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

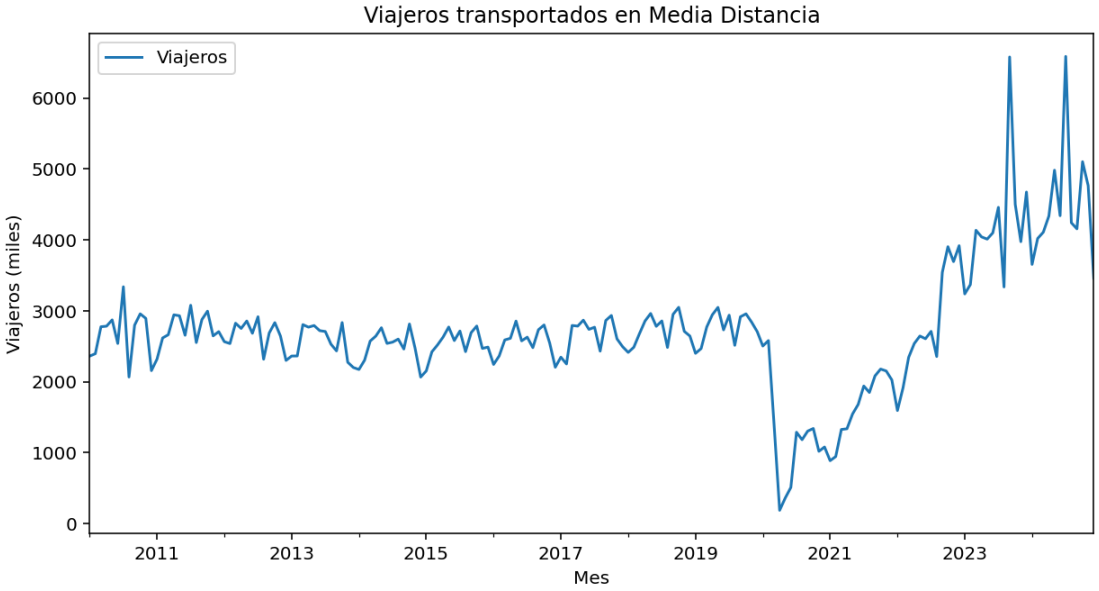

Figura 1

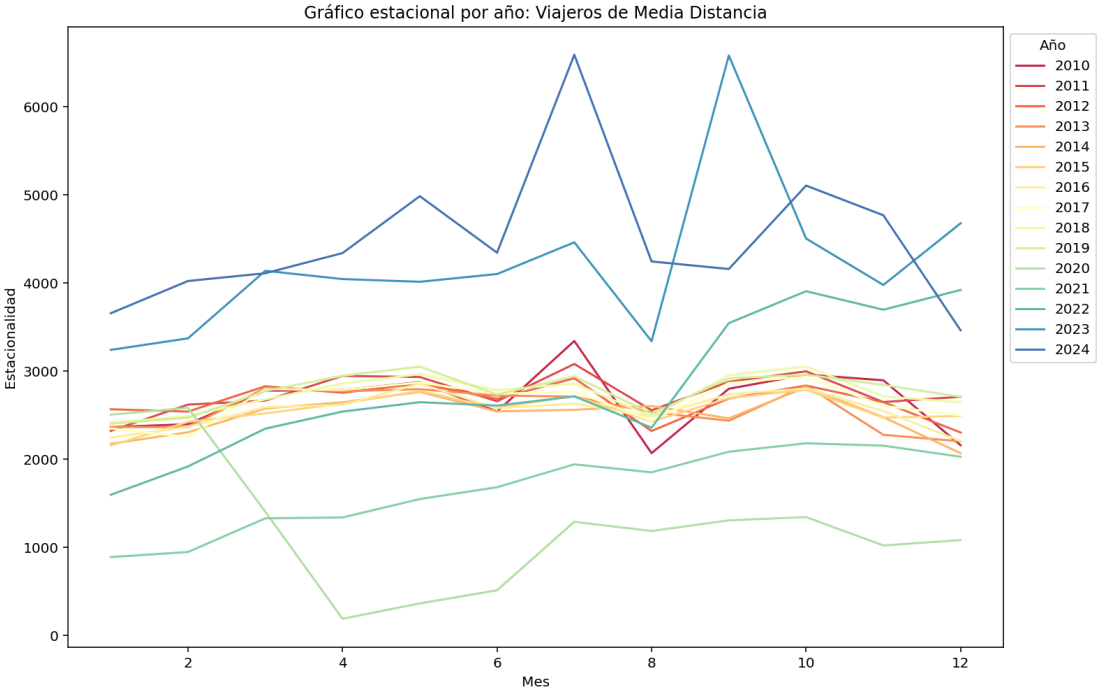

Figura 2

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.

