, se hace en el libro *"Forecasting: Principles and Practice"* de Rob J Hyndman[16] y George Athanasopoulos: a través de contrastes de hipótesis.

En este trabajo, para hacer una validación completa, se van a aplicar cuatro contrastes de hipótesis: cada uno de ellos para validar un aspecto distinto del modelo. Para este trabajo en concreto, la librería que se va a emplear en la sección de desarrollo, aplica estos contrastes directamente al modelo y retorna su estadístico; junto a su correspondiente p-valor. Por tanto, bastará con saber qué trata de contrastar cada uno de estos tests, cuáles son sus hipótesis nulas y alternativas respectivamente y, finalmente,

[16]Actualmente, una de las figuras más admiradas y menos cuestionadas en el campo de las series temporales.

cómo interpretar el p-valor asociado a cada uno de ellos.

Test Breusch-Pagan

El contraste de hipótesis Breusch-Pagan (Breusch & Pagan, 1979) trata de contrastar si el modelo ARIMA(p, d, q), con sus respectivos coeficientes, presenta homocedasticidad —o no—. Sus hipótesis son las siguientes:

- H_0: El modelo presenta homocedasticidad.
- H_A: El modelo no presenta homocedasticidad.

Para establecer un estándar en todos los contrastes de hipótesis empleados en el desarrollo, se va a establecer un $\alpha = 0.05$. En el caso concreto del contraste de Breusch-Pagan, si el p-valor resultante fuera menor que α, se rechazaría la hipótesis nula; sugiriendo, pues, que el modelo no presenta homocedasticidad y se debería tratar de encontrar otro mejor.

Test Ljung-Box

Tanto este contraste como el siguiente, evalúan, de una manera distinta, la componente residual del modelo. Esta es crucial, pues debe tener un patrón de ruido. De lo contrario, se interpreta como que el modelo no ha sido capaz de captar bien los patrones de los datos y otro modelo debería ser empleado.

Este contraste en concreto, el test Ljung-Box (Ljung & Box, 1978), es el más común de los cuatro tests de esta sección y el que se suele aplicar siempre. Lo que trata de corroborar este test es si dos residuos no están autocorrelados. Como se mencionó en apartados anteriores, una característica del ruido es que tiene autocorrelación cero. Este contraste trata de validar esto mismo. Por tanto, sus hipótesis son las siguientes:

- H_0: El residuo r_t y r_{t-l} no están autocorrelados.
- H_A: El residuo r_t y r_{t-l} están autocorrelados.

En las hipótesis de arriba, l comprende desde cero hasta k ($l \in [1, \cdots, k]$), siendo k el *lag* máximo sobre el que se quiere comprobar si r_t tiene autocorrelación con él. El valor de k suele ser uno representativo de la serie temporal. Por ejemplo, si la serie temporal muestra una estacionalidad cada m periodos, entonces, por convención $k = m$, pues interesa saber si hay autocorrelación hasta ese periodo[17].

[17] Si una serie temporal tiene estacionalidad cada m periodos, estos periodos están correlacionados.

Por tanto, para validar el modelo se debe buscar que el p-valor asociado a cada r_{t-l} con $l \in [1, \cdots, k]$ sea $> \alpha$.

Test Jarque-Bera

Este contraste (Jarque & Bera, 1980), al igual que el anterior, también observa los residuos del modelo pero esta vez, para tratar de verificar si estos siguen una distribución normal. Este test de normalidad se basa en la asimetría (*skewness*) y en el apuntamiento (*kurtosis*) para determinar si una distribución (residuos en este caso) sigue una distribución normal. Sus hipótesis, para este caso concreto, son las siguientes:

- H_0: Los residuos siguen una distribución normal.
- H_A: Los residuos no siguen una distribución normal.

Luego, al igual que en los anteriores contrastes, se busca que el p-valor asociado a este test sea $> \alpha$ para poder validar el modelo.

Test z-score

Como último contraste a emplear para la validación del modelo, está el z-score. Este test, entre otras cosas, se puede emplear para validar si un coeficiente (no parámetro) determinado en el modelo ARIMA(p, d, q) es significativo, o no. En caso de que no lo sea, se debe eliminar. En caso de que ninguno lo sea, este test sugiere que se trate de buscar otro modelo ARIMA(p, d, q). Las hipótesis de este test son las siguientes:

- H_0: El coeficiente es cero (i.e. no es significativo)
- H_A: El valor del coeficiente no es cero (i.e. es significativo).

Por tanto, este test, a diferencia de los anteriores, se trata de buscar que, al menos un p-valor asociado a un coeficiente, sea $< \alpha$.

2.1.3 LSTM

Las redes neuronales surgen en 1943, con el primer modelo matemático sugerido de una simple neurona artificial (McCulloch & Pitts, 1943). Sin embargo, el primer modelo con capacidad de auto-aprendizaje se da quince años más tarde (Rosenblatt, 1958). La idea era imitar matemáticamente el comportamiento biológico de una neurona en un dispositivo *hardware* llamado perceptrón. Esto, da lugar al perceptrón multi-capa (MLP) (Werbos & John, 1974), que consiste en un gran número

de neuronas (perceptrones), distribuidas por capas[18], todas interconectadas entre sí entre capa y capa. Cada neurona, en una determinada capa, recibe datos de todas las neuronas de la capa anterior. Estos datos son procesados a través de una función matemática que produce una salida. Esta salida es entonces, enviada a todas las neuronas de la capa siguiente.

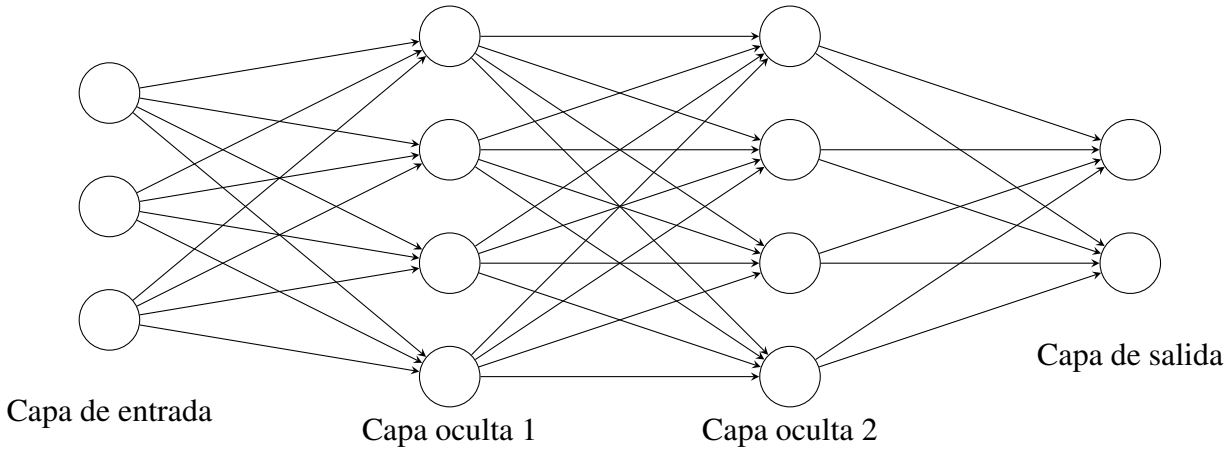

Figura 2.14: Ejemplo arquitectura de perceptrón multi-capa con dos capas ocultas.

Su uso no se popularizó hasta que no se definió un algoritmo eficaz para que estas neuronas aprendieran por sí solas. Este algoritmo fue el *backpropagation* (Rumelhart et al., 1986). Consiste en propagar hacia atrás los errores cometidos por las neuronas para que estas aprendan. Esto, se hace al aplicar una función de coste cuyo objetivo es minimizarla lo máximo posible.

Desde entonces, han surgido diversas variaciones de MLP para distintos campos de la inteligencia artificial. Algunos ejemplos son: las RNN (*Recurrent Neural Networks*) (Jordan, 1986)[19], o las CNN (*Convolutional Neural Network*) (LeCun et al., 1998). Las primeras, se dice que tienen *memoria*. Se aplican en campos como el procesamiento de lenguaje (NLP), reconocimiento de voz o **series temporales**. Las segundas fueron un gran avance en el campo de la visión por computador para procesamiento de imágenes.

Las redes neuronales LSTM (Hochreiter & Schmidhuber, 1997) son una mejora de las RNN. Estas sufrían de problemas de desvanecimiento del gradiente; o, por el contrario, del gradiente explosivo[A13] a

[18]El término ***Deep** Learning*, empleado para referirse al campo de las redes neuronales, viene del gran número de capas —i.e. profundidad → *deep*— que estas poseen.

[19]Como curiosidad, Michael I. Jordan fue alumno de Rumelhart (uno de los autores del artículo de *backpropagation*). De hecho, en ambos artículos: (Jordan, 1986) y (Rumelhart et al., 1986) se mencionan las RNN.

la hora de entrenarse. Esto comprometía el rendimiento de las RNN haciendo que tuvieran dificultades almacenando información a largo plazo. Las LSTM fueron diseñadas para que almacenar información a largo plazo fuese su comportamiento por defecto (Olah, 2015).

Antes de proceder a explicar la red neuronal LSTM, se van a explicar las funciones sigmoide y tangente.

2.1.3.1 Función sigmoide

Esta función se suele denotar con la letra griega σ. Su fórmula es $\sigma(x) = \frac{1}{1 + e^{-t}}$.

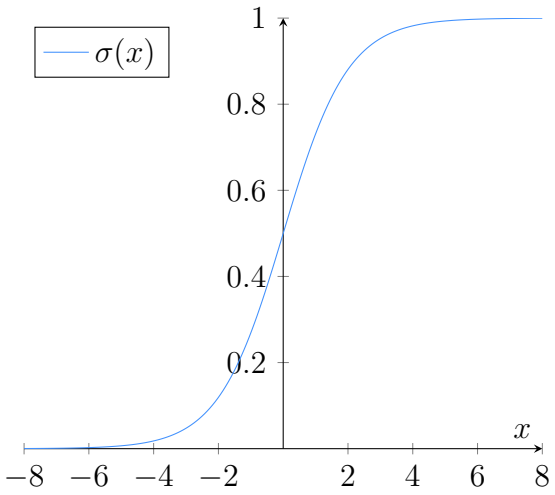

Figura 2.15: Función sigmoide.

El rango de esta función es $a \in \mathbb{R} : \ a \in [0, 1]$. Es decir, esta función se va a encargar de devolver un valor entre 0 y 1.

- Cuanto más próximo a 0, significará menos importante.
- Cuanto más próximo a 1, significará más importante.

2.1.3.2 Función tangente

Esta función tiene una forma parecida a la anterior, pero con un rango un poco más amplio; entre 1 y -1. Su fórmula es $tanh = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

Su finalidad en la LSTM es la de regular valores para simplificar las operaciones y que sea menos costoso entrenar a la red neuronal.

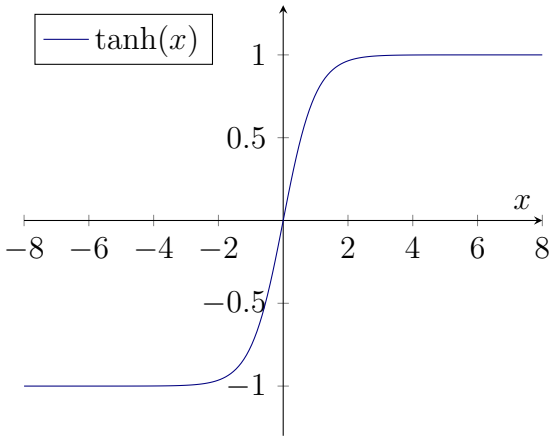

Figura 2.16: Función tangente hiperbólica.

2.1.3.3 Procedimiento LSTM

Las LSTM se pueden ver como una secuencia de neuronas interconectadas entre sí. Cada una de ellas le pasa información a la siguiente —en forma de cadena—. De manera individual, cada neurona se encargará de eliminar, actualizar, añadir información y pasarlo a la siguiente neurona. Esto se hace a través de **puertas**.

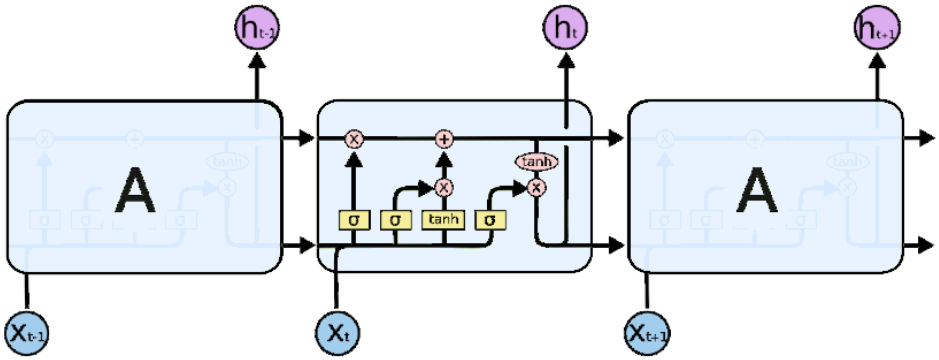

Figura 2.17: Ejemplo visual LSTM[20].

En la imagen 2.17, los círculos azules, x_{t-1}, x_t, x_{t+1}, es la nueva información que le llega a dicha neurona en un instante t determinado. En el caso de una serie temporal, serán los valores de $y_1, y_2, \cdots, y_t$.

Los círculos rosas, h_{t-1}, h_t, h_{t+1}, hacen referencia al estado oculto (*hidden state* en inglés) de la celda (celda y neurona es lo mismo en este caso). Este estado oculto se puede ver como la **memoria a corto**

[20]Fuente: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

plazo que se va transmitiendo de neurona en neurona. Nótese que la flecha del círculo rosa sale de la flecha horizontal de abajo de cada celda.

Cabe mencionar que tanto los valores $\cdots, x_{t-1}, x_t, x_{t+1}, \cdots$ como los valores $\cdots, h_{t-1}, h_t, h_{t+1}, \cdots$ no tienen por qué ser valores numéricos aislados —en el caso de este trabajo sí—. Pueden ser vectores, como ocurre con las palabras, que hay que transformarlas a vectores numéricos. A esto se le conoce como *embedding*.

La flecha horizontal de arriba de la celda se la conoce como estado de celda (*cell state* en inglés). Este estado de celda representa la **memoria a largo plazo**.

Por último, en el interior de la celda aparecen las funciones σ y $tanh$; así como operaciones de sumas y multiplicaciones. Se va a tratar de explicar paso por paso lo que ocurre dentro de una celda desde que le llega información de la celda anterior hasta que la pasa a la siguiente.

Los pasos, a *grosso modo* son los siguientes:

1) Se decide qué información eliminar de la memoria a largo plazo a través de la **puerta de olvido**.
2) Se decide qué información actualizar de la memoria a largo plazo a través de la **puerta de actualización** (también conocida como puerta de entrada).
3) Se decide qué información de la memoria a largo plazo (estado de la celda) es relevante a corto plazo (estado oculto) a través de la **puerta de salida**.

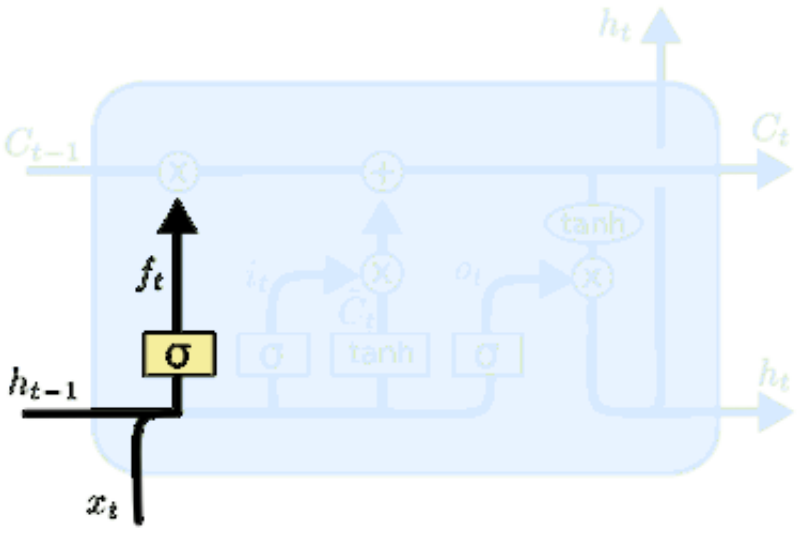

Figura 2.18: Puerta de olvido[21].

En la figura 2.18 el estado oculto (h_{t-1}) que recibe la celda actual de la celda anterior, junto con el *input* x_t, se pasan por una función sigmoide. A esta función es a la que se le conoce como **puerta de olvido**. La función sigmoide, como ya se ha debatido al principio de esta sección, tendrá una salida entre 0 y 1. Denótese a esta salida como f_t.

$$f_t = \sigma(W_{1.1} \cdot h_{t-1} + W_{1.2} \cdot x_t + b_1)$$

En donde:

- W son pesos.
- b es el *bias* (parámetro de compensación).

