Tarea Series Temporales

Alvaro Vilela Nova

Febrero 2024

1. Introducción

La serie que vamos a analizar ha sido obtenida utilizando la API Weather Underground. Contiene información diaria sobre el clima en la ciudad de Delhi entre 1 de enero de 2013 y el 24 de abril de 2017. Las variables que encontramos en los datos son los siguientes:

- **meantemp**: la temperatura media a lo largo del día.
- **humidity**: la humedad ese día.
- **wind_speed**: la velocidad del viento ese día.
- **meanpressure**: la presión atmosférica media a lo largo del día.

Nuestro objetivo va a ser predecir el valor de la temperatura media cada día en un espacio de tiempo de 31 días (y si fuese posible hasta un año). Vamos a reservar los datos del último mes registrado para comprobar cómo el modelo predice datos que ya conocemos.

Importaremos los datos y veremos cómo son utilizando el siguiente código:

```
os.chdir(r'C:\Users\alvar\Desktop\Master Big Data\Series Temporales\
    ↪ Tarea')
df = pd.read_csv(r'DailyDelhiClimateTrain.csv')

df['date'] = pd.to_datetime(df['date']).dt.date
df['date'] = pd.to_datetime(df['date'])
df.set_index('date',inplace=True)
df
```

date	meantemp	humidity	wind_speed	meanpressure
2013-01-01	10.000000	84.500000	0.000000	1015.666667
2013-01-02	7.400000	92.000000	2.980000	1017.800000
2013-01-03	7.166667	87.000000	4.633333	1018.666667
2013-01-04	8.666667	71.333333	1.233333	1017.166667
2013-01-05	6.000000	86.833333	3.700000	1016.500000
...	...	...	...	...
2016-12-28	17.217391	68.043478	3.547826	1015.565217
2016-12-29	15.238095	87.857143	6.000000	1016.904762
2016-12-30	14.095238	89.666667	6.266667	1017.904762
2016-12-31	15.052632	87.000000	7.325000	1016.100000
2017-01-01	10.000000	100.000000	0.000000	1016.000000

Figura 1

Crearemos una nueva serie con los datos de la temperatura y otra con los datos de la temperatura sin el último mes. No tendremos en cuenta el último dato, pues parece que los valores están establecidos a modo de placeholder.

```
total_temp = df['meantemp'].iloc[:-1].copy()
temp = total_temp.iloc[:-31]
```

2. Representación gráfica y descomposición

Vamos a representar la temperatura media a lo largo del tiempo en una gráfica.

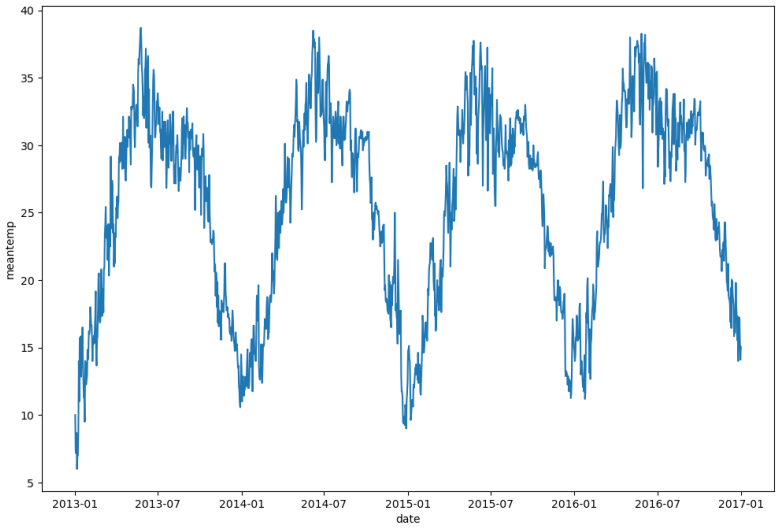


Figura 2: Temperatura por día

Vamos a hacer la descomposición estacional de la serie. Podemos apreciar a simple vista que hay una clara estacionalidad con un periodo de un año.

```
result = seasonal_decompose(total_temp, model = 'add',period=365)
```

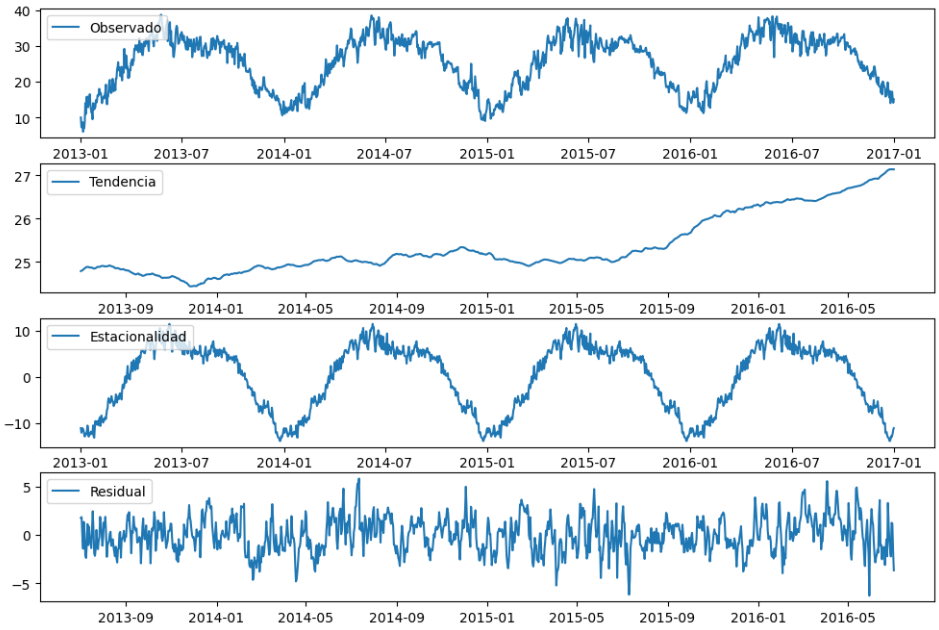


Figura 3: Descomposición de la serie temporal

Es bastante evidente que los datos van a ser estacionales. Vamos a comprobar si podemos caracterizar la estacionalidad utilizando los meses del año. Para ello agrupamos los datos por meses y establecemos como valor de cada mes la media de las temperaturas.

```
total_temp_monthly = total_temp.resample('M').mean()
result = seasonal_decompose(total_temp_monthly, model = 'multiplicative
    ↪ ',period=12)
```

