Minería de datos y modelización predictiva: Series Temporales

Hugo Gómez Sabucedo

hugogomezsabucedo@gmail.com

Máster Big Data, Data Science & Inteligencia Artificial

Curso 2024-2025

Universidad Complutense de Madrid

Índice

1. Introducción y análisis inicial

1.1. Introducción

En este ejercicio se realizará el análisis y predicciones sobre una serie temporal, con datos obtenidos a partir del Instituto Nacional de Estadística. Estos datos, disponibles en el archivo `viajerosMD.xlsx`, que contienen los datos sobre los viajeros totales en ferrocarril de media distancia en España, medidos en **miles de viajeros**, desde el año 2010 hasta el 2024, medidos mensualmente. Aunque la serie original contiene datos que se remontan al año 2000, para no tener una serie tan grande, se ha decidido elegir únicamente estos datos. Por lo tanto, la serie constará de 15 años, o lo que es lo mismo, 180 observaciones, de las cuales, en lo que se corresponde al punto 3 del enunciado de la práctica, reservaremos 12 para realizar el test de los modelos, siendo las 168 restantes los datos con los que entrenaremos el modelo.

El código completo de Python de esta práctica se adjunta en el archivo `codigoMineria3.py`, también al final de este documento, por lo que aquí sólo incluiremos *snippets* de código que sean especialmente relevantes o que no se hayan visto en clase. Antes de comenzar con el análisis, debemos naturalmente importar el archivo en Python, y realizar una serie de pequeñas modificaciones. Estas consisten en transformar la fecha, para que sea un formato de fecha legible por Python. En el archivo original el formato es *2024M12*, por lo que empleamos el siguiente código para realizar la conversión (líneas 1 y 2). Además, transformamos los datos de la serie, los valores numéricos, para que se consideren como un número y no como string (línea 3), y se establece la fecha como índice (línea 4). Por último, debemos invertir la serie (línea 5), ya que el archivo ordena las observaciones desde la más reciente a la más antigua, y para su representación nos interesa tener en primer lugar la observación más antigua. Con esto, tendremos los datos listos y correctamente cargados.

```
1   viajeros['Fecha'] = viajeros['Fecha'].str.strip()
2   viajeros['Fecha'] = pd.to_datetime(viajeros['Fecha'], format='%YM%m')
3   viajeros['Viajeros'] = viajeros['Viajeros'].apply(pd.to_numeric,
    errors='coerce')
4   viajeros.set_index('Fecha', inplace=True)
5   viajeros = viajeros.iloc[::-1]
```

1.2. Representación gráfica

En la figura 1, nos encontramos con una representación tal cual de la serie, dónde se observa una clara estacionalidad en los datos, con valores que se repiten en periodos de un año. Además, se observa un brusco descenso en el año 2020, cuadrando con la pandemia, donde el número de viajes se redujo bruscamente, hasta llegar casi a 0, debido a las medidas que limitaban la

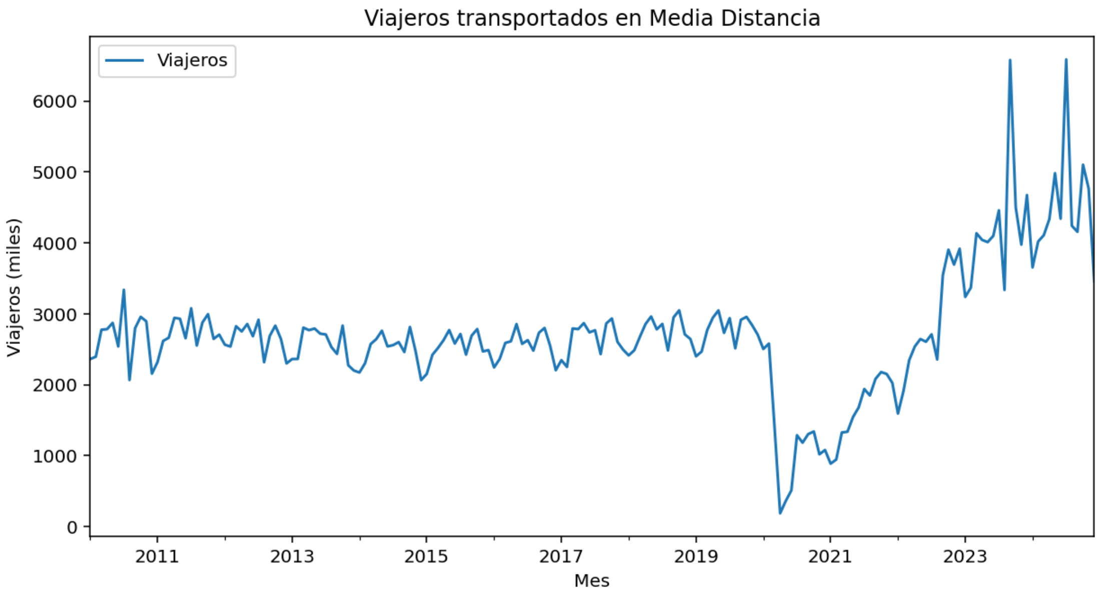

Figura 1

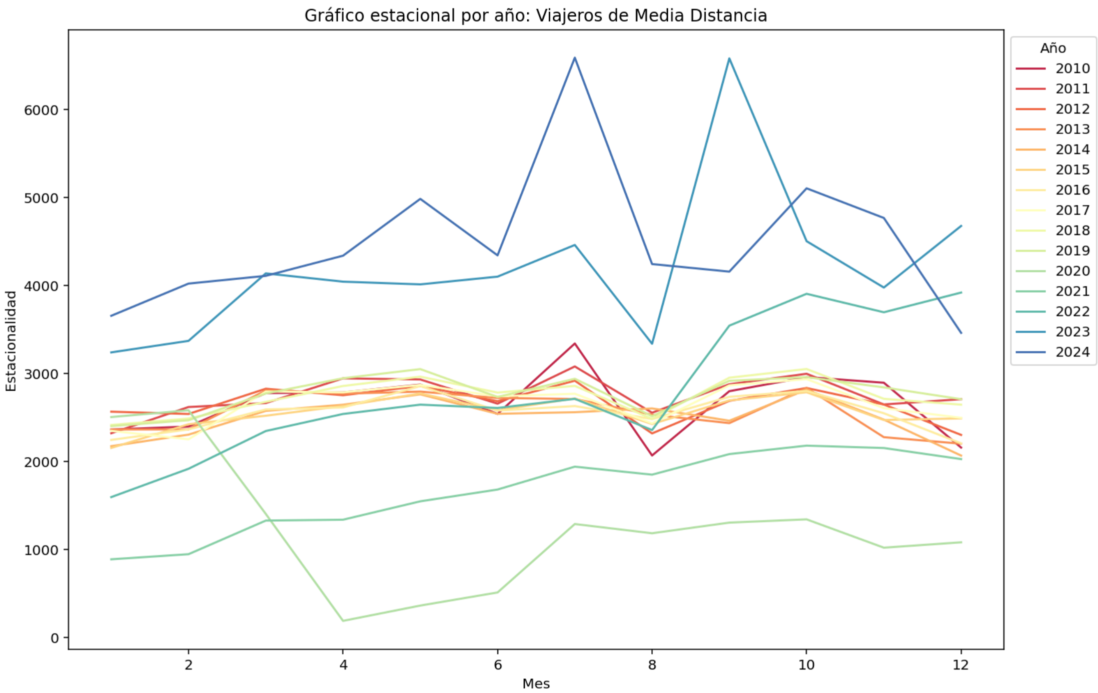

Figura 2

movilidad. En el período post-pandemia, se observa una tendencia claramente al alza, que no dura sólo ese año, sino también hasta el presente, a la vez que se observa también una clara estacionalidad, pues si bien los viajeros aumentan cada año que pasa, se sigue observando un pico en los datos a mediados de año, y un valle hacia finales de año. Esto se ve también en la figura 2, donde hemos descompuesto la serie asignando a cada año un color diferente, y hemos representado el número de viajeros en cada mes. Aquí se observa más claramente el escaso número de viajeros del año 2020, el ligero aumento que se produjo en 2021, y las cantidades tan elevadas de viajeros que vimos en los dos últimos años, así como un claro pico de viajeros que se produce en el mes de julio, seguido de una disminución brusca de los mismos en el mes de agosto, lo que se corresponde claramente con el patrón de vacaciones que estamos acostumbrados a ver en España.