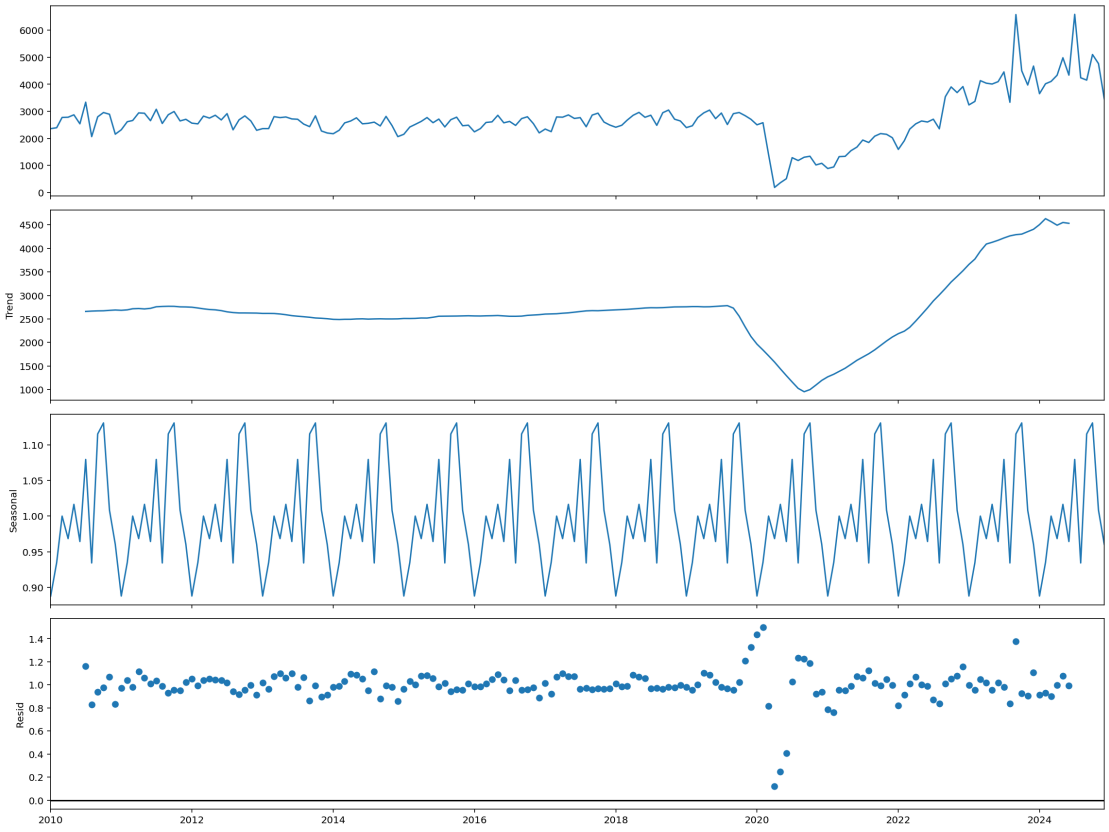

Figura 3

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

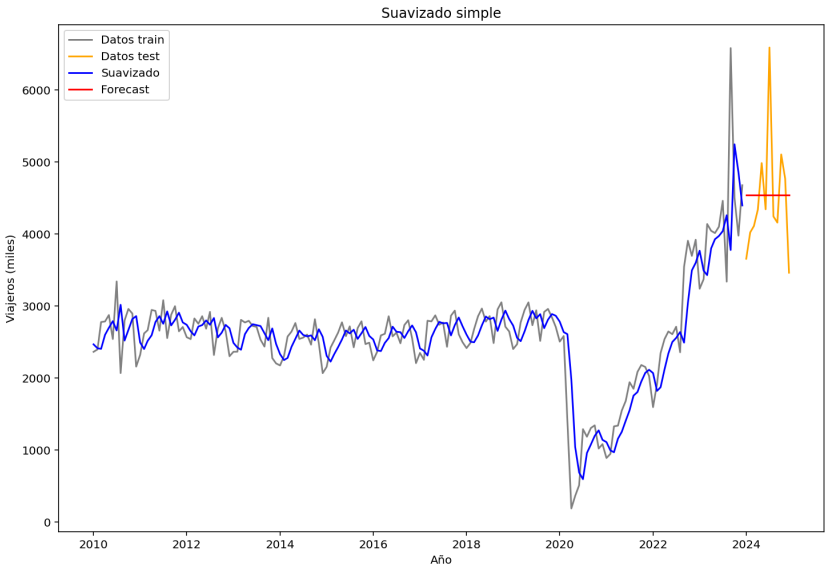

Figura 4: Modelo de suavizado simple

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

