Predicción de temperaturas medias mensuales en Santiago de Compostela. Comparación de modelos de predicción.

David Noya Couselo

UNIVERSIDAD
COMPLUTENSE
MADRID

1.- Introducción

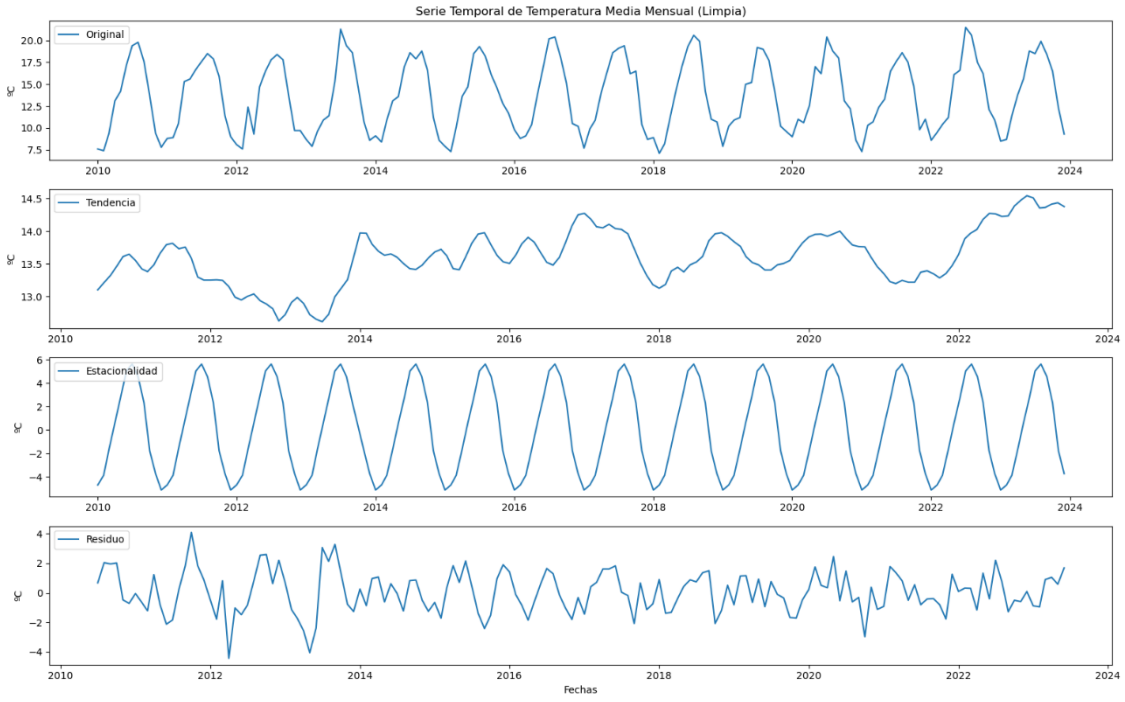
En este estudio, realizaremos un análisis de la serie temporal de temperaturas medias mensuales en la región de Santiago de Compostela, en el periodo de 2010 a 2023. La serie muestra no solo las fluctuaciones climáticas de la ciudad, sino que también ofrece la oportunidad de encontrar posibles tendencias y patrones a lo largo del tiempo. La información climática, especialmente las temperaturas medias de una zona, es muy importante para comprender cambios en el medio ambiente, planificar actividades agrícolas y urbanísticas, así como evaluar posibles impactos en la salud de la población.

La serie utilizada en este estudio contiene registros mensuales de la temperatura media de la ciudad de Santiago de Compostela (expresadas en grados centígrados). La serie temporal consta de 168 mediciones. Estos datos, recogidos por la estación Santiago-EOAS, resultan fundamentales para analizar el comportamiento climático de la región, permitiendo identificar variaciones estacionales, así como tendencias a largo plazo asociadas a fenómenos climáticos globales.

El objetivo del estudio es comparar dos modelos de predicción de temperaturas, para ver cuál de ellos resulta más adecuado para su posible utilización a posteriori. El primer modelo es el de suavizado exponencial, especialmente útil para pronosticar datos cuando existe un patrón estacional. El segundo modelo se escogerá de manera adecuada posteriormente.

2.- Descripción gráfica y descomposición

Al hacer una primera representación gráfica de la serie, encontramos un valor erróneo correspondiente al mes de octubre de 2023 (la serie marcaba un valor -9999 ºC). Este valor en la medición ya venía indicado como erróneo por una variable dicotómica en la serie. Tras eliminarlo de la serie, procedemos a la representación gráfica y su descomposición, obteniendo lo siguiente:



La gráfica anterior muestra la serie original y su descomposición en componentes principales:

- **Tendencia:** La tendencia muestra los cambios a largo plazo de manera más fidedigna. Podemos observar una ligera tendencia alcista en los últimos años, lo cual nos indica un pequeño aumento de las temperaturas medias, especialmente en 2023.
- **Estacionalidad:** El componente de estacionalidad muestra claramente los patrones recurrentes anuales correspondientes a las distintas estaciones.
- **Residuo:** El residuo muestra la variabilidad no explicada por la tendencia o estacionalidad, lo cual nos permite detectar anomalías con mayor precisión. Aparentemente existen algunos eventos inusuales entorno a los años 2012 y 2014, puede que debidos a eventos climáticos extremos o a factores relacionados con los instrumentos de medición. Es posible que en esos años hubiera algún cambio ambiental en la estación meteorológica. Haría falta un estudio más exhaustivo para determinar la causa real.

