

## Máster Big Data, Data Science & Inteligencia Artificial 2024-2025 - UCM

Profesor: Dra Juana María Alonso

Alumno: Ing Paez Sheridan, Pablo Santiago

Materia: Minería de datos y modelización predictiva III

Tarea de Minería de datos y modelización predictiva III

# Mineria de datos y modelizacion predictiva

## 3 - Tarea

Librerias usadas

```
In [61]: import pandas as pd
import matplotlib.pyplot as plt
import pmdarima as pm

from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing, Holt, ExponentialSmooth
from tabulate import tabulate
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
```

## 1 - Presentación de la serie

La serie con la que vamos a trabajar es un serie obtenida por el Servicio Meteorológico Nacional de la República Argentina que tiene todos los registros temperatura maxima y minima para todas las ciudades del país correspondiente a los últimos 365 días.

El dataset original se adjunta con el nombre original "registro\_temperatura365d\_smn.txt".

Fuente: <https://datos.gob.ar/dataset/smn-registro-temperatura-365-dias>

Para poder trabajar con datos más reducidos y cumplir con la consigna voy a trabajar con los datos de la ciudad donde vivo, Rosario, y solo con las TEMPERATURAS MAXIMAS dejando de lado las mínimas.

Uso el naming convention bronze, silver, gold para nombrar los diferentes conjuntos de datos según la madurez de los mismos. Esta convención es sacada del patrón Medallion el cual es un patrón de diseño para manipulación de datos. Bronze son los datos como vienen de origen, Silver son datos filtrados y limpiados, Gold son los datos finales que pueden tener transformación de tipos de datos así como agregados definidos por el nivel de Negocios.

Referencia: <https://www.databricks.com/glossary/medallion-architecture>

```
In [ ]: with open("registro_temperatura365d_smn.txt", "r", encoding="latin1") as f:
    bronze_data = f.read()
```

```

lines = bronze_data.strip().split("\n")[2:]
rosario_temps = [line.split()[1] for line in lines if "ROSARIO" in line]

max_temps = []
for temp in rosario_temps:
    try:
        max_temps.append(float(temp))
    except ValueError:
        max_temps.append(None)

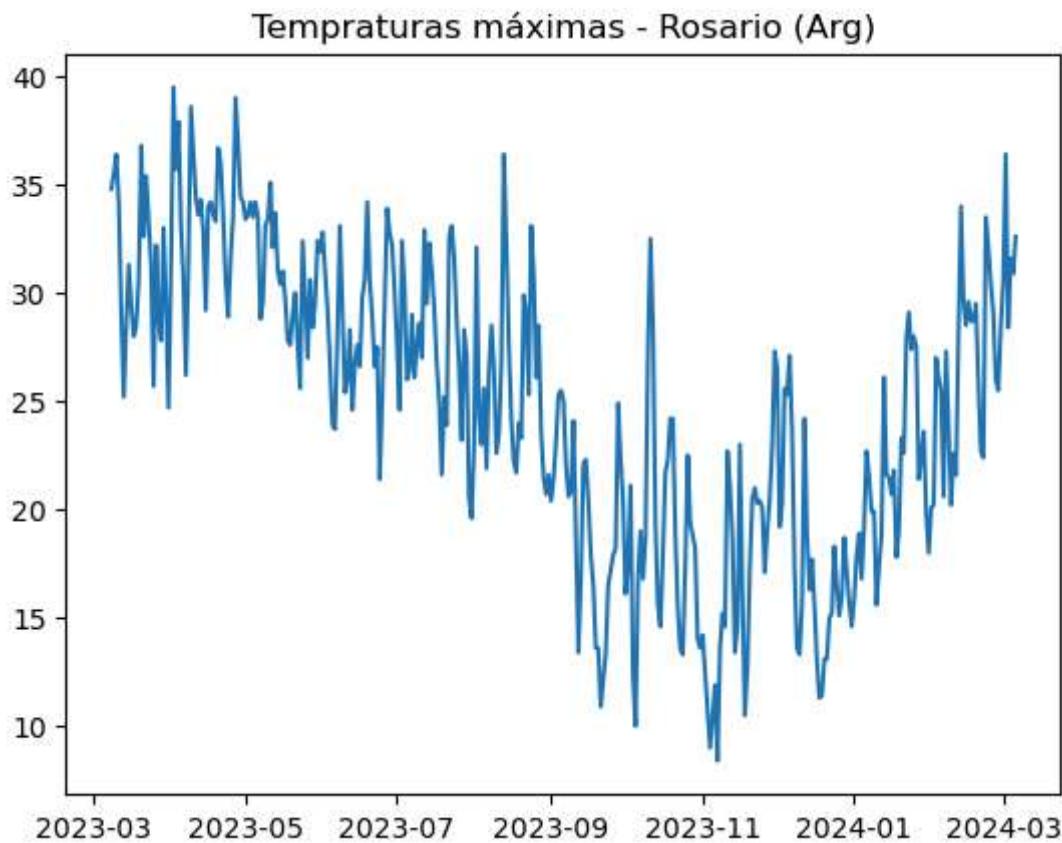
silver_dates = pd.date_range(start="2023-03-08", periods=len(max_temps))

gold_rosario_temperature_series = pd.Series(max_temps, index=silver_dates)

plt.plot(gold_rosario_temperature_series)
plt.title('Tempraturas máximas en C° - Rosario (Arg)')

