Módulo 9: Minería de datos y modelización predictiva

UNIVERSIDAD
COMPLUTENSE
MADRID

Profesor: Juana María Alonso
Alumno: Pablo Pérez Calvo

1. Introducción

La base de datos elegida para realizar la práctica recoge los datos del número de viajeros que entran en Cádiz a lo largo del año. Este conjunto de datos ha sido tomado del INE y se han seleccionado datos que abarcan desde 2010 hasta 2023, un total de 168 observaciones.

Además, como la idea principal de la selección de esta base de datos es el estudio del turismo en Cádiz se va a tener en cuenta el número total de viajeros, la suma de los residentes en España y en el extranjero.

La base de datos consta de dos categorías: "Periodo" y "Viajeros". La primera columna nos da la información del año y el mes en el que se viajó a Cádiz, la segunda nos indica el número total de viajeros a Cádiz en la fecha dada. A continuación, mostraremos las 5 primeras observaciones de la base de datos para mostrar un ejemplo del formato de estos.

Periodo	Viajeros
2010M01	68.620
2010M02	116.739
2010M03	147.463
2010M04	178.649
2010M05	223.255

Antes de comenzar con el análisis recogeremos en el siguiente cuadro todas las librerias de *Python* utilizadas en esta práctica.

```
#Importacion de librerias
import numpy as np
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt
import seaborn as sns
import os
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing
    , Holt
from tabulate import tabulate
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from sklearn.metrics import mean_absolute_error, mean_squared_error
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import pmdarima as pm
```

2. Análisis

Ejercicio 2

Representación gráfica y descomposición estacional

En primer lugar, realizamos la lectura del archivo y transformamos el formato de periodo a un formato fecha.

```
datos = pd.read_excel('TurismoCadiz.xlsx')

datos['Periodo']=datos['Periodo'].str.strip()
datos['Periodo']= pd.to_datetime(datos['Periodo'].str.replace('M', '-'),
      format='%Y-%m')
```

En segundo lugar, creamos la serie indexando la columna periodo

```
turismo_Cad= datos.set_index('Periodo')['Viajeros']
```

Por último representamos gráficamente la serie con la ayuda de la librería *matplotlib*.

```
turismo_Cad.plot()
plt.title('Numero de viajeros en la ciudad de Cadiz')
plt.xlabel('Fecha')
plt.ylabel('Viajeros')
plt.show()
```

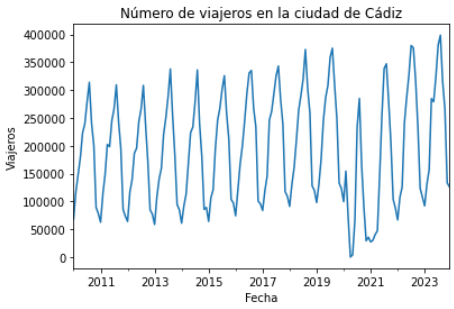

Figura 1: Representación gráfica de la serie

Esta serie claramente presenta **estacionalidad** debido a que presenta un comportamiento que se repite cada periodo.

Para realizar la descomposición estacional lo haremos por el modelo aditivo, debido a que tenemos datos igual a 0 correspondientes a la época del COVID.

```
Additive_decomposition= seasonal_decompose(turismo_Cad, model='additive'
    , period=12)
#Representamos los componentes de la serie obtenidos.
plt.rc("figure", figsize=(16, 12))
plt.rc("font", size=13)
fig = Additive_decomposition.plot()
```

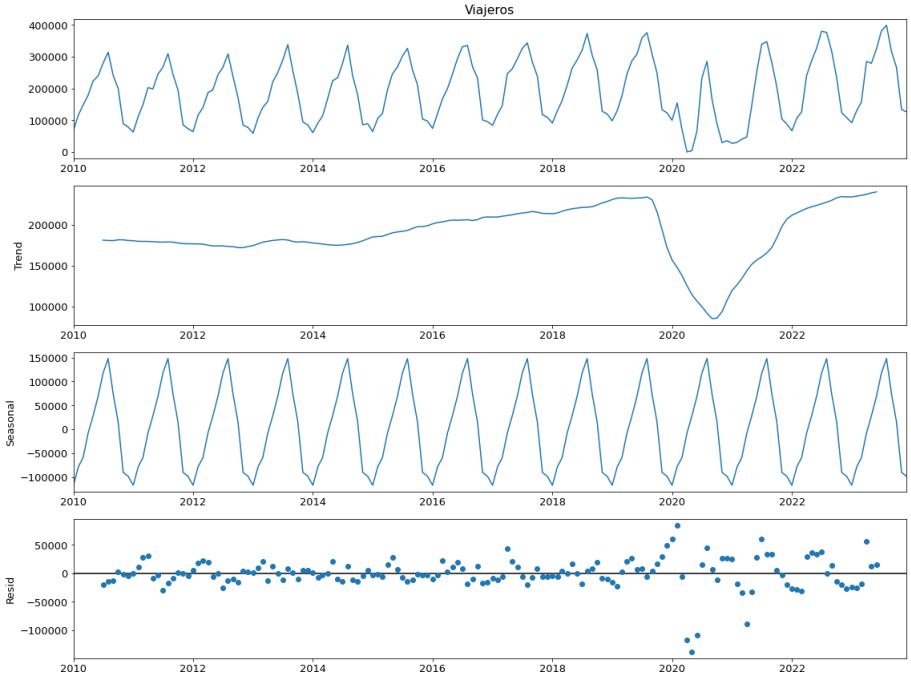

Figura 2: Descomposición estacional aditiva

Observando la tecera gráfica, la cual representa la estacionalidad, comprobamos que se repite de manera constante puesto que los coeficientes se repiten para cada periodo. La segunda gráfica representa la tendencia y la cuarta la componente irregular.