El código utilizado en este apartado fue el siguiente:

```
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
import pandas as pd

# Leemos el archivo csv
cleaned_data = pd.read_csv('cleaned_resultadoCSV.csv', encoding='latin1')

# Observamos la composición de la tabla
# cleaned_data.head()

# Nos aseguramos de que la columna 'Data' está en el formato adecuado
cleaned_data['Data'] = pd.to_datetime(cleaned_data['Data'])

# Colocamos la columna 'Data' como índice
cleaned_data.set_index('Data', inplace=True)

# Identificamos el valor erróneo con su valor dicotómico y lo eliminamos
cleaned_data = cleaned_data[cleaned_data['Código validación'] != 9]

# Realizamos la descomposición de la srie temporal
result_cleaned = seasonal_decompose(cleaned_data['Valor'], model='additive', period=12)

# Generamos la figura para graficar la serie y su descomposición
plt.figure(figsize=(16, 10))

# Serie original
plt.subplot(411)
plt.plot(cleaned_data['Valor'], label='Original')
plt.legend(loc='upper left')
plt.ylabel('ºC')
plt.title('Serie Temporal de Temperatura Media Mensual (Limpia)')

# Tendencia
plt.subplot(412)
plt.plot(result_cleaned.trend, label='Tendencia')
plt.ylabel('ºC')
plt.legend(loc='upper left')

# Estacionalidad
plt.subplot(413)
plt.plot(result_cleaned.seasonal,label='Estacionalidad')
plt.ylabel('ºC')
plt.legend(loc='upper left')

# Residuo
plt.subplot(414)
plt.plot(result_cleaned.resid, label='Residuo')
plt.ylabel('ºC')
plt.xlabel('Fechas')
plt.legend(loc='upper left')
plt.tight_layout()

# Guardamos la figura
plt.savefig('descomposición.png')
plt.show()
```

3.- Separación de datos de entrenamiento y datos de prueba

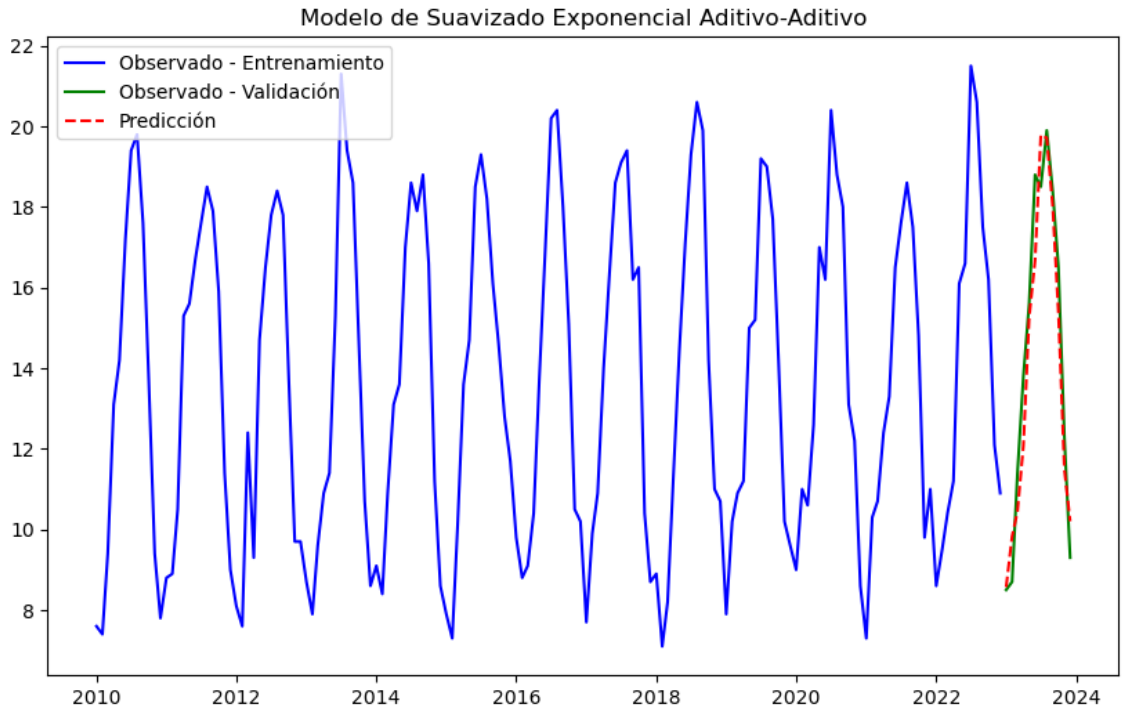
Debido a que la estacionalidad de la serie temporal es anual, reservamos un periodo total a 12 meses, correspondientes al año 2023 para realizar las pruebas de predicción. El código utilizado fue:

```
#Ajustamos los últimos 12 meses para la validación de los modelos

# Actualizamos los datos
validation_data_updated = cleaned_data.tail(12)
training_data_updated = cleaned_data.iloc[:-12]

# Marcamos los límites de ambas series de datos
cutoff_date_training_updated = training_data_updated.index.max()
cutoff_date_validation_updated = validation_data_updated.index.min()

#Comprobamos que todo está correcto
(training_data_updated.tail(), validation_data_updated, cutoff_date_training_updated, cutoff_date_vali
```

4.- Modelo de suavizado exponencial

Dado que la serie muestra una clara estacionalidad y una posible tendencia, el modelo escogido es el de Holt-Winters, indicado para estos casos. Para encontrar el modelo de suavizado exponencial más adecuado, utilizamos el Criterio de Información de Akaike (AIC), que mide la calidad del modelo teniendo en cuenta la complejidad de este. Debemos buscar el valor más bajo para cada modelo evaluado: aditivo-aditivo, aditivo-multiplicativo, multiplicativo-aditivo, multiplicativo-multiplicativo.