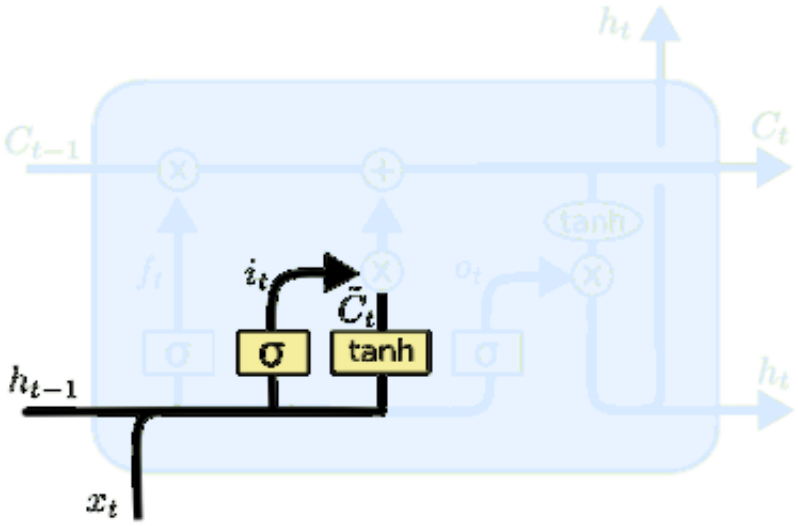

Figura 2.19: Puerta de actualización[22].

En la figura 2.19 el estado oculto h_{t-1}, junto con el *input* x_t, se pasan de nuevo por otra función sigmoide conocida como **puerta de actualización**. Además, también se operan en una función $tanh$. Denótese como i_t a la salida de la función sigmoide y como $\tilde{C}_t$ a la salida de la función $tanh$.

$$i_t = \sigma(W_{2.1.1} \cdot h_{t-1} + W_{2.1.2} \cdot x_t + b_{2.1})$$
$$\tilde{C}_t = tanh(W_{2.2.1} \cdot h_{t-1} + W_{2.2.2} \cdot x_t + b_{2.2})$$

$\tilde{C}_t$ representa la nueva información candidata para ser almacenada en memoria a largo plazo —i.e. estado de la celda—. i_t, como tiene un rango de 0 a 1, indica la cantidad de esta nueva información que

[21] Fuente: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[22] Fuente: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

va a ser parte de la memoria a largo plazo —cuanto más próximo a 1, mayor parte de esta información candidata será pasada al estado de la celda—.

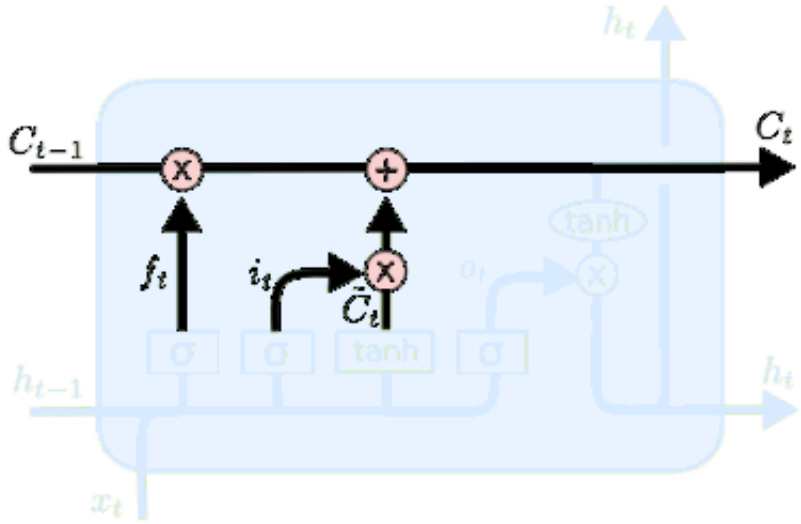

Figura 2.20: Actualización de la memoria a largo plazo[23].

Con la información recopilada hasta el momento, se puede actualizar la memoria a largo plazo o, lo que es lo mismo, el estado de la celda anterior denotado como C_{t-1}.

Lo primero, se decide qué información a largo plazo no es importante y, por tanto, se puede olvidar. Esto lo determina el valor de f_t calculado en la puerta de olvido: $f_t \in [0, 1]$. Cuanto más próximo a 0, indica que la información a largo plazo hasta el momento no es importante en el momento actual y se puede olvidar. Por el contrario, un f_t próximo a 1, indica todo lo contrario:

$$C'_t = C_{t-1} \cdot f_t$$

El siguiente paso es decidir qué cantidad de la información candidata va a pasar a formar parte de la memoria a largo plazo:

$$C''_t = i_t \cdot \tilde{C}_t$$

El paso final consiste en añadir estos resultados para obtener el nuevo estado de celda.

$$C_t = C'_t + C''_t$$

[23]Fuente https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

C_t será la memoria a largo plazo que se enviará a la siguiente neurona.

Por último, queda determinar qué memoria a corto plazo (estado oculto) será enviada también a la siguiente neurona.

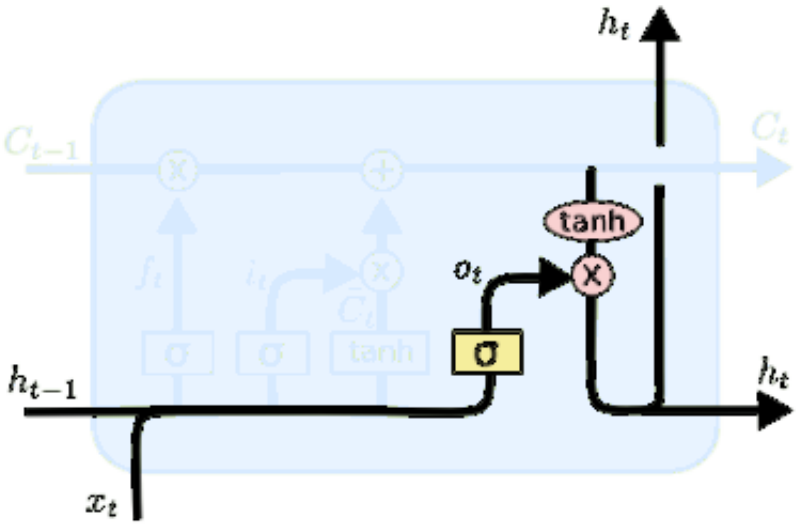

Figura 2.21: Puerta de salida[24].

La memoria a corto plazo que se envía a la siguiente neurona es una versión *filtrada* de la memoria a largo plazo.

De nuevo, el estado oculto h_{t-1}, junto con el *input* de la celda x_t, se pasan por una función sigmoide. Denótese a la salida de esta función como o_t. Por otro lado, el estado de la celda es pasado por una función tangente, que comprimirá los valores de C_t entre 1 y -1 —recuérdese que C_t podría ser un vector—. Como $o_t \in [0, 1]$, cuanto más próximo a 0, indicará que menor información de memoria a largo plazo será pasada a la siguiente neurona como estado oculto. Si, por el contrario, o_t es muy próximo a 1, indicaría que gran parte de la memoria a largo plazo (estado de celda) será pasada a la memoria a corto plazo (estado oculto).

2.1.4 Prophet

Prophet (Taylor & Letham, 2018) es un algoritmo para la predicción de series temporales desarrollado por Facebook. Es relativamente reciente, pero se ha hecho muy popular por lo intuitivo que es y su facilidad de uso en la práctica —además de su efectividad—.

[24]Fuente https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Algo a tener en cuenta, es que Facebook desarrolló este algoritmo para series temporales frecuentes en esta empresa. Estas, son series temporales con múltiples estacionalidades, varios *outliers*, etc. Son en este tipo de datos en los que Prophet obtiene su mejor rendimiento.

2.1.4.1 Cómo funciona Prophet por debajo

Los tres algoritmos mencionados hasta ahora son generativos[25]. Pese a ello, ARIMA y Prophet, por ejemplo, tienen dos enfoques muy distintos para hacer predicciones. El modelo ARIMA, como se ha visto, utiliza valores y/o errores pasados para ajustar el modelo a la serie temporal y poder generar predicciones futuras. Esto es sustancialmente distinto a lo que se hace en una **regresión**[A13]. Así es como Prophet enfoca su problema para generar un modelo y poder hacer el pronóstico futuro.

El modelo de Prophet es otro tipo de modelo estructural al ya visto con anterioridad en 2.1.2.

$$Y_t = T_t + S_t + H_t + \varepsilon_t \tag{2.1.18}$$

Aquí se ve un nuevo término: H_t. En este modelo, la H_t denota *holidays* (vacaciones) incluye tanto festivos como eventos puntuales como podrían ser La Final de la Champions o Las Fallas de Valencia.

A continuación, se va a explicar cómo Prophet ajusta la tendencia y vacaciones; componentes presentes y empleadas respectivamente en el modelo del trabajo.

Nota

Para clarificar que T_t, H_t de 2.1.18 son funciones del tiempo, se va a cambiar la notación a:

$$T_t \to T(t) \qquad\qquad H_t \to H(t)$$

Tendencia del algoritmo Prophet

Prophet cuenta con dos modelos para la tendencia: uno para tendencias no lineales con saturación y otro para tendencias lineales sin saturación. En este trabajo, la serie temporal con la que se trabaja no tiene cota superior y la tendencia es lineal. Por tanto, la tendencia que se empleará será la lineal.

Antes de explicar más en profundidad el modelo de tendencia lineal de Prophet, conviene saber qué son los puntos de inflexión. Prophet define S puntos de inflexión, $s_j,\ j = 1, \cdots, S$ a lo largo de la

[25] Un algoritmo generativo es aquel que aprende patrones y estructuras de los datos pasados y **genera** (de ahí el nombre) nuevos datos con las mismas características. Esto es debido a la predicción **extrapolar** de las series temporales.

serie temporal, en donde la tasa de crecimiento cambia. Se pone un ejemplo para ver esto visualmente y que sea más sencillo de comprender (véase figura 2.22).

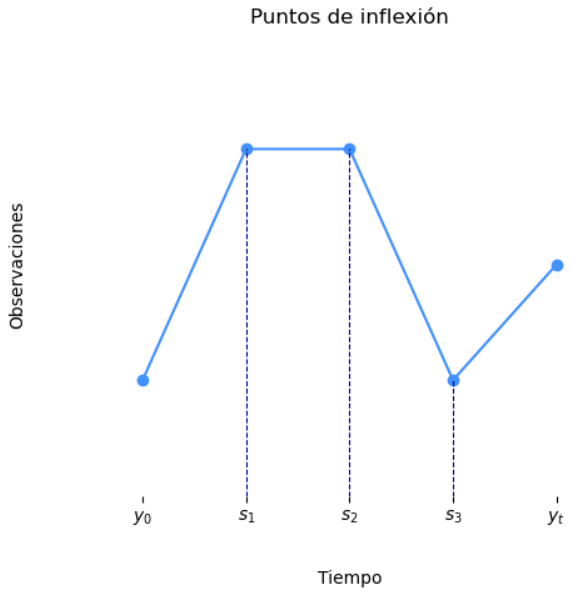

Figura 2.22: Ejemplo visual de puntos de inflexión.

En 2.22 hay $S = 3$ puntos de inflexión. Es decir, 3 rangos en los que la tasa de crecimiento de la serie temporal cambia en un determinado s_j. Se puede definir un vector de ajuste respecto a la tasa de crecimiento base k como $\delta \in \mathbb{R}^S$, donde δ_j es el cambio de tasa de crecimiento en s_j. De esta manera, la tasa de crecimiento en cualquier instante $t \in [0, T]$ es:

$$k + \sum_{j:t>s_j} \delta_j \tag{2.1.19}$$

Esto se puede representar mejor con un vector $a(t) \in \{0, 1\}^S$ tal que:

$$a_j(t) = \begin{cases} 1, & \text{si} \quad t \geq s_j, \\ 0, & \text{de lo contrario.} \end{cases} \tag{2.1.20}$$

Una representación más visual de la ec. 2.1.20, para $S = 3$ es la siguiente:

$$a(t) = \begin{cases} [0,0,0], & t \in [0, s_1), \\ [1,0,0], & t \in [s_1, s_2), \\ [1,1,0], & t \in [s_2, s_3), \\ [1,1,1], & t \in [s_3, T). \end{cases} \tag{2.1.21}$$

Luego 2.1.19 se puede reescribir como $k + a(t)^{\mathrm{T}}\delta$.

Nota

Los puntos de inflexión s_j pueden ser introducidos por el analista, o dejar a Prophet que los seleccione automáticamente, en cuyo caso, asume que $\delta_j \sim \text{Laplace}(0, \tau)$. En el desarrollo del trabajo se emplean ambos casos.

Para el modelo de tendencia lineal, Prophet divide la serie temporal en trozos (*piece-wise*). El inicio y final de cada trozo lo determinan los puntos de inflexión.

$$\underbrace{T(t) = (k + a(t)^T\delta)t + (m + a(t)^T\gamma)}_{\text{Modelo de tendencia lineal por trozos}} \tag{2.1.22}$$

En donde:

- k es la tasa de crecimiento de la tendencia.
- $a(t)$ son los puntos inflexión recientemente mencionados.
- γ es un parámetro de ajuste para que la aproximación en trozos sea continua (véase la siguiente figura).
- m es un parámetro de compensación.

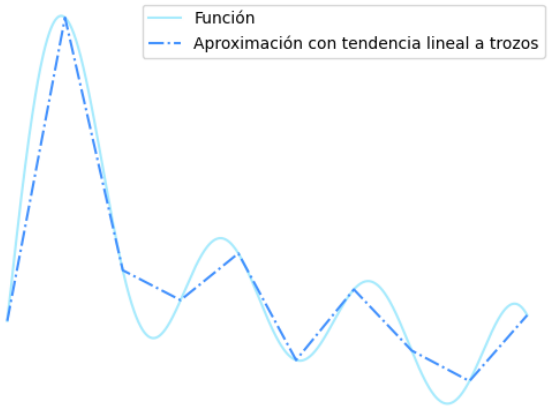

Figura 2.23: Ejemplo de una aproximación de tendencia lineal por trozos.

Para la **estimación de tendencia futura**, el modelo generativo se basa en que hay S puntos de inflexión en la serie temporal. Es decir, la probabilidad de que en un instante t cualquiera de la serie temporal —i.e. $t \in [0, T]$—, dicho instante sea un punto de inflexión es $\frac{S}{T} = P$. Cada uno de estos puntos de inflexión, como ya se ha comentado, tiene asociada una tasa de cambio $\delta_j \sim \text{Laplace}(0, \tau)$. Prophet asume que en el futuro, habrá los mismos puntos de inflexión con la misma frecuencia siguiendo la misma distribución $\text{Laplace}(0, \tau)$. De esta manera, futuros puntos de inflexión son aleatoriamente esparcidos en T^F [26]. Es decir que:

$$\forall j \in [0, T^F], \quad \begin{cases} \delta_j \sim \text{Laplace}(0, \tau) \ \text{ c.p.}^* \ P \\ \delta_j = 0 \ \text{ c.p. } 1 - P \end{cases}$$

*: con probabilidad.

Vacaciones y eventos del algoritmo Prophet

En el desarrollo, con el fin de tratar un modelo lo más preciso posible, se decide incluir a Prophet las fechas de Navidades a lo largo de los años para que las tenga en cuenta a la hora de ajustar el modelo y realizar las predicciones.

[26] T^F representa el tiempo futuro que se quiere predecir. Si el periodo es diario y se quiere predecir el siguiente mes, entonces $T^F = 30$ con $j \in [0, T^F]$. j pues, tomaría como valor cada día del mes siguiente.

2.2 Trabajos relacionados

Estudio de la efectividad del modelado de series temporales con ARIMA para la predicción del precio de acciones

Este estudio (Mondal et al., 2014) realiza predicciones del precio de acciones de 56 distintas empresas distribuidas en 7 sectores distintos: tecnológico, eléctrico, automovilístico, banca, infraestructura, acero, bienes de consumo de alta rotación.

El periodo se midió en meses y para las 56 empresas se recopilaron datos de los últimos 23 meses. El criterio empleado para la elección del mejor modelo fue AICc.

El resultado que se obtuvo fue una media de $91,35\%$ de precisión entre los siete sectores: superando el 85% de precisión en todos ellos. El sector en el que mejores resultados se obtuvieron fue el de bienes de consumo de alta rotación, con un 95.93% de precisión.

Comparación entre ARIMA y LSTM en predicción de series temporales

Este artículo (Siami-Namini et al., 2018) compara el modelo tradicional de predicción de series temporales ARIMA con el modelo LSTM.

Para ello se emplean 12 datasets distintos cuyos datos están recogidos con una periodicidad mensual y comprendidos entre enero 1985 y agosto 2018. Estos datasets se dividen en 70% entrenamiento y 30% test para su correspondiente entrenamiento del modelo LSTM.

Los resultados muestran que, de media, un algoritmo basado en LSTM mejora en un 85% los resultados en comparación con ARIMA.

Análisis y predicción de series temporales de la enfermedad del coronavirus en Indonesia aplicando ARIMA y Prophet

Este artículo (Satrio et al., 2021) realiza, a través de un dataset descargado en *kaggle*, una comparación entre los algoritmos ARIMA y Prophet.

El dataset que se utiliza contiene información diaria de datos de COVID de distintos estados del país de Indonesia.