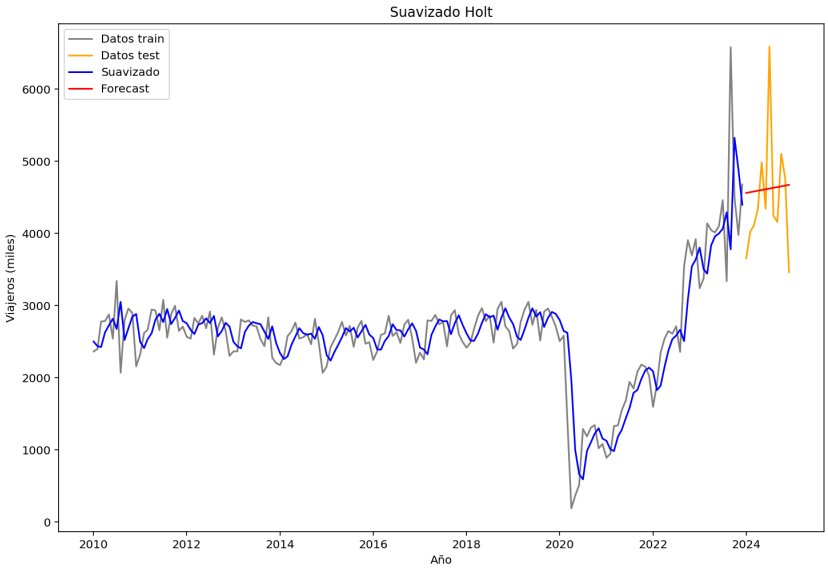

Figura 5: Modelo de suavizado de Holt

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

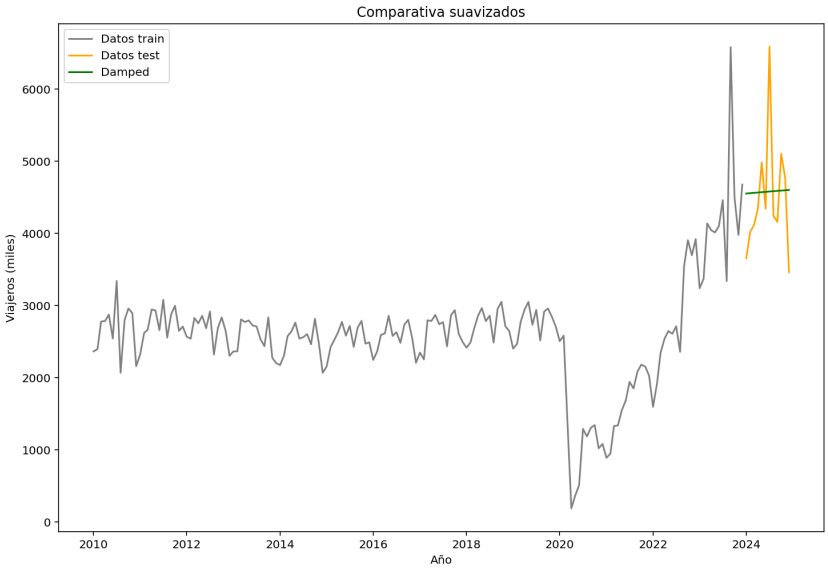

Figura 6: Modelo de tendencia amortiguada

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.