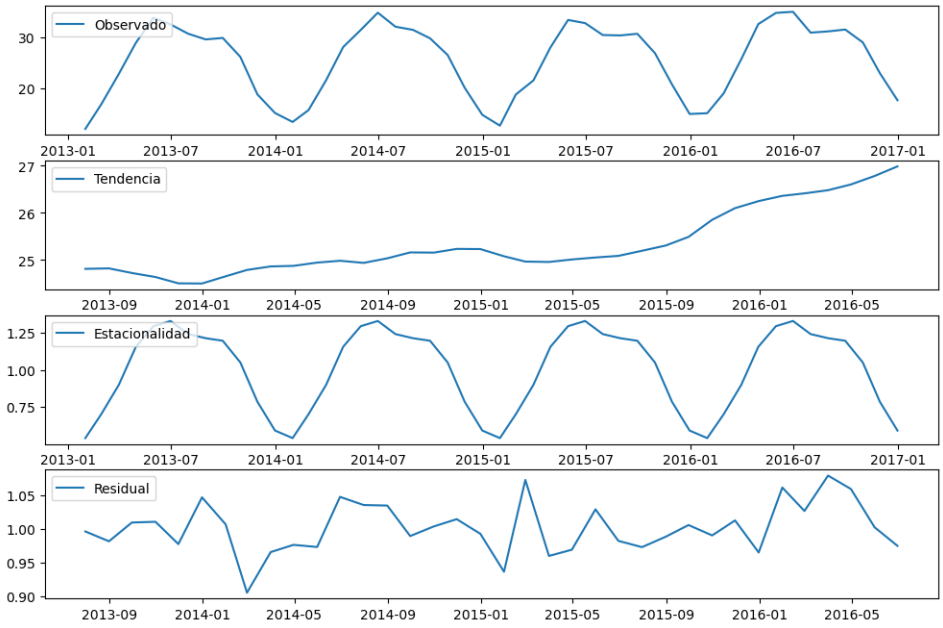


Figura 4: Descomposición de la serie temporal por mes

Podemos comprobar que los datos son efectivamente estacionales y que se corresponde con un periodo de 12 meses. Además, tienen una tendencia positiva, se comprueba utilizando el test de Dickey-Fuller, que nos da un p-valor de 0.15.

```
# H_0: la serie no es estacionaria
# H_1: la serie es estacionaria
adfuller(temp)[1]
```

3. Modelo de suavizado exponencial

Probaremos cuatro tipos de modelos de suavizado exponencial, además, de crear un modelo de suavizado exponencial para la serie diferenciada, intentando así eliminar la tendencia de los datos. Los códigos para cada modelo son los siguientes:

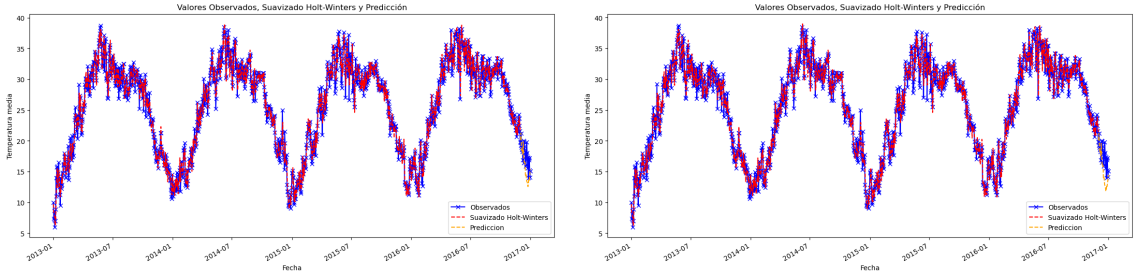
```
hw_mm = sm.tsa.ExponentialSmoothing(temp, trend='mul', seasonal='mul',
    ↪ seasonal_periods=365).fit()
hw_aa = sm.tsa.ExponentialSmoothing(temp, trend='add', seasonal='add',
    ↪ seasonal_periods=365).fit()
hw_ma = sm.tsa.ExponentialSmoothing(temp, trend='mul', seasonal='add',
    ↪ seasonal_periods=365).fit()
hw_am = sm.tsa.ExponentialSmoothing(temp, trend='add', seasonal='mul',
    ↪ seasonal_periods=365).fit()

# Modelo para serie diferenciada
```

```python
def subtract_previous(series):
    """
    Devuelve la serie diferenciada de una serie.

    Input parameters:
    series: Pandas Series

    Output:
    result_series: Pandas Series de la serie diferenciada
    """
    series2 = series.copy()
    previous_values = series2.shift(1)  # Shift the series by 1 to get
        ↪ the previous values
    result_series = series - previous_values
    return result_series

temp_dif = subtract_previous(temp) # Esta es la serie diferenciada
temp_dif = temp_dif.iloc[1:]

hw_dif = sm.tsa.ExponentialSmoothing(temp_dif, trend='add', seasonal='
    ↪ add', seasonal_periods=365).fit()
```

Modelo	AIC	BIC	SSE
mm	1727.164	3670.107	2855.929
aa	1716.679	3659.622	2835.066
ma	1716.227	3659.171	2834.171
am	1727.386	3670.329	2856.373
dif	1763.453	3706.138	2928.752

Cuadro 1: Estadísticos de los modelos

Todos los modelos tienen resultados muy parecidos, en todo caso elegiríamos entre utilizar el modelo aa o el modelo ma. Vamos a decidir entre estos dos fijándonos en la predicción que hacen del último mes que conocemos.



(a) Modelo hw_aa

(b) Modelo hw_ma

Figura 5: Comparación de la predicción de los modelos

La predicción entre ambos es muy cercana, pero parece que en el caso del modelo hw_aa se adecúa algo más así que escogeremos este.

Vamos a aplicar este modelo a los datos completos y hacer una predicción de la temperatura para cada día del siguiente mes.

```
# Aplicar suavizado Holt-Winters.
hw_aa = sm.tsa.ExponentialSmoothing(total_temp, trend='add', seasonal='
    ↪ add', seasonal_periods=365).fit()

# Obtener predicciones para 14 dias
pred_hw_aa = hw_aa.forecast(steps=31)
```

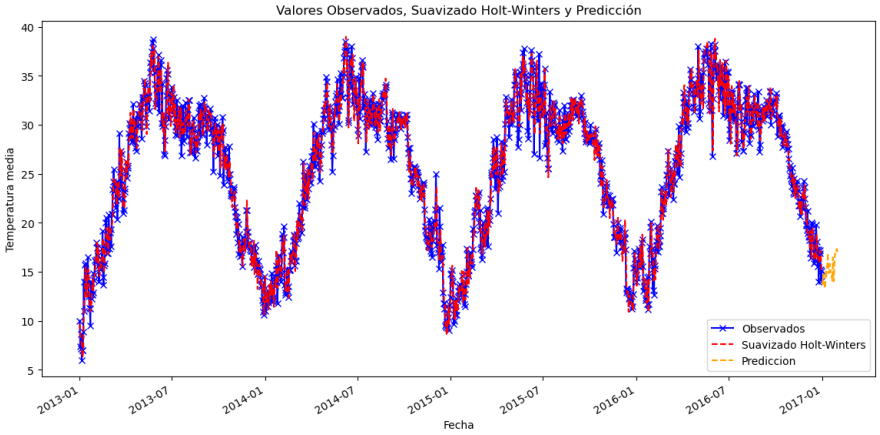


Figura 6: Predicción con modelo de suavizado exponencial

La expresión del modelo teórico de Holt-Winters aditivo es:

$$X_t = (L_t + b_t) + S_t + z_t$$

donde cada elemento se calcula de la siguiente manera

$$L_t = \alpha(x_t - S_{t-s}) + (1-\alpha)(L_{t-1} + b_{t-1})$$

$$b_t = \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1}$$

$$S_t = \gamma(x_t - L_t) + (1-\gamma)S_{t-s},$$

donde s es la cantidad de índices estacionales (o la longitud de un periodo). Los valores iniciales se calculan tal que:

$$L_0 = \frac{x_1 + \ldots + X_s}{s}$$

$$b_0 = \frac{1}{s}\left(\frac{(x_{s+1} - x_1)}{s} + \frac{(x_{s+2} - x_2)}{s} + \ldots + \frac{(x_{s+s} - x_s)}{s}\right)$$

$$S_1 = x_1 - L_0,\ S_2 = x_2 - L_0, \ldots, S_s = x_s - L_0$$

La expresión usada para predecir valores futuros es:

$$\hat{x}_{n+m} = (L_n + b_n m) + S_{n+m-s},\ m \geq 1$$

Los valores de estos coeficientes en nuestro caso pueden encontrarse utilizando el método .summary() en nuestro modelo. No incluyo el output porque es demasiado largo.

4. Modelos con autocorrelación

Vamos a utilizar los gráficos de auto-correlación y auto-correlación parcial para determinar los coeficientes que utilizaremos en el ARIMA. También utilizaremos la función auto-arima() para encontrar un modelo ARIMA posiblemente mejor que el que determinemos manualmente. Inicialmente intenté hacer un modelo SARIMA con los datos diarios, sin embargo, no pude completarlo pues añadir 365 índices de estacionalidad es demasiado para el módelo y rápidamente se agota la memoria.

Vamos a representar las funciones de autocorrelación para los datos diarios de todas maneras con el objetivo de ilustrar cómo interpretar las funciones en este caso particular.

```
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = plot_acf(temp,lags=50,ax=ax1)
ax2 = fig.add_subplot(212)
fig = plot_pacf(temp,lags=50,ax=ax2)
```

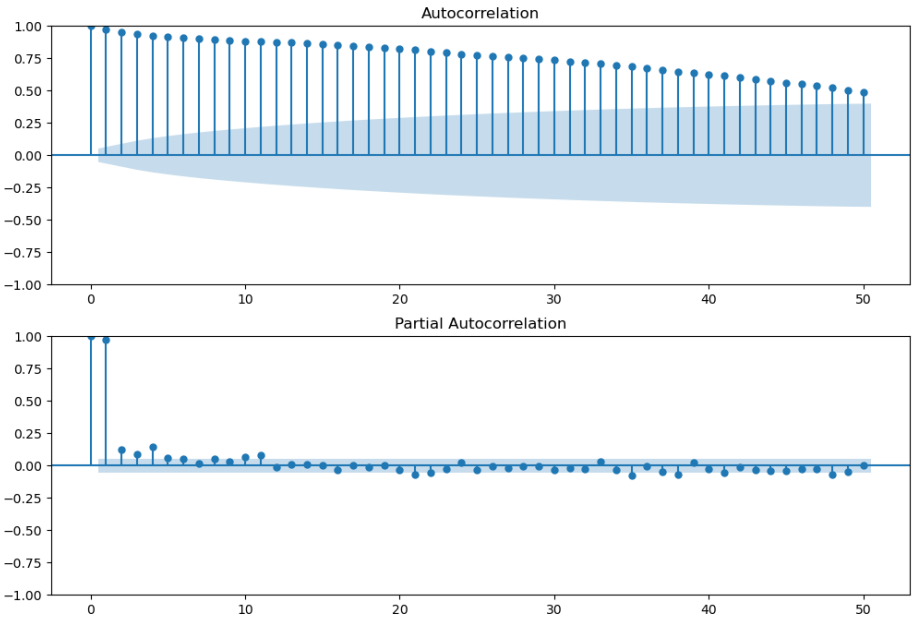


Figura 7: Auto-correlación y auto-correlación parcial

La función de auto-correlación de los datos diarios de temperatura disminuye lentamente, esto lo interpretamos como que los datos no son estacionarios, esto ya lo habíamos comprobado más arriba usando el test de Dickey-Fuller. También podemos comprobar lo mismo para la serie diferenciada.

```
# Test de Dickey-Fuller y KPSS
adfuller(temp_dif)[1] # 2.564669665816475e-29
kpss(temp_dif)[1] # 0.1 (p-valor mayor que este)

# Funciones de auto-correlacion
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = plot_acf(temp_dif,lags=50,ax=ax1)
ax2 = fig.add_subplot(212)
fig = plot_pacf(temp_dif,lags=50,ax=ax2)
```

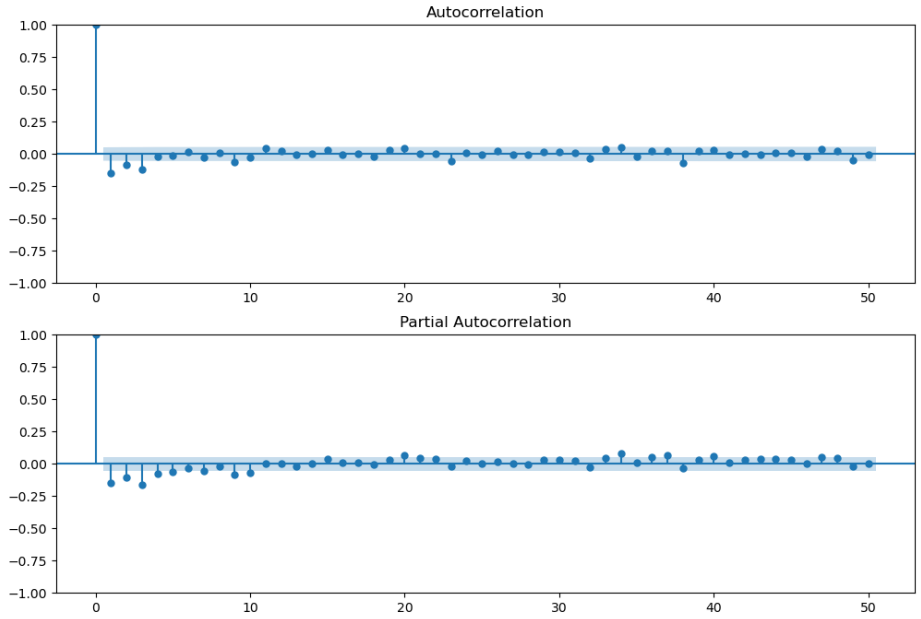


Figura 8: Auto-correlación y auto-correlación parcial de la serie diferenciada

Podemos comprobar que los datos diarios de temperatura diferenciados sí son estacionarios y que podríamos utilizar un modelo SARIMA para modelar la serie temporal. Como ya comentamos antes, los datos diarios tienen dificultad para ser modelados de esta manera por la cantidad de índices de estacionalidad que habría que calcular.