Los resultados muestran que Prophet obtiene mejores resultados que ARIMA teniendo el primero un 91% de precisión en los resultados obtenidos, mientras que ARIMA únicamente un 25% de precisión.

3 Aspectos metodológicos

Las técnicas y metodologías aplicadas en este trabajo tienen su porqué. En los siguientes apartados se discutirán los distintos enfoques, procedimientos y tecnologías empleadas en la realización de este trabajo.

3.1 Metodología

El desarrollo de este trabajo ha conllevado una planificación previa. Así como un consenso sobre qué algoritmos aplicar y por qué.

3.1.1 Técnicas empleadas

Una vez se decidió el tema de este trabajo, lo primero fue determinar qué algoritmos se iban a emplear para desarrollo de este. Los motivos por los que se han elegido los algoritmos ARIMA, LSTM y Prophet son los siguientes:

ARIMA se decidió aplicarlo a este trabajo por varios motivos. El primero de todos es que fue uno de los primeros modelos en aparecer para la predicción de series temporales. Asienta las bases para el análisis y pronóstico de series temporales y, por ello, aún se sigue aplicando en el estado del arte de hoy en día (Schaffer et al., 2021) (Benvenuto et al., 2020). El segundo motivo es que tiene una componente matemática por detrás muy interesante de explicar y de conocer. Este conocimiento es necesario para entender algoritmos que surgieron posteriormente y que también se emplean a día de hoy, como pueden ser los algoritmos ARCH y GARCH (muy comunes en áreas financieras).

El motivo por el que se ha elegido el algoritmo **LSTM** es porque, por lo general, se obtienen mejores resultados con él que con ARIMA (Siami-Namini et al., 2018) (Siami-Namini et al., 2019). Otro algoritmo muy parecido al LSTM (una variante) es GRU (*Gated Recurrent Units*). Este tiene un mejor ratio tiempo-resultados que LSTM y es más viable para datasets pequeños y otros casos concretos (Yang et al., 2020). Sin embargo, para este trabajo en concreto, debido a que, por lo general, LSTM obtiene mejores resultados y la computación no es un problema, se ha visto oportuno elegir LSTM por encima de GRU. También cabe mencionar que LSTM tiene muchas variantes. Sin embargo, ninguna de estas variantes mejora significativamente la versión más simple; explicada en la sección 2.1.3 (Greff et al., 2016).

Prophet, al ser creado por Facebook, una red social, es un algoritmo pensado para series temporales

que presentan estacionalidad, con valores extremos y que son sensibles a eventos especiales y/o vacaciones. Además, Prophet es relativamente sencillo de usar, ya que se trata de un algoritmo con un nivel alto de abstracción. Simplemente especificando algunos parámetros intuitivos, se pueden llegar a modelizar, de una forma bastante precisa, las series temporales.

De todas maneras, el motivo de peso fundamental de la elección de estos algoritmos no es su rendimiento —que también— sino que surgen en épocas totalmente distintas. Se ha visto de interés considerar las predicciones de los tres algoritmos y ver cómo de diferentes son debido a que, emplean técnicas de aprendizaje totalmente distintas.

3.1.2 Estudio y planificación

Una vez se han elegido los algoritmos a implementar en el trabajo, se ha necesitado un estudio riguroso de todos ellos. En especial del algoritmo de ARIMA, pues es el más interesante de explicar matemáticamente.

Una vez finalizada la etapa de estudio, se procedió a la implementación de todos estos conceptos adquiridos a la hora de analizar y predecir la serie temporal elaborada en el punto 4.

3.2 Tecnologías empleadas

3.2.1 Python

El lenguaje de programación seleccionado para la realización de este trabajo ha sido Python con la versión 3.7.13.

Se ha elegido este lenguaje de programación debido a que implemente librerías muy útiles para el desarrollo de este trabajo. Las librerías que se han implementado en el desarrollo de este trabajo son:

Numpy

La librería de Numpy permite realizar transformaciones matemáticas a los datos de manera simple e intuitiva. La versión de esta librería en el trabajo es 1.21.6.

Pandas

Esta librería es esencial a día de hoy si se quiere trabajar con datos en Python que están en formato .csv (como era el caso). La versión empleada de esta librería es la 1.3.5.

Matplotlib

Principal librería de Python a la hora de mostrar gráficas. La mayoría de gráficas en el trabajo se han elaborado a partir de esta librería. Su versión es 3.5.2.

Seaborn

Seaborn es una librería de visualización de datos de Python que está construida sobre Matplotlib. Permite realizar gráficas sencillas de una manera fácil y rápida. La versión utiliza en este trabajo de esta librería ha sido la 0.11.2

Statsmodels

Esta librería, entre otras muchas cosas, tiene implementadas muchas de las funcionalidades vitales de este trabajo para el modelo ARIMA: contrastes de hipótesis, construcción de modelos ARIMA, etc. La versión utilizada es 0.11.0.

pmdarima

Esta librería, específica para ARIMA, sirve el mismo propósito que la anterior librería. Esta librería se ha empleado entonces por el método `auto_arima()` . Este método, utilizado en el trabajo, devuelve el mejor modelo ARIMA posible —dentro de un rango de posibilidades—. La versión de esta librería en el trabajo es 2.0.4 .

Sklearn

Librería de aprendizaje automático de muy alto nivel. Algunos métodos, para la preparación de datos para el modelo LSTM, han sido empleados por esta librería. Su versión es 1.0.2.

Keras

Librería especializada en redes neuronales de un alto nivel (aunque no tan alto como la anterior). Esta librería ha sido fundamental para la construcción del modelo LSTM. La versión empleada en el trabajo de esta librería ha sido 2.11.0.

Tensorflow

Librería de aprendizaje automático de un nivel medio sobre la que se sostiene la anterior. Al igual que la anterior, ha sido fundamental para la construcción del modelo LSTM. Su versión es 2.11.0.

Prophet

Librería para la construcción del modelo Prophet. Su versión es 1.1.5.

3.2.2 Jupyter notebook

Para el entorno de desarrollo se ha optado por utilizar Jupyter notebook. Esto es debido a que este es un entorno perfecto si se quiere combinar texto con código. En el caso concreto de este trabajo, en el siguiente punto, Desarrollo, está lo más esencial del proceso. Sin embargo, en el código del desarrollo, al estar aplicado en Jupyter notebook, todo el proceso está con mucho más detalle. Es por esta razón por la que se ha decidido emplear este IDE.

4 Desarrollo del trabajo

4.1 Datos

Para el análisis temporal de la productividad se ha optado por elegir, entre varios *datasets*, el encontrado en OECD data. Este *dataset*, además de contener suficientes datos para el análisis temporal, contiene datos de varios países en los que la variable evaluada es únicamente la productividad. Información adicional como el tiempo y la frecuencia de los datos están incluidos en este *dataset*. En este caso, se tratan de datos desde el año 1960 hasta el año 2021 con una frecuencia trimestral.

Para el valor de la productividad, según la fuente de datos, se ha medido de la siguiente manera:

> *"Se calcula multiplicando el número medio de horas trabajadas por la medida de empleo obtenida para cada país considerado."*

Algunas transformaciones, como la limpieza de datos, exploración de estos y selección de países europeos, han sido necesarias. Respecto a esto último, **se ha considerado muy interesante poder medir la productividad de los países cercanos a España** y hacer una **media** de esta variable para, de esta manera, poder obtener una productividad media europea.

Por último, el código con una explicación más detallada del proceso se encuentra alojado en https://github.com/JCOQUE/TFG-matematicas.git.

4.2 Modelos

El *dataframe* resultante de las transformaciones realizadas con el fin de obtener datos temporales es el siguiente:

Tabla 6: Dataframe desarrollo.

Fecha	Productividad
1960-03-01	0.284764
1960-06-01	0.287928
1960-09-01	0.291303
⋮	⋮

Aquí se puede ver de una manera más clara la periodicidad trimestral de los datos en la columna de `Fecha` (YYYY:MM:DD), y la productividad media europea en la columna `Productividad`.

Para una visión más amplia de cómo evolucionan los datos a lo largo del tiempo, se puede ver la siguiente imagen, en donde se muestra la serie temporal asociada a los datos de la tabla 6.

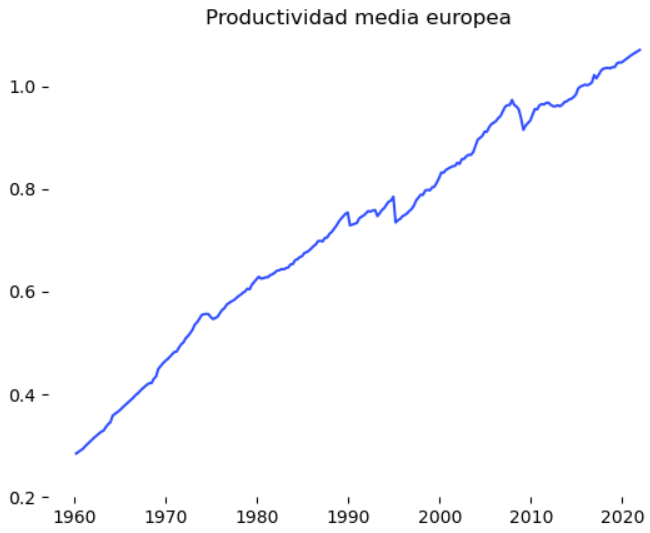

Figura 4.1: Serie temporal productividad.

Los algoritmos para la predicción de esta serie temporal se han aplicado en el mismo orden que la explicación de estos en este documento: ARIMA, LSTM y Prophet.

A continuación se explica ARIMA, donde el análisis y la evaluación de la serie temporal es un paso fundamental.

ARIMA

Para crear un modelo ARIMA, hay una serie de pasos, muy bien definidos, que deben hacerse. Estos pasos son los siguientes:

Paso 1) Analizar la serie temporal y sus componentes.

La serie temporal de la figura 4.1 tiene una clara tendencia ascendente; por lo que seguramente sea necesario diferenciarla mínimo una vez.

Por otro lado, no parece mostrar ningún patrón que se repita en un intervalo regular de tiempo; lo que indica que seguramente un modelo ARIMA (y no SARIMA) sea el más apropiado.

Por último, la serie temporal sí parece mostrar dos ciclos: uno a finales del siglo XX y otro alrededor de 2008. Estos picos sugieren un análisis adicional que podría explicar por qué surgieron y cuáles fueron sus causas y sus consecuencias. Dicho análisis va más allá del contenido de este trabajo.

Paso 2) Si la varianza no es constante a lo largo del tiempo, transformar la serie temporal aplicando la transformación de Box-Cox: el logaritmo.

En el caso de la serie temporal en la gráfica 4.1, en una primera instancia en el desarrollo, se decidió no aplicar esta transformación debido a que, aparentemente, la serie temporal no mostraba heterocedasticidad.

No obstante, a la hora de proseguir con los siguientes pasos y generar el modelo, el contraste de hipótesis de **Breusch-Pagan** sugiere aplicar una transformación logarítmica a la serie temporal. Esto se puede hacer de una manera muy sencilla con el comando `np.log` de la librería `numpy`.

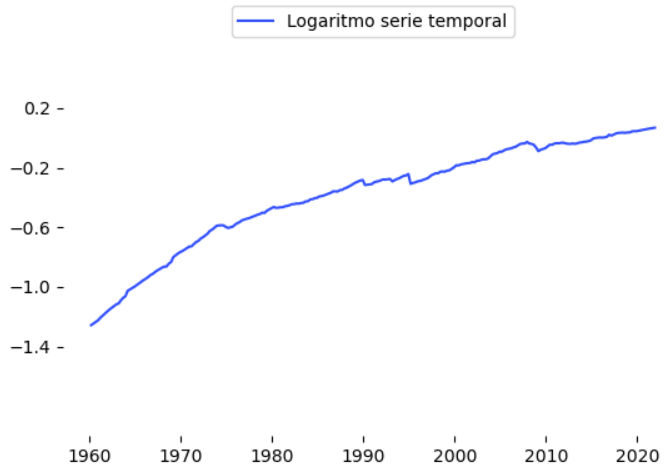

Figura 4.2: Serie temporal tras aplicar la transformación Box-Cox.

Aunque tenga una forma muy parecida a la serie temporal original, el eje y es completamente distinto, y con una tendencia menos pronunciada.

Paso 3) Aplicar los test de ADF y KPSS para determinar si la serie temporal es estacionaria.

Al aplicar estos contrastes de hipótesis a la gráfica 4.2, estos discrepan sobre si es necesario aplicar una diferencia de primer orden a la serie temporal. O lo que es lo mismo, si la serie temporal es estacionaria.

El test ADF indica que la serie es estacionaria y, por lo tanto, no hace falta aplicar ninguna diferencia. Por otro lado, el KPSS sugiere todo lo contrario. Debido a la forma de la serie temporal en la gráfica 4.2, se decide aplicar una primera diferencia (i.e. resultado KPSS). Esto se puede hacer a través del comando `diff()` .

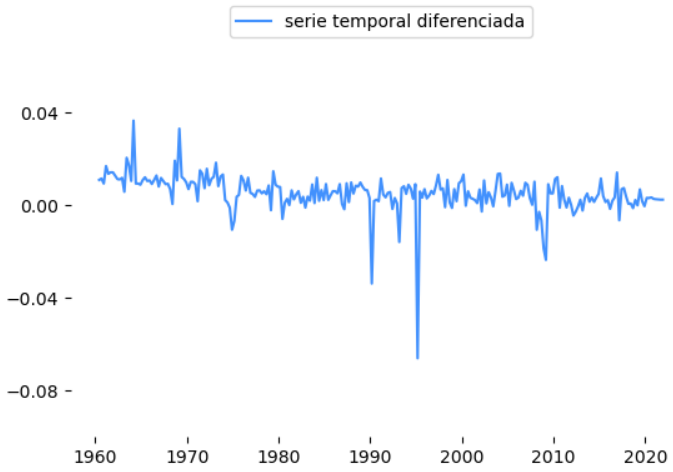

Figura 4.3: Serie temporal 4.2 tras haberle aplicado una diferencia de primer orden.

Una vez aplicada esta diferencia, ambos test ADF y KPSS vuelven a discernir sobre si la serie temporal necesita ser diferenciada por segunda vez. Se decide esta vez no aplicar ninguna diferencia adicional debido a que la serie temporal de la figura 4.3 sí parece cumplir mejor con las propiedades de una serie estacionaria (media y varianza constante; sin autocorrelación). Luego, esta vez se decide seguir la sugerencia del test ADF y no diferenciar esta serie temporal. Por tanto, en este paso 3), se sabe ya el primero de los 3 parámetros ARIMA(p, d, q): $d = 1$.

De todas maneras, al ser una decisión intuitiva y empírica, si el modelo ARIMA resultante no pasase los test de validación (paso 7), lo primero a probar para mejorar el modelo, sería aplicarle una segunda diferencia a la serie temporal 4.2.

Paso 4) Examinar los gráficos ACF y PACF y sacar conclusiones acerca de posibles modelos a emplear.

Los gráficos ACF y PACF indican, entre otro tipo de información, posibles y/o aproximados valores de los parámetros p y q del modelo ARIMA(p, d, q).

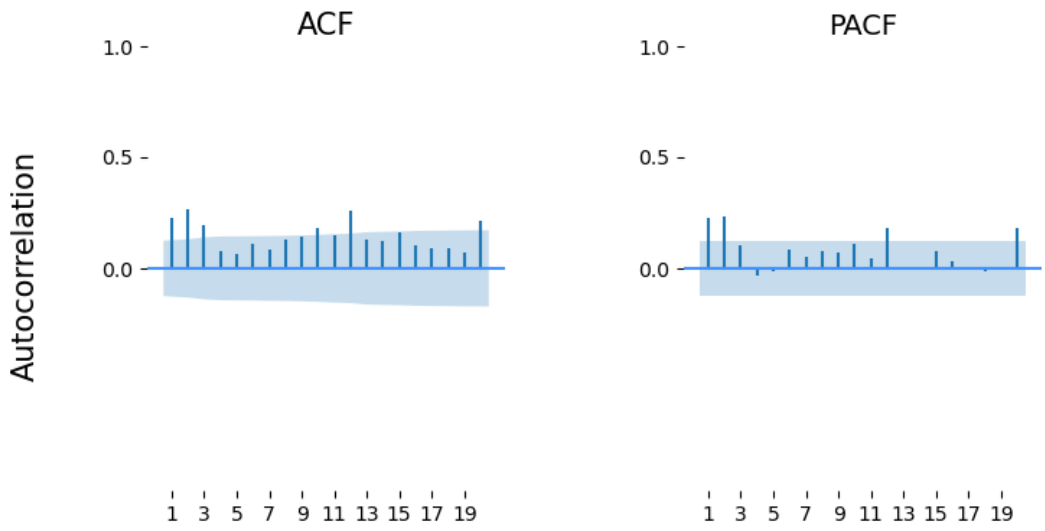

Figura 4.4: Gráficos ACF y PACF de la serie temporal 4.3.

De la figura 4.4 se extraen las siguientes conclusiones:

ACF

- Debido a que no presenta un patrón claro (ni decreciente exponencialmente, ni sinusoidal) y tampoco decrece rápidamente a cero, no se puede extraer información concluyente con respecto a si el valor de p pueda ser —seguramente— mayor a cero. Esto se determinará con ayuda del PACF.
- Los tres primeros *lags* sobresalen la zona azul. Esto indica que como posibles valores a probar del parámetro q deberían ser $1, 2$ y 3.