Periodo	2010-01-01	2010-02-01	2010-03-01	2010-04-01	2010-05-01	2010-06-01	2010-07-01	2010-08-01	2010-09-01	2010-10-01	2010-11-01	2010-12-01
Coeficientes de estacionalidad	-117297.201656	-78125.140759	-59671.563835	-7898.252938	27983.391293	69234.509882	117539.490652	148186.593216	73317.154113	15617.997062	-90382.679220	-98504.297810

Esta tabla representa los coeficientes de estacionalidad. Por un lado, Enero tiene el menor de los coeficientes lo que indica que en ese mes hay $117297,2$ viajeros menos que la media. Por otro lado, Julio y Agosto son los que mayor coeficiente de estacionalidad presentan.

Gracias a estas componentes podemos representar la tendencia y la serie ajustada estacionalmente, es decir, eliminando la componente estacional. Al ser el modelo aditivo se obtiene restando el coeficiente estacional correspondiente a cada mes a la serie.

```
S_Ajustada_Est=turismo_Cad-Additive_decomposition.seasonal
plt.figure(figsize=(12, 8))
# Serie original
plt.plot(turismo_Cad, label='Datos', color='gray')
# Tendencia
plt.plot(Additive_decomposition.trend, label='Tendencia', color='blue')
# Serie estacionalmente ajustada
plt.plot(S_Ajustada_Est, label='Estacionalmente ajustada', color='red')
plt.xlabel('Fecha')
plt.ylabel('Viajeros')
plt.title('Viajeros en Cadiz')
plt.legend()
plt.show()
```

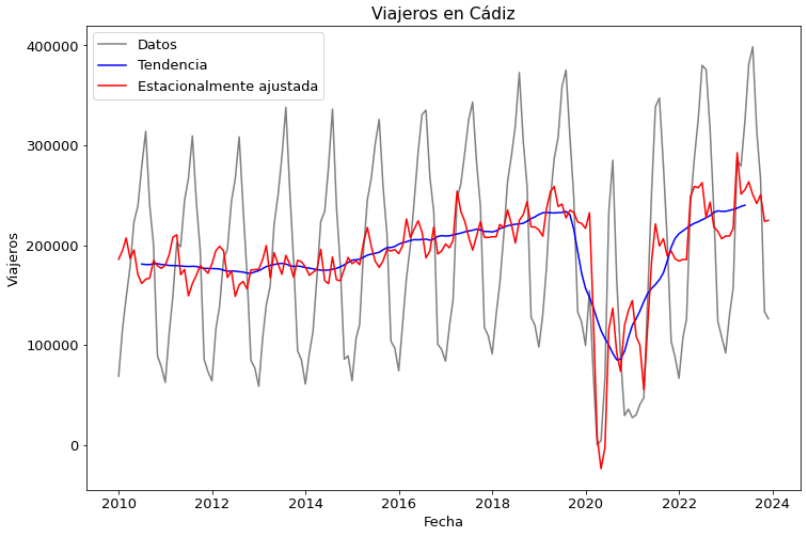

Figura 3: Comparación tendencia y estacionalmente ajustada

Otra forma de representar gráficamente la estacionalidad es dibujar los valores de la serie separando cada año con un color diferente.

```
plt.figure(figsize=(12, 8))
sns.lineplot(x= turismo_Cad.index.month, y= turismo_Cad,
hue= turismo_Cad.index.year, palette='viridis')
plt.xlabel('Mes')
plt.ylabel('Estacionalidad')
plt.title('Grafico estacional por anio: Viajeros en Cadiz')
plt.legend(title='Anio',loc='upper left', bbox_to_anchor=(1, 1))
plt.show()
```

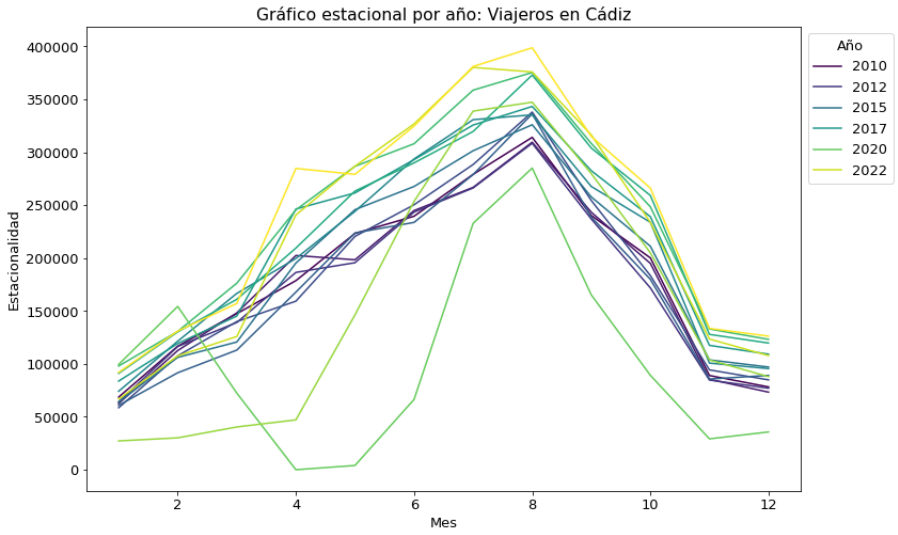

Figura 4: Gráfico estacional por año

Ejercicio 3

Para comprobar la eficacia de los métodos de predicción que vamos a hacer en los siguientes apartados reservamos los últimos datos observados (TEST)(un periodo en las series estacionales o aproximadamente 10 observaciones) para comparar con las predicciones realizadas por cada uno de los métodos. Luego ajustamos los modelos sobre la serie sin esos últimos datos (TRAIN) en los siguientes apartados.