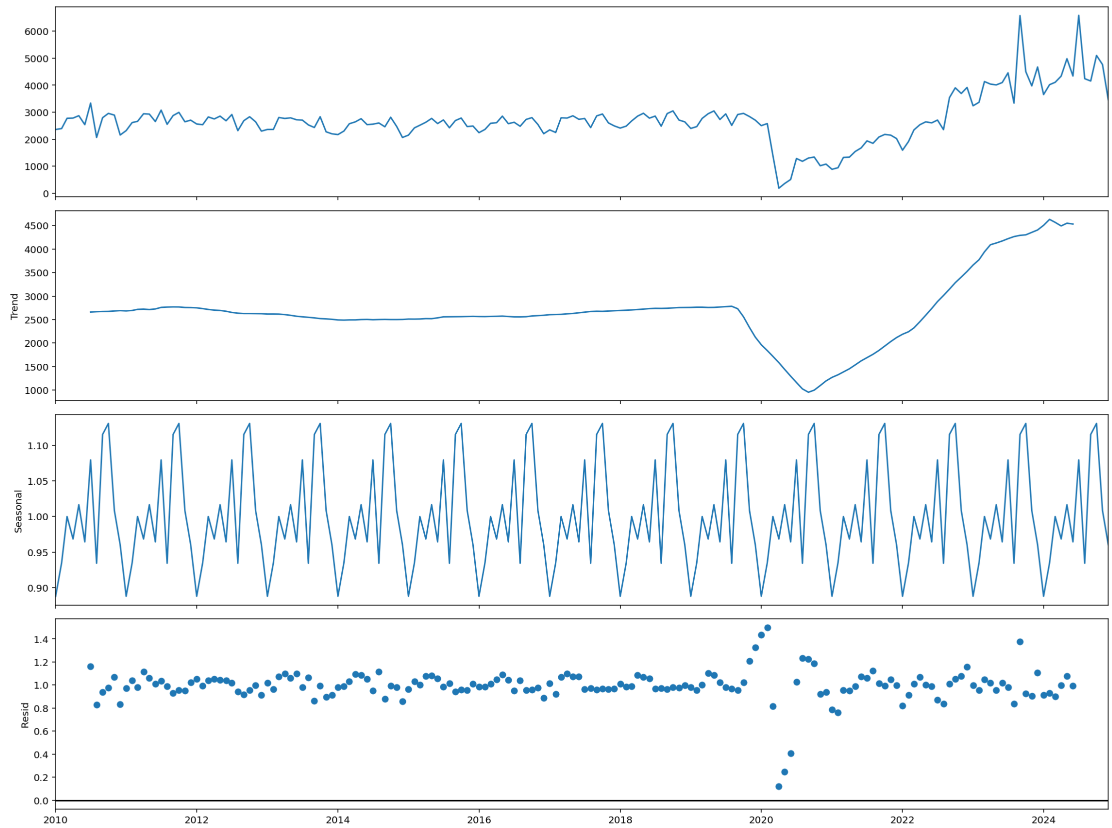

Figura 3

Ya que la serie tiene un claro comportamiento estacional, podemos realizar su descomposición estacional, mediante el método `seasonal_decompose`. Para hacer este método, podemos utilizar un modelo aditivo, en el que los valores desestacionalizados se obtienen sumando las correcciones estacionales a la serie; o un modelo multiplicativo, donde la serie corregida se obtiene multiplicando la serie original por el componente estacional. Este último será el que usaremos, ya que en nuestros datos no tenemos ningún valor que sea 0. Si observamos la gráfica que se produce al realizar esta descomposición, en la figura 3, vemos que el componente estacional (tercera gráfica) está claramente marcado, lo que quiere decir que la serie tiene un comportamiento estacional. Además, este toma valores alrededor de 1, ya que nos indica cuánto aumenta o disminuye el número de viajeros en un mes concreto con respecto a la media. En la cuarta gráfica, vemos el componente irregular, mientras que en la tercera observamos la tendencia, donde se ve claramente que la serie tenía una tendencia constante hasta que llegó la pandemia, donde se observa un brusco descenso, seguido de una marcada tendencia al alza.

2. Modelos de suavizado exponencial

En esta sección aplicaremos distintos modelos de suavizado, con el objetivo de determinar finalmente cuál de ellos es mejor. Estos modelos estiman los valores de los componentes de la serie en función del tiempo, usando los valores anteriores y suavizándolos empleando coeficientes que minimicen el error producido. Crearemos cuatro modelos diferentes partiendo de los datos de train, con el objetivo de realizar diferentes predicciones y compararlas con los datos de test, para evaluar que modelo es el que mejor se ajusta.

2.1. Modelo de suavizado simple

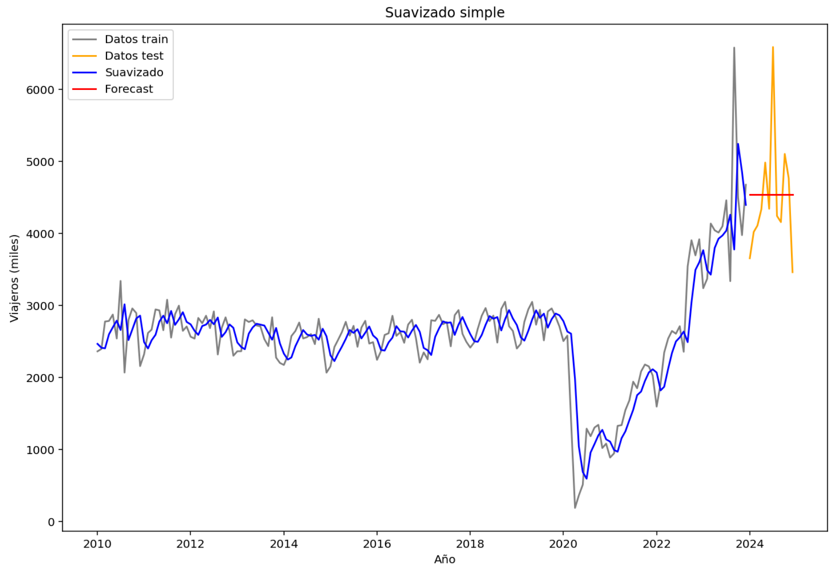

Figura 4: Modelo de suavizado simple

El modelo de suavizado simple suele usarse cuando la serie no presenta una tendencia relevante, por lo que es casi seguro que podremos desestimarlo de entre los modelos candidatos a ser ganadores. Aún así, en la gráfica 4 se muestra el resultado de la predicción para el último año de este modelo, donde se ve que claramente no es correcta, ya que muestra un valor fijo para las predicciones de los 12 meses ya que, como dijimos, este modelo sólo se usa para series que no tienen una tendencia muy marcada. Si vemos el parámetro alfa, este es de 0.523406, mientras que el valor inicial es de 2465.023001.

2.2. Modelo de suavizado de Holt

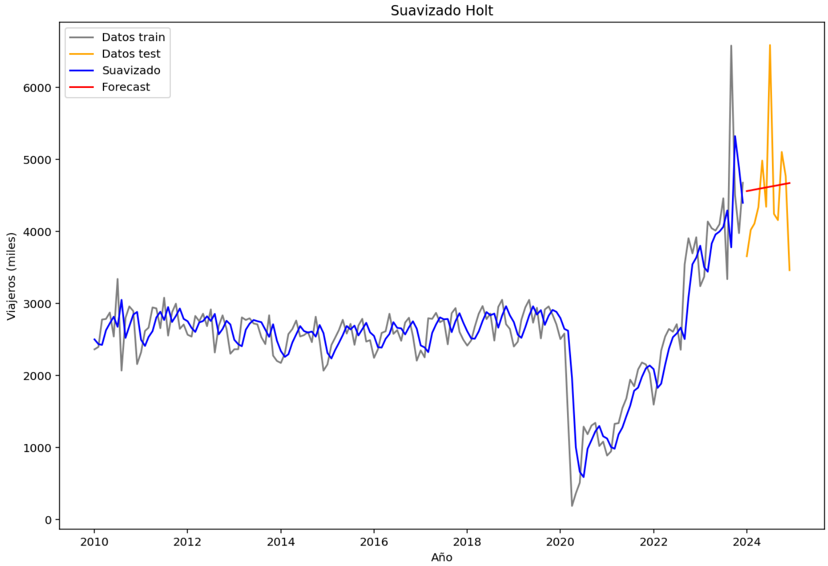

Figura 5: Modelo de suavizado de Holt

El método de Holt es similar al anterior, pero en este caso presupone que la tendencia es lineal, es decir, que tiene una pendiente variable. De esta forma, en este método obtendremos dos parámetros, α y β, que se corresponden con el factor de suavización del nivel y la tendencia, respectivamente, así como el valor inicial de dicho nivel y tendencia. Si aplicamos el modelo y realizamos las predicciones, obtenemos la gráfica de 5, donde vemos que, aunque las predicciones siguen siendo totalmente erróneas, se aprecia la tendencia creciente que veíamos en los datos. En este caso, los parámetros que obtuvimos del modelo son $\alpha = 0.547596$, $\beta = 0.000100$, $L_0 = 2489.217917$ y $B_0 = 10.117447$.