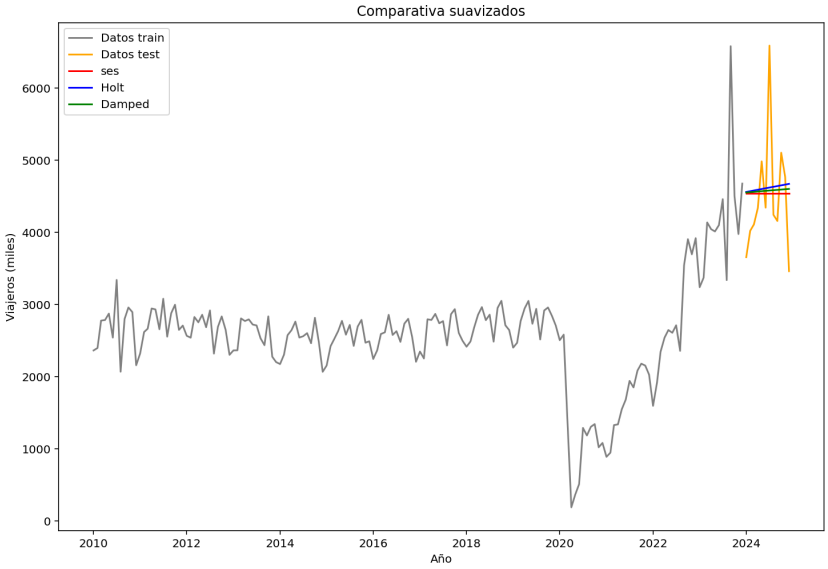

Figura 7: Comparación: suavizado simple, Holt y *damped*

2.4. Modelo de suavizado de Holt-Winters

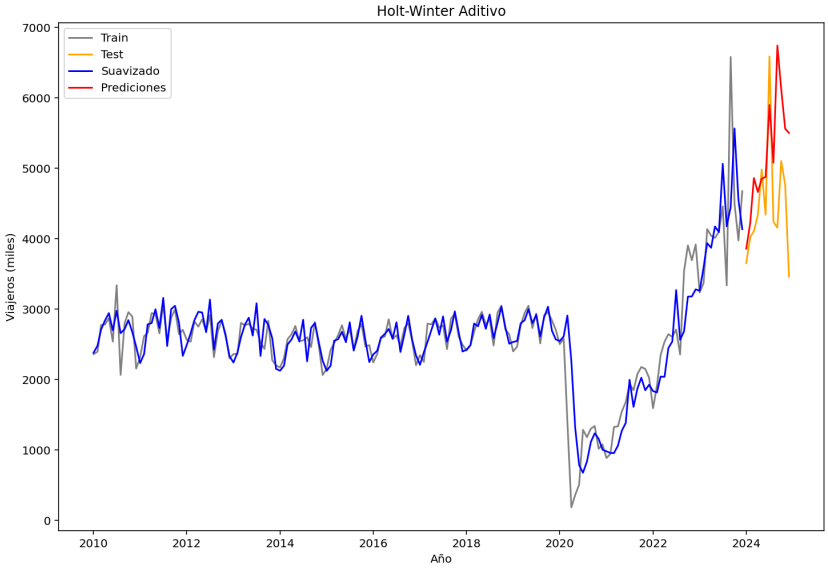

Figura 8: Modelo de tendencia amortiguada

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.

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
```

```
44
45  # Hacemos la descomposición estacional multiplicativa
46  mult_decomp = seasonal_decompose(viajeros, model='multiplicative', period
    =12)
47  plt.rc("figure", figsize=(16,12))
48  fig = mult_decomp.plot()
49
50  # Representamos la tendencia y la serie ajustada estacionalmente
51  viajeros_ajustada = viajeros['Viajeros'] - mult_decomp.seasonal
52  plt.figure(figsize=(12, 8))
53  plt.plot(viajeros, label='Datos', color='gray')
54  plt.plot(mult_decomp.trend, label='Tendencia', color='blue') # Tendencia
55  plt.plot(viajeros_ajustada, label='Estacionalmente ajustada', color='red'
    ) # Serie ajustada
56  plt.xlabel('Fecha')
57  plt.ylabel('Viajeros (miles)')
58  plt.title('Viajeros en Media Distancia')
59  plt.legend(loc='best')
60  plt.show()
61
62  # Separamos los datos en train y test, guardando el último año como test
63  train = viajeros[:-12]
64  test = viajeros[-12:]
65
66  plt.figure(figsize=(12, 8))
67  plt.plot(train, label='Train', color='gray')
68  plt.plot(test, label='Test', color='yellow')
69  plt.legend()
70  plt.xlabel('Fecha')
71  plt.ylabel('Viajeros')
72  plt.show()
73
74  # Aplicamos el suavizado exponencial simple
75  model = SimpleExpSmoothing(train, initialization_method="estimated").fit
    ()
76  print(model.params_formatted)
77  fcast = model.forecast(12)
78
79  plt.figure(figsize=(12, 8))
80  plt.plot(train, label='Datos train', color='gray')
81  plt.plot(test, label='Datos test', color='orange')
82  plt.plot(model.fittedvalues, label='Suavizado', color='blue')
83  plt.plot(fcast, label='Forecast', color='red')
84  plt.xlabel('Año')