El modelo que resultó más adecuado fue el aditivo-aditivo. A continuación, se muestra una representación gráfica de la predicción con este modelo y su comparación con los valores reales de validación:



Podemos observar que la predicción se ajusta bastante los datos observados en el periodo de validación. Los pasos seguidos para la elección de este modelo y su justificación fueron:

- **Evaluación de la Serie:** Antes de ajustar un modelo, se realizó un análisis de la serie temporal para determinar la presencia de tendencias y estacionalidad. Esto se hace para decidir si es adecuado un modelo de suavizado simple, doble o triple.

- **Selección del Modelo:** Seleccionamos un modelo de suavizado exponencial que se ajusta a la serie temporal evaluada. Si la serie mostraba tendencia y estacionalidad, el modelo de suavizado exponencial de Holt-Winters sería apropiado.

- **Estimación de Parámetros:** Los parámetros del modelo se estiman a partir de los datos. Para el suavizado de Holt-Winters, se estiman los niveles, tendencias y factores estacionales que mejor se ajustan a los datos históricos.

- **Suavizado de la Serie:** Se suaviza la serie temporal aplicando el modelo elegido para obtener una curva que refleje la tendencia y la estacionalidad sin la variabilidad aleatoria de los datos originales.

- **Validación del Modelo:** Se valida el modelo comparando las predicciones del modelo con un conjunto de datos de validación, que no se utilizaron para estimar los parámetros del modelo.

  La expresión matemática para el modelo aditivo es la siguiente:

$$\hat{y}_{t+h|t} = l_t + hb_t + s_{t-m+h_m^+}$$

  Dónde:

- $\hat{y}_{t+h|t}$ es el valor predicho h periodos adelante basados en la información hasta el periodo t.
- l_t representa el nivel (componente local) de la serie en el periodo t.
- b_t representa la tendencia de la serie en el periodo t.
- $s_{t-m+h_m^+}$ es el componente estacional, con m siendo la longitud de la temporada y h_m^+ es el índice del componente estacional que se ajusta en función de la predicción a realizar.

  El código utilizado en este apartado fue el siguiente:

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Ajuste del modelo de suavizado exponencial
# Seleccionamos el modelo óptimo probando diferentes combinaciones de tendencia y estacionalidad

# Modelo aditivo para tendencia y estacionalidad
model_add_add = ExponentialSmoothing(training_data_updated['Valor'], trend="add", seasonal="add", seasonal_periods=12).fit()

# Modelo aditivo para tendencia y multiplicativo para estacionalidad
model_add_mul = ExponentialSmoothing(training_data_updated['Valor'], trend="add", seasonal="mul", seasonal_periods=12).fit()

# Modelo multiplicativo para tendencia y aditivo para estacionalidad
model_mul_add = ExponentialSmoothing(training_data_updated['Valor'], trend="mul", seasonal="add", seasonal_periods=12).fit()
```

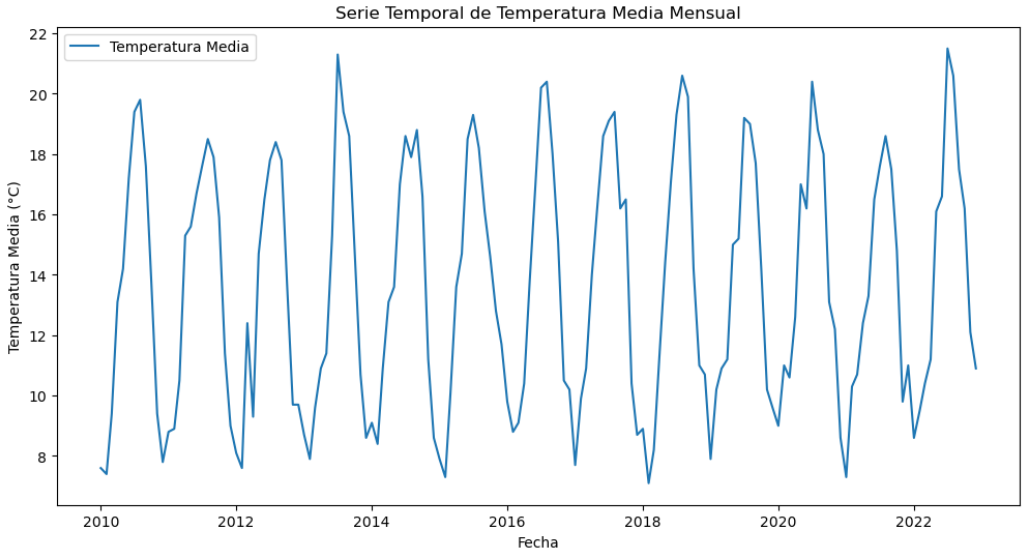
```
# Modelo multiplicativo para tendencia y estacionalidad
model_mul_mul = ExponentialSmoothing(training_data_updated['Valor'], trend="mul", seasonal="mul", seasonal_periods=12).fit()