A pesar de esto, una solución que sería interesante explorar es utilizar una serie de Fourier para modelar los índices de estacionalidad y después utilizar esta serie como la componente de estacionalidad del modelo SARIMA. No estoy seguro de cómo implementar esto en Python, pero en R es relativamente sencillo de implementar con un par de líneas de código. Con esta solución podríamos implementar el modelo SARIMA con la serie temporal de los datos diarios.

Procedemos utilizando los datos de la media de temperatura mensual para realizar nuestro análisis. Vamos a analizar su serie diferenciada de primer orden para comprobar si podríamos aplicar un modelo SARIMA.

```
monthly_dif = subtract_previous(total_temp_monthly) # Esta es la serie
    ↪ diferenciada
monthly_dif = monthly_dif.iloc[1:-6]

plt.figure(figsize=(12,8))
sns.lineplot(monthly_dif)
plt.show()
```

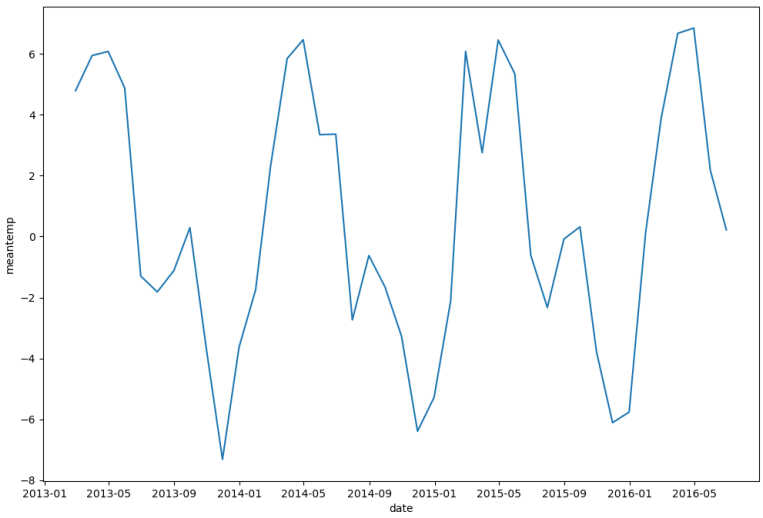


Figura 9: Serie temporal de la media de temperatura por mes diferenciada

```
# Test de Dickey-Fuller y KPSS
adfuller(monthly_dif)[1] # 1.1367724819218345e-09
kpss(monthly_dif)[1] # 0.1

fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = plot_acf(monthly_dif,lags=20,alpha = 0.05,ax=ax1)
ax2 = fig.add_subplot(212)
fig = plot_pacf(monthly_dif,lags=18,alpha = 0.05,ax=ax2)
```

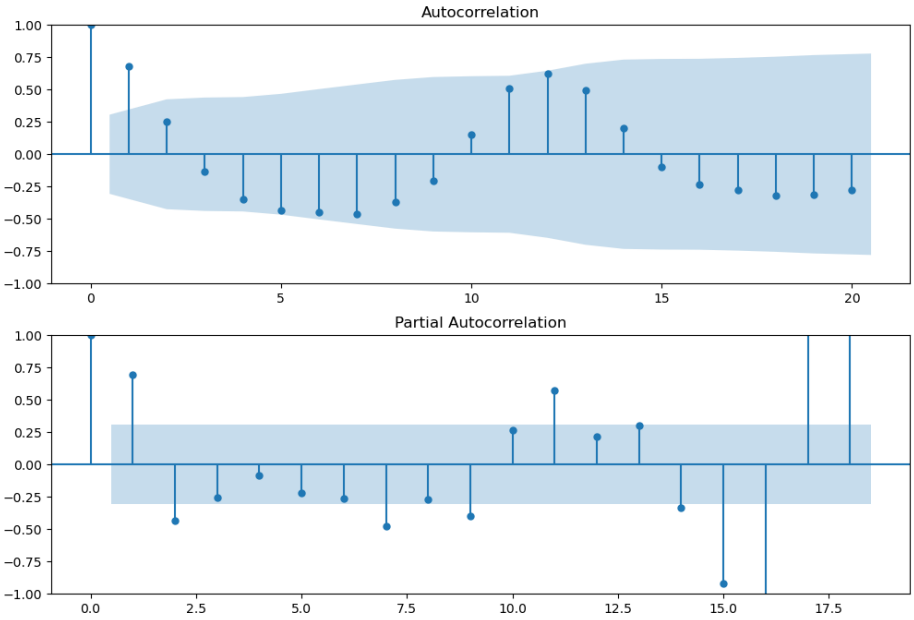


Figura 10: Auto-correlación y auto-correlación parcial de la serie de meses diferenciada

Es claro que la serie es estacionaria, tanto las funciones de auto-correlación como los tests nos lo aseguran. Un modelo que tenga en cuenta la auto-correlación sería apropiado, el correlograma muestra una función de forma sinusoidal, que nos indica que un modelo AR es adecuado. Ambas funciones muestran además un decrecimiento rápido y los valores se anulan rápidamente. Concluimos así que un modelo SARIMA (pues los datos son también estacionales) sería adecuado.

Vamos a probar un modelo SARIMA con los coeficientes escogidos de manera manual.

```
modelo = SARIMAX(endog=total_temp_monthly[:-12], order=(0,1,0),
    ↪ seasonal_order=(0,1,0,12))
modelo_res = modelo.fit(disp=None)
model1_pred = modelo_res.get_prediction(start = total_temp_monthly.
    ↪ index[-12],end = total_temp_monthly.index[-1], dynamic = True)

# Crear un grafico con matplotlib
plt.figure(figsize=(14, 6))
plt.plot(total_temp_monthly.index, total_temp_monthly,
         label='Observados', marker='x', linestyle='-', color='blue')
plt.plot(total_temp_monthly[:-12].index, modelo_res.fittedvalues,
         label='SARIMA fit', linestyle='--', color='red')
plt.plot(model1_pred.predicted_mean.index, model1_pred.predicted_mean,
         label='Forecast', linestyle='--', color='green')
plt.xlabel('Fecha')
plt.ylabel('Temperatura media del mes')
plt.title('Serie, modelo SARIMA y prediccion de 12 meses')
plt.legend()
```

```
plt.xticks(rotation=30, ha='right')
plt.show()
```