Vamos a dividir los datos en dos conjuntos y reservaremos para el test las últimas 12 observaciones, es decir, los datos correspondientes al último año que presenta la

base de datos (2023).

```
train = turismo_Cad[:156]
test = turismo_Cad[156:]

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('defunciones')
plt.show()
```

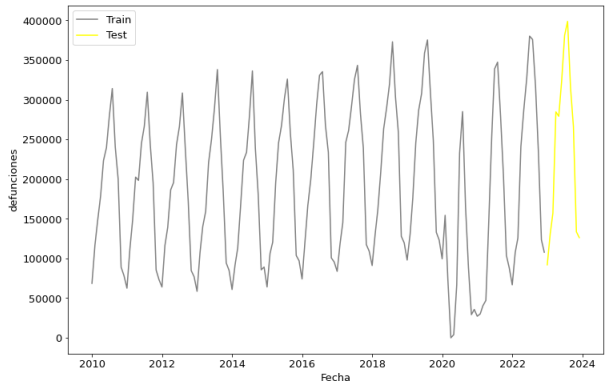

Figura 5: División Train-Test

Ejercicio 4

Encontrar el modelo de suavizado exponencial más adecuado, mostrando una tabla con los estimadores de los parámetros del modelo elegido. Para dicho modelo, representar gráficamente la serie observada y la suavizada con las predicciones para el periodo TEST. Mostrar una tabla con las predicciones

Como nuestra serie es estacional, el modelo de suavizado exponencial más adecuado es el de Holt-Winters con el modelo aditivo porque tenemos valores cero. El código utilizado para esto es el siguiente.

```
# Aplicamos suavizado
model1 = ExponentialSmoothing(train, seasonal_periods=12,trend="add",
seasonal="add", initialization_method="estimated").fit()
# Queremos predecir las 12 ultimas observaciones
fcast = model1.forecast(12)

# Representacion grafica
plt.figure(figsize=(12, 8))
plt.plot(train, label='Train', color='gray')
plt.plot(test, label='Test', color='yellow')
plt.plot(model1.fittedvalues, label='suavizado', color='blue')
plt.plot(fcast,color='red', label="prediciones")
plt.xlabel('Anio')
plt.ylabel('Viajeros')
plt.title('Holt-Winter Aditivo')
plt.legend()
```

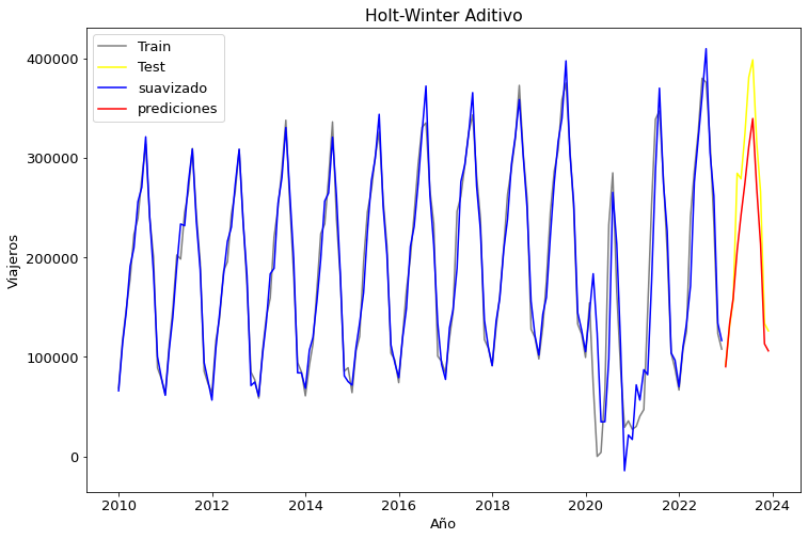

Figura 6: Suavizado Holt-Winter

Para mostrar una tabla con los estimadores de los parámetros del modelo, utilizamos el siguiente código.

```
# Encabezados de la tabla
headers = ['Name', 'Param', 'Value', 'Optimized']

# Imprimir la tabla con formato
table_str = tabulate(model1.params_formatted, headers, tablefmt='
    fancy_grid')
# Mostrar la tabla formateada
print(table_str)
```

Name	Param	Value	Optimized
smoothing_level	alpha	0.959643	True
smoothing_trend	beta	0.0001	True
smoothing_seasonal	gamma	0.0403571	True
initial_level	l.0	181420	True
initial_trend	b.0	-179.063	True
initial_seasons.0	s.0	-115258	True
initial_seasons.1	s.1	-69893.5	True
initial_seasons.2	s.2	-42105.3	True
initial_seasons.3	s.3	2028.62	True
initial_seasons.4	s.4	32823.4	True
initial_seasons.5	s.5	66297.7	True
initial_seasons.6	s.6	96715.4	True
initial_seasons.7	s.7	139430	True
initial_seasons.8	s.8	66267.4	True
initial_seasons.9	s.9	10612.1	True
initial_seasons.10	s.10	-88478.4	True
initial_seasons.11	s.11	-98438.8	True

Figura 7: Estimadores de los parámetros

Las ecuaciones que representan al modelo de suavizado exponencial de Holt-Winters en su forma aditiva son las siguientes.

$$L_t = 0{,}9596(x_t - S_{t-s}) + (1 - 0{,}9596)(L_t - 1 + b_t - 1)$$
$$b_t = 0{,}0001(L_t - L_{t-1}) + 0{,}9999 b_{t-1}$$
$$S_t = 0{,}04(x_t - L_{t-1} - b_{t-1}) + (1 - 0{,}04) S_{t-s}$$
$$x_{t+h} = (L_t + b_t h) + S_{t+h-s}$$

Ejercicio 5

Representar la serie y los correlogramas. Según el resultado de los correlogramas, decidir qué modelo puede ser ajustado. Ajustar el modelo adecuado comprobando que sus residuales están incorrelados. (Sintaxis, tablas de los parámetros estimados y gráficos)