2.3. Modelo de tendencia amortiguada

El modelo o método de tendencia amortiguada es una variación del modelo ed Holt que acabamos de ver en 2.2, que introduce un factor de amortiguación para que las predicciones no sean una simple recta, sino que tomen una forma ajustada más a una curva. Esto vendrá dado por el parámetro ϕ del modelo, que devuelve además los dos factores anteriores, así como los valores iniciales. Podemos ver las predicciones que ha realizado en la figura 6. De esta forma, tenemos un $\alpha = 0.521807$ y un $\beta = 0.001862$, muy similares a los parámetros del anterior modelo, con un $L_0 = 2488.744544$ y $B_0 = 8.973355$, también similares a los anteriores. Sin embargo, si analizamos el ϕ, vemos que es $\phi = 0.990017$, prácticamente 1, lo cual indica que no tiene apenas efecto. Es decir, que el modelo de tendencia amortiguada nos producirá los mismos resultados casi que el modelo de Holt. Esto podemos verlo fácilmente en la figura 7, donde comparamos las predicciones del modelo de suavizado simple, el de Holt y este de la tendencia amortiguada. En él, vemos en rojo las predicciones del modelo simple, que eran

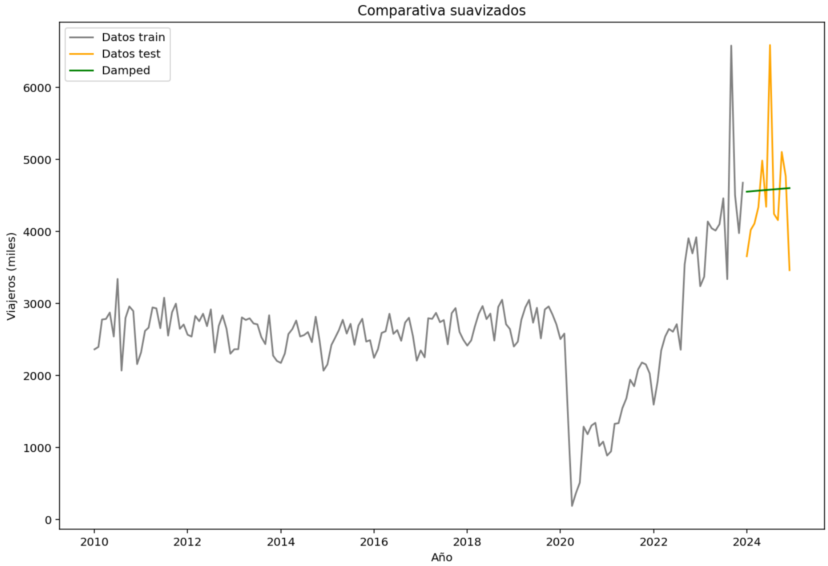

Figura 6: Modelo de tendencia amortiguada

una simple línea sin pendiente, y tenemos en azul las predicciones del método de Holt, y en verde las del método de tendencias amortiguadas, donde vemos que ambas toman unos valores bastantes similares, estando la pendiente de este último modelo un poco menos pronunciada que la del anterior.

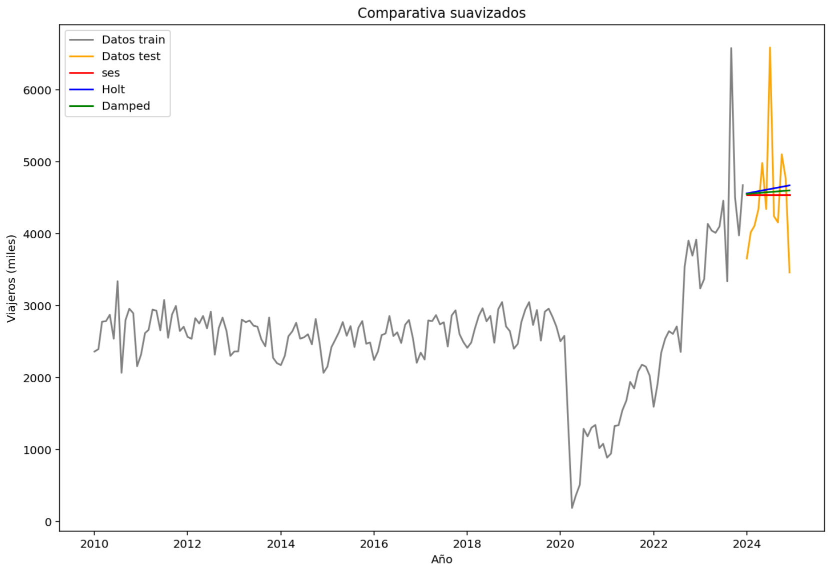

Figura 7: Comparación: suavizado simple, Holt y *damped*

2.4. Modelo de suavizado de Holt-Winters

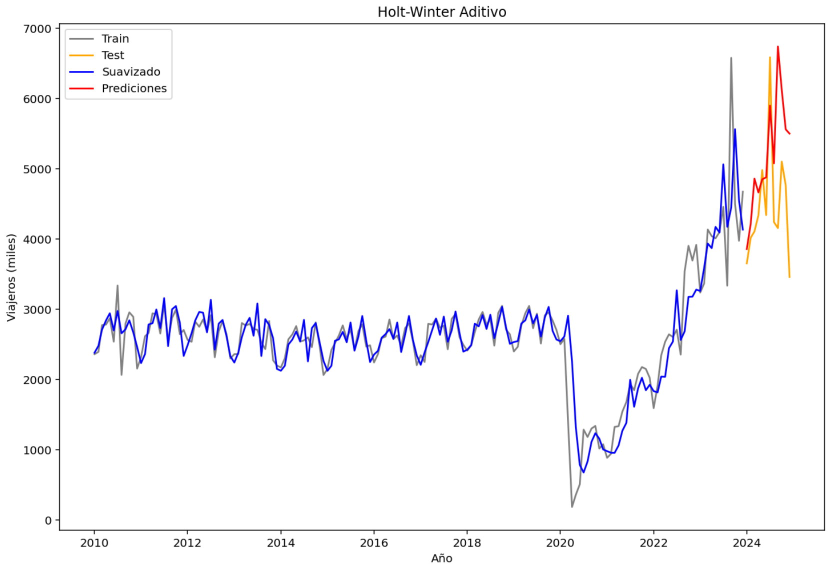

Figura 8: Modelo de tendencia amortiguada

Por último, aplicaremos el modelo de suavizado de Holt-Winters, un modelo que a priori debería ser más adecuado, ya que incorpora la estacionalidad mediante un coeficiente que multiplica a la tendencia. Los valores iniciales de la tendencia son estimados a partir de la media de los valores del primer ciclo; la pendiente se estima a partir de las diferencias en dos ciclos completos; y los índices estacionales, con los valores del primer periodo. En este caos, tenemos como parámetros los que se ven en la tabla 1. Tenemos los parámetros de antes ($\alpha, \beta, L_0 y B_0$), a los que se suma γ, que representa cuánto afecta la información nueva al patrón estacional; y los distintos parámetros iniciales de los factores estacionales S_0 a S_{11}, que, ya que empleamos un modelo multiplicativo, nos indica cuánto se desvía ese mes en concreto de la media, ya sea con más viajeros (mayor que 1) o menos viajeros.

Está claro que este es el modelo de suavizado más adecuado, ya que, como se ve en la figura 8, las predicciones se ajustan casi perfectamente con los datos que teníamos de test, sólo se observa que, para los últimos meses de 2024, no percibe a la perfección la disminución de viajeros del mes de septiembre, y obtiene unos valores más elevados.

En la tabla 2, podemos ver, por una parte, la predicción que ha realizado este modelo y por otro, los resultados que teníamos reservados de los datos de test. Vemos que las diferencias, salvo en estos últimos meses que comentamos, donde son de 2500 miles de viajeros que ha estimado de más en el mes de septiembre, o 2000 miles de viajeros de más en diciembre, en el resto de meses la estimación se ajusta bastante a la realidad.