PACF

- Presenta un patrón ligeramente sinusoidal. Por tanto, q seguramente sea mayor a cero (en este caso, por el ACF, no había duda).
- Los dos primeros *lags* sobresalen la zona azul; luego como posibles valores de p están 1 y 2.

Luego, los modelos ARIMA(p, d, q) que se van a probar son los siguientes:

- ARIMA $(1, 1, 1)$, ARIMA $(1, 1, 2)$, ARIMA $(1, 1, 3)$, ARIMA $(2, 1, 1)$, ARIMA $(2, 1, 2)$, ARIMA $(2, 1, 3)$.

Paso 5) Creación y ajuste de modelos.

Para crear los modelos, se debe llamar a la función `ARIMA` del módulo `statsmodels`, e indicar el orden de cada modelo (i.e. parámetros p, d y q.). Una vez creados los modelos, se deben ajustar los datos. Para ello, se emplea el comando `modelo.fit()`. Por último, se puede ver un resumen muy completo del modelo con el método `modelo.summary()`. En este resumen, viene información acerca del orden del modelo, su valor AIC, BIC, valores de los coeficientes del modelo ARIMA, cómo de significativos son estos coeficientes, varios contrastes de hipótesis junto con sus respectivos p-valor (muy útiles la validación del modelo), etc.

Paso 6) Elegir el mejor modelo según su AIC y BIC.

Para ver el correspondiente valor AIC (o BIC) de un modelo se puede hacer de dos maneras: `modelo.summary()` ó `modelo.aic` (`modelo.bic` para BIC). El segundo método es mucho más directo.

El modelo cuyo valor AIC es el más bajo es el modelo ARIMA$(2, 1, 3)$; mientras que el modelo con el BIC más bajo es el modelo ARIMA$(1, 1, 1)$. Debido a que la diferencia entre los valores AIC es muy pequeña entre ambos modelos, y muy grande respecto al valor BIC, **se decide escoger como mejor modelo al modelo** ARIMA$(1, 1, 1)$.

Antes de continuar con la validación de este modelo, se decide emplear la función `auto_arima()` del módulo `pmdarima`. Esta función, dado un rango de valores para los parámetros p y q (en este caso), realiza una búsqueda exhaustiva con todas las posibles combinaciones de modelos entre los rangos proporcionados. **Devuelve** ARIMA$(1, 1, 1)$ **como mejor modelo, corroborando que el estudio ARIMA hecho hasta ahora es correcto**.

Paso 7) Validación del modelo.

Para validar el modelo se ha decidido realizar cuatro contrastes de hipótesis. Se pueden utilizar otros para los mismos fines. Se ha decidido emplear estos por ser los que la función `ARIMA` realiza y proporciona sus respectivos valores. Los valores de los estadísticos y p-valor asociados a cada contraste, se pueden ver haciendo `modelo.summary()`.

El primer contraste de hipótesis que se realiza es el test de Breusch-Pagan. Este contraste trata de determinar, con cierto grado de confianza, si el modelo presenta homocedasticidad (recuérdese que en el **paso 2** se mencionó que en un principio no se aplicó la transformación Box-Cox a la serie temporal, y luego se vio que era necesario. Fue por el p-valor de este test). El p-valor (`Prob(H)`) asociado al

modelo ARIMA$(1, 1, 1)$ de este contraste de hipótesis es de $0, 50$. Como la H_0 de este test es que la serie temporal presenta homocedasticidad, **no se rechaza la hipótesis nula**.

Una vez pasado el primer test de validación, se continúa con los siguientes. Los dos próximos tests se centran en algo fundamental: los residuos del modelo. Estos residuos deberán ser ruido. De lo contrario, se interpreta que el modelo no ha sido capaz de recopilar y aprender toda la información de todas las componentes de la serie temporal; y, por tanto, que se debería probar con otro modelo, o a hacer alguna transformación necesaria a la serie temporal dada.

Para poder comprobar que los residuos se pueden considerar ruido, deben seguir una distribución normal con media cero y además no debe haber autocorrelación entre ellos.

Para lo primero, se puede ver cómo están distribuidos los residuos en las siguientes gráficas:

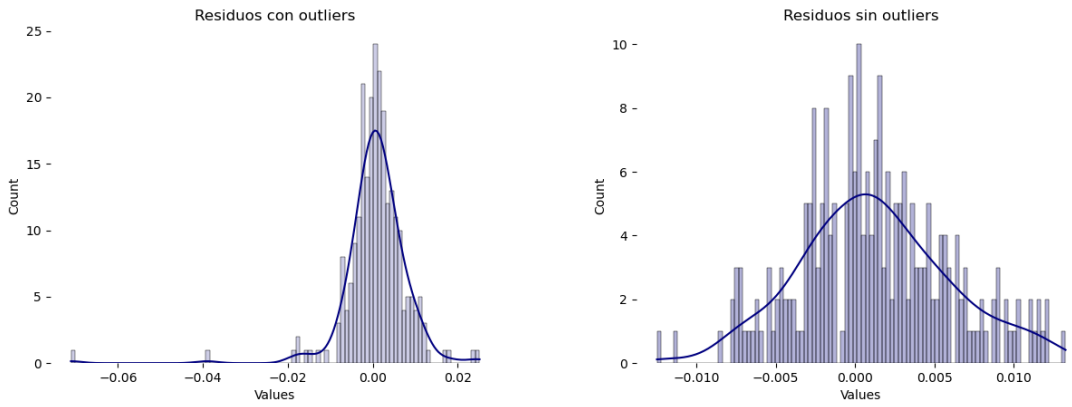

Figura 4.5: Residuos modelo ARIMA(1,1,1).

El contraste de hipótesis que se ha empleado para determinar si se puede aceptar que los residuos siguen una distribución normal es el test de Jarque-Bera. En la imagen de la izquierda de la figura 4.5, se puede ver una forma de campana de Gauss casi perfecta. Sin embargo, unos pocos *outliers* generan una distribución asimétrica. Esta asimetría conduce al test Jarque-Bera (Prob(JB)) a rechazar que los residuos sigan una distribución normal. Para tratar de solventar esto, se ha decidido eliminar los residuos que eran *outliers*. En total, eran 11 (se mantienen el 95.55% de los residuos originales). Una vez eliminados estos residuos, en la gráfica de la derecha de la figura 4.5, ahora ya sí, la campana de Gauss es simétrica y, por tanto, **el test Jarque-Bera (con un p-valor** $= 0.62$**) sugiere que los residuos siguen una distribución normal**.

Respecto a la autocorrelación de los residuos, se emplea el contraste de hipótesis de Ljung-Box[27]. Este

test mide la correlación entre un residuo y los *lags* anteriores. Este número de *lags* suele ser un número representativo de la serie temporal. Es decir, si la serie temporal tiene estacionalidad cada 12 periodos, entonces $lags = 12$. Esta serie temporal, al tener datos trimestrales durante varios años, un posible candidato sería $lags = 4$ (pues hay 4 trimestres al año). Los p-valores (Prob(Q)) asociados a cada uno de estos *lags* llevan a la conclusión de que los residuos no presentan autocorrelación entre ellos[28]. Véase la siguiente figura.

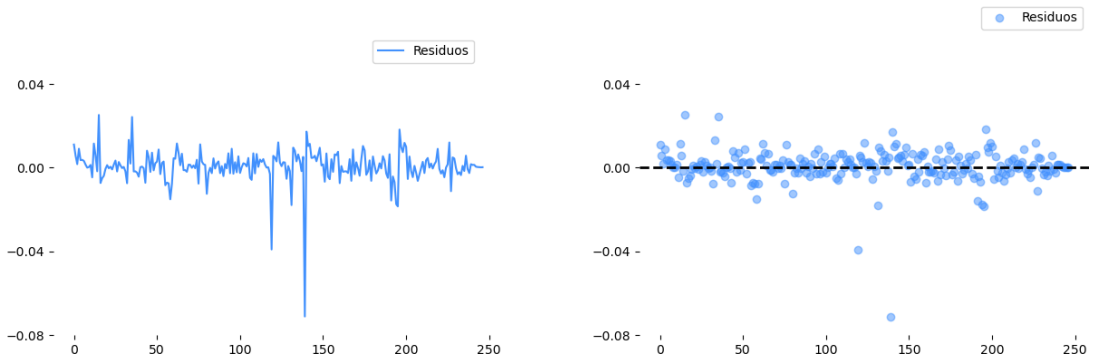

Figura 4.6: Dos maneras comunes de visualizar los residuos.

Por último, hay que revisar si todos los coeficientes de ARIMA$(1, 1, 1)$ son significativos. En caso de que no lo sean, deberán ser eliminados del modelo. La ecuación asociada al modelo ARIMA$(1, 1, 1)$ es:

$$\text{ARIMA}(1, 1, 1) = c + \phi y_{t-1} + \epsilon_t - \theta\epsilon_{t-1}$$

En donde, recuérdese que:

- c es una constante estimada por el modelo.
- ϕ el parámetro AR(1) estimado por el modelo.
- θ parámetro MA(1) estimado por el modelo.
- y_{t-1} es el valor de la serie temporal en el periodo anterior.
- $\epsilon_t, \epsilon_{t-1}$ son el error del ajuste actual y previo respectivamente por el modelo.

[27]Este test es el más común, y en algunos casos únicamente se emplea este contraste de hipótesis en la validación.

[28]De hecho, el test ADF, indica, con mucha confianza, que estos residuos son una serie estacionaria.

El valor de cada parámetro del modelo, se puede revisar con el comando `modelo.params`. En este caso, el modelo obtenido es el siguiente:

$$\text{ARIMA}(1,1,1) = 0 + 0.96y_{t-1} + \epsilon_t - -0.74\epsilon_{t-1} \tag{4.2.1}$$

Respecto a si estos coeficientes son significativos o no, se aplica el z-score. La H_0 de este test es que el coeficiente no es significativo. Por tanto, un p-valor por debajo de 0.05 sugiere que el coeficiente es significativo y, por tanto, debería tenerse en cuenta. Este p-valor se puede observar con el comando `modelo.summary().tables[1]`, ó, de manera más directa, con `modelo.pvalues`. Este comando muestra que el p-valor asociado a todos los coeficientes es de cero. Por tanto, esto lleva a pensar que **todos los coeficientes del modelo son significativos**.

En la siguiente imagen se muestra cómo el modelo de la ec. 4.2.1 se ajusta a los datos. Para ello, se emplea el comando `predict(start, end)`.

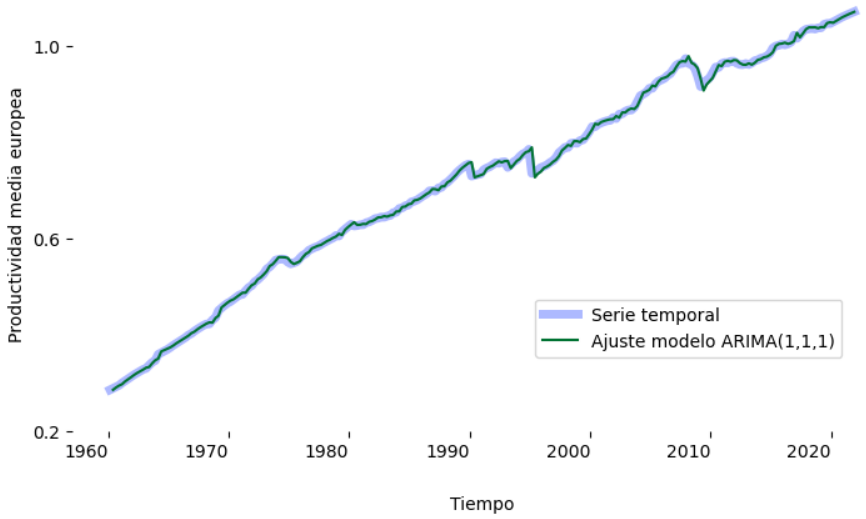

Figura 4.7: Ajuste modelo ARIMA(1,1,1).

En la gráfica 4.7 se puede ver cómo el modelo ajusta, de manera bastante precisa, la serie temporal. No se puede definir si el modelo está sobreajustado porque pese a que ajusta muy bien la recta, se trata de un modelo simple. Además, es el mejor encontrado por `auto_arima()` a través de las métricas AIC y BIC, que penalizan modelos muy complejos y, por tanto, suelen seleccionar modelos que no generen *overfitting*.

Paso 8) Realizar las predicciones.

Para realizar las predicciones, se puede hacer tanto con el comando `predict(start,end)`, como con el comando `forecast(periods)`. Se ha decidido usar el segundo por ser específico para esta tarea. **Tanto en este modelo ARIMA, como en LSTM y Prophet, se ha considerado adecuado predecir los siguientes 8 periodos**. Como el periodo es trimestral, se están prediciendo los siguientes dos años.

Véase la siguiente figura donde se muestra esta predicción.

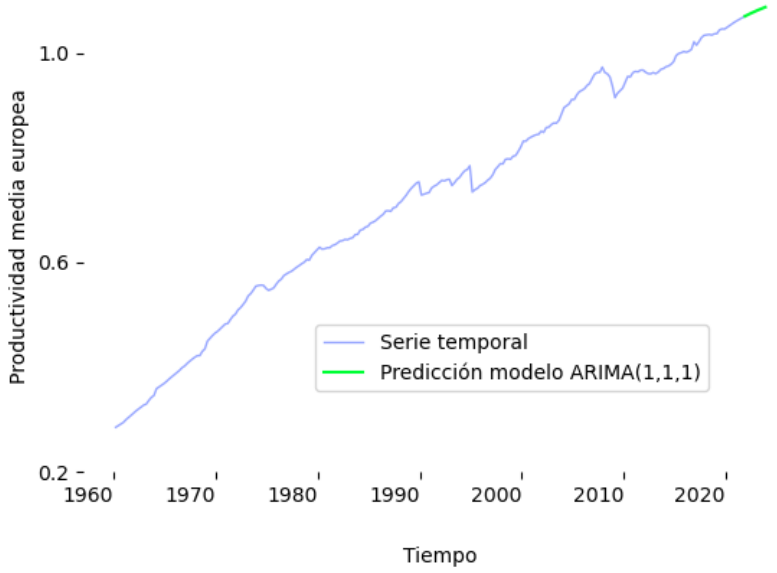

Figura 4.8: Predicción modelo ARIMA(1,1,1).

LSTM

Los pasos seguidos para la realización de este modelo son propios de las redes neuronales. Estas son un tema extenso y muchos de los conceptos que no se han explicado en esta memoria están en el propio desarrollo. Aquí se mencionarán los pasos fundamentales para la construcción del modelo LSTM obtenido.

Paso 1) Normalización de los datos.

Un paso no necesario, pero si muy recomendable a la hora de entrenar una red neuronal, es la normalización de los datos. Esto simplifica el entrenamiento de la red neuronal al trabajar con números pequeños. Para esta tarea, se ha empleado la función `MinMaxScaler`.

Paso 2) División de datos en train y test.

Mientras en el modelo ARIMA, a pesar de poder validar el modelo particionando los datos en *train* y en *test*, no es lo común, este proceso en redes neuronales es fundamental a la hora de validar un modelo (hay más como *cross-validation*). Se ha establecido una de las divisiones estándares: 80% de los datos en *train* y el restante 20% en *test*.

Algo muy importante a destacar es que, esta división no debe ser aleatoria (como muy frecuentemente se hace). Al ser una serie temporal lo que se está tratando de modelar, el orden de los datos importa; y mucho. Por ello hay que indicar a la función `train_test_split` el parámetro `shuffle = False` para que esta división en los conjuntos *train* y *test* sea de manera ordenada.

Paso 3) Formateo de los datos.

La red neuronal LSTM espera los datos en un formato concreto: `(batch_size, timesteps, channels)`. Para poder realizar la predicción y, al mismo tiempo, formatear los datos a tres dimensiones, se emplea la clase `TimeseriesGenerator`. Esta clase utiliza `timesteps` valores pasados para realizar la predicción del siguiente periodo. Debido a la tendencia de la serie temporal (lo cual indica que valores próximos son muy útiles para la predicción), se decide emplear únicamente tres valores pasados para la predicción: es decir, `timesteps = 3`.

Como al modelo únicamente se le van a pasar un lote de 3 datos, `batch_size = 1`. Finalmente, al tratarse de una serie temporal univariante: `channels = 1`.

Una vez definido esto, se puede pasar a definir la arquitectura de la red neuronal.

Paso 4) Arquitectura de la red neuronal.

La arquitectura se compone de:

- **1 neurona en la capa de entrada**: Al haber una única variable de entrada (valores pasados de la serie temporal), el número de neuronas en la capa de entrada es 1.
- **Capas ocultas: La arquitectura final tiene dos capas ocultas**: de 64 y 32 neuronas respectivamente. Tanto el número de capas como el número de neuronas en cada capa se ha establecido de forma empírica[29].
- **1 neurona en la capa de salida**: Como el problema en cuestión es una predicción de una serie temporal, debe haber una única neurona en la capa de salida. La función de activación de esta

neurona (lo que determina la salida y, por ende, el rendimiento de la red neuronal), pese a que ReLu teóricamente debería de funcionar mejor para este problema, se ha visto que la función Lineal obtenía mejores resultados.