La visualización de los correlogramas nos permitirá conocer la estacionariedad, el tipo de modelo y los retardos que son significativamente diferentes de cero. Los representaremos con la ayuda del siguiente código.

```
# Crear subgraficos para ACF y PACF
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))
# Graficar la funcion de autocorrelacipn (ACF)
plot_acf(train, lags=30, ax=ax1)
ax1.set_title('Funcipn de Autocorrelacion (ACF)')

# Graficar la funcion de autocorrelacion parcial (PACF)
plot_pacf(train, lags=30, ax=ax2)
ax2.set_title('Funcion de Autocorrelacion Parcial (PACF)')

plt.tight_layout()
plt.show()
```

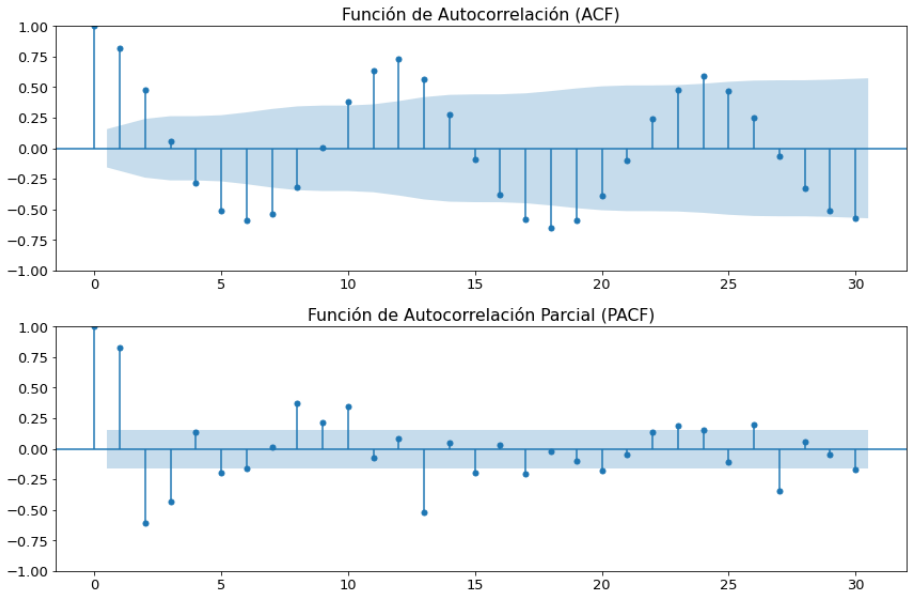

Figura 8: Representación de correlogramas

Podemos observar claramente su estructura estacional, debida al comportamiento repetitivo cada 12 meses en la gráfica ACF, la autocorrelación más fuerte es en los retardos múltiplos de 12. Aunque como podemos ver en la Figura 15 no presenta estacionariedad en media.

Podemos obtener una serie estacionaria, diferenciando mediante diferenciación de orden estacional.

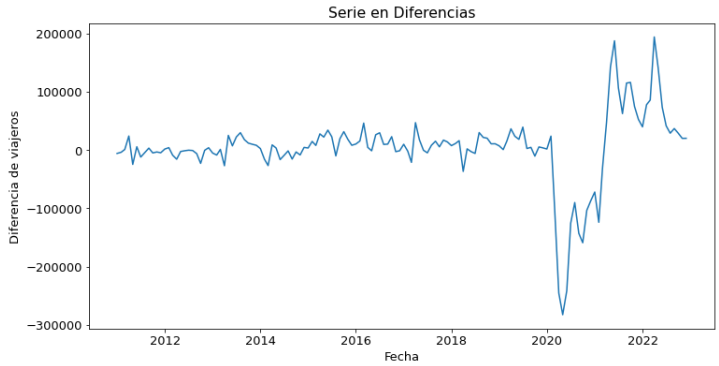

Figura 9: Serie en diferencias

Ahora calculamos de nuevo los correlogramas para la serie diferenciada.

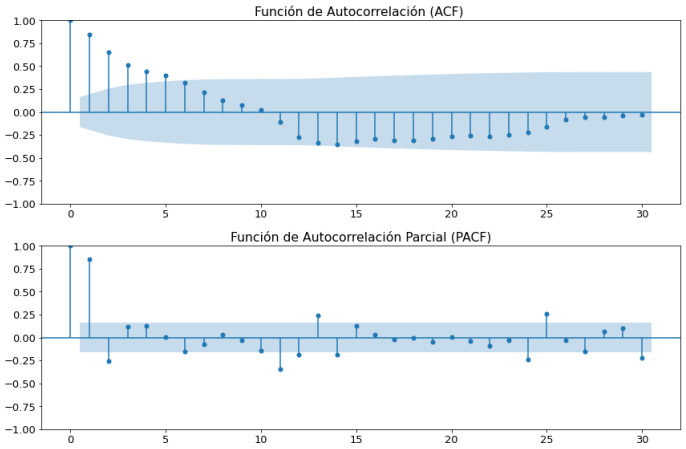

Figura 10: Correlogramas serie diferenciada

El proceso ya es estacionario porque la correlaciones simples decrecen rápidamente. La PACF muestra cortes significativos en los rezagos 1 y 2 y se repite periódicamente cada 12. Por lo que, un posible candidato a modelo es un $ARIMA(2,0,0)(0,1,1)_{12}$

```
modelo_arima = sm.tsa.ARIMA(train, order=(2, 0, 0),
seasonal_order=(0, 1, 1, 12))
resultados = modelo_arima.fit()
```