Como es evidente, no va a ser una predicción exacta, ni mucho menos, porque tampoco es lo que busca este método, pero sí que se corresponde de una forma bastante fiel a los datos que

Parámetro	**Abreviatura**	**Valor**
smoothing_level	α	0.464643
smoothing_trend	β	0.0244549
smoothing_seasonal	γ	0.178452
initial_level	L_0	2650.66
initial_trend	B_0	1.00441
initial_seasons.0	S_0	0.894502
initial_seasons.1	S_1	0.931432
initial_seasons.2	S_2	1.03039
initial_seasons.3	S_3	1.06354
initial_seasons.4	S_4	1.10814
initial_seasons.5	S_5	1.02364
initial_seasons.6	S_6	1.15761
initial_seasons.7	S_7	0.974111
initial_seasons.8	S_8	1.1079
initial_seasons.9	S_9	1.1417
initial_seasons.10	S_{10}	1.05023
initial_seasons.11	S_{11}	0.930051

Cuadro 1: Parámetros del modelo de Holt-Winters

se han observado, teniendo en cuenta también que los datos presentan una tendencia creciente que es siempre más difícil de predecir.

Mes	**Predicción**	**Datos test**
Enero 2024	3857.368419	3654
Febrero 2024	4220.702156	4020
Marzo 2024	4860.868766	4108
Abril 2024	4663.238474	4337
Mayo 2024	4850.431828	4983
Junio 2024	4878.574812	4341
Julio 2024	5899.436039	6588
Agosto 2024	5077.671698	4242
Septiembre 2024	6741.756478	4156
Octubre 2024	6124.685444	5103
Noviembre 2024	5562.474281	4766
Diciembre 2024	5501.244563	3460

Cuadro 2: Predicciones del modelo de Holt-Winters

3. Modelos ARIMA

En esta sección crearemos, por una parte, un modelo ARIMA de forma manual, ajustando los parámetros en base a lo observado en los correlogramas; y por otra parte, un modelo ARIMA de forma automática, empleando las funciones proporcionadas en Python.

3.1. Autocorrelogramas y modelo ARIMA manual

Para decidir que modelo ARIMA ajustaremos, primero vamos a analizar los autocorrelogramas, analizando la función de autocorrelación simple (ACF) y la función de autocorrelación parcial (PACF). La primera, la ACF, mide la correlación entre la serie y sus valores pasados, y se usa para identificar si la serie sigue un modelo MA o de medias móviles. De esta forma, una disminución lenta de la misma nos indicaría que la serie no es estacionaria. La segunda función, la PACF, se utiliza para identificar si la serie sigue un modelo AR o autoregresivo, ya que representa la relación de la serie entre el valor k y el actual, pero omitiendo el efecto de los valores intermedios entre k-1 y el actual. Así, los valores iniciales serán elevados, y el momento en que se corte será el orden del modelo.

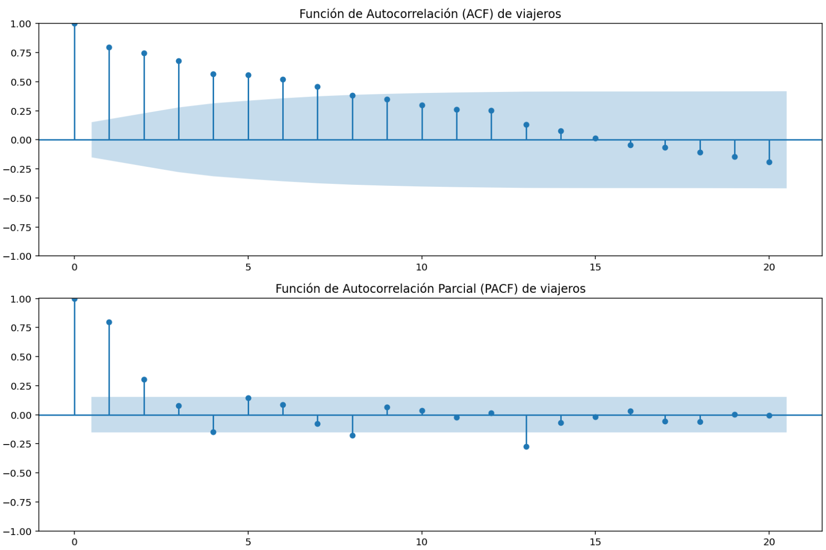

Figura 9: Autocorrelogramas

En nuestro caso, si analizamos el ACF, vemos que decrece lentamente, lo que nos indica que la serie no es estacionaria y que requeriría diferenciación (es decir, un $d > 0$). Respecto al PACF, vemos un descenso brusco en el primer retardo, mientras que a partir del segundo se puede considerar que es 0, por lo que nos sugiere un modelo `AR(1)`. Sin embargo, como hemos visto que el ACF desciende lentamente, esto nos sugería que es necesario diferenciar la serie. Esto lo haremos con el siguiente código:

```
diferencias = train.diff().dropna()
```

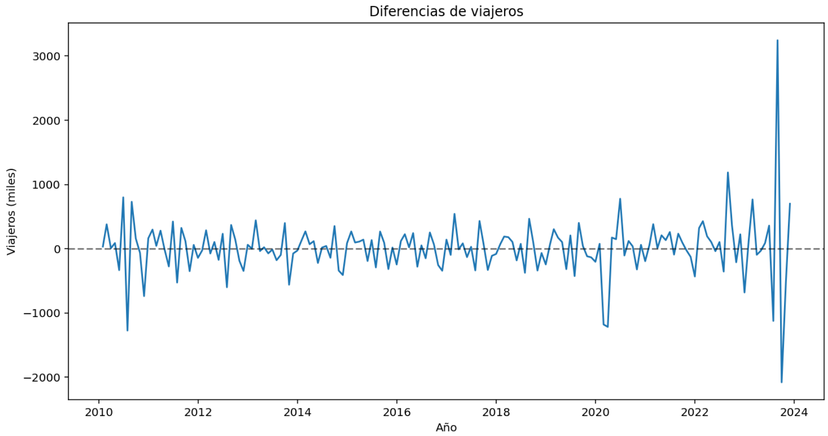

Figura 10: Diferenciación de la serie

En la figura 10 vemos la diferenciación que acabamos de hacer, donde ya se ve que la serie tiene una media constante. Vamos a analizar de nuevo el ACF y el PACF, para determinar los parámetros del modelo, en la figura 11. En este caso, vemos que la serie tiene ya un comportamiento estacionario, ya que cae rápidamente tras el primer retardo, mientras que los valores de la PACF son casi todos 0, por lo que podemos concluir que la diferenciacón ha sido efectiva. Esto nos dice que es un proceso integrado de orden $d = 1$.

Por lo tanto, para elegir el modelo, aplicaremos la metodología Box-Jenkins, que se resume en cuatro etapas:

1. Identificar el modelo.
2. Estimar los parámetros.
3. Probar el modelo.
4. Realizar predicciones.

Para estimar el modelo, se emplean los resultados de los ACF y los PACF, para crear el modelo $ARIMA(p,q,d)(P,Q,D)s$. Tenemos que tener en cuenta, también, que hemos tenido que realizar una diferenciación para hacer que la serie fuese estacionaria, lo cual nos implica

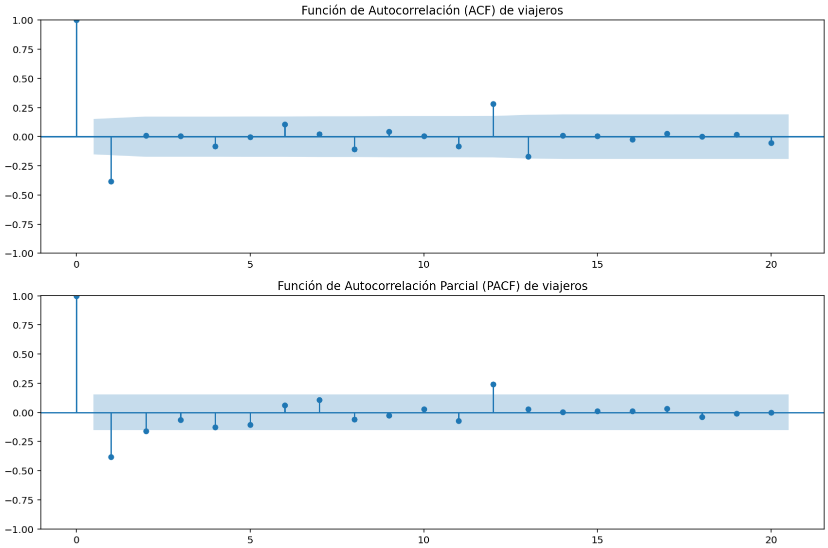

Figura 11: Autocorrelogramas de la serie diferenciada

un $d = 1$, y se observó un comportamiento estacional en el mismo, por lo que $D = 1$. De los gráficos de la ACF y la PACF deducimos que p y q deberían ser 1. Por lo tanto, se selecciona un modelo ARIMA$(1, 1, 1)(0, 1, 1)_{12}$. Ajustamos el modelo en Python mediante el siguiente código.

```
modelo_arima = sm.tsa.ARIMA(train, order=(1, 1, 1), seasonal_order
=(0, 1, 1, 12))
resultados = modelo_arima.fit()
print(resultados.summary())
```