Adicionalmente, se añade `Dropout = 0.2` en cada capa oculta como medida de regularización del modelo. Y, además, una capa de `batchNormalization` en la salida de cada capa oculta; lo cual mejora el rendimiento de la red neuronal.

Paso 5) Definición de la función de pérdida y optimizador.

Como función de pérdida se establece `loss = mean_squared_error` por su popularidad en problemas de este tipo. Adicionalmente, se elige, también porque es muy común utilizarlo, el optimizador `Adam`. Por último, se establece un `learning_rate = 0.1`, pues se ha visto que era con el que mejores resultados se obtenía.

Paso 6) Ajuste del modelo LSTM.

En este paso, se indica el número de `epochs`, el cual es establecido a 100 (estándar). El `batch_size` no hace falta definirlo porque ya fue definido en el paso 3.

A lo recientemente mencionado, se le añaden dos características muy importantes:

- `ModelCheckpoint` para guardar el mejor modelo generado durante el entrenamiento (más entrenamiento $\neq$ mejor rendimiento).
- `EarlyStopping` donde se especifica que si pasadas 10 *epochs* `patience = 10`, el valor de la función de pérdida del conjunto de validación[30] no ha mejorado, se pare el entrenamiento. Esto con el fin de ahorrar entrenamiento innecesario y que el modelo desaprenda.

Paso 7) Validación del modelo.

Para validar el modelo, se ha decidido emplear la métrica *MAE (Mean Absolute Error)* por su popularidad y su fácil interpretación. Como se ha dividido el conjunto de datos en *train* y *test*, se ha obtenido un MAE de estos dos conjuntos:

[29] Se ha seguido una regla no escrita en la arquitectura de redes neuronales: número de neuronas como potencia de 2; y cada capa disminuir esta potencia en uno.

[30] Conjunto de datos empleado para evaluar el rendimiento del entrenamiento de una red neuronal en cada *epoch*.

- **MAE train**: 0.2580
- **MAE test**: 0.1101

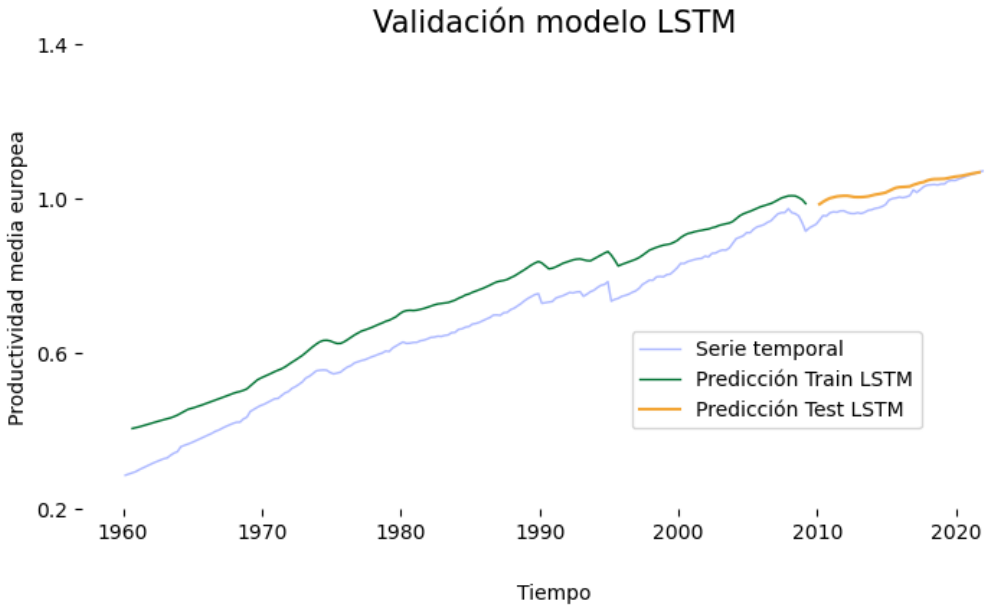

Figura 4.9: Ajuste modelo LSTM.

En la figura 4.9, se puede ver por qué `MAE test` es menor al `MAE train`: se ajusta mejor a sus respectivos datos. `MAE train`, pese a que tiene una forma casi idéntica a la serie temporal, sobreestima un poco los datos.

De los +100 modelos que se han probado, este ha sido el mejor que se ha obtenido.

Paso 8) Predicciones LSTM.

Para realizar predicciones en más de un periodo en el futuro con `TimeseriesGenerator`, cada predicción se debe concatenar a lista de valores empleados para la siguiente predicción. Es decir, como en este caso se emplean 3 valores para predecir el siguiente, esto se puede ver de esta manera:

- $[1, \cdots, 3] \Rightarrow 4$
- Se actualiza
- $[2, \cdots, 4] \Rightarrow 5$
- $\vdots$

Véase la siguiente figura para ver las predicciones con el modelo LSTM.

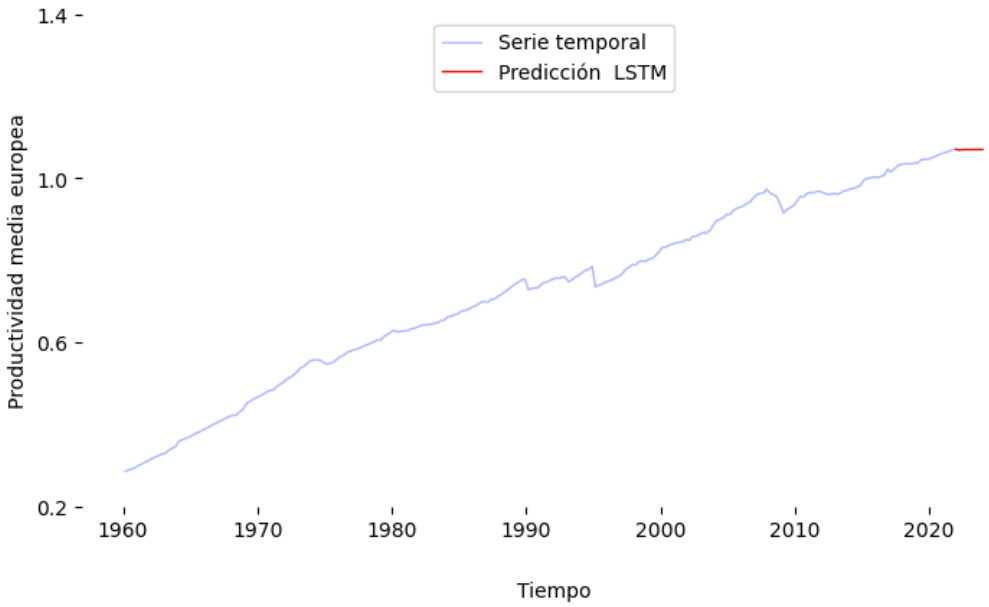

Figura 4.10: Predicción modelo LSTM.

Prophet

Para modelizar el algoritmo de Prophet, se deben seguir unos pasos distintos a los de los modelos anteriores. Se trata del algoritmo más sencillo de emplear de los tres, pues simplemente indicando algunos parámetros intuitivos, se puede llegar a modelizar la serie temporal con bastante efectividad.

Paso 1) Transformaciones necesarias.

Prophet, uno de los requisitos para poder modelar series temporales, es tener los datos en un *dataframe* en un formato determinado: tener una columna con las fechas (ordenadas) denominada `ds` y otra columna con los datos de la variable objetivo con el nombre de `y`. Esto se vería de la siguiente manera:

Tabla 7: Dataframe Prophet.

ds	y
1960-03-01	0.284764
1960-06-01	0.287928
1960-09-01	0.291303
⋮	⋮

En este *dataframe* estarán incluidos los datos de entrenamiento.

Además, aunque no haya una capacidad máxima (que se podría especificar si la hubiera), sí hay una capacidad mínima para la columna `y` : 0. El valor de la columna `Productividad` — del *dataframe* original— no puede ser negativo. Esto se le indica al modelo añadiendo la columna `floor` al *dataframe* de la tabla 7.

Por último, se ha decidido añadir las fiestas de Navidad por si pudiera ayudar a Prophet a ajustar el modelo de una manera más correcta.

Paso 2) Creación del modelo Prophet.

Para crear el modelo Prophet, se hace con la función `Prophet` y se le indican, para un primer modelo, los parámetros de `growth = linear` indicando que la serie temporal tiene un crecimiento lienal (y no logístico); y las Navidades. Sin embargo, se vio que este modelo, a pesar de obtener unos valores de `MAE train` y de `MAE test` relativamente bajos, no llegaba a ajustar del todo bien el pico de la serie temporal que hay alrededor del año 2008. Esto ocasionaba que el modelo sobreestimase los valores en el conjunto de *test*. Véase la siguiente figura.

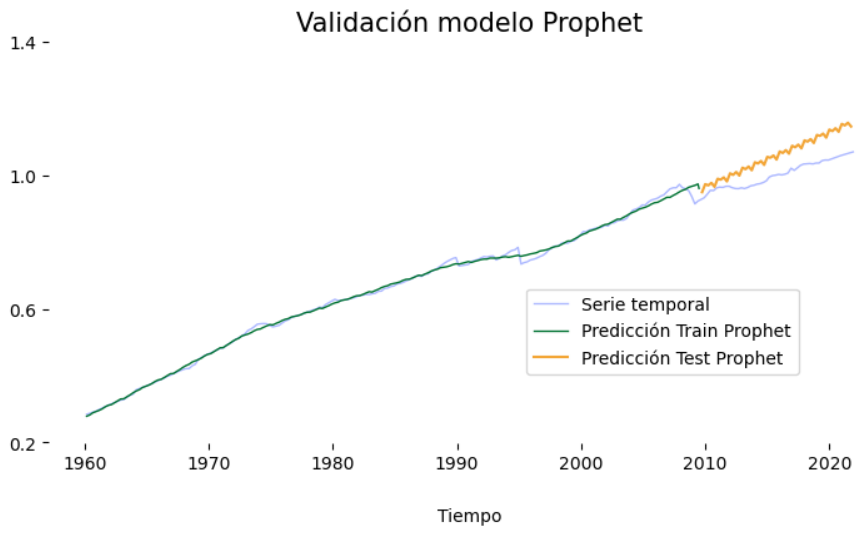

Figura 4.11: Ajuste modelo Prophet original.

Esto se debe a que Prophet detecta cambios en la tendencia:

1) En los primeros 80% de los datos (y este pico está al final).

2) Con cierta rigidez.

Véase la siguiente figura.

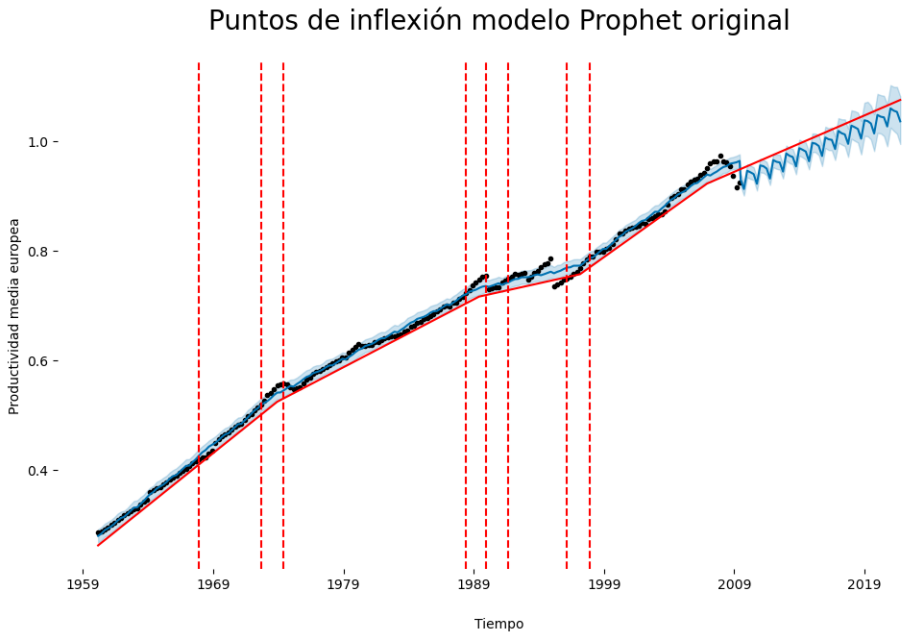

Figura 4.12: Puntos de inflexión modelo Prophet original.

En la figura 4.12, las líneas verticales rojas indican los puntos de inflexión donde Prophet detecta un cambio en la tasa de crecimiento de la tendencia. Nótese que, como se indicaba recientemente, no hay ninguna línea vertical en el último 20% de los datos de *train* y, además, detecta cambios de tendencia mínimos que parecen innecesarios. Para cambiar esto y tratar de conseguir un mejor modelo, además del tipo de tendencia y las vacaciones, se especifica el rango de puntos de inflexión con `changepoint_range = 0.99` (0.8 por defecto). Adicionalmente, con `changepoint_prior_scale = 0.03` (0.05 por defecto), se indica al modelo que sea más rígido a la hora de detectar cambios en la tendencia. Luego, con esto último, se está tratando de crear un modelo más simple.

Paso 3) Ajuste del modelo.

Para ajustar el modelo de Prophet a los datos se debe usar la función `fit()`. En la siguiente figura se puede ver cómo este ajuste es mejor al mostrado en la figura 4.11.

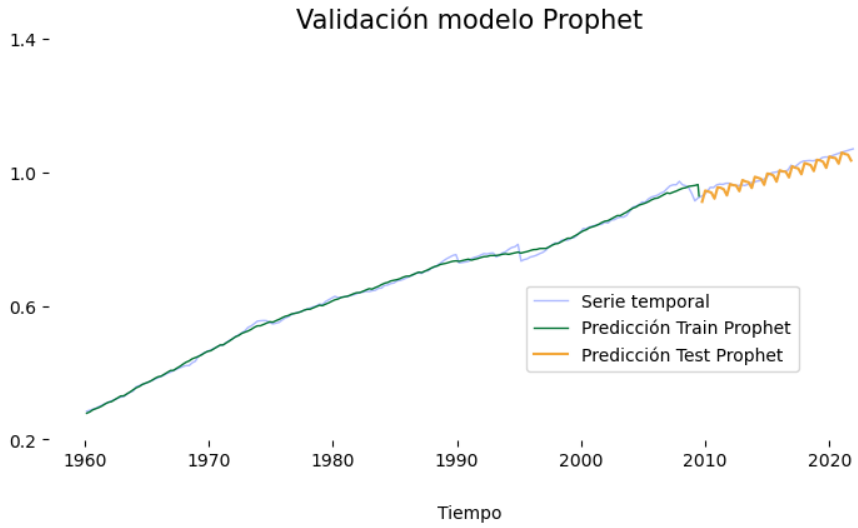

Figura 4.13: Ajuste modelo Prophet mejorado.

En la figura 4.13, se puede ver cómo ahora sí detecta puntos de inflexión en un rango mayor y, además, detecta muchos menos: solo los necesarios y suficientes. Como consecuencia, este modelo se ajusta mucho mejor a los datos de *test*.

Paso 3) Validación del modelo Prophet.

En este paso, al igual que con el modelo LSTM, se procede a obtener las métricas de `MAE train` y `MAE test`.

Los resultados, junto con la comparativa respecto al modelo anterior de Prophet, se pueden ver en la figura 4.14.

Modelo Prophet	MAE train	MAE test
Original	0.0065	0.0563
Mejorado	0.0062	0.0138

Figura 4.14: Comparativa modelos Prophet.

En la figura 4.14 se aprecia un valor de `MAE test` mucho menor en este segundo modelo de Prophet.

Paso 4) Predicciones con Prophet.

Para este último paso, Prophet crea un *dataframe* con las fechas a predecir y su frecuencia (trimestral en este caso). Estas fechas se le indican en la función `make_future_dataframe`. En la siguiente figura, se pueden ver las predicciones realizadas por Prophet.

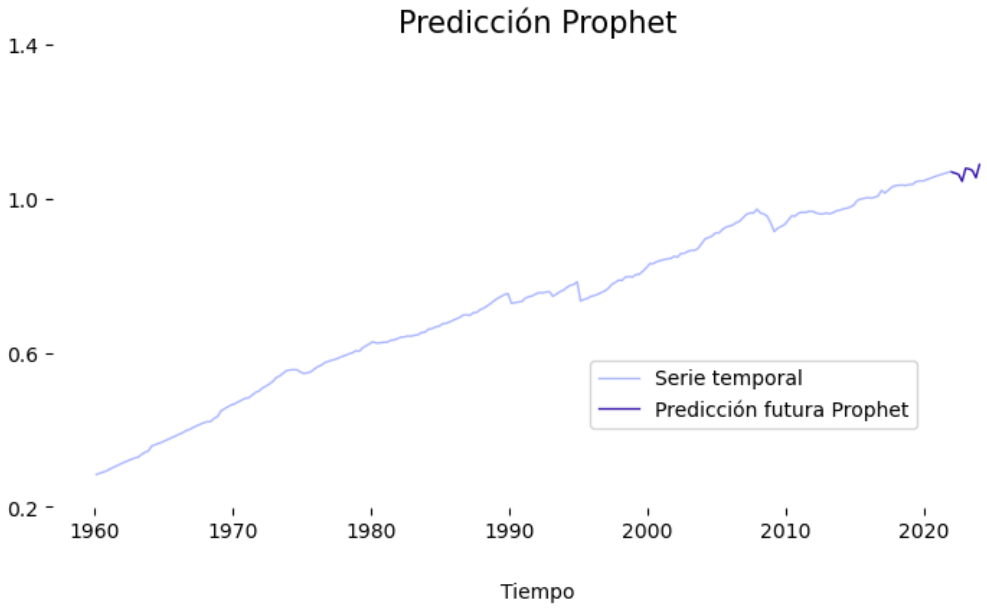

Figura 4.15: Predicción modelo Prophet.