El resumen del modelo es el siguiente.

```
 SARIMAX Results
==========================================================================
Dep. Variable:                    Viajeros   No. Observations:       156
Model:      ARIMA(2, 0, 0)x(0, 1, [1], 12)   Log Likelihood   -1679.971
Date:     Sun, 16 Mar 2025                   AIC     3367.942
Time:     13:08:30                           BIC     3379.822
Sample:   01-01-2010                         HQIC    3372.770
          - 12-01-2022
Covariance Type:                       opg
==========================================================================
             coef    std err       z      P>|z|      [0.025    0.975]
--------------------------------------------------------------------------
ar.L1       1.1247    0.070     16.028      0.000      0.987     1.262
ar.L2      -0.2778    0.066     -4.225      0.000     -0.407    -0.149
ma.S.L12   -0.7188    0.081     -8.860      0.000     -0.878    -0.560
sigma2   9.925e+08  4.3e-11   2.31e+19      0.000   9.92e+08  9.92e+08
==========================================================================
Ljung-Box (L1) (Q):     0.00   Jarque-Bera (JB):  144.62
Prob(Q):                0.97   Prob(JB):            0.00
Heteroskedasticity (H): 2.66   Skew:               -0.63
Prob(H) (two-sided):    0.00   Kurtosis:            7.74
==========================================================================
```

Vamos a realizar un diagnóstico del modelo para ver si cumple los supuestos básicos:

Todo los coeficientes tienen $p - valor < 0{,}05$, por lo que son estadísticamente significativos.

Ahora tenemos que comprobar que los residuos están incorrelados, lo haremos con la ayuda de un gráfico que muestra el comportamiento de los residuos.

```
# Graficar los residuos del modelo
resultados.plot_diagnostics(figsize=(12, 8))
plt.show()
```

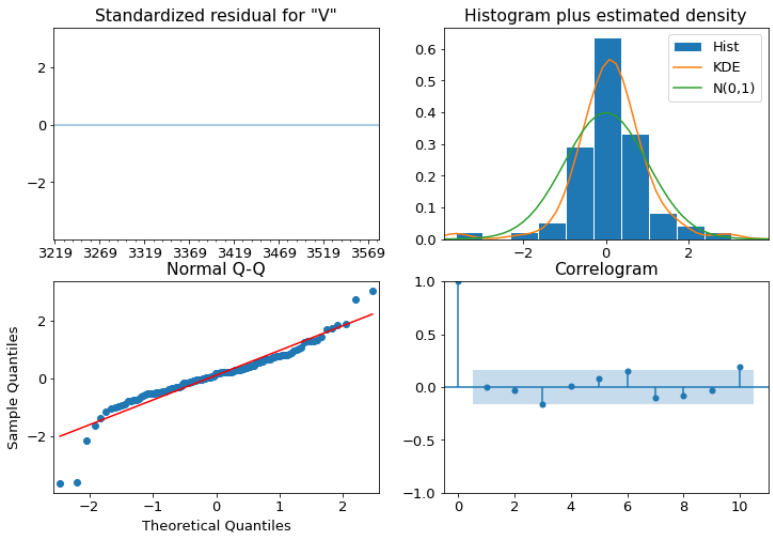

Figura 11: Gráfico de los residuos del modelo manual

Podemos observar en el gráfico del correlograma que todos los residuos menos uno se encuentran dentro de las bandas de confianza, situadas paralelamente a una distancia de $\frac{2}{\sqrt{T}}$ respecto al origen.

En el gráfico del Q-Q plot, los valores de los extremos se alejan un poco de la línea debido a los valores nulos de la época de la pandemia.

Volviendo al resumen del modelo escrito anteriormente, el p-valor de la prueba Ljung-Box es de $0{,}97 > 0{,}05$ lo que nos indica que no hay autocorrelación significativa en los residuos, estos son independientes.

Otra forma de evaluar el modelo es utilizando el Error Absoluto Medio y la Media de los Errores al Cuadrado.

```
print(resultados.mse)      929353009.6546615
print(resultados.mae)      21655.023808617883
```

Aparte de presentar un error reducido, para que un modelo sea bueno es importante que sea lo más sencillo posible. Para esto se hace uso del criterio de información de Akaike y el criterio bayesiano de Schwarz, ambos nos lo proporciona el resumen del modelo mostrado anteriormente. **AIC:** $3367{,}942$ y **BIC:** $3379{,}822$.

Ambos criterios buscan encontrar el modelo óptimo, equilibrando el ajuste del

modelo y el número de parámetros utilizados. Cuánto menor sean estos valores mejor será el modelo.

Ejercicio 6

Ajustar un modelo ARIMA con ajuste automático. Comparar los resultados con el manual y elegir el mejor.

Una vez creado el modelo manualmente, vamos a ajustar un modelo ARIMA mediante ajuste automático utilizando la función *pm.auto_arima*