# Seleccionamos el modelo con el AIC más bajo
models = [model_add_add, model_add_mul, model_mul_add, model_mul_mul]
model_names = ['Aditivo-Aditivo', 'Aditivo-Multiplicativo', 'Multiplicativo-Aditivo', 'Multiplicativo-Multiplicativo']
aic_values = [model.aic for model in models]
best_model_index = aic_values.index(min(aic_values))
best_model = models[best_model_index]
best_model_name = model_names[best_model_index]

# Predicciones para el periodo de validación
preds = best_model.forecast(12)

# Realizamos una representación gráfica de la serie observada y la serie suavizada con predicciones
plt.figure(figsize=(10, 6))
plt.plot(training_data_updated['Valor'], label='Observado - Entrenamiento', color='blue')
plt.plot(validation_data_updated.index, validation_data_updated['Valor'], label='Observado - Validación', color='green')
plt.plot(validation_data_updated.index, preds, label='Predicción', color='red', linestyle='--')
plt.title(f'Modelo de Suavizado Exponencial {best_model_name}')
plt.legend()
plt.show()

(best_model_name, best_model.summary())
```

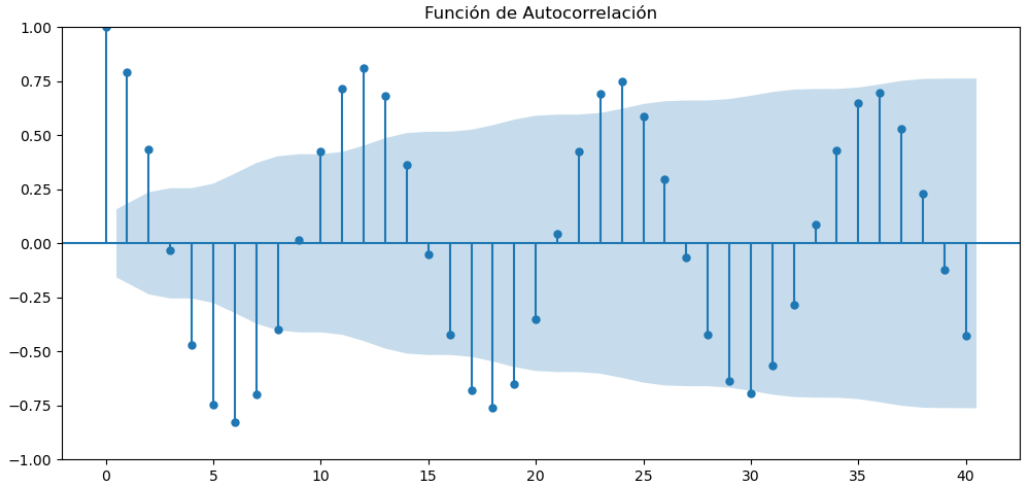
Mostrando los siguientes resultados en pantalla:

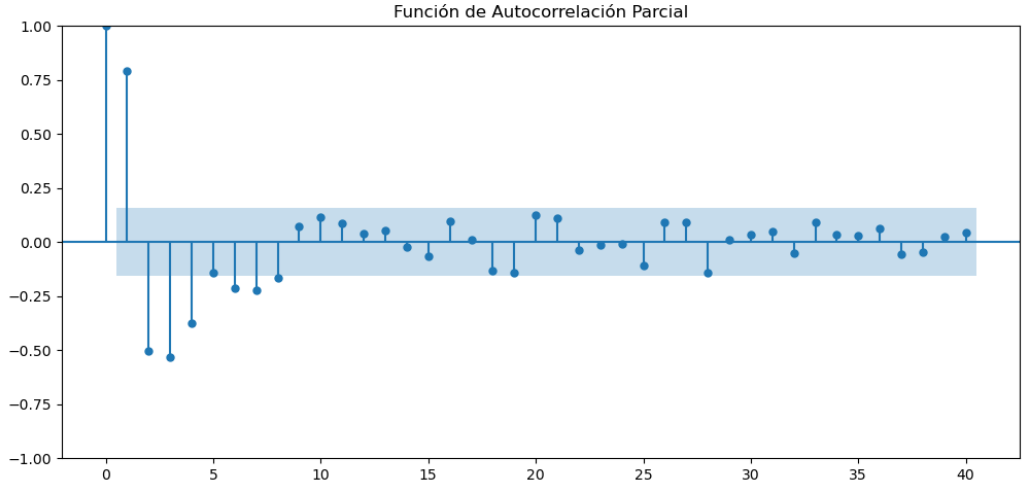
```
('Aditivo-Aditivo',
 <class 'statsmodels.iolib.summary.Summary'>
 """
                       ExponentialSmoothing Model Results
 ==============================================================================
 Dep. Variable:                   Valor   No. Observations:                  155
 Model:             ExponentialSmoothing   SSE                           335.595
 Optimized:                         True   AIC                           151.734
 Trend:                         Additive   BIC                           200.429
 Seasonal:                      Additive   AICC                          156.764
 Seasonal Periods:                  12   Date:                 Mon, 26 Feb 2024
 Box-Cox:                          False   Time:                         00:46:55
 Box-Cox Coeff.:                    None
 ==============================================================================
                          coeff                 code              optimized
 ------------------------------------------------------------------------------
 smoothing_level             1.4901e-08               alpha                 True
 smoothing_trend             1.1825e-08                beta                 True
 smoothing_seasonal           0.2744159               gamma                 True
 initial_level                13.143216                 1.0                 True
 initial_trend                0.0051568                 b.0                 True
 initial_seasons.0           -4.9895223                 s.0                 True
 initial_seasons.1           -4.7525959                 s.1                 True
 initial_seasons.2           -2.3782222                 s.2                 True
 initial_seasons.3           -0.1670612                 s.3                 True
 initial_seasons.4            1.7870902                 s.4                 True
 initial_seasons.5            4.0413445                 s.5                 True
 initial_seasons.6            5.5469075                 s.6                 True
 initial_seasons.7            5.3848659                 s.7                 True
 initial_seasons.8            3.8300186                 s.8                 True
 initial_seasons.9           -0.1981650                 s.9                 True
 initial_seasons.10          -3.4666375                s.10                 True
 initial_seasons.11          -4.7318164                s.11                 True
 ------------------------------------------------------------------------------
 """)
```

5.- Segundo modelo de predicción

Procedemos a representar la serie temporal junto con la función de autocorrelación y autocorrelación parcial:







Vemos que la función de autocorrelación muestra cómo los valores de la serie se correlacionan con sus propios retrasos. Cuando la ACF muestra una disminución gradual, esto indica que podemos usar un modelo ARIMA. Los picos significativos en intervalos graduales nos muestran estacionalidad.

La PACF o función de autocorrelación parcial muestra la correlación entre los valores de la serie y sus retrasos después de eliminar los efectos de los retrasos intermedios. Los picos nos ayudan a identificar el orden del componente AR en un modelo ARIMA.

Tras analizar ambas funciones, consideramos que el modelo adecuado para ajustar es el **ARIMA estacional**, también conocido como SARIMA (Seasonal ARIMA). Los parámetros SARIMA son (p, d, q) para la parte no estacional y (P, D, Q, m) para la parte estacional, donde:

- p y P se refieren al número de términos autorregresivos.
- d y D al número de diferenciaciones necesarias para alcanzar la estacionariedad.
- q y Q al número de términos de media móvil.
- m representa la periodicidad de la estacionalidad (en este caso 12).

Para ajustar el modelo SARIMA existe una librería llamada pmdarima, cuya función auto_arima calcula los parámetros. Tras ejecutar la función, obtenemos los parámetros: **SARIMAX (1, 1, 0) (0, 1, 1, 12).** El código utilizado para esto fue:

```
from pmdarima import auto_arima

# Utilizamos auto_arima para encontrar el mejor modelo ARIMA estacional automáticamente
auto_arima_model = auto_arima(training_data_updated['Valor'], start_p=0, start_q=0,
                              max_p=5, max_q=5, m=12,
                              seasonal=True, d=1, D=1, trace=True,
                              error_action='ignore',
                              suppress_warnings=True,
                              stepwise=True)

auto_arima_model.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(1,1,1)[12]             : AIC=577.006, Time=0.10 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=639.792, Time=0.02 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=581.076, Time=0.06 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=inf, Time=0.31 sec
 ARIMA(0,1,0)(0,1,1)[12]             : AIC=575.803, Time=0.08 sec
 ARIMA(0,1,0)(0,1,2)[12]             : AIC=576.867, Time=0.21 sec
 ARIMA(0,1,0)(1,1,0)[12]             : AIC=598.959, Time=0.05 sec
 ARIMA(0,1,0)(1,1,2)[12]             : AIC=578.774, Time=0.53 sec
 ARIMA(1,1,0)(0,1,1)[12]             : AIC=558.616, Time=0.11 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=616.033, Time=0.02 sec
 ARIMA(1,1,0)(1,1,1)[12]             : AIC=560.271, Time=0.14 sec
 ARIMA(1,1,0)(0,1,2)[12]             : AIC=560.174, Time=0.24 sec
 ARIMA(1,1,0)(1,1,2)[12]             : AIC=inf, Time=1.01 sec
 ARIMA(2,1,0)(0,1,1)[12]             : AIC=559.441, Time=0.13 sec
 ARIMA(1,1,1)(0,1,1)[12]             : AIC=inf, Time=0.40 sec
 ARIMA(2,1,1)(0,1,1)[12]             : AIC=inf, Time=0.61 sec
 ARIMA(1,1,0)(0,1,1)[12] intercept   : AIC=560.616, Time=0.13 sec