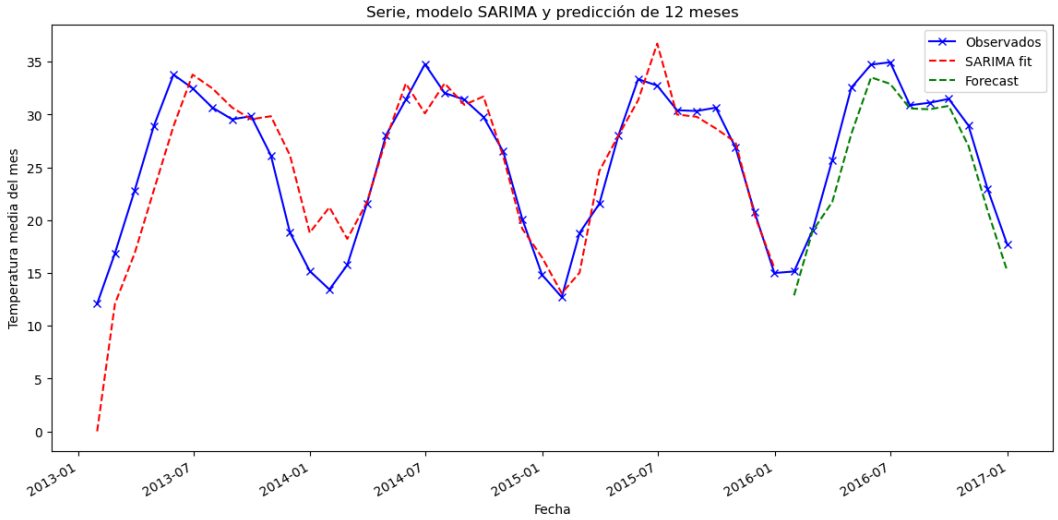


Figura 11: SARIMA con coeficientes manuales

Ahora vamos a probar utilizando la función auto_arima() del paquete pmdarima. Los parámetros utilizados son los siguientes:

```
modelo_auto = auto_arima(
y = total_temp_monthly[:-12],
start_p = 0,
start_q = 0,
max_p = 3,
max_q = 3,
seasonal = True,
test = 'adf',
m = 12, # periodicidad de la estacionalidad
d = 1,
D = None, # El algoritmo determina 'D'
trace = True,
error_action = 'ignore',
suppress_warnings = True,
stepwise = True)
```

Tras jugar un poco con los parámetros de la función he decidido escoger $d = 1$ pues daba mejores resultados que permitir que el algoritmo de la función escogiese el valor.

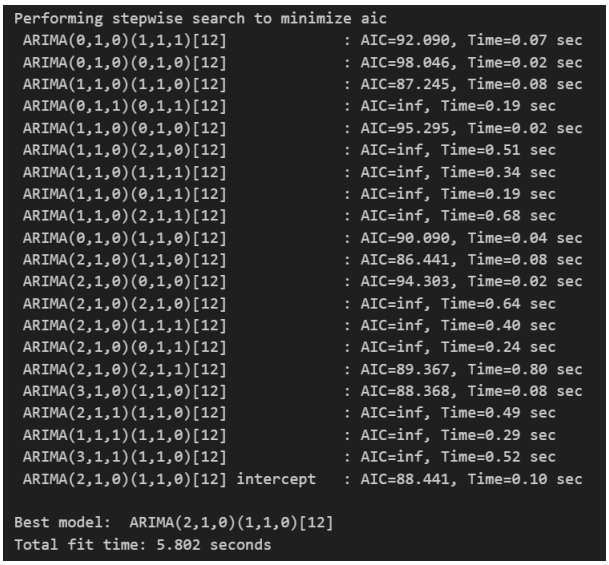


Figura 12: Output auto_arima

Comprobamos que el modelo ganador es el que tiene como parámetros $p = 2$, $d = 1$, $q = 0$, $P = 1$, $D = 1$, $Q = 0$ y $s = 12$ que son una elección de coeficientes más razonable cuando nos fijamos en los correlogramas. Vemos como se adapta a los datos con los que ha entrenado y cómo de cercana es a los datos del último año es la predicción.

```
fitted_auto = modelo_auto.fit(y=total_temp_monthly[:-12], disp= None)
auto_pred = fitted_auto.predict(12)

# Crear un grafico con matplotlib
plt.figure(figsize=(14, 6))
plt.plot(total_temp_monthly.index, total_temp_monthly,
         label='Observados', marker='x', linestyle='-', color='blue')
plt.plot(total_temp_monthly[:-12].index, fitted_auto.fittedvalues(),
         label='SARIMA fit', linestyle='--', color='red')
plt.plot(auto_pred.index, auto_pred,
         label='Forecast', linestyle='--', color='green')
plt.xlabel('Fecha')
plt.ylabel('Temperatura media del mes')
plt.title('Serie, modelo SARIMA y prediccion de 12 meses')
plt.legend()
plt.xticks(rotation=30, ha='right')
plt.show()
```

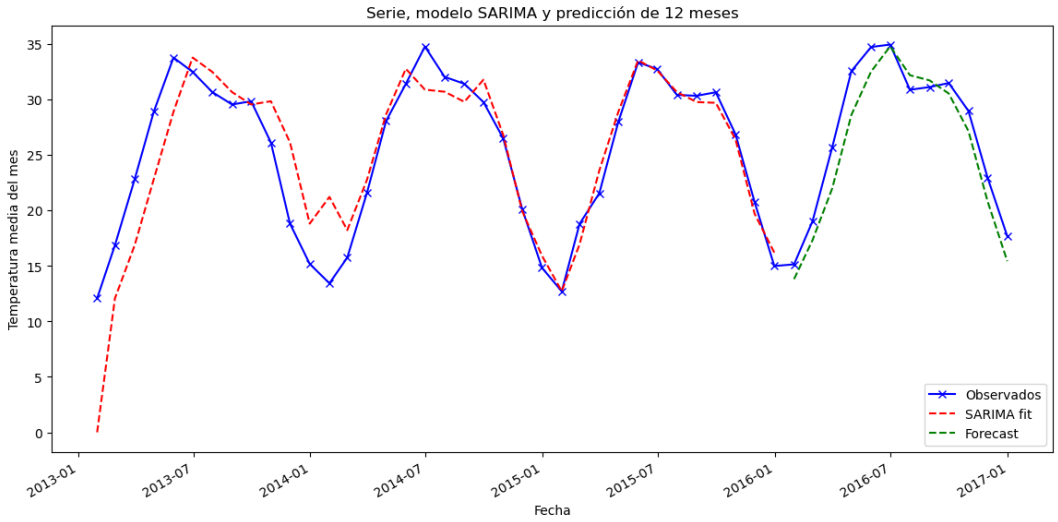


Figura 13: SARIMA con coeficientes automáticos

Comparamos los dos modelos utilizando sus estadísticos.

Modelo	AIC	BIC
Manual	98.046	99.181
auto_arima	86.441	90.983

Cuadro 2: Comparación de estadísticos de los modelos

Escogeremos como modelo ganador el modelo que obtuvimos usando la función auto_arima(). Con los parametros de nuestro modelo SARIMA su expresión algebraica será

$$\left(1-\Phi_1 B^{12}\right)\left(1-\phi_1 B-\phi_2 B^2\right)\left(1-B^{12}\right)(1-B)\,X_t = Z_t$$

.