```

Out[ ]: Text(0.5, 1.0, 'Tempraturas máximas - Rosario (Arg)')



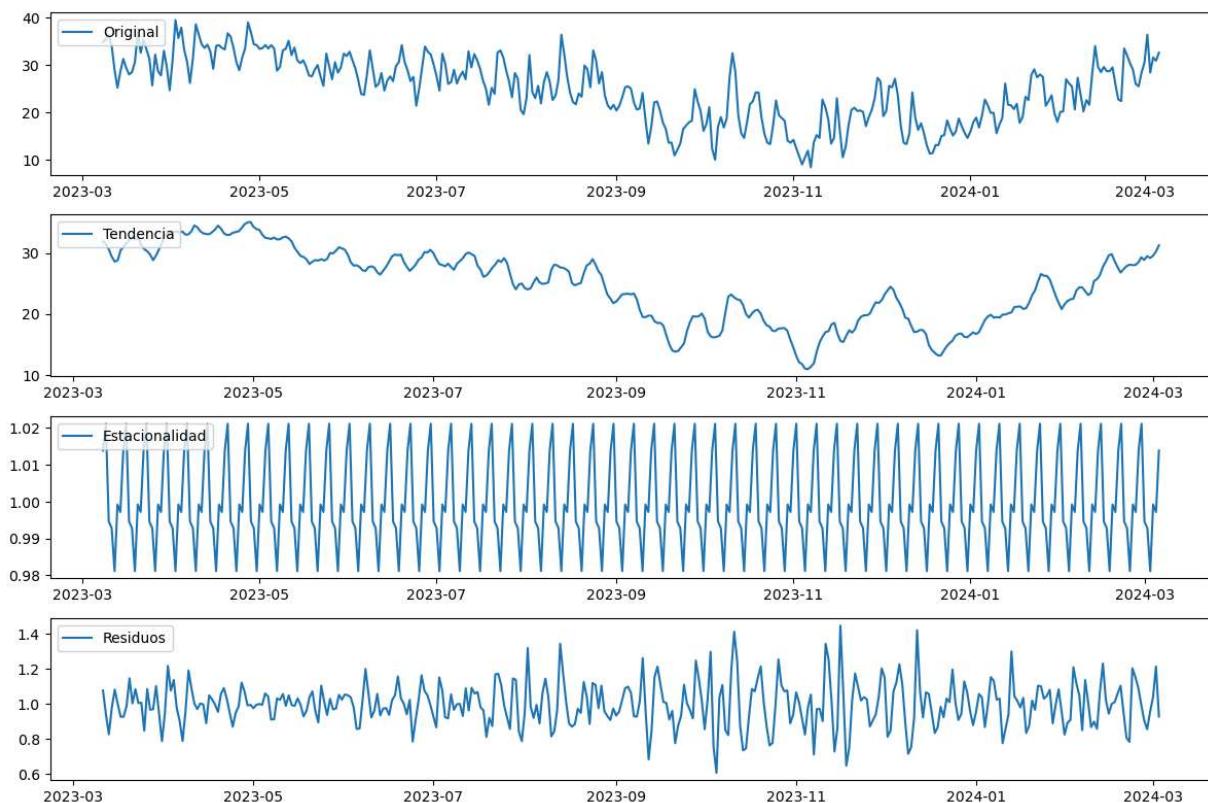
## 2 - Representación gráfica y descomposición estacional