Best model:  ARIMA(1,1,0)(0,1,1)[12]
Total fit time: 4.147 seconds
```

Con los siguientes resultados:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	155
Model:	SARIMAX(1, 1, 0)x(0, 1, [1], 12)	**Log Likelihood**	-276.308
Date:	Mon, 26 Feb 2024	**AIC**	558.616
Time:	00:44:46	**BIC**	567.484
Sample:	0	**HQIC**	562.219
	- 155		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.3572	0.078	-4.556	0.000	-0.511	-0.204
ma.S.L12	-0.7257	0.083	-8.789	0.000	-0.887	-0.564
sigma2	2.6903	0.288	9.347	0.000	2.126	3.254

Ljung-Box (L1) (Q):	0.17	**Jarque-Bera (JB):**	6.37
Prob(Q):	0.68	**Prob(JB):**	0.04
Heteroskedasticity (H):	0.56	**Skew:**	-0.29
Prob(H) (two-sided):	0.05	**Kurtosis:**	3.86

La elección del modelo SARIMA sobre otras opciones se justifica debido a la flexibilidad de este modelo. Es capaz de adaptarse a diversas características de la serie, como tendencias no lineales y estacionalidad, por lo que es una buena herramienta para pronósticos precisos.

Diferenciación entre SARIMA y SARIMAX: Aunque en el código esté utilizando la librería SARIMAX, realmente estoy aplicando el modelo SARIMA o ARIMA estacional. La principal diferencia es que el modelo SARIMAX también incluye variables exógenas, haciendo el análisis más preciso aún. Como en este estudio estamos utilizando únicamente la variable temperatura, sólo podemos utilizar el ARIMA estacional.

6.- Expresión algebraica del modelo ajustado

A continuación, vamos a expresar algebraicamente el modelo utilizado anteriormente:

Parte no estacional (ARIMA (1,1,0)):

AR (1): El coeficiente para el término autoregresivo de primer orden es -0,3572, indicando una relación inversa con el valor anterior en la serie.

$$\Phi(L)_{y_t} = -0{,}3572 y_{t-1} + \epsilon_t$$

donde Φ(L) es el operador polinómico para el término AR(1), y_t es el valor de la serie en el tiempo t y ϵ_t es el término de error.

Parte estacional ((0, 1, 1, 12)):

Diferenciación estacional (D=1): Se aplica una diferenciación de orden 1 a la serie cada 12 periodos para manejar la estacionalidad.

MA estacional de primer orden (MA.S. L12): El coeficiente para el término de media móvil estacional es -0,7257, indicando un ajuste inverso basado en el error del término estacional previo.

$$\Theta(L^{12})\Delta_{12}y_t = -0{,}7257\epsilon_{t-12} + \epsilon_t$$

donde $\Theta(L^{12})$ es el operador polinómico para el término MA estacional, Δ_{12} representa la diferenciación estacional, y ϵ_{t-12} es el término de error en el tiempo $t - 12$.

Varianza del término de error (σ^2):

La varianza estimada del término de error es 2,6903, proporcionando una medida de la variabilidad de los errores del modelo.

Este modelo con sus parámetros estimados nos ofrece una buena herramienta para la predicción y análisis de series con patrones estacionales claros.

7.- Predicción e intervalos de confianza

Tras el ajuste del modelo y su expresión algebraica, realizamos predicciones siguiendo los siguientes pasos:

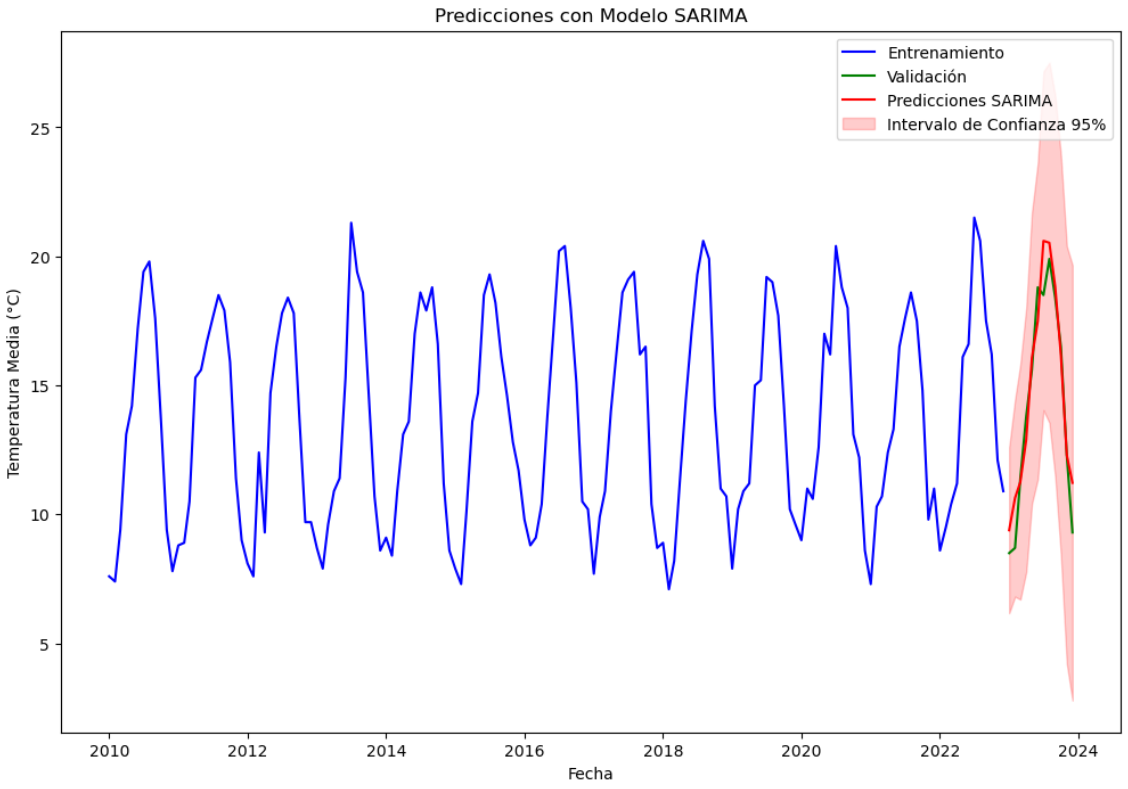
1. **Predicciones:** Generamos predicciones para el último año (el periodo reservado para validación), incluyendo los intervalos de confianza del 95%.

2. **Evaluación Numérica:** Calculamos el MSE y el MAE para cuantificar la precisión de las predicciones comparadas con los valores reales de validación.

3. **Representación Gráfica:** Graficamos tanto los datos de entrenamiento como los de validación, junto con las predicciones y los intervalos de confianza, para visualizar el desempeño del modelo.

La longitud del periodo de predicción es de 12 meses, correspondiente al ciclo estacional completo. De esta manera podremos obtener una visión completa de la estacionalidad esperada. El intervalo de confianza será del 95% significa que con un 95 % de probabilidad, no podremos descartar la predicción.

El resultado es el siguiente:



Por lo que se puede observar, obtenemos un resultado satisfactorio para la predicción de temperaturas en el año 2023. Las métricas de error fueron:

- **MSE** = 1,3424
- **MAE** = 0,9211

El código utilizado en este apartado fue el siguiente:

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Ajustamos el modelo SARIMA con los parámetros proporcionados
model_sarimax = SARIMAX(training_data_updated['Valor'], order=(1, 1, 0), seasonal_order=(0, 1, 1, 12))
model_sarimax_fitted = model_sarimax.fit()

# Realizamos predicciones para el periodo de validación
preds_sarimax = model_sarimax_fitted.get_forecast(steps=12)
preds_sarimax_mean = preds_sarimax.predicted_mean
preds_sarimax_conf_int = preds_sarimax.conf_int()

# Cálculo de métricas para el modelo SARIMA
mse_sarimax = mean_squared_error(validation_data_updated['Valor'], preds_sarimax_mean)
mae_sarimax = mean_absolute_error(validation_data_updated['Valor'], preds_sarimax_mean)

# Representación gráfica de las predicciones y los valores reales
plt.figure(figsize=(12, 8))
plt.plot(training_data_updated.index, training_data_updated['Valor'], label='Entrenamiento', color='blue')
plt.plot(validation_data_updated.index, validation_data_updated['Valor'], label='Validación', color='green')
plt.plot(validation_data_updated.index, preds_sarimax_mean, label='Predicciones SARIMA', color='red')
plt.fill_between(validation_data_updated.index, preds_sarimax_conf_int.iloc[:, 0], preds_sarimax_conf_int.iloc[:, 1], color='red'
plt.title('Predicciones con Modelo SARIMA')
plt.xlabel('Fecha')
plt.ylabel('Temperatura Media (°C)')
plt.legend()
plt.show()

(mse_sarimax, mae_sarimax)
```