En la figura 4.15, se puede ver que Prophet, con respecto a los anteriores modelos, es el único que no ha estimado una tendencia recta.

4.3 Comparativa

Los tres modelos que se han empleado, cada uno ha obtenido unos resultados distintos. Una práctica muy común a la hora realizar predicciones, es emplear varios algoritmos y, posteriormente, comparar los modelos generados por cada uno de estos.

Para ello, se emplean las **técnicas de evaluación**. Estas difieren según el tipo de predicción. Concretamente, para series temporales, las dos más comunes son MAE y MSE (Maen Squared Error). Hay más y se pueden emplear varias de ellas. En este trabajo, se ha decidido únicamente emplear la medida de MAE para comparar los tres modelos. Se ha elegido esta métrica por su sencillez a la hora de su interpretación.

Algo que cabe destacar es que no se ha calculado el MAE del modelo ARIMA. Esto es porque se trata de un algoritmo estadístico y, pese a que se puede dividir en *train* y en *test* —como se ha hecho con LSTM y Prophet— la manera más correcta de evaluar este modelo estadístico es con estadística. Por ello, lo más correcto es emplear los contrastes de hipótesis utilizados en este trabajo para validar un modelo ARIMA.

No obstante, se va a calcular el MAE de ARIMA (tanto el de *train* como el de *test*) con el fin de poder comparar los tres modelos de una manera más sencilla.

Los resultados son los siguientes:

Modelos	MAE train	MAE test
ARIMA	0.0034	0.0031
LSTM actual	0.2580	0.1101
LSTM TFG	0.2580	0.1101
Prophet	0.0062	0.0138

Figura 4.16: Comparativa de todos los modelos.

En la figura 4.16 se pueden ver los resultados, tanto de `MAE train` como de `MAE test` de los modelos obtenidos. Cabe recalcar que el modelo `LSTM actual` es el ejecutado por el usuario (con los parámetros y modificaciones que este quiera) y `LSTM TFG` es el modelo explicado en este trabajo. Aunque sean únicamente 4 modelos, se pueden ver de manera mucho más directa los mejores y peores resultados en figura 4.17.

Modelos	MAE train	MAE test
ARIMA	0.0034	0.0031
LSTM actual	**0.2580**	**0.1101**
LSTM TFG	**0.2580**	**0.1101**
Prophet	0.0062	0.0138

Modelos	MAE train	MAE test
ARIMA	**0.0034**	**0.0031**
LSTM actual	0.2580	0.1101
LSTM TFG	0.2580	0.1101
Prophet	0.0062	0.0138

Figura 4.17: Peor (izquierda) y mejor (derecha) resultado de cada métrica de evaluación.

Las gráficas de cada modelo mostradas en el desarrollo daban indicios de cuáles iban a ser el mejor y peor modelo. No obstante, recuérdese que la comparativa ARIMA es algo artificial, pues no se ha entrenado ni evaluado de la misma manera.

Una posible razón por la que el modelo LSTM haya sido el peor, es porque este modelo suele aplicarse en secuencias con muchos datos (más que los 248 que tenía esta serie temporal) y, además, donde valores muy distanciados entre sí en la secuencia tienen correlación. Esto es debido a su capacidad de retener información de los datos secuenciales en un largo periodo de tiempo. En este caso, al tratarse de una serie temporal con una tendencia tan clara, con pocas fluctuaciones, puede haber sido el motivo de por qué LSTM ha obtenido el pero de los resultados.

Algo que también llama la atención, es que su `MAE train` es más alto que su `MAE test`. Esto no es común y una de las posibles explicaciones es que, los datos de entrenamiento sean más difíciles que los de *test*. En este caso, ambos ciclos (picos) que presentaba la serie temporal, entraban en el conjunto de *train*. Quizá haya sido este uno de los motivos.

Por otra parte, el algoritmo de Prophet, pese a tener unos valores de `MAE train` y `MAE test` relativamente bajos, no es el algoritmo más adecuado para esta serie temporal. Prophet, fue diseñado por Facebook, para modelar de la mejor manera posible series temporales frecuentes en Facebook. Estas son series estacionales con múltiples estacionales, varios *outliers*, falta de datos, etc. Sin embargo, esta serie temporal no presentaba ninguna de estas características. Aun así, Prophet ha sido capaz de modelar y ajustarse a los datos de una manera precisa, como se vio en la imagen 4.13.

Prophet, lo bueno que tiene respecto a los otros dos modelos, es que es más fácil e intuitivo de utilizar. En ARIMA, pese a utilizar el comando `auto_arima()`, si hubiera que retocar el modelo para mejorarlo, se necesitan conocimientos matemáticos y estadísticos como se ha podido ver en este trabajo. Por el otro lado, LSTM, al ser una red neuronal, gran parte del ajuste de parámetros se hace de manera empírica. No solo los resultados de la LSTM, al ser una red neuronal, tienen una interpretación difícil; sino que además a la hora de querer mejorar el modelo es difícil saber qué cambiar. Por no hablar de la complejidad y extensión base de las redes neuronales.

Por último, ARIMA, pese a ser el modelo más antiguo, ha sido el que mejor ha logrado obtener la información de los datos. No obstante, esta serie temporal era una muy buena candidata para ARIMA: no presentaba *outliers*, falta de datos, ni estacionalidad (en ese caso se aplica SARIMA).

El punto de esta comparativa es que, estos resultados no son extrapolables a otras series temporales,

y se debe estudiar cada una de ellas por separado y llegar a la conclusión de qué algoritmos podrían llegar a generar los mejores modelos para los datos dados.

Por último, se muestra una gráfica que contiene las predicciones de los 3 modelos.

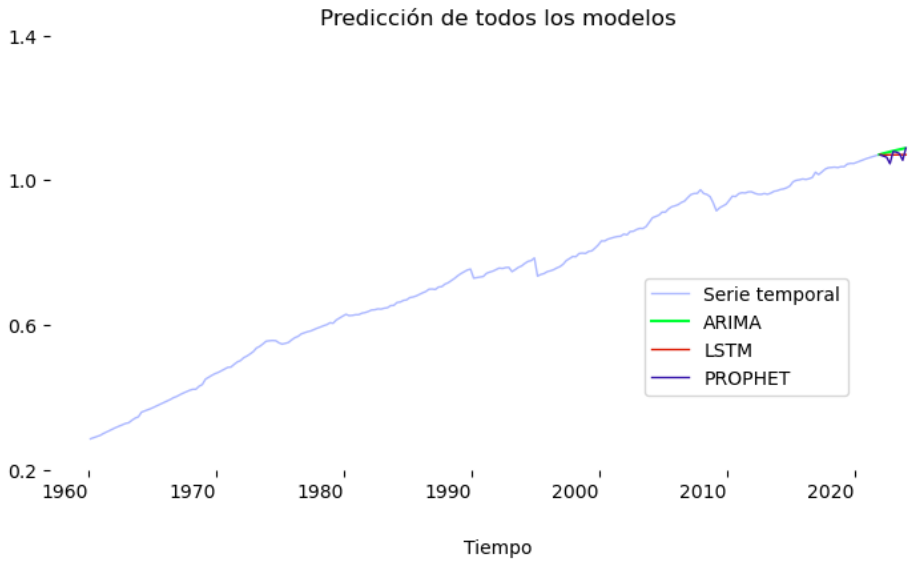

Figura 4.18: Predicciones de los tres modelos.

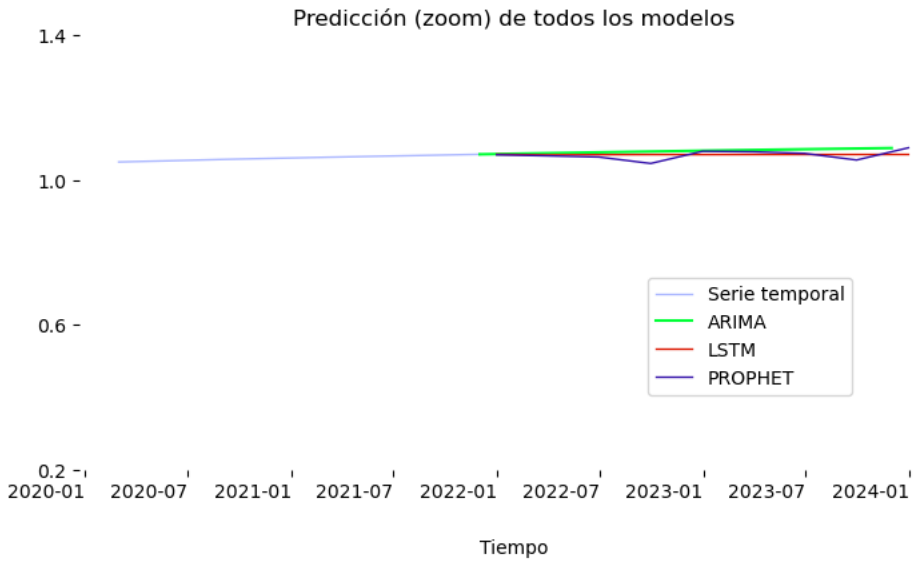

Figura 4.19: Predicciones (con zoom) de los tres modelos.

5 Conclusiones

En este estudio se ha podido analizar y predecir la productividad media europea entre los años 1960 y 2021, ambos incluidos. Para ello, se han empleado tres algoritmos de tres épocas distintas: ARIMA, LSTM y Prophet. En este trabajo en concreto, ARIMA ha sido el algoritmo que mejor ha conseguido capturar la información de los datos (véase 4.7, 4.17). Por otro lado, LSTM es el que peor métrica ha obtenido (4.17).

No obstante, en la figura 4.19 se observa que la predicción de los tres algoritmos ofrecen resultados bastante similares. Prophet es el único que predice una tendencia con cierta fluctuación, a diferencia de ARIMA y LSTM cuyas predicciones son más lineales.

Es interesante comprobar cómo, algoritmos tan distintos entre sí, con unas matemáticas por detrás que no tienen nada que ver de un algoritmo con otro, desarrollados en épocas tan distintas y con propósitos tan diferentes, han conseguido obtener resultados parecidos.

Este estudio, al ser tan amplio y abarcar diferentes posibilidades, permite plantear varias líneas futuras de trabajo. En primer lugar, se podría tratar de refinar los modelos obtenidos. Por ejemplo, una opción no empleada en este trabajo es el uso de librerías como `Optuna` o `GridSearch` para encontrar un modelo LSTM con los mejores hiperparámetros posibles a base de una búsqueda exhaustiva de estos. Además de mejorar los modelos, se pueden probar otros algoritmos muy empleados a día de hoy en series temporales como puede ser la red neuronal Transformer (Vaswani et al., 2017) o TCN (*Temporal Convolutional Network*) (Bai et al., 2018). También, si en un futuro apareciesen datos con mayor frecuencia (e.g. mensual), al tenerse más datos, se podría hacer un estudio más profundo y preciso. En quinto lugar, se podría hacer una comparativa de series temporales multivariantes. Es decir, comparar, a lo largo del tiempo, la productividad media europea con alguna otra variable de interés. Este trabajo daría para mucho y sería muy interesante si la productividad y alguna otra variable tienen alguna relación en el tiempo. Por último, se puede comparar la productividad media europea con las productividades medias de otros continentes. Esto último se pretende hacer en un futuro no muy lejano.

6 Apéndice

6.1 ¿Qué es una variable aleatoria?

Una variable aleatoria (X) es la variable de estudio en un análisis estadístico. Las propiedades de esta variable son desconocidas —i.e. media, varianza, etc.—.

Si el análisis estadístico se trata de evaluar la altura media de personas españolas mayores de edad, entonces:

$$X \sim \text{Altura de una persona española mayor de edad}$$

Este estudio, para que los resultados no sean sesgados, se deben obtener alturas de personas aleatorias —de ahí viene el término de variable *aleatoria*—.

Para este análisis estadístico hay que determinar cuántas alturas de personas aleatorias se van a obtener. Este número es representado por la letra n. Luego se obtiene un vector de n variables aleatorias: $X_1, X_2, \cdots, X_n$.

Cada una de estas variables aleatorias se puede ver como una muestra del análisis estadístico. Cada muestra observada tiene un valor asociado, i.e. la altura de cada una de las personas del estudio. Entonces, el vector de variables aleatorias *tiene asociado* un vector de valores observados denotado como $x_1, x_2, \cdots, x_n$.

6.2 Propiedades básicas de Esperanza

Sean X e Y dos variables aleatorias con esperanza finita y $c \in \mathbb{R}$ una constante.

1) $E[c] = c$.

2) $E[cX] = c \cdot E[X]$.

3) $E[X + Y] = E[X] + E[Y]$.

4) $E[XY] = E[X] \cdot E[Y]$ solo si X e Y son variables aleatorias independientes.

6.3 Propiedades básicas de Varianza

$$\begin{aligned}\mathrm{Var}(X) &= E[(X-\mu)^2] \\ &= E[X^2 - 2XE[X] + E[X]^2] \Rightarrow \text{identidad notable} \\ &= E[X^2] - 2E[X]E[X] + E[X]^2 \\ &= E[X^2] - E[X]^2.\end{aligned}$$

Sean X e Y dos variables aleatorias con esperanza finita y $c \in \mathbb{R}$ una constante.

1) $\mathrm{Var}[X] \geq 0$.

2) $\mathrm{Var}[c] = 0$.

3) $\mathrm{Var}[cX] = c^2 \cdot \mathrm{Var}[X]$.

4) $\mathrm{Var}[X+Y] = \mathrm{Var}[X] + \mathrm{Var}[Y]$ sólo si X e Y son variables aleatorias independientes.

6.4 Ejemplo de interdependencia de correlación

Sea Tim, un chico al que le gusta jugar al fútbol. Tim tiene un partido de fútbol todos los sábados. Para mantenerse algo activo el día de antes, Tim decide que los viernes sale a correr suave; un poco. En cambio, los domingos descansa y no hace deporte porque el partido de fútbol del sábado le deja muy cansado.

Supongamos que esta rutina, Tim la repite durante 8 semanas. En estas 8 semanas hubo 2 de ellas en las que Tim no salió a correr los viernes porque estaba lloviendo.

ACF interpretará que 6 de 8 semanas que Tim ha salido a correr los viernes, ha descansado el domingo y, por tanto, asignará una correlación alta entre salir a correr y descansar el domingo. Sin embargo, el **PACF** se 'daría cuenta' de que en esas dos semanas que no ha salido a correr, ha seguido descansando el domingo. Y que la correlación entre salir a correr los viernes (y_{t-2}) y descansar los domingos (y_t) era porque ambos estaban conectados a jugar el partido de fútbol el sábado (y_{t-1}). Por ende, el PACF asigna una correlación baja entre y_{t-2} e y_t (a diferencia del ACF).

6.5 Intervalo 95% en distribución gaussiana

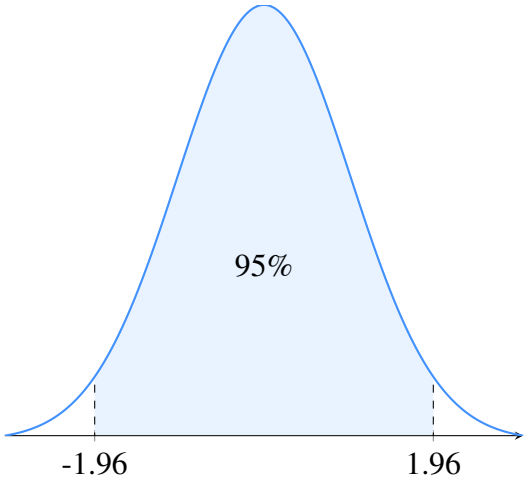

Figura 6.1: Intervalo de confianza al 95% - Distribución Normal tipificada.

6.6 Contrastes de hipótesis

Un contraste de hipótesis es una técnica estadística que se utiliza para comprobar la validez de una afirmación basándose en la información recogida en una muestra (serie temporal, en el caso de este trabajo) de observaciones. Se basa en dos hipótesis:

- Hipótesis Nula (H_0): Es la afirmación que se desea contrastar. La afirmación que inicialmente responde a cierta "creencia previa".
- Hipótesis alternativa (H_1 ó H_A): Se da por cierta cuando la información que proporciona la muestra no apoya la afirmación de la hipótesis nula.

Es importante considerar que, por convención matemática, nunca se menciona *aceptar* la H_A, sino que se *rechaza* la H_0.