Vamos a proceder a hacer la descomposición de la serie. El valor mínimo observado es 8.4 C° por lo que podemos usar una descomposición multiplicativa ya que no tenemos valores negativos o 0.

```
In [14]: def decompose_time_series(time_series):
    decomposition = seasonal_decompose(time_series, model='multiplicative')
    trend = decomposition.trend
    seasonal = decomposition.seasonal
    residual = decomposition.resid

    plt.figure(figsize=(12, 8))
    plt.subplot(411)
    plt.plot(time_series, label='Original')
    plt.legend(loc='upper left')
    plt.subplot(412)
    plt.plot(trend, label='Tendencia')
    plt.legend(loc='upper left')
    plt.subplot(413)
    plt.plot(seasonal, label='Estacionalidad')
    plt.legend(loc='upper left')
    plt.subplot(414)
    plt.plot(residual, label='Residuos')
    plt.legend(loc='upper left')
    plt.tight_layout()
    plt.show()

decompose_time_series(gold_rosario_temperature_series)
```



Analizando la descomposicion de la serie podemos observar que es una serie que cuenta con una tendencia muy marcada y tambien muy variable lo que seguro dificulta su modelizacion a futuro, por otro lado el componente estacional existe pero el rango entre el valor minimo y maximo es muy pequeno por lo que nos indica que la estacionalidad a penas si esta presente en nuestro modelo. Con respecto a los residuos estos son aceptables

aunque son incluso mas significativos que la estacionalidad. No observamos ciclos ni patrones de ningun tipo.

Si los datos se hubieses extendido en el tiempo es probable que la serie hubiese presentando una estacionalidad anual pero al tener solo un año de datos esa estacionalidad no se visualiza en el conjunto de datos

### 3 - Muestras de Train y Test

Vamos a dividir nuestro conjunto de datos en dos subconjuntos de train y test. Reservo el 10% de las observaciones para pruebas quedandome con el 90% para entrenamiento

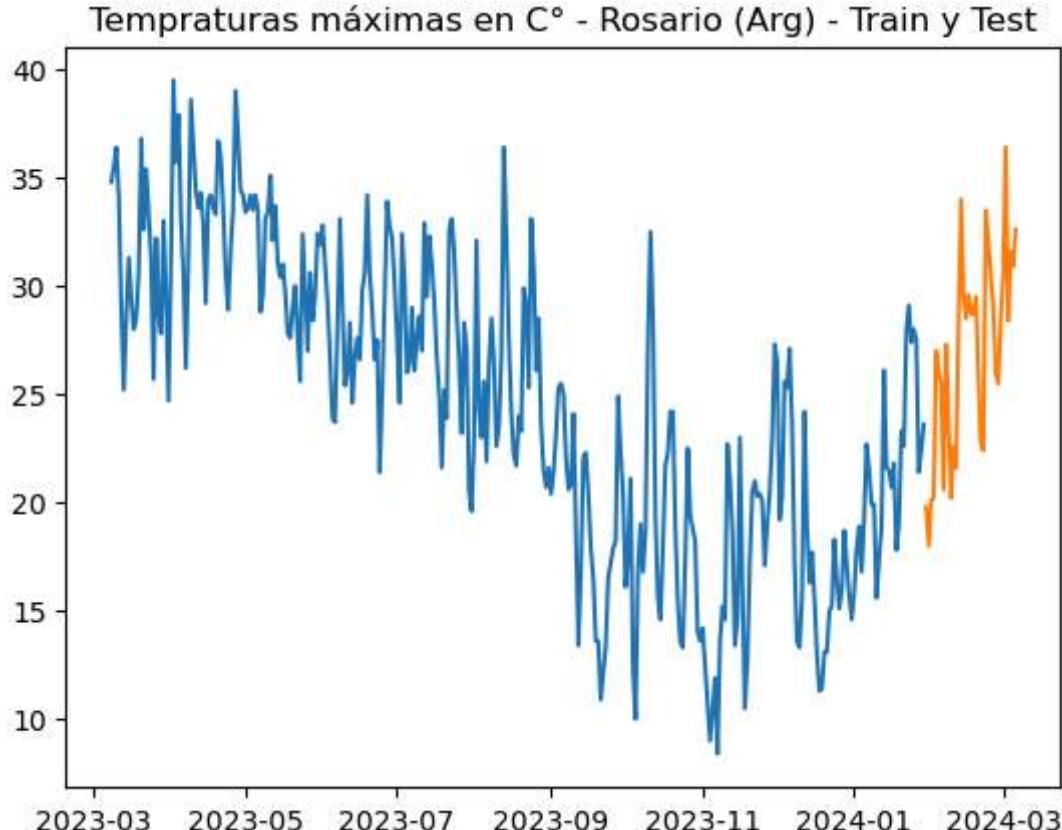
```
In [17]: def split_train_test(time_series, test_size):
    split_index = int(len(time_series) * (1 - test_size))
    train = time_series[:split_index]
    test = time_series[split_index:]
    return train, test

gold_rosario_temperature_train, gold_rosario_temperature_test= split_train_test(gold_rosario_temperature)

plt.plot(gold_rosario_temperature_train)
plt.title('Tempraturas máximas en C° - Rosario (Arg) - Train y Test')

plt.plot(gold_rosario_temperature_test)
```