Sin embargo, al analizar los parámetros, se ve que, si bien el modelo captura bien la dinámica de la serie, los residuos no son normales, y tenemos un p-valor para el AR(1) de 0.575, lo que nos indica que este coeficiente no es significativo. Respecto a los residuos, vemos que el test de Jarque-Bera, que nos permite comprobar si los datos siguen la asimetría y curtosis de una normal, tiene un p-valor de 0, menor que 0.05, por lo que se rechaza la hipótesis de la normalidad. Además, tenemos un AIC de 2293.991, un BIC de 2306.164 y un HQIC de 2298.935, lo que nos indica un modelo bueno, pero ligeramente complejo. Vamos a ajustar, por tanto, otro modelo, eliminando el parámetro AR(1), haciendo un ARIMA$(0, 1, 1)(0, 1, 1)_{12}$.

En este modelo, tenemos los parámetros que se ven en la figura 13, que se estiman mediante máxima verosimilitud. Como todos los p-valores son menores que 0.05, podemos concluir que, ahora sí, todos son estadísticamente significativos, y se incluyen en el modelo. Por otra parte, en el gráfico 12, donde analizamos los residuos, vemos que, no muestran un patrón, se ajustan más o menos a la normal (aunque con una ligera asimetría), son incorrelados (pues están dentro de las bandas de confianza), y se observan algunas desviaciones en el gráfico Q-Q, que se deben con toda seguridad al año de la pandemia. Respecto a las métricas de ajuste, el AIC es de

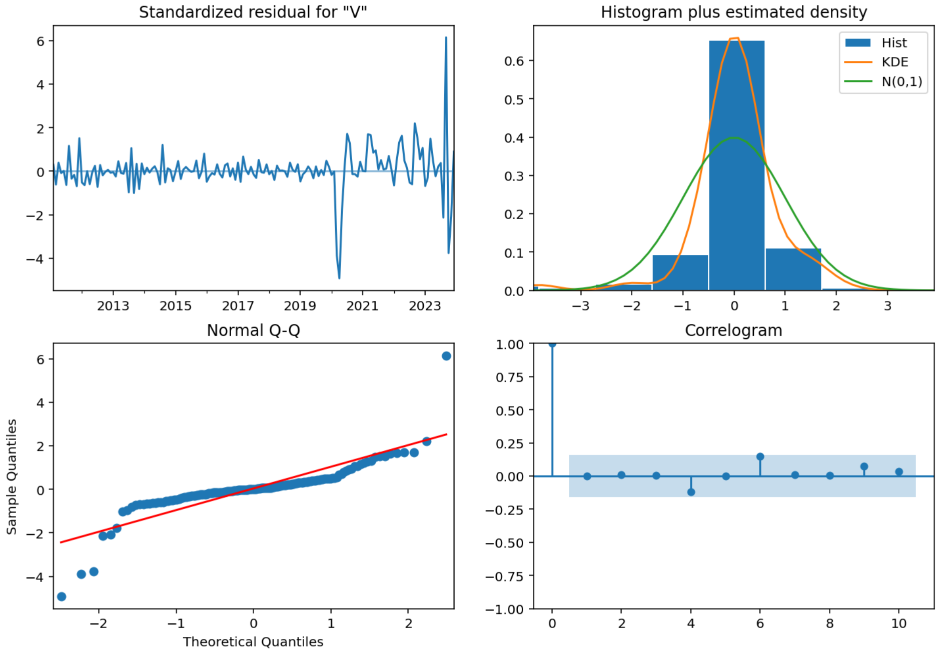

Figura 12: Estudio de los residuos del modelo ARIMA

2292.115, el BIC es de 2301.246 y el HQIC es de 2295.824, todos menores que en el anterior modelo, lo que nos indica que es mejor en términos de parsimonia y de ajuste. Es decir, es un modelo más sencillo, y que ajusta mejor. Si analizamos también el contraste de Ljung-Box, este toma un valor de 0.02 con una probabilidad de 0.90. Ya que la probabilidad es alta, mayor que 0.05, esto nos indica que los residuos son independientes (es decir, ruido blanco) y, por tanto, que el modelo es adecuado.

```
                                    SARIMAX Results
==========================================================================================
Dep. Variable:                          Viajeros   No. Observations:                   168
Model:             ARIMA(0, 1, 1)x(0, 1, 1, 12)   Log Likelihood                -1143.058
Date:                          Thu, 20 Mar 2025   AIC                            2292.115
Time:                                  19:56:23   BIC                            2301.246
Sample:                              01-01-2010   HQIC                           2295.824
                                   - 12-01-2023
Covariance Type:                            opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.3774      0.031    -12.148      0.000      -0.438      -0.317
ma.S.L12      -0.6293      0.058    -10.795      0.000      -0.744      -0.515
sigma2      1.449e+05   6411.747     22.601      0.000    1.32e+05    1.57e+05
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):              1241.02
Prob(Q):                              0.90   Prob(JB):                         0.00
Heteroskedasticity (H):               9.69   Skew:                             0.04
Prob(H) (two-sided):                  0.00   Kurtosis:                        16.86
===================================================================================
```

Figura 13: Parámetros del modelo ARIMA

3.2. Modelo autoARIMA

Ahora ajustaremos el modelo automático, lo cual haremos mediante el siguiente comando:

```
modelo_auto = pm.auto_arima(train, m=12, d=0, D=1, start_p=0, max_p=3, start_q=0, max_q=3,seasonal=True, trace=True,error_action='ignore', suppress_warnings=True,stepwise=True)
```