6.6.1 p-valor

A *grosso modo*, cada contraste de hipótesis tiene un estadístico (de contraste). Un estadístico es una determinada función matemática que recibe como entrada los datos de muestra. La salida de esta función es el valor asociado al **p-valor**.

$$0 \leq p\text{-}valor \leq 1$$

El p-valor se puede definir de muchas maneras:

- Grado de compatibilidad de los datos con la H_0. Cuanto más próximo sea el p-valor a 0, más incompatible será la muestra observada con H_0.
- Probabilidad de que el estadístico de contraste tome el valor obtenido; o incluso muestre aún más discrepancia con H_0 cuando la realidad es que H_0 es cierta.
- Probabilidad de rechazar H_0 cuando es cierta.

Dependiendo del contraste de hipótesis que se esté haciendo y el ámbito en el que uno se encuentre, se podrá optar a asumir más o menos incertidumbre si se rechaza la H_0. Este grado de incertidumbre que se está dispuesto a asumir viene dado por α.

$$0 \leq \alpha \leq 1$$

El valor de α lo decide el propio analista, pero el valor por defecto es $\alpha = 0.05$.

El p-valor obtenido se comparará con α dando lugar a los siguientes escenarios posibles:

- $p\text{-}valor < \alpha$: Se rechaza la H_0.
- $p\text{-}valor = \alpha$ (o en torno a α): Es necesario un análisis más profundo. Se toma una decisión con mucho cuidado.
- $p\text{-}valor > \alpha$: No se rechaza la H_0.

6.7 Raíces unitarias

Nota

Se recomienda haber visto la familia ARIMA con anterioridad para entender mejor este concepto.

Para simplificar el procedimiento, se introduce el Operador *Lag* denotado con la letra B. Este operador hace más sencilla la manera denotar observaciones pasadas:

$$y_{t-1} = By_t$$

Puesto que la estacionariedad de un proceso ARIMA depende únicamente de la estacionariedad de su proceso AR —pues el proceso MA es siempre estacionario— se parte de la fórmula general de un proceso AR(p).

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} + \cdots + \phi_p y_{t-p} + \varepsilon_t \tag{6.7.1}$$

Aplicando el Operador *Lag* a 6.7.1, se llega al polinomio característico:

$$\begin{aligned} y_t &= \phi_1 B y_t + \phi_2 B^2 y_t + \cdots + \phi_p B^p y_t + \varepsilon_t \\ y_t &= y_t(\phi_1 B + \phi_2 B^2 + \cdots + \phi_p B^p) + \varepsilon_t \\ y_t &- y_t(\phi_1 B + \phi_2 B^2 + \cdots + \phi_p B^p) = \varepsilon_t \\ y_t &\underbrace{(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p)}_{\text{polinomio característico}} = \varepsilon_t \end{aligned} \tag{6.7.2}$$

Es común denotar al polinomio característico como $\phi(B)$. Luego, la ec. 6.7.2 de arriba se quedaría como:

$$y_t \underbrace{\phi(B)}_{\text{polinomio característico}} = \varepsilon_t$$

Igualando este polinomio a 0 se obtiene la **ecuación característica**:

$$\phi(B) = \underbrace{1 - \phi_1 B - \phi_2 B^2 - \phi_3 B^3 - \cdots - \phi_p B^p = 0}_{\text{ecuación característica}} \tag{6.7.3}$$

Las raíces de esta ecuación característica —i.e. valores de B que cumplen la ecuación— deben caer **fuera** del círculo unidad para que el proceso se considere estacionario. Es decir, que $|B| > 1$.

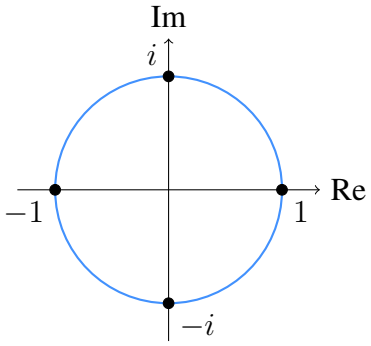

Figura 6.2: Círculo unidad en $\mathbb{C}$.

Ahora, sea $\lambda = B^{-1}$. Si a la ec. 6.7.3 se la multiplica a ambos lados por B^{-p} se obtiene:

$$\begin{aligned} B^{-p}(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p) &= B^{-p} \cdot 0 \\ \lambda^p(1 - \phi_1 \lambda^{-1} - \phi_2 \lambda^{-2} - \cdots - \phi_p \lambda^{-p}) &= 0 \\ \lambda^p - \phi_1 \lambda^{p-1} - \phi_2 \lambda^{p-2} - \cdots - \phi_p &= 0 \end{aligned} \tag{6.7.4}$$

Por el cambio de variable $\lambda = B^{-1}$, como $|B| > 1$, implica que $|\lambda| < 1$. Luego, las raíces de la ec. 6.7.4 deben caer **dentro** del círculo unidad (de ahí viene lo de raíces **unitarias**).

Como las raíces pueden ser complejas, El Teorema Fundamental de Álgebra[A8] demuestra que existen p raíces para la ecuación característica con un polinomio de grado p.

Es decir, sean $\lambda_1, \lambda_2, \cdots, \lambda_p$ las raíces de ec. 6.7.4. Entonces esta ecuación se puede reescribir como:

$$(\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_p) = 0 \tag{6.7.5}$$

Luego, para que el proceso sea estacionario, las p raíces $\lambda_1, \lambda_2, \cdots, \lambda_p$ deben caer dentro del círculo unidad de los complejos.

Por tanto, el contraste de hipótesis ADF, como su H_0 es que la serie temporal **no es estacionaria**, lo que trata de constatar es: si se puede afirmar que alguna de estas raíces cae **fuera** del círculo unidad. Por el contrario, como la H_0 del contraste de hipótesis es justo la contraria —que la serie temporal **es estacionaria**— lo que trata de verificar este contraste es: si se puede afirmar que todas las raíces caen **dentro** del círculo unidad.

Se dejan dos ejemplos adicionales de la restricción de valores que pueden tomar los parámetros ϕ en un proceso AR(1) y en un proceso AR(2) respectivamente.

Ejemplo 1

Considérese el proceso AR(1). Para simplificar los cálculos, se impone que $c = 0$:

$$y_t = \phi y_{t-1} + \varepsilon_t$$

La ecuación característica es la siguiente:

$$\underbrace{(1 - \phi B)}_{\text{polinomio característico}} y_t = \varepsilon_t$$

Siguiendo los pasos de 6.7.3:

$$\underbrace{\lambda - \phi_1 = 0}_{\text{ecuación característica}}$$

$$\lambda = \phi_1 \tag{6.7.6}$$

Como la raíz debe caer dentro del círculo unidad, implica que $|\lambda| < 1$. Luego, para que AR(1) sea estacionario $|\phi_1| < 1$.

Ejemplo 2

Considérese ahora un proceso AR(2). La ecuación característica es la siguiente:

$$\underbrace{(1 - \phi B - \phi_2 B^2)}_{\text{polinomio característico}} y_t = \varepsilon_t$$

Siguiendo los pasos de 6.7.3:

$$\underbrace{\lambda^2 - \phi_1 \lambda - \phi_2 = 0}_{\text{ecuación característica}}$$

Como las raíces son complejas, el Teorema Fundamental de Álgebra demuestra que esta ecuación tiene dos raíces: λ_1, λ_2.

$$\lambda_1, \lambda_2 = \frac{\phi_1 \pm \sqrt{\phi_1^2 + 4\phi_2}}{2}$$

Pueden suceder dos casos:

1) Las raíces son reales. Luego, como $|\lambda| < 1$, la ecuación está acotada.

$$-1 < \frac{\phi_1 \pm \sqrt{\phi_1^2 + 4\phi_2}}{2} < 1$$
$$-2 < \phi_1 \pm \sqrt{\phi_1^2 + 4\phi_2} < 2$$

El λ_i más grande está acotado por 2. Lo que implica:

$$\phi_1 + \sqrt{\phi_1^2 + 4\phi_2} < 2$$
$$\phi_1^2 + 4\phi_2 < (2 - \phi_1)^2$$
$$\cancel{\phi_1^2} + 4\phi_2 < 4 - 4\phi_1 + \cancel{\phi_1^2}$$
$$\phi_2 < 1 - \phi_1$$

De manera análoga, el λ_i más pequeño está acotado por -2. Luego:

$$-2 < \phi_1 - \sqrt{\phi_1^2 + 4\phi_2}$$
$$(-2 - \phi_1)^2 < \left(-\sqrt{\phi_1^2 + 4\phi_2}\right)^2$$
$$4 + \cancel{\phi_1^2} + 4\phi_1 < \cancel{\phi_1^2} + 4\phi_2$$
$$1 + \phi_1 < +\phi_2$$

2) El segundo caso que puede suceder es que ambas raíces λ_1, λ_2 sean complejas. Esto implica que $\phi_1^2 < -4\phi_2$. Entonces $|\lambda_i|$ al $\in \mathbb{C}$ ya no es un valor absoluto, sino un módulo[A9]. Luego $|\lambda_1| = |\lambda_2|$. Como λ_1, λ_2 son raíces conjugadas (Herrera, n.d.) se pueden escribir de la siguiente manera:

$$\lambda_1, \lambda_2 = \underbrace{\tfrac{\phi_1}{2}}_{\text{p. real}} \underbrace{\pm i \tfrac{\sqrt{-(\phi_1^2 + 4\phi_2)}}{2}}_{\text{p. e imaginaria}}$$

Teniendo en cuenta la restricción $|\lambda| < 1$:

$$|\lambda_1| = |\lambda_2| = \left(\frac{\phi_1}{2}\right)^2 + \left(\frac{\sqrt{-(\phi_1^2 + 4\phi_2)}}{2}\right)^2$$

$$|\lambda_1| = |\lambda_2| = \frac{\cancel{\phi_1^2} - \cancel{\phi_1^2} - 4\phi_2}{4} = -\phi_2$$

Luego, como $|\lambda| = -\phi_2 < 1$. Es lo mismo que $\phi_2 > -1$. También de las primeras dos restricciones:

$$\phi_2 < 1 - \phi_1 \quad \Rightarrow \quad \phi_2 + \phi_1 < 1$$
$$1 + \phi_1 < \phi_2 \quad \Rightarrow \quad \phi_2 - \phi_1 < 1$$

Se deduce que $\phi_2 < 1$ y junto con la restricción de arriba se obtiene que $|\phi_2| < 1$.
Por tanto, las restricciones que se obtienen para el modelo AR(2) son:

$$\phi_1 + \phi_2 < 1$$
$$\phi_1 - \phi_2 < 1$$
$$|\phi_2| < 1$$

No puede haber más casos porque las raíces complejas son conjugadas. Es decir, no puede haber un número impar de raíces complejas porque si $z \in \mathbb{C}$ es raíz, entonces $\overline{z}$ también lo es.

6.8 Teorema Fundamental de Álgebra

Teorema Fundamental de Álgebra

Todo polinomio $p(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z^1 + a_0 \in \mathbb{C}[z]$, con $n \in \mathbb{N} \setminus \{0\}$ y $a_i \neq 0$ para $i \in \{0, 1, \ldots, n\}$, tiene n raíces en $\mathbb{C}$.

Corolario para el Teorema Fundamental de Álgebra

Todo polinomio $p(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z^1 + a_0 \in \mathbb{C}[z]$, con $n \in \mathbb{N} \setminus \{0\}$ y $a_i \neq 0$ para $i \in \{0, 1, \ldots, n\}$ se puede expresar de la forma:

$$p(z) = a_n (z - z_1)(z - z_2) \cdots (z - z_n)$$

En donde $z_1, z_2, \cdots, z_n$ son raíces de $p(z)$.

Para demostrar este teorema, se va a demostrar antes el teorema de Liouville

Teorema de Liouville (análisis complejo)

Sea $f : \mathbb{C} \to \mathbb{C}$ una función entera[31] y acotada. Es decir $\exists\, M > 0$ tal que:

$$|f(z)| < M \quad \forall z \in \mathbb{C} \tag{6.8.1}$$

entonces, resulta que f es constante.

[31] Una función entera en complejos es lo mismo que decir que es derivable en todo $\mathbb{C}$, i.e. holomorfa en todo $\mathbb{C}$

Demostración: teorema de Liouville

La fórmula integral de Cauchy dice que $\forall z_0 \in$ en una curva γ que a su vez $\gamma \subseteq D$ definido por:

$$D = \{z : |z - z_0| \leq r\}$$

Entonces:

$$f'(z_0) = \frac{1}{2\pi i} \oint_{|z-z_0|=r} \frac{f(z)}{(z - z_0)^2}\, dz = \frac{1}{2\pi i} \oint_{\gamma} \frac{f(z)}{r^2}\, dz \text{ siendo } \gamma \text{ la curva.}$$

De modo que si en vez de $f(z_0)$, se valora M que por 6.8.1 es $> |f(z_0|\ \ \forall z_0 \in \mathbb{C}$

$$|f'(z_0)| \leq \frac{1}{2\pi i} \oint_\gamma \frac{M}{r^2}\, dz = \frac{1}{\cancel{2\pi i}} \frac{M}{r^2} \cancel{2\pi i r} = \frac{M}{r}$$

Como $f(z_0)$ es entera, se puede elegir un radio r tan grande como se quiera y por ende se concluye que $f'(z_0) = 0 \quad \forall z_0 \in \mathbb{C}$. Luego se acaba de demostrar que entonces $f(z_0)$ es constante en todo $\mathbb{C}$.

■

Ahora sí, se procede a demostrar el Teorema Fundamental de Álgebra:

Demostración: Teorema Fundamental de Álgebra

Supóngase que $p(z)$ no tuviera ninguna raíz compleja. Entonces $p(z) \neq 0 \quad \forall z \in \mathbb{C}$. Esto implicaría que la función $f(z) = \dfrac{1}{p(z)}$ sería analítica en $\mathbb{C}$. Con lo cual:

$$\begin{aligned} |f(z)| = \frac{1}{|p(z)|} &= \frac{1}{|a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0|} \\ &= \frac{1}{|z^n (a_n + \frac{a_{n-1}}{n} + \cdots + \frac{a_1}{z^{n-1}} + \frac{a_0}{z^n})|} \\ &= \frac{1}{|z|^n |a_n + \frac{a_{n-1}}{n} + \cdots + \frac{a_1}{z^{n-1}} + \frac{a_0}{z^n}|} \end{aligned}$$

Entonces $lim_{z\to\infty}|f(z)| = 0$, lo que implica que $f(z)$ está acotada en $\mathbb{C}$ y por el teorema de Liouville, $f(z)$ entonces sería constante. Esto implicaría que $p(z)$ también es constante. Pero $p(z)$ no puede ser constante porque $\deg(p(z)) \geq 1$.

Luego, se ha contradicho que $p(z)$ es imposible que no tenga raíces, lo que implica que tiene al menos una raíz z_1 —lo contrario de que no tenga raíces **no es** que tenga n raíces, sino que tiene al menos una raíz—.

Luego, como z_1 es raíz del polinomio $p(z)$, este se puede reescribir como:

$$p(z) = a_n(z - z_1)q(z)$$

Volviendo a aplicar el teorema de Liouville, se sabe que el polinomio $q(z)$ tendrá al menos una raíz z_2.

Luego:

$$p(z) = a_n(z - z_1)q(z) = a_n(z - z_1)(z - z_2)r(z)$$

Se puede ver, que si se aplica el teorema de Liouville en cadena, al final se obtendría que:

$$p(z) = a_n(z - z_1)(z - z_2)\cdots(z - z_n)$$

Luego, se acaba de demostrar que un polinomio $p(z)$ de grado n con $z \in \mathbb{C}$ tiene exactamente n raíces.

■

6.9 Módulo de un número complejo

Un número complejo se puede representar de varias maneras: de forma polar, como un par ordenado, de forma binómica, etc. Quizá la más intuitiva para explicar en módulo de un número complejo sea la forma binómica. Si $z \in \mathbb{C}$, su forma binómica es $z = a + bi$ donde a es la parte *real* y b la *imaginaria*. Esto se corresponde a un punto en el plano complejo.

El módulo de un número complejo se denota como $|z|$ y no es más que la distancia desde el $(0, 0)$ hasta z.

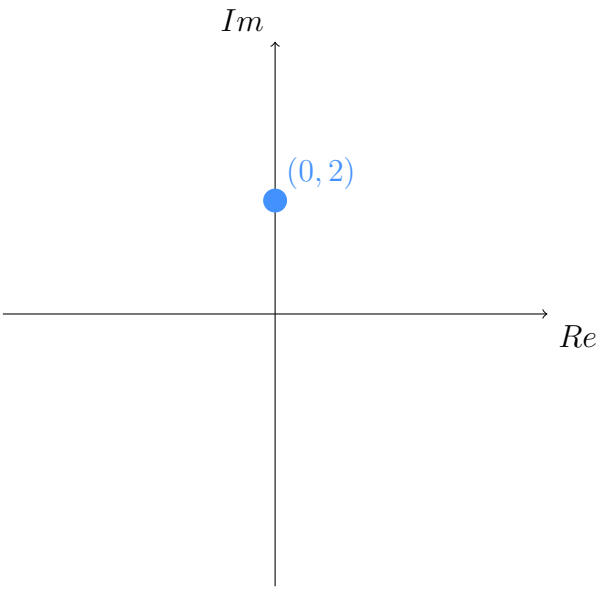

Figura 6.3: Representación $z = 2i$ en el plano complejo.