Out[17]: [`<matplotlib.lines.Line2D at 0x261f324ec70>`]



## 4 - Suavizado Exponencial

Vamos a proceder a realizar las diferentes tecnicas de suavizado exponenciales dados en clase a saber:

- Metodo exponencial simple
- Metodo de alisado doble de holt
- Metodo de tendencia amortiguada de Holt
- Método de Holt-Winters

```
In [48]: def simple_exp_smoothing_forecast(time_series_train, time_series_test, alpha=0.2):
    model = SimpleExpSmoothing(time_series_train).fit(smoothing_level=alpha, optimi
    forecast = model.forecast(steps=len(time_series_test))
    return model.fittedvalues, forecast

def holt_trend_smoothing_forecast(time_series_train, time_series_test, alpha=0.2, b
    model = Holt(time_series_train).fit(smoothing_level=alpha, smoothing_trend=beta
    forecast = model.forecast(steps=len(time_series_test))
    return model.fittedvalues, forecast

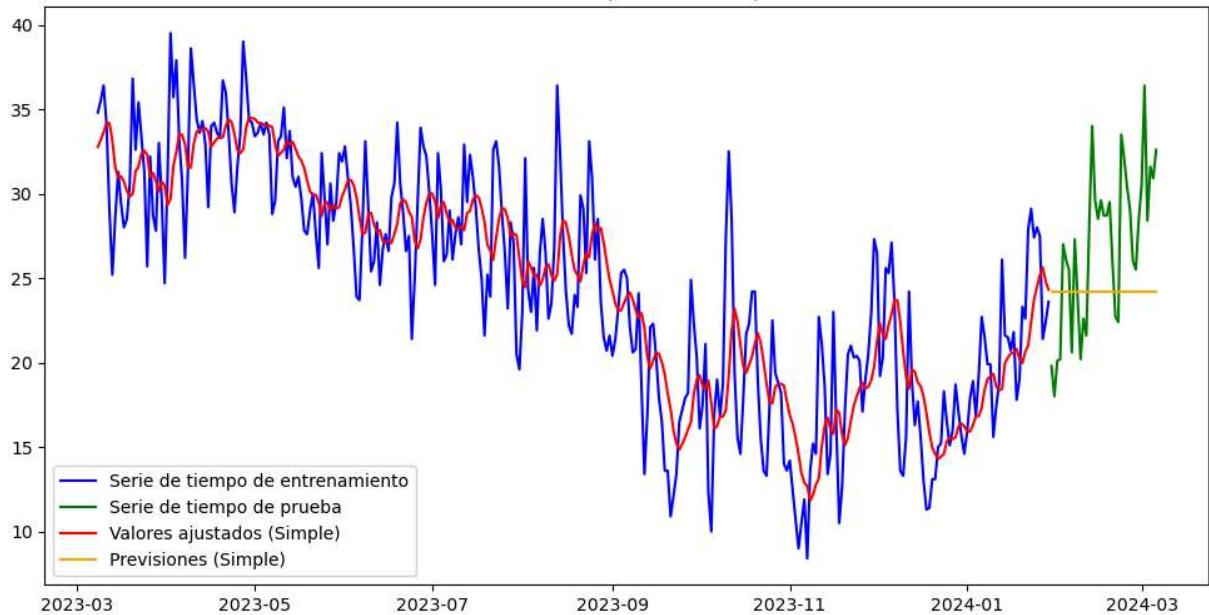
def holt_trend_damped_smoothing_forecast(time_series_train, time_series_test, alpha
    model = Holt(time_series_train, damped_trend = True).fit(smoothing_level=alpha,
    forecast = model.forecast(steps=len(time_series_test))
    return model.fittedvalues, forecast

def holt_winters_smoothing_forecast(time_series_train, time_series_test, seasonal_p
    model = ExponentialSmoothing(time_series_train, seasonal_periods=seasonal_perio
    forecast = model.forecast(steps=len(time_series_test))
    return model.fittedvalues, forecast

simple_fitted, simple_forecast = simple_exp_smoothing_forecast(gold_rosario_tempera
holt_fitted, holt_forecast = holt_trend_smoothing_forecast(gold_rosario_temperature
holt_damped_fitted, holt_damped_forecast = holt_trend_damped_smoothing_forecast(gol
holt_winters_fitted, holt_winters_forecast = holt_winters_smoothing_forecast(gold_r
```

```
In [49]: plt.figure(figsize=(12, 6))
plt.plot(gold_rosario_temperature_train, label='Serie de tiempo de entrenamiento',
plt.plot(gold_rosario_temperature_test, label='Serie de tiempo de prueba', color='g
plt.plot(gold_rosario_temperature_train.index, simple_fitted, label='Valores ajusta
plt.plot(gold_rosario_temperature_test.index, simple_forecast, label='Previsiones (
plt.title('Suavizado exponencial simple')
plt.legend()
plt.show()
```