Esto nos proporciona el siguiente output:

```
Performing stepwise search to minimize aic
ARIMA(0,0,0)(1,1,1)[12] intercept   : AIC=2532.302, Time=0.16 sec
ARIMA(0,0,0)(0,1,0)[12] intercept   : AIC=2529.758, Time=0.02 sec
ARIMA(1,0,0)(1,1,0)[12] intercept   : AIC=2326.242, Time=0.29 sec
ARIMA(0,0,1)(0,1,1)[12] intercept   : AIC=2430.942, Time=0.32 sec
ARIMA(0,0,0)(0,1,0)[12]             : AIC=2531.202, Time=0.02 sec
ARIMA(1,0,0)(0,1,0)[12] intercept   : AIC=2330.791, Time=0.12 sec
ARIMA(1,0,0)(2,1,0)[12] intercept   : AIC=2327.524, Time=0.78 sec
ARIMA(1,0,0)(1,1,1)[12] intercept   : AIC=2323.603, Time=0.62 sec
ARIMA(1,0,0)(0,1,1)[12] intercept   : AIC=2324.114, Time=0.41 sec
ARIMA(1,0,0)(2,1,1)[12] intercept   : AIC=2325.603, Time=1.15 sec
ARIMA(1,0,0)(1,1,2)[12] intercept   : AIC=2325.603, Time=1.19 sec
ARIMA(1,0,0)(0,1,2)[12] intercept   : AIC=2324.340, Time=0.99 sec
ARIMA(1,0,0)(2,1,2)[12] intercept   : AIC=2326.882, Time=1.45 sec
ARIMA(2,0,0)(1,1,1)[12] intercept   : AIC=2316.942, Time=0.81 sec
ARIMA(2,0,0)(0,1,1)[12] intercept   : AIC=2317.061, Time=0.56 sec
ARIMA(2,0,0)(1,1,0)[12] intercept   : AIC=2323.214, Time=0.35 sec
ARIMA(2,0,0)(2,1,1)[12] intercept   : AIC=2318.875, Time=1.30 sec
ARIMA(2,0,0)(1,1,2)[12] intercept   : AIC=2318.933, Time=1.34 sec
ARIMA(2,0,0)(0,1,0)[12] intercept   : AIC=2330.780, Time=0.12 sec
ARIMA(2,0,0)(0,1,2)[12] intercept   : AIC=2317.058, Time=1.02 sec
ARIMA(2,0,0)(2,1,0)[12] intercept   : AIC=2322.958, Time=0.75 sec
ARIMA(2,0,0)(2,1,2)[12] intercept   : AIC=inf, Time=1.71 sec
ARIMA(3,0,0)(1,1,1)[12] intercept   : AIC=2318.228, Time=1.01 sec
ARIMA(2,0,1)(1,1,1)[12] intercept   : AIC=2318.370, Time=1.01 sec
ARIMA(1,0,1)(1,1,1)[12] intercept   : AIC=2316.413, Time=0.72 sec
ARIMA(1,0,1)(0,1,1)[12] intercept   : AIC=2316.201, Time=0.58 sec
ARIMA(1,0,1)(0,1,0)[12] intercept   : AIC=2331.178, Time=0.05 sec
ARIMA(1,0,1)(0,1,2)[12] intercept   : AIC=2316.479, Time=1.48 sec
ARIMA(1,0,1)(1,1,0)[12] intercept   : AIC=2323.333, Time=0.36 sec
ARIMA(1,0,1)(1,1,2)[12] intercept   : AIC=2318.390, Time=1.37 sec
ARIMA(2,0,1)(0,1,1)[12] intercept   : AIC=2318.191, Time=0.87 sec
ARIMA(1,0,2)(0,1,1)[12] intercept   : AIC=2318.191, Time=0.84 sec
ARIMA(0,0,0)(0,1,1)[12] intercept   : AIC=2530.616, Time=0.09 sec
ARIMA(0,0,2)(0,1,1)[12] intercept   : AIC=2388.834, Time=0.64 sec
ARIMA(2,0,2)(0,1,1)[12] intercept   : AIC=inf, Time=1.00 sec
ARIMA(1,0,1)(0,1,1)[12]             : AIC=2314.686, Time=0.38 sec
ARIMA(1,0,1)(0,1,0)[12]             : AIC=2329.101, Time=0.04 sec
ARIMA(1,0,1)(1,1,1)[12]             : AIC=2315.315, Time=0.48 sec
ARIMA(1,0,1)(0,1,2)[12]             : AIC=2315.310, Time=0.77 sec
ARIMA(1,0,1)(1,1,0)[12]             : AIC=2321.435, Time=0.18 sec
```

```
 ARIMA(1,0,1)(1,1,2)[12]             : AIC=2317.277, Time=1.15 sec
 ARIMA(0,0,1)(0,1,1)[12]             : AIC=2431.070, Time=0.21 sec
 ARIMA(1,0,0)(0,1,1)[12]             : AIC=2322.608, Time=0.26 sec
 ARIMA(2,0,1)(0,1,1)[12]             : AIC=2316.682, Time=0.47 sec
 ARIMA(1,0,2)(0,1,1)[12]             : AIC=2316.683, Time=0.46 sec
 ARIMA(0,0,0)(0,1,1)[12]             : AIC=2531.484, Time=0.14 sec
 ARIMA(0,0,2)(0,1,1)[12]             : AIC=2388.587, Time=0.42 sec
 ARIMA(2,0,0)(0,1,1)[12]             : AIC=2315.577, Time=0.36 sec
 ARIMA(2,0,2)(0,1,1)[12]             : AIC=2317.649, Time=0.89 sec

Best model:  ARIMA(1,0,1)(0,1,1)[12]
Total fit time: 31.727 seconds
```

De esta forma, el método autoARIMA ha determinado que el mejor modelo es un $\texttt{ARIMA}(1,0,1)(0,1,1)_{12}$, el cual es ligeramente parecido al que habíamos probado nosotros en el método manual, tanto el primero como el segundo. Sus parámetros están en la figura 14. De nuevo, todos los parámetros son significativos, puesto que su p-valor es 0. Además, en la figura vemos el análisis de los residuos, que son prácticamente iguales a los del modelo manual. Obtenemos también una probabilidad alta en la prueba de Ljunb-Box, por lo que los residuos son ruido blanco, y podemos concluir que el modelo es también adecuado.

```
                               SARIMAX Results
==========================================================================================
Dep. Variable:                                  y   No. Observations:                  168
Model:             SARIMAX(1, 0, 1)x(0, 1, 1, 12)   Log Likelihood               -1153.343
Date:                            Thu, 20 Mar 2025   AIC                           2314.686
Time:                                    20:28:19   BIC                           2326.886
Sample:                                01-01-2010   HQIC                          2319.641
                                     - 12-01-2023
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9434      0.027     35.513      0.000       0.891       0.995
ma.L1         -0.3244      0.047     -6.913      0.000      -0.416      -0.232
ma.S.L12      -0.6038      0.100     -6.042      0.000      -0.800      -0.408
sigma2      1.456e+05   6422.107     22.664      0.000    1.33e+05    1.58e+05
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):              1356.65
Prob(Q):                              0.98   Prob(JB):                         0.00
Heteroskedasticity (H):               6.28   Skew:                             0.36
Prob(H) (two-sided):                  0.00   Kurtosis:                        17.43
===================================================================================
```

Figura 14: Parámetros del modelo autoARIMA

3.3. Comparación y conclusiones

Para determinar el mejor modelo, tenemos que analizar los resultados del ARIMAman y del autoARIMA (ya que está claro que los modelos de suavizado no van a ser los mejores, ya que vimos que no ajustaban bien los datos [exceptuando el de Holt-Winters, que en cualquier caso

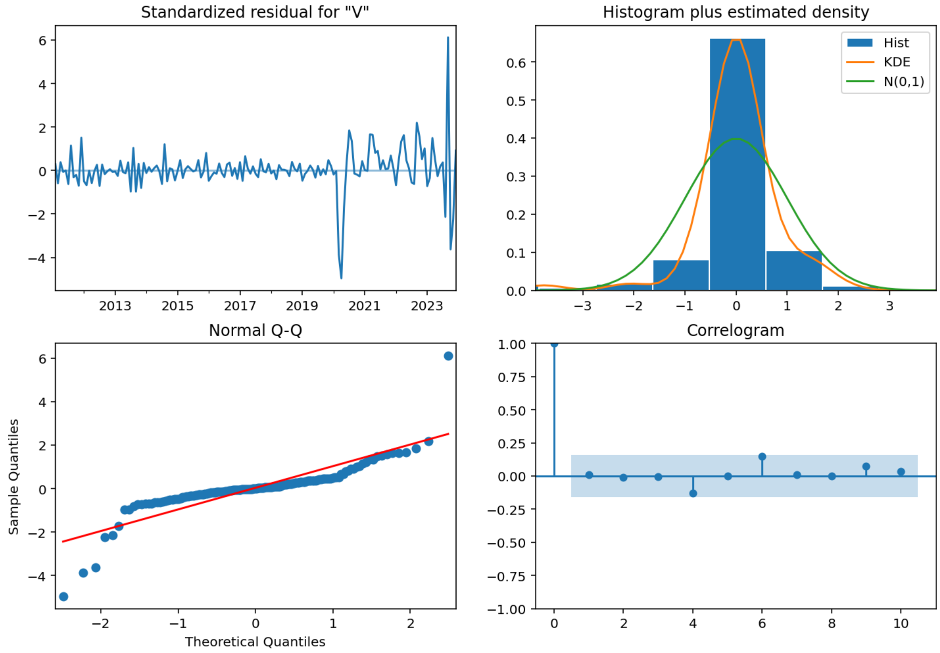

Figura 15: Estudio de los residuos del modelo autoARIMA

tiene peor ajuste que los modelos que venimos de analizar con el modelado de ARIMA]). Para determinar el mejor modelo, realizaremos una comparación basada en los parámetros de ambos, que recordamos que eran un ARIMA$(0,1,1)(0,1,1)_{12}$ en el manual y un ARIMA$(1,0,1)(0,1,1)_{12}$ en el automático. Si vemos los valores de AIC, tenemos 2292.115 para el manual y 2314.686 para el automático; para el BIC, 2301.246 para el manual y 2326.886 para el automático; y para el HQIC, 2295.824 para el manual y 2319.641 para el automático. Aunque la diferencia no es elevada, los tres valores son más bajos en el caso del manual, lo que nos sugiere que este podría ser el mejor modelo. Además, los parámetros en ambos modelos son estadísticamente significativos, ya que su p-valor es menor que 0.05. Sin embargo, en el modelo automático, el coeficiente del término autorregresivo AR(1) es de 0.94, muy próximo a 1, lo que podría sugerir que el modelo está captando una tendencia en vez de una dependencia temporal. La prueba de Ljung-Box es satisfactoria en ambos casos, al igual que la de Jarque-Bera; esta última, sin embargo, es mejor en el caso del modelo manual, ya que indica que se ajusta mejor a una distribución normal. Por lo tanto, y basándonos en todo esto, determinamos que el modelo manual es el mejor de entre los dos.