Podemos ver este forecast de test con sus intervalos de confianza correspondientes.

```
[pred_ajustado, conf_int] = fitted_auto.predict(12, return_conf_int=
    ↪ True)
pred_ajustado = pd.DataFrame(pred_ajustado)
pred_ajustado['l_con'] = conf_int[:,0]
pred_ajustado['h_con'] = conf_int[:,1]

plt.figure(figsize=(14, 6))
plt.plot(total_temp_monthly.index, total_temp_monthly,
         label='Observados', marker='x', linestyle='-', color='blue')
plt.plot(pred_ajustado[0], label='Pronostico ARIMA ajustado', linestyle
    ↪ ='--', color='green', alpha=.6)
plt.plot(pred_ajustado['h_con'], label='Cota superior intervalo de
    ↪ confianza', color='orange', alpha=.6)
```

```
plt.plot(pred_ajustado['l_con'], label='Cota inferior intervalo de
    ↪ confianza', color='red', alpha=.6)
plt.fill_between(pred_ajustado.index,
                 pred_ajustado['h_con'],
                 pred_ajustado['l_con'], color='k', alpha=.2)
plt.legend()
plt.show()
```

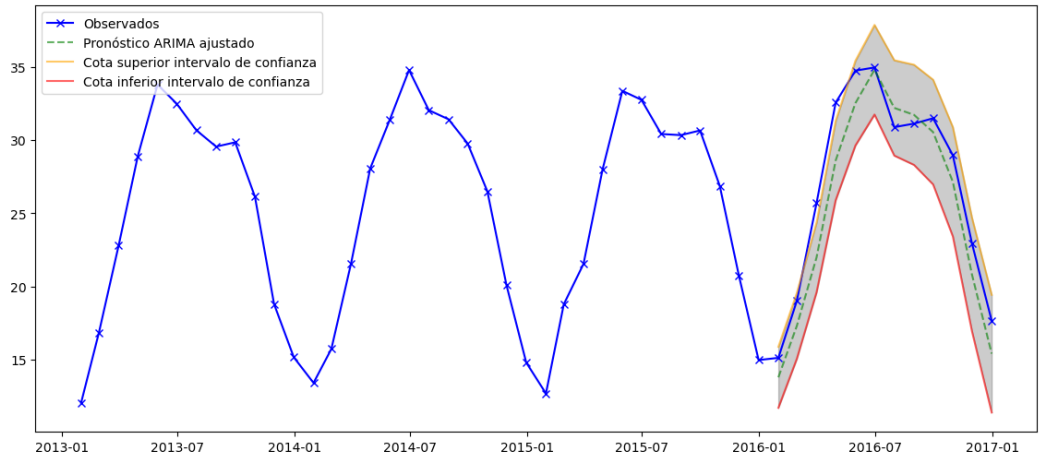


Figura 14: Predicción de prueba con intervalos de confianza

5. Comparación de modelos y predicciones

Tenemos dos modelos que hemos escogido hasta ahora, el modelo obtenido de auto_arima para la temperatura media de los meses y el modelo de suavizado exponencial para la temperatura diaria. Vamos a realizar predicciones de las temperaturas de todo el año siguiente a nuestros datos y compararemos los resultados.

```
# Obtener predicciones para 14 dias
pred_hw_aa = hw_aa.forecast(steps=365)

# Crear un grafico con matplotlib
plt.figure(figsize=(14, 6))
plt.plot(total_temp.index, total_temp,
         label='Observados', marker='x', linestyle='-', color='blue')
plt.plot(total_temp.index, hw_aa.fittedvalues,
         label='Suavizado Holt-Winters', linestyle='--', color='red')
plt.plot(pred_hw_aa.index, pred_hw_aa,
         label='Prediccion', linestyle='--', color='orange')
plt.xlabel('Fecha')
plt.ylabel('Temperatura media')
plt.title('Valores Observados, Suavizado Holt-Winters y Prediccion')
```

```
plt.legend()
plt.xticks(rotation=30, ha='right')
plt.show()
```

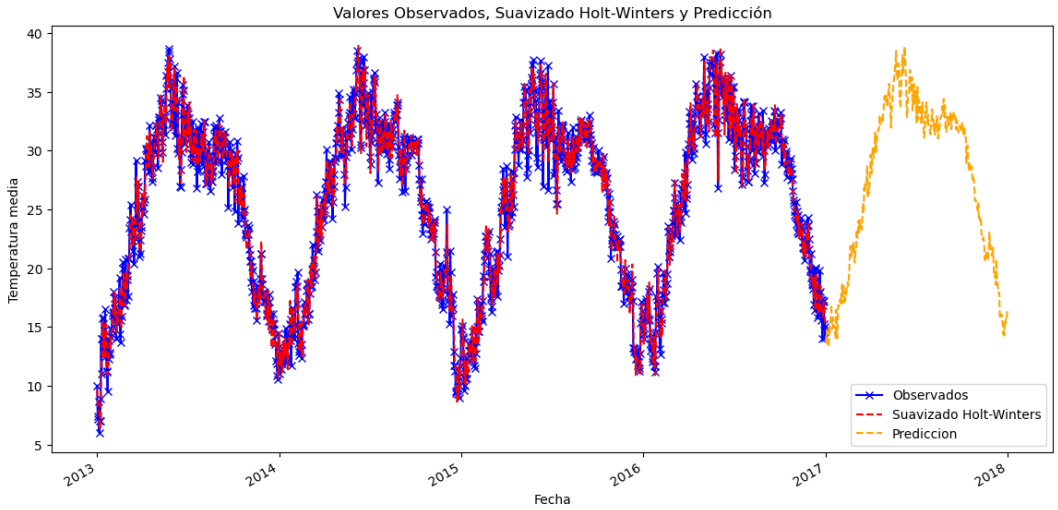


Figura 15: Predicción del 2018 con Holt-Winters

Para obtener los intervalos de confianza de esta predicción necesitamos utilizar una versión más moderna de HoltWintersResultsWrapper disponible en statsmodels, ETSResults, obtenida de ETSModel(). El código utilizado y su resultado es el siguiente:

```
from statsmodels.tsa.exponential_smoothing.ets import ETSModel

HWTES_Model = ETSModel(endog=total_temp,  trend= 'add', seasonal='add',
    ↪  seasonal_periods=365).fit()

point_forecast = HWTES_Model.forecast(365)

# Calculo del intervalo de confianza
ci = HWTES_Model.get_prediction(start=1461,end=1461+364)

lower_conf_forecast = ci.pred_int(alpha=0.1).iloc[:,0]
upper_conf_forecast = ci.pred_int(alpha=0.1).iloc[:,1]

plt.figure(figsize=(14, 6))
plt.plot(total_temp.index, total_temp,
         label='Observados', marker='x', linestyle='-', color='blue')
plt.plot(point_forecast, label='Pronostico Holt-Winters', linestyle='--
    ↪ ', color='green', alpha=.6)
plt.plot(upper_conf_forecast, label='Cota superior intervalo de
    ↪ confianza', color='orange', alpha=.6)
```

```
plt.plot(lower_conf_forecast, label='Cota inferior intervalo de
    ↪ confianza', color='red', alpha=.6)
plt.fill_between(point_forecast.index,
                 upper_conf_forecast,
                 lower_conf_forecast, color='k', alpha=.2)
plt.legend()
plt.show()
```