8.- Comparación de modelos

Ahora que tenemos los ajustes y predicciones de ambos modelos, podemos realizar una comparación para evaluar cuál de ellos resulta más efectivo. Para ello, utilizaremos primero métricas numéricas específicas que cuantifican el error de las predicciones. Estas incluyen el Error Cuadrático Medio (MSE) y el Error Absoluto Medio (MAE). Estas métricas proporcionan una manera de calcular la precisión de los modelos de manera numérica. Como ya hemos calculado el MSE y el MAE del modelo SARIMA, sólo debemos calcular sus respectivos en el modelo de suavizado exponencial.

Comparación Numérica de los Modelos:

Para el modelo de suavizado exponencial, los valores de las métricas son:

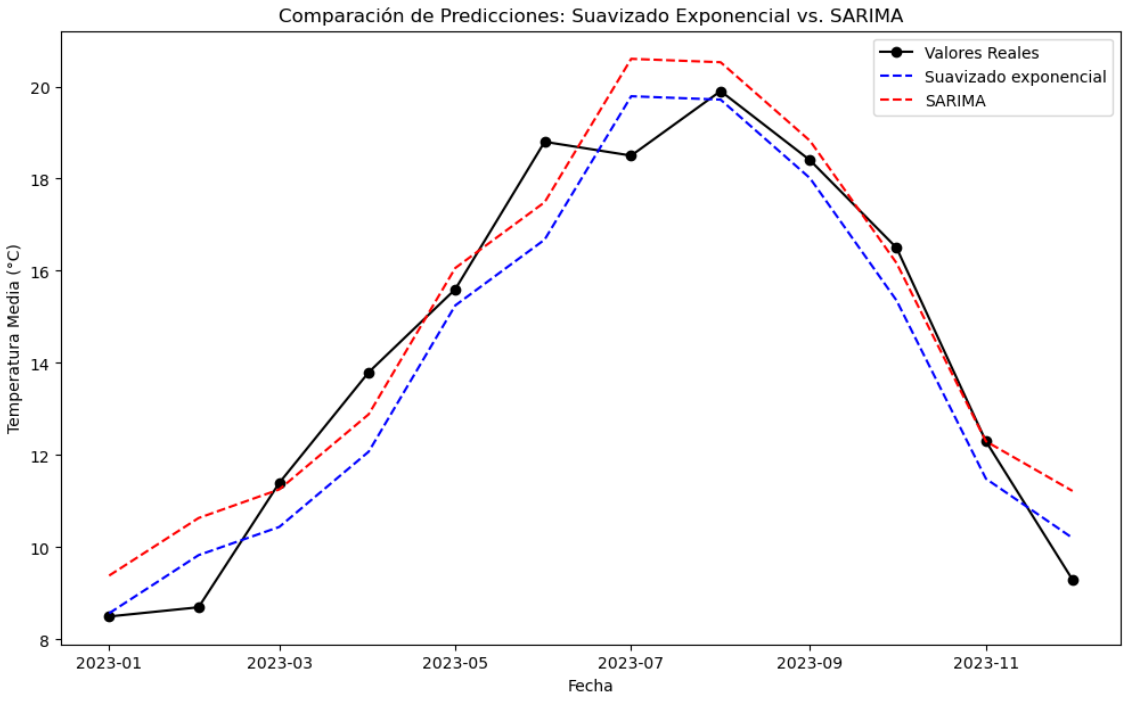
- MSE (Suavizado Exponencial): 1,20
- MAE (Suavizado Exponencial): 0,92