85  plt.ylabel('Viajeros (miles)')
86  plt.title('Suavizado simple')
87  plt.legend()
88  plt.show()
89
90
91
```

```
92  # Aplicamos el método de Holt
93  model1 = Holt(train, initialization_method="estimated").fit()
94  fcast1 = model1.forecast(12)
95  print(model1.params_formatted)
96  print(fcast1)
97
98  plt.figure(figsize=(12, 8))
99  plt.plot(train, label='Datos train', color='gray')
100 plt.plot(test, label='Datos test', color='orange')
101 plt.plot(model1.fittedvalues, label='Suavizado', color='blue')
102 plt.plot(fcast1, label='Forecast', color='red')
103 plt.xlabel('Año')
104 plt.ylabel('Viajeros (miles)')
105 plt.title('Suavizado Holt')
106 plt.legend()
107 plt.show()
108
109
110 # Aplicamos el método de la tendencia amortiguada
111 model2 = Holt(train,damped_trend=True, initialization_method="estimated")
    .fit()
112 fcast2 = model2.forecast(12)
113 print(model2.params_formatted)
114
115 plt.figure(figsize=(12, 8))
116 plt.plot(train, label='Datos train', color='gray')
117 plt.plot(test, label='Datos test', color='orange')
118 plt.plot(fcast, label='ses', color='red')
119 plt.plot(fcast1, label='Holt', color='blue')
120 plt.plot(fcast2, label="Damped",color='green')
121 plt.xlabel('Año')
122 plt.ylabel('Viajeros (miles)')
123 plt.title('Comparativa suavizados')
124 plt.legend()
125 plt.show()
126
127
128 # Aplicamos el método de Holt-Winters
129 model3 = ExponentialSmoothing(train, seasonal_periods=12, trend="add",
130                                seasonal="add", initialization_method="
                                   estimated").fit()
131 fcast3 = model3.forecast(12)
132 print(fcast3)
133
134 plt.figure(figsize=(12, 8))
135 plt.plot(train, label='Train', color='gray')
136 plt.plot(test, label='Test', color='blue')
137 plt.plot(model3.fittedvalues, label='Suavizado', color='blue')
138 plt.plot(fcast3,color='red', label="Prediciones")
139 plt.xlabel('Año')
140 plt.ylabel('Viajeros (miles)')
```