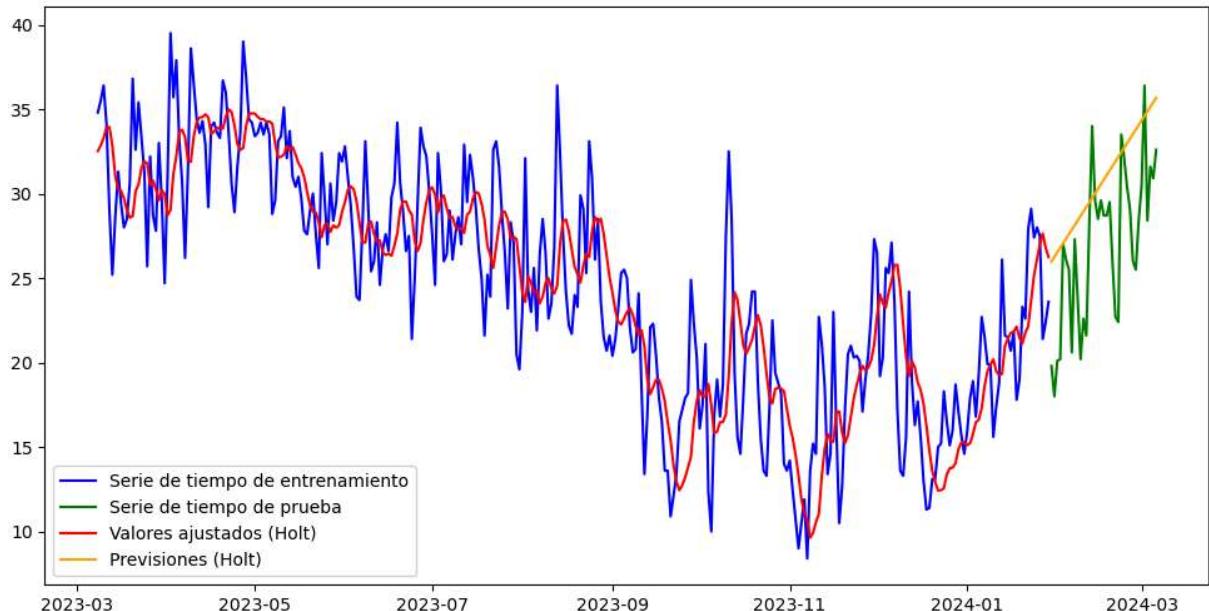
Suavizado exponencial simple



El suavizado exponencial simple es una tecnica que no tiene en cuenta la tendencia por lo que no modela bien la tendencia variable ni la estacionalidad, nuestra serie cuenta con ambos componentes por lo que el comportamiento de la misma no es bien predicho por esta tecnica de suavizado