La expresión algebraica del modelo es, por tanto, la siguiente:

$$(1-B)y_t = (1+0.3774B)(1+0.6293B^{12})e_t \tag{1}$$

Donde el término $(1-B)y_t$ es la diferencia de primer orden de la serie; $(1+0.3774B)$ es la parte de la media móvil no estacional, y $(1+0.6293B^{12})$ la parte de la media móvil estacional.

Calculamos las predicciones para este modelo manual, empleando para ello el método `get_forecast`, para el año 2024 (que es el que reservamos para los datos de test), y graficamos tanto la predicción del número de viajeros como los datos de test que teníamos, para compararlos. En la figura 16 vemos estas predicciones en la serie global, con todos los datos, mientras que en la figura 17 vemos también los intervalos de confianza, representados de forma gráfica, como se solicitaba en el enunciado del trabajo. Como podemos ver, las predicciones realizadas se ajustan bastante bien a los datos de test que teníamos. En los primeros meses del año capta los resultados y las tendencias bastante bien, detectando también el pico de viajeros del mes de julio. Si bien es cierto que, por ejemplo, el descenso de viajeros de septiembre o diciembre no los captura tan bien. Sin embargo, es importante fijarse en un hecho curioso: los datos de test tienen tres meses con datos que sobresalen de los intervalos de confianza, lo cual puede significar que, aunque el modelo se ajuste bien, se han producido unas variaciones atípicas en los números de viajeros en estos meses, que el modelo no sería capaz de detectar.

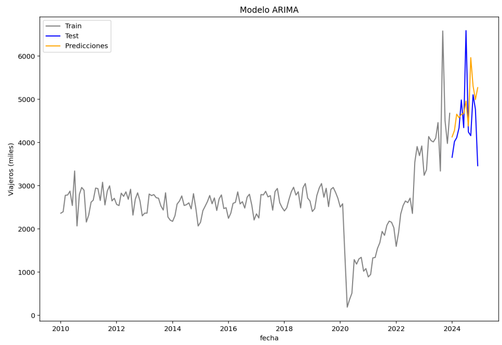

Figura 16: Predicciones del modelo manual

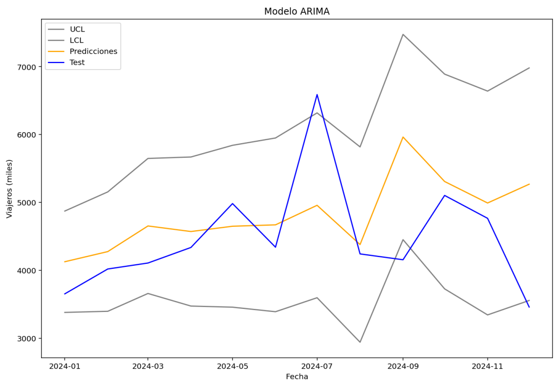

Figura 17: Predicciones del modelo manual con intervalos de confianza

Sin embargo, en resumen, podemos decir que el modelo es de una calidad bastante aceptable. Hemos representado, en la figura 18 una predicción a mayores, para los próximos 5 años (72 meses, ya que tenemos que contar los 12 meses del año 2024). Estas predicciones captan, además de la estacionalidad de los datos, la tendencia al alza que vinimos comentando que se observaba desde la pandemia. Con estas predicciones será interesante analizar, una vez que se tengan los datos de este periodo, compararlos, para ver si el modelo realizó unas predicciones adecuadas.

A modo de conclusión, tenemos un modelo $\texttt{ARIMA}(0,1,1)(0,1,1)_{12}$ que hemos creado de forma manual, que hemos determinado que era el ganador, por ser el que mejor parsimonia y ajuste tiene en cuanto a los datos. Tras realizar las predicciones, además, se observan unos resultados bastante bien ajustados. Los residuos del modelo pueden considerarse ruido blanco, lo que indica también que el modelo ha capturado correcatmente las dependencias temporales. Además, son estables, por lo que el modelo no estaría sesgado. Las métricas de AIC y BIC obtenidas apoyan esto, ya que sugieren que el modelo no está sobreajustado y que no es excesivamente complejo. Las predicciones que realiza son adecuadas a corto plazo, pero una

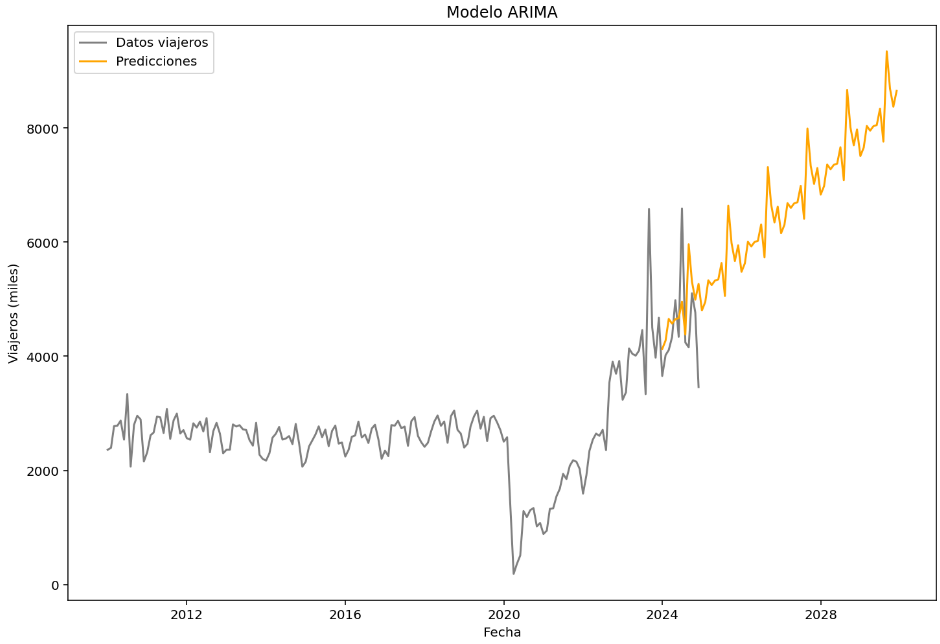

Figura 18: Predicciones a futuro

posible mejora para este ejercicio podría ser probar con otro tipo de modelos, o incorporar algunas variables que se sepa que afectan especialmente al número de viajeros, para poder realizar mejores predicciones a largo plazo.

A. Anexo: Código de la práctica

```
import warnings
import numpy as np
import pandas as pd
import seaborn as sns
import datetime as dt
import pmdarima as pm
import statsmodels.api as sm
import matplotlib.pyplot as plt

from tabulate import tabulate
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing,
 Holt

warnings.simplefilter(action='ignore', category=FutureWarning)

# Cargamos y formateamos la fecha
viajeros = pd.read_excel('viajerosMD.xlsx')
viajeros['Fecha'] = viajeros['Fecha'].str.strip()
viajeros['Fecha'] = pd.to_datetime(viajeros['Fecha'], format='%YM%m')
viajeros['Viajeros'] = viajeros['Viajeros'].apply(pd.to_numeric, errors='
coerce')
viajeros.set_index('Fecha', inplace=True)
viajeros.dropna(inplace=True)
viajeros = viajeros.iloc[::-1]  # Invertimos la serie, ya que los datos
van de 2024 para atras

# Hacemos un gráfico con la serie
viajeros.plot(y='Viajeros', figsize=(10,5))
plt.title("Viajeros transportados en Media Distancia")
plt.xlabel("Mes")
plt.ylabel("Viajeros (miles)")
plt.show()

# Representamos los valores por año
viajeros['Año'] = viajeros.index.year.astype(str)
plt.figure(figsize=(12, 8))
sns.lineplot(x=viajeros.index.month, y=viajeros.Viajeros, hue = viajeros[
'Año'], palette='Spectral')
plt.xlabel('Mes')
plt.ylabel('Estacionalidad')
plt.title('Gráfico estacional por año: Viajeros de Media Distancia')
plt.legend(title='Año', loc='upper left', bbox_to_anchor=(1, 1))
plt.show()
viajeros.drop('Año', axis=1, inplace=True) # Eliminamos la columna que
habíamos añadido
```

```python

# Hacemos la descomposición estacional multiplicativa
mult_decomp = seasonal_decompose(viajeros, model='multiplicative', period=12)
plt.rc("figure", figsize=(16,12))
fig = mult_decomp.plot()

# Representamos la tendencia y la serie ajustada estacionalmente
viajeros_ajustada = viajeros['Viajeros'] - mult_decomp.seasonal
plt.figure(figsize=(12, 8))
plt.plot(viajeros, label='Datos', color='gray')
plt.plot(mult_decomp.trend, label='Tendencia', color='blue') # Tendencia
plt.plot(viajeros_ajustada, label='Estacionalmente ajustada', color='red') # Serie ajustada
plt.xlabel('Fecha')
plt.ylabel('Viajeros (miles)')
plt.title('Viajeros en Media Distancia')
plt.legend(loc='best')
plt.show()

# Separamos los datos en train y test, guardando el último año como test
train = viajeros[:-12]
test = viajeros[-12:]

plt.figure(figsize=(12, 8))
plt.plot(train, label='Train', color='gray')
plt.plot(test, label='Test', color='yellow')
plt.legend()
plt.xlabel('Fecha')
plt.ylabel('Viajeros')
plt.show()

# Aplicamos el suavizado exponencial simple
model = SimpleExpSmoothing(train, initialization_method="estimated").fit()
print(model.params_formatted)
fcast = model.forecast(12)

plt.figure(figsize=(12, 8))
plt.plot(train, label='Datos train', color='gray')
plt.plot(test, label='Datos test', color='orange')
plt.plot(model.fittedvalues, label='Suavizado', color='blue')
plt.plot(fcast, label='Forecast', color='red')
plt.xlabel('Año')
plt.ylabel('Viajeros (miles)')
plt.title('Suavizado simple')
plt.legend()
plt.show()