```
141 plt.title('Holt-Winter Aditivo')
142 plt.legend()
143
144 #Mostramos los parámetros
145 headers = ['Name', 'Param', 'Value', 'Optimized']
146 table_str = tabulate(model3.params_formatted, headers, tablefmt='
    fancy_grid')
147 print(table_str)
148
149 # Mostramos la evolución de los componentes
150 fig, axes = plt.subplots(nrows=3, ncols=1, figsize=(12, 8))
151 axes[0].plot(model3.level)
152 axes[0].set_title('Level')
153 axes[1].plot(model3.trend)
154 axes[1].set_title('Trend')
155 axes[2].plot(model3.season)
156 axes[2].set_title('Season')
157 plt.tight_layout()
158 plt.show()
159
160
161 # Dibujamos el correlograma
162 fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))
163 plot_acf(train, lags=20, ax=ax1)
164 ax1.set_title('Función de Autocorrelación (ACF) de viajeros')
165 plot_pacf(train, lags=20, ax=ax2)
166 ax2.set_title('Función de Autocorrelación Parcial (PACF) de viajeros')
167 plt.tight_layout()
168 plt.show()
169
170 # Creamos el modelo manual
171 modelo_arima = sm.tsa.ARIMA(train, order=(1, 1, 1), seasonal_order=(0, 1,
     1, 12))
172 resultados = modelo_arima.fit()
173 print(resultados.summary())
174
175 # Comprobamos que los residuos estén incorrelados
176 resultados.plot_diagnostics(figsize=(12, 8))
177 plt.show()
178
179 print(resultados.mse)
180
181 # Calculamos predicciones
182 prediciones = resultados.get_forecast(steps=12)
183 predi_test=prediciones.predicted_mean
184 print(predi_test)
185
186 # Graficamos las predicciones
187 plt.figure(figsize=(12, 8))
188 plt.plot(train, label='Train', color='gray')
189 plt.plot(test, label='Test', color='blue')
```

```
plt.plot(predicciones.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 = predicciones.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=None, D=1,
                           start_p=0, max_p=3, start_q=0, max_q=3,
                           start_P=0, max_P=2, start_Q=0, max_Q=2,
                           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')
```

```
236 plt.ylabel('Viajeros (miles)')
237 plt.title('Modelo ARIMA')
238 plt.legend()
239 plt.show()
240
241
242 plt.figure(figsize=(12, 8))
243 plt.plot(intervalos_confianza_a['lower Viajeros'], label='UCL', color='
    gray')
244 plt.plot(intervalos_confianza_a['upper Viajeros'], label='LCL', color='
    gray')
245 plt.plot(predi_test, label='Predicciones', color='orange')
246 plt.plot(test, label='Test', color='blue')
247 plt.xlabel('Fecha')
248 plt.ylabel('Viajeros (miles)')
249 plt.title('Modelo ARIMA')
250 plt.legend()
251 plt.show()
```