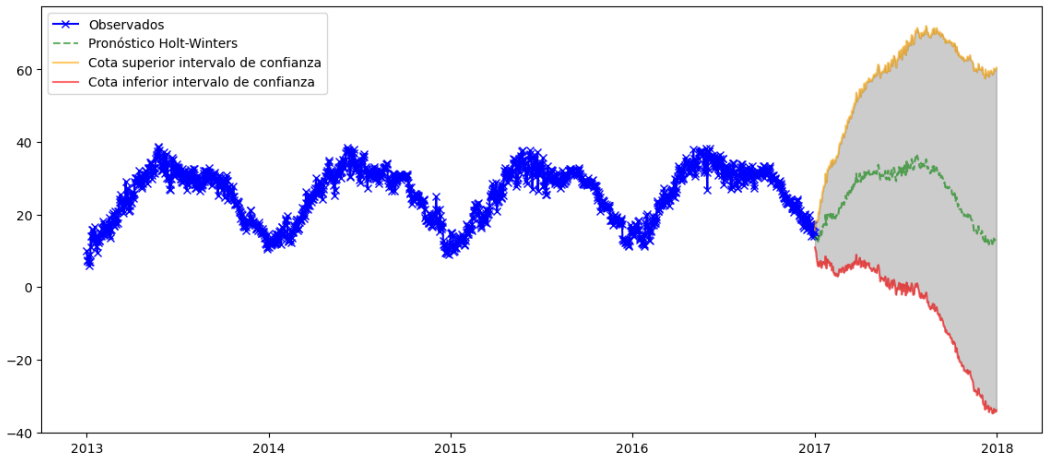


Figura 16: Predicción de 2018 con intervalos de confianza

Los intervalos de confianza de la predicción se hacen evidentemente excesivamente grandes conforme aumenta la ventana en la que predecimos; en este caso, el intervalo de confianza tiene un alfa de 0.1.

Ahora, la predicción de un año con el modelo ARIMA.

```
modelo_ARIMA = ARIMA(order=(2,1,0), seasonal_order=(1,1,0,12)).fit(y=
    ↪ total_temp_monthly)
[pred, conf_int] = modelo_ARIMA.predict(12, return_conf_int=True)
pred = pd.DataFrame(pred)
pred['l_con'] = conf_int[:,0]
pred['h_con'] = conf_int[:,1]

plt.figure(figsize=(14, 6))
plt.plot(total_temp_monthly.index, total_temp_monthly,
         label='Observados', marker='x', linestyle='-', color='blue')
plt.plot(pred[0], label='Pronostico ARIMA', linestyle='--', color='
    ↪ green', alpha=.6)
plt.plot(pred['h_con'], label='Cota superior intervalo de confianza',
    ↪ color='orange', alpha=.6)
plt.plot(pred['l_con'], label='Cota inferior intervalo de confianza',
    ↪ color='red', alpha=.6)
plt.fill_between(pred.index,
```

```
                pred['h_con'],
                pred['l_con'], color='k', alpha=.2)
plt.legend()
plt.show()
```

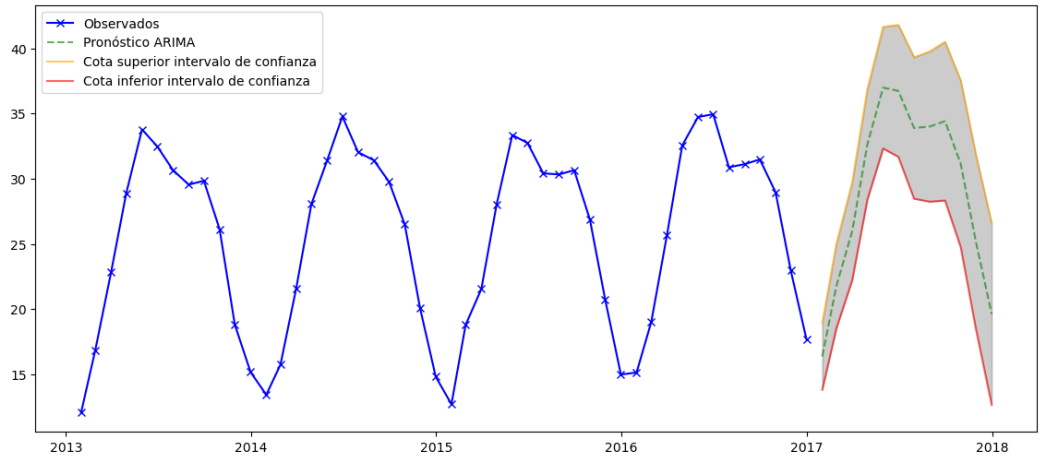


Figura 17: Predicción del modelo SARIMA con intervalos de confianza

Podemos comparar las dos predicciones si pasamos la predicción del modelo Holt-Winters a medias por mes, de la misma manera con los datos de los intervalos de confianza.

```
hw_pred_monthly = pred_hw_aa.resample('M').mean()
upper_conf_forecast_m =upper_conf_forecast.resample('M').mean()
lower_conf_forecast_m = lower_conf_forecast.resample('M').mean()

plt.figure(figsize=(14, 6))
plt.plot(total_temp_monthly.index, total_temp_monthly,
         label='Observados', marker='x', linestyle='-', color='blue')
plt.plot(pred[0], label='Pronostico ARIMA', linestyle='--', color='
    ↪ green', alpha=.6)
plt.plot(pred['h_con'], label='Cota superior intervalo de confianza',
    ↪ color='orange', alpha=.6)
plt.plot(pred['l_con'], label='Cota inferior intervalo de confianza',
    ↪ color='red', alpha=.6)
plt.plot(hw_pred_monthly, label='Prediccion Holt-Winters mensual',
    ↪ linestyle='--',
         color='purple')
plt.plot(upper_conf_forecast_m, label='Cota superior intervalo de
    ↪ confianza HW', alpha=.6)
plt.plot(lower_conf_forecast_m, label='Cota inferior intervalo de
    ↪ confianza HW', alpha=.6)
plt.fill_between(pred.index,
                pred['h_con'],
```

```
                pred['l_con'], color='k', alpha=.2)
plt.legend()
plt.show()
```