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

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,1,1)[12] intercept   : AIC=3370.086, Time=0.55 sec
 ARIMA(0,0,0)(0,1,0)[12] intercept   : AIC=3586.485, Time=0.08 sec
 ARIMA(1,0,0)(1,1,0)[12] intercept   : AIC=3395.580, Time=0.34 sec
 ARIMA(0,0,1)(0,1,1)[12] intercept   : AIC=3480.592, Time=0.37 sec
 ARIMA(0,0,0)(0,1,0)[12]             : AIC=3585.454, Time=0.08 sec
 ARIMA(1,0,1)(0,1,0)[12] intercept   : AIC=3403.082, Time=0.17 sec
 ARIMA(1,0,1)(1,1,1)[12] intercept   : AIC=3370.863, Time=0.59 sec
 ARIMA(1,0,1)(0,1,2)[12] intercept   : AIC=3370.632, Time=0.97 sec
 ARIMA(1,0,1)(1,1,0)[12] intercept   : AIC=3387.643, Time=0.46 sec
 ARIMA(1,0,1)(1,1,2)[12] intercept   : AIC=3371.898, Time=2.75 sec
 ARIMA(1,0,0)(0,1,1)[12] intercept   : AIC=3377.962, Time=0.38 sec
 ARIMA(2,0,1)(0,1,1)[12] intercept   : AIC=3370.835, Time=0.61 sec
 ARIMA(1,0,2)(0,1,1)[12] intercept   : AIC=3369.458, Time=0.59 sec
 ARIMA(1,0,2)(0,1,0)[12] intercept   : AIC=3403.480, Time=0.27 sec
 ARIMA(1,0,2)(1,1,1)[12] intercept   : AIC=3370.399, Time=0.87 sec
 ARIMA(1,0,2)(0,1,2)[12] intercept   : AIC=3370.273, Time=1.45 sec
 ARIMA(1,0,2)(1,1,0)[12] intercept   : AIC=3387.473, Time=0.75 sec
 ARIMA(1,0,2)(1,1,2)[12] intercept   : AIC=3371.588, Time=4.60 sec
 ARIMA(0,0,2)(0,1,1)[12] intercept   : AIC=3432.031, Time=0.45 sec
 ARIMA(2,0,2)(0,1,1)[12] intercept   : AIC=3370.579, Time=1.38 sec
 ARIMA(1,0,3)(0,1,1)[12] intercept   : AIC=3371.613, Time=0.89 sec
 ARIMA(0,0,3)(0,1,1)[12] intercept   : AIC=3416.182, Time=0.56 sec
 ARIMA(2,0,3)(0,1,1)[12] intercept   : AIC=3372.789, Time=1.14 sec
 ARIMA(1,0,2)(0,1,1)[12]             : AIC=3368.261, Time=0.58 sec
 ARIMA(1,0,2)(0,1,0)[12]             : AIC=3401.516, Time=0.21 sec
 ARIMA(1,0,2)(1,1,1)[12]             : AIC=3369.416, Time=0.76 sec
 ARIMA(1,0,2)(0,1,2)[12]             : AIC=3369.198, Time=1.23 sec
 ARIMA(1,0,2)(1,1,0)[12]             : AIC=3385.607, Time=0.49 sec
 ARIMA(1,0,2)(1,1,2)[12]             : AIC=3370.159, Time=3.10 sec
 ARIMA(0,0,2)(0,1,1)[12]             : AIC=3431.332, Time=0.32 sec
 ARIMA(1,0,1)(0,1,1)[12]             : AIC=3368.835, Time=0.41 sec
 ARIMA(2,0,2)(0,1,1)[12]             : AIC=3369.056, Time=1.15 sec
 ARIMA(1,0,3)(0,1,1)[12]             : AIC=3370.366, Time=0.79 sec
 ARIMA(0,0,1)(0,1,1)[12]             : AIC=3480.148, Time=0.25 sec
 ARIMA(0,0,3)(0,1,1)[12]             : AIC=3415.319, Time=0.52 sec
 ARIMA(2,0,1)(0,1,1)[12]             : AIC=3369.790, Time=0.54 sec
 ARIMA(2,0,3)(0,1,1)[12]             : AIC=3371.245, Time=3.17 sec

Best model:  ARIMA(1,0,2)(0,1,1)[12]
Total fit time: 33.865 seconds
```

El modelo ganador que nos devuelve esta función es un $ARIMA(1,0,2)(0,1,1)[12]$, basándose en el menor AIC.

El resumen del modelo ganador mediante ajuste automático es el siguiente.

```
 SARIMAX Results
==========================================================================
Dep. Variable:                    y   No. Observations:                  156
Model: SARIMAX(1, 0, 2)x(0, 1, [1], 12)   Log Likelihood           -1679.131
Date:   Mon, 17 Mar 2025                  AIC                       3368.261
Time:   13:19:48                          BIC                       3383.110
Sample: 01-01-2010                        HQIC                      3374.295
        - 12-01-2022
Covariance Type:                                 opg
==========================================================================
               coef    std err      z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------
ar.L1      0.7454      0.078     9.540     0.000         0.592    0.899
ma.L1      0.4141      0.093     4.450     0.000         0.232    0.597
ma.L2      0.2069      0.118     1.760     0.078        -0.023    0.437
ma.S.L12  -0.7263      0.082    -8.810     0.000        -0.88    -0.565
sigma2     9.95e+08   8.44e-11  1.18e+19  0.000      9.95e+08    9.95e+08
==========================================================================
Ljung-Box (L1) (Q):       0.09   Jarque-Bera (JB):       135.45
Prob(Q):                  0.77   Prob(JB):                 0.00
Heteroskedasticity (H):   2.59   Skew:                    -0.70
Prob(H) (two-sided):      0.00   Kurtosis:                 7.54
==========================================================================
```

Vamos a estudiar los residuos del modelo como hicimos con el modelo manual.