```

```
# Aplicamos el método de Holt
model1 = Holt(train, initialization_method="estimated").fit()
fcast1 = model1.forecast(12)
print(model1.params_formatted)
print(fcast1)

plt.figure(figsize=(12, 8))
plt.plot(train, label='Datos train', color='gray')
plt.plot(test, label='Datos test', color='orange')
plt.plot(model1.fittedvalues, label='Suavizado', color='blue')
plt.plot(fcast1, label='Forecast', color='red')
plt.xlabel('Año')
plt.ylabel('Viajeros (miles)')
plt.title('Suavizado Holt')
plt.legend()
plt.show()


# Aplicamos el método de la tendencia amortiguada
model2 = Holt(train,damped_trend=True, initialization_method="estimated")
.fit()
fcast2 = model2.forecast(12)
print(model2.params_formatted)

plt.figure(figsize=(12, 8))
plt.plot(train, label='Datos train', color='gray')
plt.plot(test, label='Datos test', color='orange')
plt.plot(fcast, label='ses', color='red')
plt.plot(fcast1, label='Holt', color='blue')
plt.plot(fcast2, label="Damped",color='green')
plt.xlabel('Año')
plt.ylabel('Viajeros (miles)')
plt.title('Comparativa suavizados')
plt.legend()
plt.show()


# Aplicamos el método de Holt-Winters
model3 = ExponentialSmoothing(train, seasonal_periods=12, trend="mul",
                              seasonal="mul", initialization_method="
                              estimated").fit()
fcast3 = model3.forecast(12)
print(fcast3)

plt.figure(figsize=(12, 8))
plt.plot(train, label='Train', color='gray')
plt.plot(test, label='Test', color='orange')
plt.plot(model3.fittedvalues, label='Suavizado', color='blue')
plt.plot(fcast3,color='red', label="Prediciones")
plt.xlabel('Año')
plt.ylabel('Viajeros (miles)')
```

```
plt.title('Holt-Winter Aditivo')
plt.legend()

#Mostramos los parámetros
headers = ['Name', 'Param', 'Value', 'Optimized']
table_str = tabulate(model3.params_formatted, headers, tablefmt='
fancy_grid')
print(table_str)

# Mostramos la evolución de los componentes
fig, axes = plt.subplots(nrows=3, ncols=1, figsize=(12, 8))
axes[0].plot(model3.level)
axes[0].set_title('Level')
axes[1].plot(model3.trend)
axes[1].set_title('Trend')
axes[2].plot(model3.season)
axes[2].set_title('Season')
plt.tight_layout()
plt.show()


# Dibujamos el correlograma
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))
plot_acf(train, lags=20, ax=ax1)
ax1.set_title('Función de Autocorrelación (ACF) de viajeros')
plot_pacf(train, lags=20, ax=ax2)
ax2.set_title('Función de Autocorrelación Parcial (PACF) de viajeros')
plt.tight_layout()
plt.show()

# Diferenciamos la serie
diferencias = train.diff().dropna()
plt.figure(figsize=(12, 6))
plt.plot(diferencias)
plt.axhline(y=0, color='black', linestyle='--', alpha=0.5)
plt.title('Diferencias de viajeros')
plt.xlabel('Año')
plt.ylabel('Viajeros (miles)')
plt.show()

# Dibujamos el correlograma
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))
plot_acf(diferencias, lags=20, ax=ax1)
ax1.set_title('Función de Autocorrelación (ACF) de viajeros')
plot_pacf(diferencias, lags=20, ax=ax2)
ax2.set_title('Función de Autocorrelación Parcial (PACF) de viajeros')
plt.tight_layout()
plt.show()

# Creamos el modelo manual
```

```
modelo_arima = sm.tsa.ARIMA(train, order=(0, 1, 1), seasonal_order=(0, 1,
 1, 12))
resultados = modelo_arima.fit()
print(resultados.summary())

# Comprobamos que los residuos estén incorrelados
resultados.plot_diagnostics(figsize=(12, 8))
plt.show()

print(resultados.mse)

# Calculamos predicciones
prediciones = resultados.get_forecast(steps=12)
predi_test=prediciones.predicted_mean
print(predi_test)

# Graficamos las predicciones
plt.figure(figsize=(12, 8))
plt.plot(train, label='Train', color='gray')
plt.plot(test, label='Test', color='blue')
plt.plot(prediciones.predicted_mean, label='Predicciones', color='orange'
)
plt.xlabel('fecha')
plt.ylabel('Viajeros (miles)')
plt.title('Modelo ARIMA')
plt.legend()
plt.show()

# Graficamos añadiendo los intervalos de confianza
intervalos_confianza = prediciones.conf_int()
plt.figure(figsize=(12, 8))
plt.plot(intervalos_confianza['lower Viajeros'], label='UCL', color='gray
')
plt.plot(intervalos_confianza['upper Viajeros'], label='LCL', color='gray
')
plt.plot(predi_test, label='Predicciones', color='orange')
plt.plot(test, label='Test', color='blue')
plt.xlabel('Fecha')
plt.ylabel('Viajeros (miles)')
plt.title('Modelo ARIMA')
plt.legend()
plt.show()


# Ahora creamos el modelo automático
modelo_auto = pm.auto_arima(train, m=12, d=0, D=1,
                           start_p=0, max_p=3, start_q=0, max_q=3,
                           seasonal=True, trace=True,
                           error_action='ignore', suppress_warnings=True,
                           stepwise=True)

```

```
# Imprimimos el modelo y estudiamos sus residuos
print(modelo_auto.summary())
best_arima = sm.tsa.ARIMA(train, order=(0, 1, 1), seasonal_order=(0, 1,
1, 12))
resultados_a = best_arima.fit()

resultados_a.plot_diagnostics(figsize=(12, 8))
plt.show()

# Calculamos las predicciones y las comparamos con los datos test
prediciones_a = resultados_a.get_forecast(steps=12)
predi_test_a=prediciones_a.predicted_mean
intervalos_confianza_a = prediciones_a.conf_int()
plt.figure(figsize=(12, 8))
plt.plot(train, label='Train', color='gray')
plt.plot(test, label='Test', color='blue')
plt.plot(prediciones_a.predicted_mean, label='Predicciones', color='
orange')
plt.xlabel('Periodo')
plt.ylabel('Viajeros (miles)')
plt.title('Modelo ARIMA')
plt.legend()
plt.show()


plt.figure(figsize=(12, 8))
plt.plot(intervalos_confianza_a['lower Viajeros'], label='UCL', color='
gray')
plt.plot(intervalos_confianza_a['upper Viajeros'], label='LCL', color='
gray')
plt.plot(predi_test, label='Predicciones', color='orange')
plt.plot(test, label='Test', color='blue')
plt.xlabel('Fecha')
plt.ylabel('Viajeros (miles)')
plt.title('Modelo ARIMA')
plt.legend()
plt.show()

# Calculamos predicciones a futuro (para los próximos cinco años)
prediciones = resultados.get_forecast(steps=72)

# Graficamos las predicciones
plt.figure(figsize=(12, 8))
plt.plot(viajeros, label='Datos viajeros', color='gray')
plt.plot(prediciones.predicted_mean, label='Predicciones', color='orange'
)
plt.xlabel('Fecha')
plt.ylabel('Viajeros (miles)')
plt.title('Modelo ARIMA')
plt.legend()
plt.show()
```