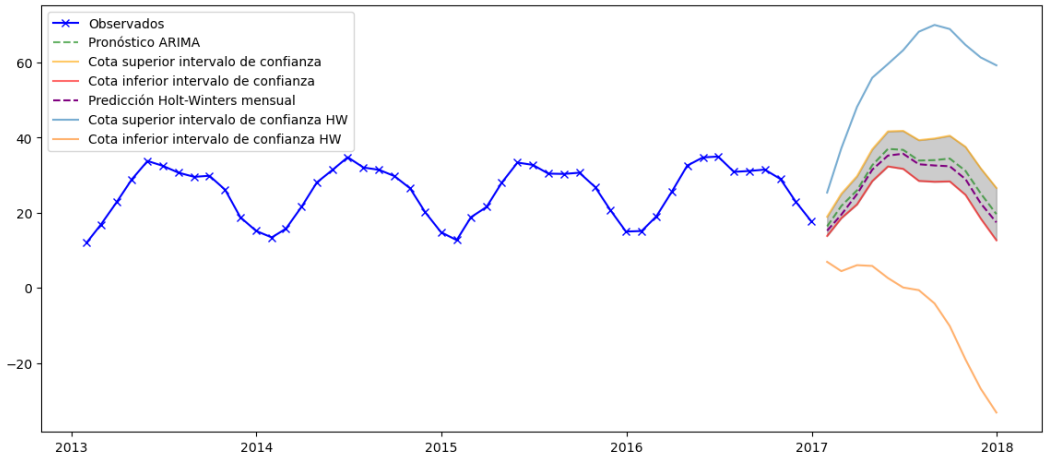


Figura 18: Comparación de la predicción con ambos intervalos de confianza

Por lo amplios que son los intervalos de confianza del modelo Holt-Winters, vamos a representarlo utilizando tan solo el intervalo confianza del modelo SARIMA.

```
plt.figure(figsize=(14, 6))
plt.plot(total_temp_monthly.index, total_temp_monthly,
         label='Observados', marker='x', linestyle='-', color='blue')
plt.plot(pred[0], label='Pronostico ARIMA', linestyle='--', color='
    ↪ green', alpha=.6)
plt.plot(pred['h_con'], label='Cota superior intervalo de confianza',
    ↪ color='orange', alpha=.6)
plt.plot(pred['l_con'], label='Cota inferior intervalo de confianza',
    ↪ color='red', alpha=.6)
plt.plot(hw_pred_monthly, label='Prediccion Holt-Winters mensual',
    ↪ linestyle='--',
         color='purple')
plt.fill_between(pred.index,
                pred['h_con'],
                pred['l_con'], color='k', alpha=.2)
plt.legend()
plt.show()
```

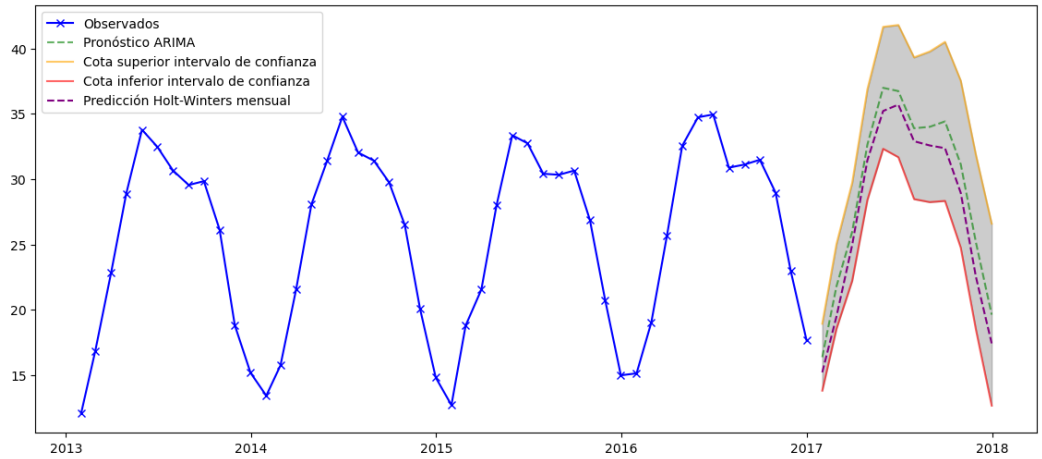


Figura 19: Comparación de la predicción con intervalo de confianza

Vemos que no hay gran diferencia entre las predicciones una vez están ajustadas a los datos por mes, de hecho, la predicción del modelo Holt-Winters se mantiene completamente dentro del intervalo de confianza. La diferencia entre predicciones oscila entre 2.3 y 1 grado.

```
pred[0] - hw_pred_monthly
```

```
2017-01-31    1.161050
2017-02-28    2.305920
2017-03-31    1.009761
2017-04-30    1.194625
2017-05-31    1.772050
2017-06-30    1.035893
2017-07-31    0.991134
2017-08-31    1.426849
2017-09-30    2.049543
2017-10-31    2.193209
2017-11-30    2.639959
2017-12-31    2.195979
```

Figura 20: Diferencia entre predicciones

6. Conclusiones

Hemos podido obtener dos buenos modelos para nuestros datos. Si bien no están prediciendo sobre las mismas unidades de tiempo, podemos utilizar el modelo Holt-Winters para predecir días y el modelo SARIMA para predecir en espacios de tiempo más grandes. En espacios de tiempo más grandes es posible que el modelo SARIMA sea más adecuado. Pues parece que el modelo Holt-Winters pierde precisión cuando se le pide predecir mucho tiempo, mientras que el SARIMA con datos mensuales se comporta muy bien.

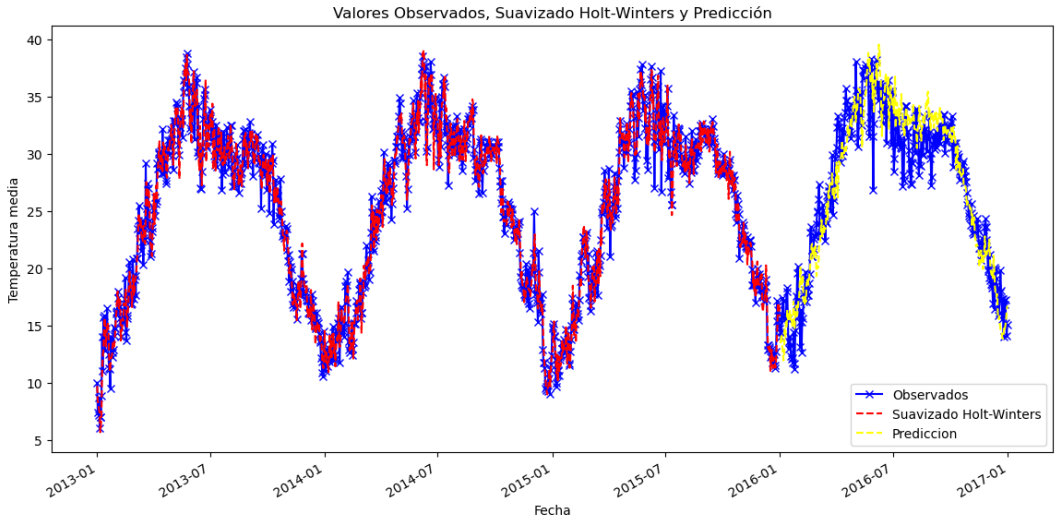


Figura 21: Holt-Winters con 1 año de test

Una ventaja sustancial del modelo Holt-Winters es que lo podemos utilizar para predecir y trabajar con datos diarios, mientras que el SARIMA tiene algunos problemas por la cantidad de índices de estacionalidad que necesita calcular en estos casos. Como ya comenté más arriba, esto podría solventarse utilizando una serie de Fourier que modela los índices de estacionalidad (y es significativamente más sencilla de calcular computacionalmente) y sustituir esta serie en la componente estacional del modelo.