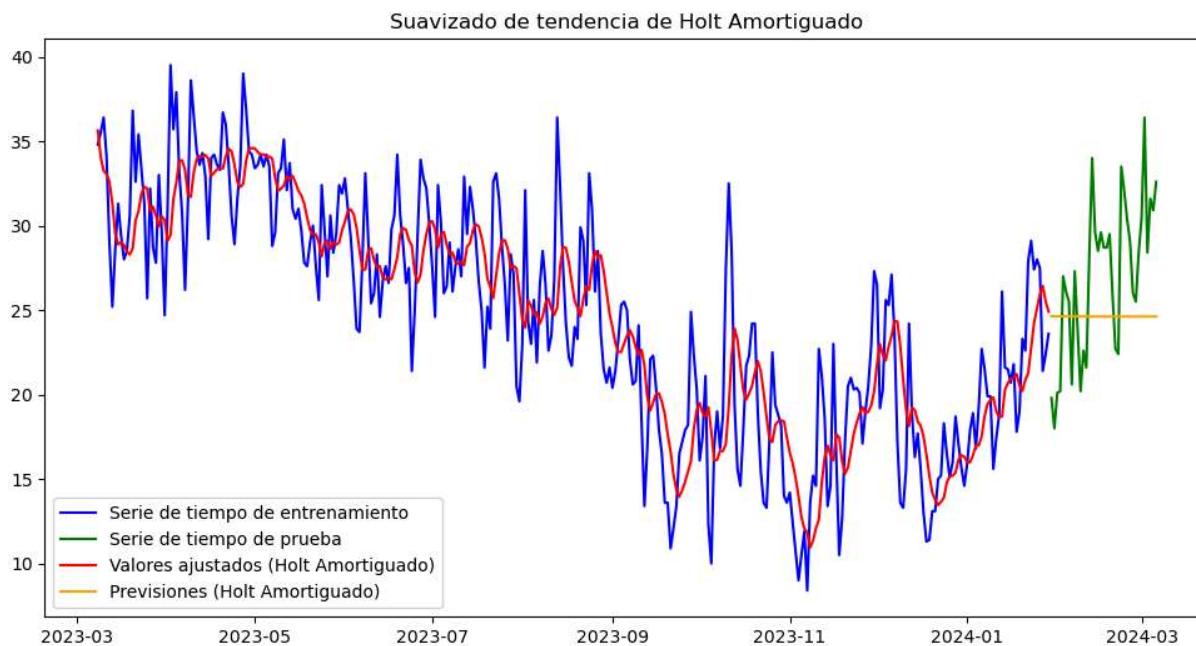
```
In [50]: plt.figure(figsize=(12, 6))
plt.plot(gold_rosario_temperature_train, label='Serie de tiempo de entrenamiento',
plt.plot(gold_rosario_temperature_test, label='Serie de tiempo de prueba', color='g'
plt.plot(gold_rosario_temperature_train.index, holt_fitted, label='Valores ajustados (Holt)')
plt.plot(gold_rosario_temperature_test.index, holt_forecast, label='Previsiones (Holt)')
plt.title('Suavizado de tendencia de Holt')
plt.legend()
plt.show()
```