```
best_arima = sm.tsa.ARIMA(train, order=(1, 0, 2),
seasonal_order=(0, 1, 1, 12))
resultados_a = best_arima.fit()
# Graficar los residuos del modelo
resultados_a.plot_diagnostics(figsize=(12, 8))
plt.show()
```

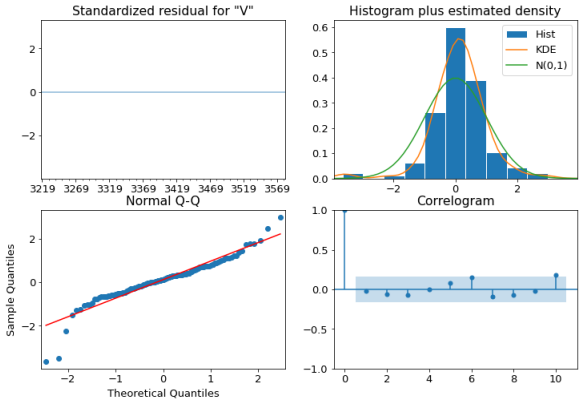

Figura 12: Residuos del modelo ajustado automáticamente

Podemos observar que los gráficos son bastante similares al del modelo manual, casi todos los residuos se encuentran dentro de las bandas de confianza y el gráfico del Q-Q plot presenta problemas en los valores extremos.

Para determinar cual es el mejor modelo, analizaremos la siguiente tabla resumen.

Comparativa de Modelos	**Modelo Manual ARIMA(2,0,0)(0,1,1,12)**	**Modelo Automático ARIMA(1,0,2)(0,1,1,12)**
AIC	**3367.942**	**3379.822**
BIC	**3368.261**	**3383.110**
Ljung-Box (p-valor)	**0.97**	**0.77**
Kurtosis	**7.74**	**7.54**

- Los valores AIC y BIC son ligeramente menores en el primer modelo, esot indica que es un poco mejor en el balance de ajuste y complejidad.
- Ambos modelos presentan un $p-valor > 0{,}05$ para el contraste Ljung-Box lo que indica que sus residuos son independientes pero el modelo manual tiene mayor p-valor por lo que es mejor para la validez del modelo.
- Ambos modelos no presentan normalidad, aunque el segundo presenta ligeramente menor Kurtosis que el primero, se acerca a la normal.
- Por último, en el resumen de cada modelo podemos observar que el modelo ajustado automáticamente utiliza una variable más que el modelo manual y además, una de ellas (ma.L2) tiene $p-valor = 0{,}078 > 0{,}05$ lo que indica que no es significativo. En el modelo manual ya vimos que todos los coeficientes son significativos.

Por todo lo comentado anteriormente, nos quedaremos con el modelo manual $ARIMA(2,0,0)(0,1,1)_{12}$ como el mejor modelo.

Ejercicio 7

Escribir la expresión algebraica del modelo ajustado con los parámetros estimados.

Retomando el resumen de los coeficientes del modelo ganador tenemos.

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          1.1247      0.070     16.028      0.000       0.987       1.262
ar.L2         -0.2778      0.066     -4.225      0.000      -0.407      -0.149
ma.S.L12      -0.7188      0.081     -8.860      0.000      -0.878      -0.560
sigma2     9.925e+08   4.3e-11   2.31e+19      0.000    9.92e+08    9.92e+08
==============================================================================
```

Los coeficientes de la parte autorregresiva son $1{,}1247$ y $-0{,}2778$ y el coeficiente de la parte de medias móviles estacional es $-0{,}7188$. También hay que tener en cuenta la diferenciación estacional.

Nuestra expresión del modelo queda:

$$(1 - 1{,}12B + 0{,}28B^2)(1 - B^{12})X_t = (1 - 0{,}72B^{12})Z_t$$

donde B es el operador de retardo, X_t la serie de tiempo en el instante t y Z_t un proceso de ruido blanco.

Desarrollando la parte izquierda de la ecuación tenemos:

$$(1 - B^{12} - 1{,}12B + 1{,}12B^{13} + 0{,}28B^2 - 0{,}28B^{14})X_t =$$

$$= X_t - B^{12}X_t - 1{,}12BX_t + 1{,}12B^{13}X_t + 0{,}28B^2X_t - 0{,}28B^{14}X_t$$

Aplicando los retardos y desarrollando la parte derecha de la ecuación, tenemos la siguiente expresión final.

$$X_t - X_{t-12} - 1{,}12X_{t-1} + 1{,}12X_{t-13} + 0{,}28X_{t-2} - 0{,}28X_{t-14} = Z_t - 0{,}72Z_{t-12}$$

Finalmente despejando X_t, el modelo nos queda

$$X_t = 1{,}12X_{t-1} - 0{,}28X_{t-2} + X_{t-12} - 1{,}12X_{t-13} + 0{,}28X_{t-14} + Z_t - 0{,}72Z_{t-12}$$

Ejercicio 8

Calcular las predicciones y los intervalos de confianza para las unidades de tiempo que se considere oportuno, dependiendo de la serie, siguientes al último valor observado. Representarlas gráficamente.

Recordamos que en anteriores apartados dividimos el conjunto de datos en train y test, así que vamos a predecir los valores de 2023, el año siguiente al último observado que coincide con lo que guardamos en el fichero test.

```
prediciones = resultados.get_forecast(steps=12)
predi_test=prediciones.predicted_mean
print(predi_test)
```

```
Fecha          Prediccion
2023-01-01     84578.172400
2023-02-01    114783.401062
2023-03-01    119593.984191
2023-04-01    163022.363028
2023-05-01    209807.156994
2023-06-01    260780.383346
2023-07-01    330267.256791
2023-08-01    348289.521629
2023-09-01    274651.875805
2023-10-01    206424.090831
2023-11-01    103064.728149
2023-12-01     93300.706119
```

Ahora representamos gráficamente estas predicciones para compararlas con los valores de los datos test.

```
plt.figure(figsize=(12, 8))
# Serie Original
plt.plot(train, label='Train', color='gray')
plt.plot(test, label='Test', color='yellow')
# Predicciones
plt.plot(prediciones.predicted_mean, label='Predicciones', color='blue')
plt.xlabel('Fecha')
plt.ylabel('Viajeros')
plt.title('Modelo ARIMA')
plt.legend()
plt.show()
```