Para el modelo SARIMAX (1, 1, 0) (0, 1, 1, 12), basado en tu ejecución más reciente, los valores de las métricas son:

- MSE (SARIMA): 1,3424
- MAE (SARIMA): 0,9211

Ambos modelos tienen un rendimiento similar en términos de precisión, con el modelo de suavizado exponencial teniendo un MSE ligeramente menor, lo que indica un mejor ajuste general a los datos de validación. Sin embargo, ambas métricas son prácticamente igual de precisas.

Comparación Gráfica de los Modelos:



Podemos ver que ambos modelos se ajustan bastante bien a los valores de validación reales. El modelo de suavizado exponencial se ajusta mejor a los datos reales en 7/12 meses, mientras que el modelo SARIMA se ajusta mejor en 5/12 meses. El código utilizado en este apartado fue:

```
import matplotlib.pyplot as plt

# Generamos la figura para graficar las tres series
plt.figure(figsize=(12, 7))
plt.plot(validation_data_updated.index, validation_data_updated['Valor'], label='Valores Reales', color='black', marker='o')
plt.plot(validation_data_updated.index, preds, label='Suavizado exponencial', color='blue', linestyle='--')
plt.plot(validation_data_updated.index, preds_mean, label='SARIMA', color='red', linestyle='--')
plt.title('Comparación de Predicciones: Suavizado Exponencial vs. SARIMA')
plt.xlabel('Fecha')
plt.ylabel('Temperatura Media (°C)')
plt.legend()
plt.show()
```

Conclusiones Principales

Efectividad de los Modelos: Tanto el modelo de suavizado exponencial como el modelo SARIMA (y SARIMAX sin variables exógenas) han demostrado ser efectivos para modelar la serie temporal de temperatura. Cada uno de estos modelos tiene sus fortalezas y se ajusta de manera diferente a los datos, capturando la tendencia y la estacionalidad de la serie.

Comparación de Modelos: La comparación numérica y visual entre los modelos de suavizado exponencial y SARIMA reveló que ambos modelos proporcionan predicciones precisas, con el modelo de suavizado exponencial mostrando un ligero mejor ajuste según las métricas de MSE y MAE. Sin embargo, el modelo SARIMA ofrece una mayor flexibilidad para modelar dinámicas complejas, incluyendo la estacionalidad.

Importancia de la Estacionalidad: El análisis destacó la importancia de considerar la estacionalidad en la serie temporal. El modelo SARIMA, ajustado para incluir componentes estacionales, permitió capturar y predecir con éxito las variaciones estacionales en la temperatura, lo que subraya la relevancia de utilizar modelos que puedan manejar patrones estacionales para series temporales como esta.

Selección de Modelos y Parámetros: La selección cuidadosa de los modelos y la optimización de sus parámetros son cruciales para obtener predicciones precisas. La función 'auto_arima' de 'pmdarima' y el análisis de los componentes de la serie temporal fueron herramientas valiosas en este proceso, permitiendo identificar el mejor ajuste para los datos.

Uso de Intervalos de Confianza: Los intervalos de confianza generados para las predicciones aportan una visión crucial sobre la incertidumbre asociada a estas. Estos intervalos son fundamentales para la toma de decisiones informadas, especialmente en aplicaciones prácticas donde el riesgo y la incertidumbre deben ser gestionados.

Flexibilidad y Adaptabilidad: La capacidad para adaptar el análisis y el modelado a los resultados de la exploración de datos inicial y a la evaluación de los modelos es clave para el éxito en el análisis de series temporales. Las herramientas y métodos utilizados ofrecen la flexibilidad necesaria para ajustar los modelos a medida que se adquiere una comprensión más profunda de la serie.

Para futuros análisis, sería recomendable incluir variables exógenas, utilizando el modelo SARIMAX, si se dispone de datos que puedan ser relevantes e influyan en la temperatura. Esto permitirá reducir aún más los errores. Este análisis proporciona una base sólida para la comprensión y predicción de series temporales complejas, demostrando la utilidad de una variedad de técnicas de modelado y la importancia de una selección cuidadosa de modelos y parámetros. La capacidad para predecir con precisión la temperatura media mensual en Santiago de Compostela tiene aplicaciones valiosas en planificación urbana, gestión de recursos y preparación para eventos climáticos extremos.