En la figura 6.3, el punto marcado corresponde a $z = 2i$. Su módulo es 2.

En el caso en el que el punto no esté sobre un eje, habrá que aplicar el teorema de Pitágoras.

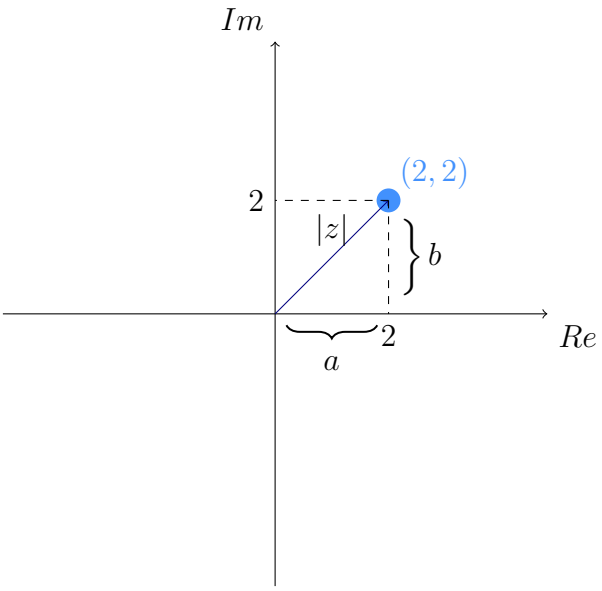

Figura 6.4: Representación visual del módulo de un número complejo en plano.

Luego si $z = a + bi$, su módulo. en la figura 6.4, por teorema de Pitágoras es:

$$|z| = \sqrt{a^2 + b^2}$$

6.10 Demostraciones familia ARIMA

6.10.1 Demostraciones procesos AR

Demostración: Patrón decreciente en procesos AR

Sea el modelo AR(1) un proceso estacionario. Luego por 6.7 se sabe que $|\phi| < 1$. Sin pérdida de la generalidad, considérese $c = 0$.

$$y_t = \phi y_{t-1} + \varepsilon_t$$

$$y_t - \phi y_{t-1} = \varepsilon_t$$

Se multiplica y_{t-k} a ambos lados y se extrae la esperanza.

$$E[y_t \cdot y_{t-k}] - \phi E[y_{t-1} \cdot y_{t-k}] = E[y_{t-k} \cdot \varepsilon_t]$$

Por la ec. 6.10.1 se sabe que si $c = 0$, entonces $\mu = 0$. Luego, por la ec. 2.1.4:

$$\gamma_k - \phi\gamma_{k-1} = E[y_{t-k} \cdot \varepsilon_t]$$

Como ε_t es ruido $E[y_{t-k} \cdot \varepsilon_t] = 0$ Luego:

$$\gamma_k = \phi\gamma_{k-1}$$

Siguiendo esta relación, se obtiene que:

$$\gamma_{k-1} = \phi\gamma_{k-2}$$

$$\gamma_{k-2} = \phi\gamma_{k-3}$$

Luego:

$$\gamma_k = \phi\gamma_{k-1} = \phi^2\gamma_{k-2} = \phi^3\gamma_{k-3} = \cdots = \phi^k\gamma_0$$

Por la ec. 2.1.5, la autocorrelación del proceso AR(1) es:

$$\rho_k = \phi^k$$

Como por ec. 6.7.6 $|\phi| < 1$, pueden darse dos casos:

- Si $\phi > 0$, entonces, al ser positivo da lugar al patrón que decae exponencialmente.
- Si $\phi < 0$, entonces al ser negativo, da lugar al patrón sinusoidal.

En cualquiera de los casos, se da que a mayor k —i.e. a mayor distancia entre tiempo observado y pasado— la autocorrelación es menor. Lo cual también tiene lógica (Mills, 2019).

Demostración: c no es la media de un proceso AR(p)

Supóngase un AR(p):

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

Para hallar la media de y_t se extrae la esperanza a ambos lados de la ecuación:

$$
\begin{aligned}
E[y_t] &= E[c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} + \cdots + \phi_p y_{t-p} + \varepsilon_t] \\
\mu &= E[c] + E[\phi_1 y_{t-1}] + E[\phi_2 y_{t-2}] + E[\phi_3 y_{t-3}] + \cdots + E[\phi_p y_{t-p}] + E[\varepsilon_t] \\
\mu &= c + \phi_1 E[y_{t-1}] + \phi_2 E[y_{t-2}] + \phi_3 E[y_{t-3}] + \cdots + \phi_p E[y_{t-p}] + 0 \\
\mu &= c + \phi_1 \cdot \mu + \phi_2 \cdot \mu + \phi_3 \cdot \mu + \cdots + \phi_p \cdot \mu \\
c &= \mu - \mu(\phi_1 + \phi_2 + \phi_3 + \cdots + \phi_p) \\
c &= \mu \left(1 - \sum_{j=1}^{p} \phi_p\right)
\end{aligned}
$$

Luego:

$$
\mu = \frac{c}{\left(1 - \sum_{j=1}^{p} \phi_p\right)} \tag{6.10.1}
$$

■

Demostración: proceso AR(p) no es necesariamente estacionario

Considérese AR(1):

$$
y_t = \phi_1 y_{t-1} + \varepsilon_t
$$

Esto se puede reescribir como:

$$
\begin{aligned}
y_t &= \phi_1 y_{t-1} + \varepsilon_t \\
&= \phi_1 (\underbrace{\phi_1 y_{t-2} + \varepsilon_{t-1}}_{y_{t-1}}) + \varepsilon_t
\end{aligned}
$$

Siguiendo este proceso de manera recursiva...

$$
\begin{aligned}
y_t &= \phi_1^2 y_{t-2} + \phi_1 \varepsilon_{t-1} + \varepsilon_t \\
&= \phi_1^2 (\underbrace{\phi_1 y_{t-3} + \varepsilon_{t-2}}_{y_{t-2}}) + \phi_1 \varepsilon_{t-1} + \varepsilon_t \\
&= \phi_1^3 y_{t-3} + \phi_1^2 \varepsilon_{t-2} + \phi_1 \varepsilon_{t-1} + \varepsilon_t \\
&\vdots
\end{aligned} \tag{6.10.2}
$$

Se puede ver que si $|\phi_1| > 1$, al estar elevado cada vez a una potencia mayor, se iría a infinito y la condición de estacionariedad 2.1.10 no se cumpliría.
Se acaba de demostrar, pues, que para que AR(1) sea un proceso estacionario $|\phi_1| < 1$.

■

6.10.2 Demostraciones procesos MA

MA es un proceso estacionario **siempre**. Para ello hay que demostrar que ni la media ni la varianza dependen del tiempo[32].

Demostración: μ es la media de un proceso MA(q)

Esta primera demostración tiene dos propósitos:

1) Demostrar que la media de una proceso MA(q) es constante.
2) Demostrar que la media de un proceso MA(q) es μ y por eso la constante se denota con la letra μ —y no con otra letra genérica como sí pasaba con AR(p).

Al igual que en **??** se extrae la media a ambos lados de 2.1.13:

$$\begin{aligned} E[y_t] &= E[\mu + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots + \theta_q\varepsilon_{t-q}] \\ \mu &= E[\mu] + E[\varepsilon_t] + E[\theta_1\varepsilon_{t-1}] + E[\theta_2\varepsilon_{t-2}] + \cdots + E[\theta_q\varepsilon_{t-q}] \\ \mu &= \mu + E[\varepsilon_t] + \theta_1 E[\varepsilon_{t-1}] + \theta_2 E[\varepsilon_{t-2}] + \cdots + \theta_q E[\varepsilon_{t-q}] \end{aligned}$$

Por la propiedad de ruido explicada en [2.1.4]: $E[\varepsilon_{t-k}] \quad \forall k \in [0, T] \; = 0$. Luego:

$$\begin{aligned} \mu &= \mu + 0 \\ &= \mu \end{aligned} \tag{6.10.3}$$

Se acaba de demostrar que la media de un proceso MA(q) es constante.

■

[32] Que la autocorrelación sea 0 entre todas las observaciones es trivial, pues se trata de una combinación lineal de ruidos, la autocorrelación de MA(q) es 0.

Demostración: Varianza de MA(q) es constante

En esta segunda demostración se va a comprobar que la varianza de un proceso MA(q) también es constante —y por ende no depende del tiempo—.

$$\begin{aligned}
\text{Var}[y_t] &= \text{Var}[\mu + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots + \theta_q\varepsilon_{t-q}] \\
&= \text{Var}[\mu] + \text{Var}[\varepsilon_t] + \text{Var}[\theta_1\varepsilon_{t-1}] + \text{Var}[\theta_2\varepsilon_{t-2}] + \cdots + \text{Var}[\theta_q\varepsilon_{t-q}] \\
&= 0 + \text{Var}[\varepsilon_t] + \theta_1^2\text{Var}[\varepsilon_{t-1}] + \theta_2^2\text{Var}[\varepsilon_{t-2}] + \cdots + \theta_q^2\text{Var}[\varepsilon_{t-q}] \\
&= \sigma^2 + \theta_1^2\sigma^2 + \theta_2^2\sigma^2 + \cdots + \theta_q^2\sigma^2 \\
&= \sigma^2\left(1 + \sum_{i=1}^{q}\theta_i^2\right) = C
\end{aligned}$$

■

6.11 Criterios de selección en modelos de aprendizaje automático

En *machine learning*, una de las características a tener en cuenta a la hora de determinar si un modelo es mejor que otro es la diferencia de simplicidad entre ambos. De esta manera (por poner un ejemplo simple), aunque pueda parecer ilógico, un modelo AR(1) puede ser mejor que un modelo AR(2) —pese a que este es una extensión del primero—.

Un efecto secundario de que los criterios AIC y BIC penalicen modelos con altos parámetros p y q es que se evitan modelos demasiado complejos y específicos, dando lugar a modelos más simples y generales que tienen menor probabilidad de dar problemas de *overfitting* a la hora de hacer un pronóstico (Allamy, 2014).

6.12 ¿Cómo funcionan AIC y BIC para elegir los mejores modelos?

El procedimiento de estos criterios —y el que usan los software por detrás— es el siguiente:

Se determina un p_{max} y un q_{max}. Se hacen todas las combinaciones posibles de AIC($\bar{p}, \bar{q}$) con $\bar{p} \in [0, 1, 2, \cdots, p_{max}]$ y con $\bar{q} \in [0, 1, 2, \cdots, q_{max}]$.

Los valores p, q óptimos se determinan de tal manera que:

$$AIC(p, q) = \min AIC(\bar{p}, \bar{q})$$

El proceso es análogo para BIC.

Una posible dificultad reside en que no hay método ni estrategia para determinar p_{max} y q_{max}; aunque se asume que son lo suficientemente grandes como para contener el 'verdadero' modelo de la serie temporal (Mills, 2019).

6.13 Desvanecimiento del gradiente y gradiente explosivo

El mecanismo por el que las redes neuronales aprenden, es a través de la optimización de sus pesos. Estos pesos se actualizan después de un determinado número de entrenamientos (se especifica con el batch size). La optimización de estos pesos se hace mediante el algoritmo de *Backpropagation* (Rumelhart et al., 1986).

Este algoritmo revolucionó el campo de la inteligencia artificial. Se basa en la regla de la cadena para la optimización de pesos. De manera muy simplificada, esta regla de la cadena son multiplicaciones. En muchas ocasiones, la actualización de un peso implica muchas multiplicaciones (seguramente miles).

Ahora, el problema del desvanecimiento del gradiente surge cuando muchos de estos valores que se están multiplicando $\in (-1, 1)$. Cuando se multiplican muchas veces (miles de veces) números $\in (-1, 1)$, el resultado que se obtiene es un número extremadamente pequeño. Luego este peso que se está tratando de actualizar, no lo hace de una manera eficiente —pues se actualiza en una cantidad mínima—.

Por el contrario, el problema del gradiente descendente se da cuando la mayoría de estas multiplicaciones $\notin (-1, 1)$. El resultado de multiplicar muchas veces números $\notin (-1, 1)$ da lugar a un número muy grande.

Luego el peso que se está tratando de actualizar lo hace de manera muy brusca y no llega a converger a un valor óptimo.

Las RNN, por su naturaleza, pueden caer en estos problemas.

6.14 Ejemplo básico de regresión

En una regresión, el estadístico (persona) elige distintas variables que puedan tener un impacto en la variable estudiada. Cada una de estas variables tiene un peso asociado, que, junto con un término independiente, forman una recta. El objetivo en este problema es, por tanto, averiguar cuál es el

valor de cada uno de estos pesos, de tal manera que la recta generada se ajuste lo mejor posible a la recta de los datos (esto se consigue con el método de mínimos cuadrados). Luego la regresión es un *curve-fitting problem* (problema de ajuste de curva). Un ejemplo donde se puede apreciar de una manera clara un problema de regresión es el siguiente:

$$\text{Precio vivienda} = \beta_0 + \beta_1 \cdot \text{ubicación} + \beta_2 \cdot \text{metros}^2 + \beta_3 \cdot \text{n.º habitaciones} + \cdots + \beta_n \cdot \text{n.º habitaciones} \tag{6.14.1}$$

En la ec. 6.14.1 se está tratando de estimar el precio de una vivienda a partir de unas variables que el estadístico elige. La dificultad está en encontrar los parámetros $\beta_0, \beta_1, \cdots, \beta_n$ que mejor se ajusten a los datos que ya se tienen.

Referencias

Adejumo, A. O., Odetunmibi, O., Adejumo, O., Oguntunde, P., Ikoba, N., & Obalowu, J. (2018). Modelling of enugu monthly rainfall using box and jenkins methodology.

Allamy, H. (2014). Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study).

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Benvenuto, D., Giovanetti, M., Vassallo, L., Angeletti, S., & Ciccozzi, M. (2020). Application of the arima model on the covid-2019 epidemic dataset. *Data in brief*, *29*, 105340.

Box, G. E., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *26*(2), 211–243.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: Forecasting and control*. John Wiley & Sons.

Bozdogan, H. (1987). Model selection and akaike's information criterion (aic): The general theory and its analytical extensions. *Psychometrika*, *52*(3), 345–370.

Breusch, T. S., & Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. *Econometrica: Journal of the econometric society*, 1287–1294.

Brockwell, P. J., & Davis, R. A. (1991). *Time series: Theory and methods*. Springer science & business media.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, *74*(366a), 427–431.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, *28*(10), 2222–2232.

Harvey, A. C., & Peters, S. (1990). Estimation procedures for structural time series models. *Journal of forecasting*, *9*(2), 89–108.

Herrera, A. (n.d.). *Introducción a los números complejos*. https://www.uv.mx/personal/aherrera/files/2014/08/01a.-INTRODUCCION-A-LOS-NUMEROS-COMPLEJOS.pdf

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Holt, C. C. (1957). Forecasting trends and seasonals by exponentially weighted averages. *ONR Memorandum*, *52*, 1957.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice* (2nd). OTexts.

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd). OTexts.

Jarque, C. M., & Bera, A. K. (1980). Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics letters*, *6*(3), 255–259.

Jordan, M. (1986). *Serial order: A parallel distributed processing approach. technical report, june 1985-march 1986* (tech. rep.). California Univ., San Diego, La Jolla (USA). Inst. for Cognitive Science.

Kwiatkowski, D., Phillips, P. C., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, *54*(1), 159–178. https://doi.org/10.1016/0304-4076(92)90104-Y

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Ljung, G. M., & Box, G. E. (1978). On a measure of lack of fit in time series models. *Biometrika*, *65*(2), 297–303.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*, 115–133.

Mills, T. C. (2019). *Applied time series analysis: A practical guide to modeling and forecasting*. Academic press.

Mondal, P., Shit, L., & Goswami, S. (2014). Study of effectiveness of time series modeling (arima) in forecasting stock prices. *International Journal of Computer Science, Engineering and Applications*, *4*(2), 13.

Myung, I. J. (2003). Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, *47*(1), 90–100.

Olah, C. (2015). *Understanding lstm networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Satrio, C. B. A., Darmawan, W., Nadia, B. U., & Hanafiah, N. (2021). Time series analysis and forecasting of coronavirus disease in indonesia using arima model and prophet. *Procedia Computer Science*, *179*, 524–532.

Schaffer, A. L., Dobbins, T. A., & Pearson, S.-A. (2021). Interrupted time series analysis using autoregressive integrated moving average (arima) models: A guide for evaluating large-scale health interventions. *BMC medical research methodology*, *21*(1), 1–12.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, 461–464.

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2018). A comparison of arima and lstm in forecasting time series. *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, 1394–1401.

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019). The performance of lstm and bilstm in forecasting time series. *2019 IEEE International Conference on Big Data (Big Data)*, 3285–3292. https://doi.org/10.1109/BigData47090.2019.9005997

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, *72*(1), 37–45.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Vrieze, S. I. (2012). Model selection and psychological theory: A discussion of the differences between the akaike information criterion (aic) and the bayesian information criterion (bic). *Psychological methods*, *17*(2), 228.

Werbos, P., & John, P. (1974). Beyond regression : New tools for prediction and analysis in the behavioral sciences /.

Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management science*, *6*(3), 324–342.

Yang, S., Yu, X., & Zhou, Y. (2020). Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. *2020 International workshop on electronic communication and artificial intelligence (IWECAI)*, 98–101.