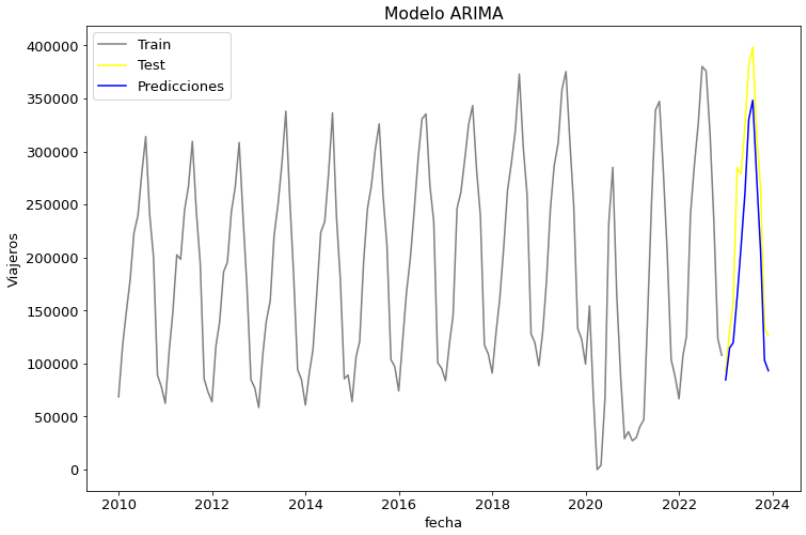

Figura 13: Comparación de predicciones con datos originales

A primera vista podemos observar una buena predicción. Para poder compararlos mejor, vamos a representar gráficamente solo los valores de 2023 y además, calcularemos los intervalos de confianza. Esto nos indicará que a un $95\,\%$ el número de viajeros estará dentro de ese rango.

```
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='blue')
plt.plot(test, label='Test', color='yellow')
plt.xlabel('Fecha')
plt.ylabel('Viajeros')
plt.title('Modelo ARIMA')
plt.legend()
plt.show()
```

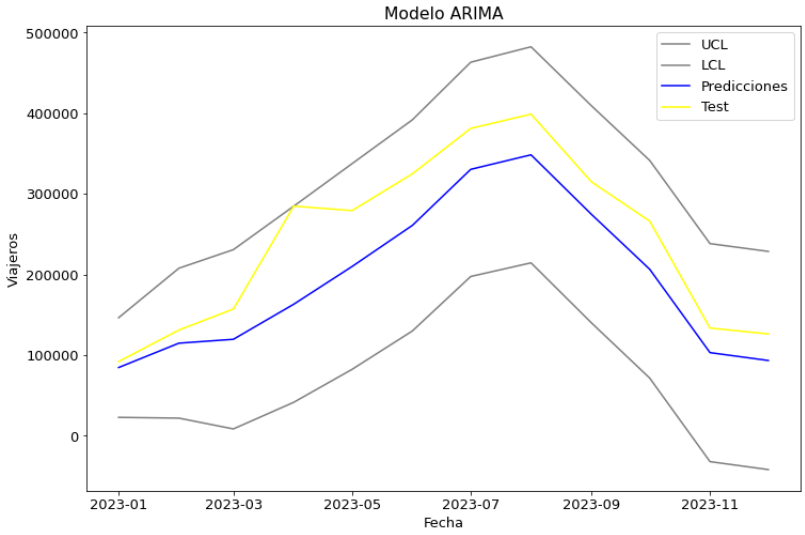

Figura 14: Predicciones vs datos test con intervalos de confianza

Podemos ver que las predicciones se asemejan bastante a los valores originales por lo que concluimos que el modelo ARIMA presentado es un modelo adecuado para esta serie temporal. Observamos como el número de viajeros en Cádiz aumenta cuando llega la temporada de verano.

Ejercicio 9

Comparar las predicciones obtenidas con cada uno de los métodos (suavizado y ARIMA) con los valores observados que habíamos reservado antes. Conclusiones

Recuperando las variables donde guardamos las predicciones mediante el suavizado Holt-Winters y las predicciones con el modelo ARIMA, creamos un gráfico con ambas predicciones para compararlas.

```
plt.plot(test, label='Test', color='yellow')
plt.plot(fcast,color='red', label="Prediciones suavizado")
plt.plot(predi_test, label='Predicciones ARIMA', color='blue')
plt.legend()
plt.title("Comparacion de Predicciones")
plt.xlabel('Fecha')
plt.ylabel('Viajeros')
plt.show()
```

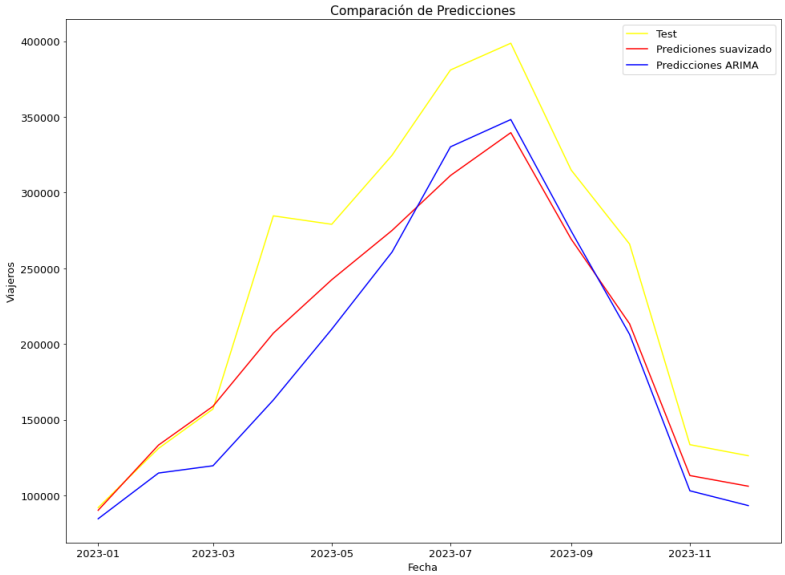

Figura 15: Predicciones Suavizado vs ARIMA

Ambas predicciones son bastante cercanas a los valores originales. Veamos cual es mejor calculando su error absoluto medio y el error cuadrático medio.

```
print(f"MAE Suavizado:{mae_suavizado}, RMSE Suavizado:{rmse_suavizado}")
print(f"MAE ARIMA: {mae_arima}, RMSE ARIMA: {rmse_arima}")
MAE Suavizado: 36429.806868201675, RMSE Suavizado: 44581.12643486132
MAE ARIMA: 48370.86330456089, RMSE ARIMA: 56063.87923734708
```

Concluimos que el modelo de suavizado exponencial es más preciso que ARIMA para estos datos, ya que presenta menor MAE y RMSE. Sus predicciones se aproximan más a los datos originales.