Suavizado de tendencia de Holt



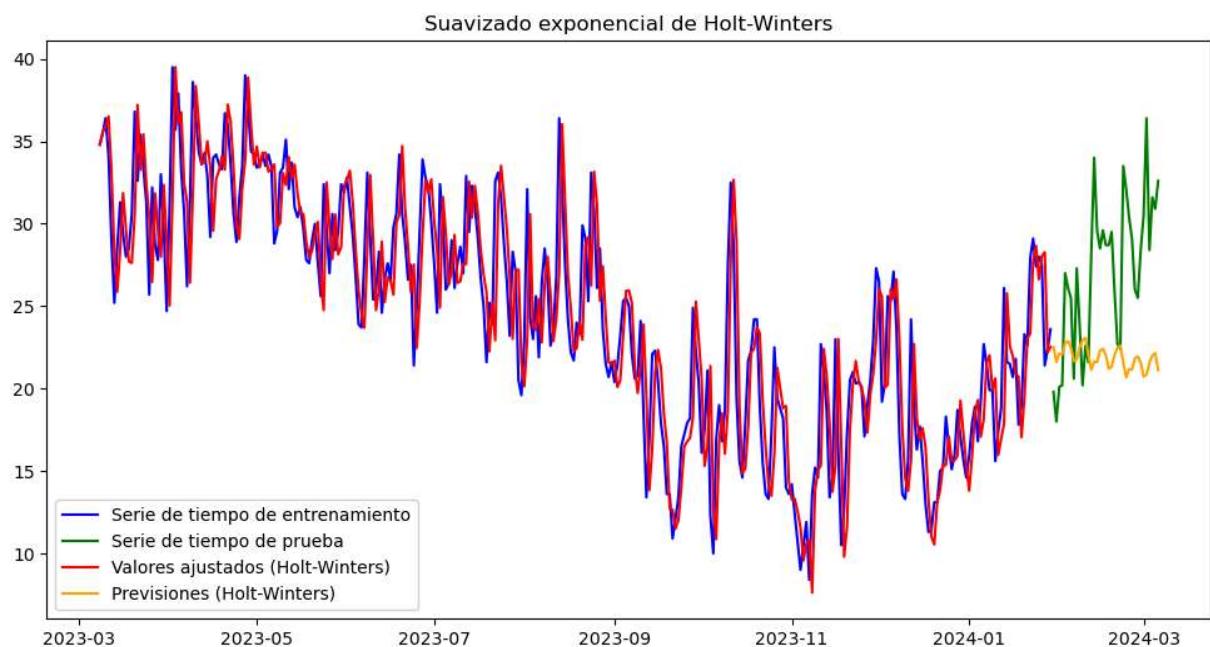
El suavizado por tendencia de Holt tiene en cuenta cambios en la tendencia y este es principalmente util cuando la funcion de la tendencia de la serie no es lineal sino que puede experimentar cambios como es el caso de esta serie que tiene una tendencia que cambia incluso de signo mas de una vez en la misma grafica. Sin embargo este metodo no modela muy bien la estacionalidad por lo que las predicciones no son del todo correctas.

```
In [51]: plt.figure(figsize=(12, 6))
plt.plot(gold_rosario_temperature_train, label='Serie de tiempo de entrenamiento',
         plt.plot(gold_rosario_temperature_test, label='Serie de tiempo de prueba', color='green')
plt.plot(gold_rosario_temperature_train.index, holt_damped_fitted, label='Valores ajustados (Holt Amortiguado)')
plt.plot(gold_rosario_temperature_test.index, holt_damped_forecast, label='Previsiones (Holt Amortiguado)')
plt.title('Suavizado de tendencia de Holt Amortiguado')
plt.legend()
plt.show()
```



El suavizado con tendencia de Holt amortiguado es util para modelar series que tienen una tendencia que disminuye con el paso del tiempo y que no tengan estacionalidad. Como nuestra serie tiene estacionalidad y tambien tendencia pero que no disminuye con el tiempo nos brinda unas predicciones pobres, identicas a el suavizado exponencial simple.

```
In [52]: plt.figure(figsize=(12, 6))
plt.plot(gold_rosario_temperature_train, label='Serie de tiempo de entrenamiento',
         plt.plot(gold_rosario_temperature_test, label='Serie de tiempo de prueba', color='green')
plt.plot(gold_rosario_temperature_train.index, holt_winters_fitted, label='Valores ajustados (Holt-Winters)')
plt.plot(gold_rosario_temperature_test.index, holt_winters_forecast, label='Previsiones (Holt-Winters)')
plt.title('Suavizado exponencial de Holt-Winters')
plt.legend()
plt.show()
```



El suavizado exponencial de Holt-Winters es una tecnica que se utiliza para suavizar series con tendencia y estacionalidad, en nuestro caso seria el adecuado para nuestra serie, sin embargo las predicciones son malas. Mi hipotesis para este rendimiento es que la estacionalidad tiene un comportamiento estacional anualizado por lo que si partimos el año que tenemos en nuestra serie y construimos el modelo con una porcion de los datos es improbable que el comportamiento estacional se encuentre bien representado y por consecuencia que la serie sea bien suavizada. Si tendriamos un conjunto de datos de varios años creo que esta tecnica pudiera haber sido una buena opcion

```
In [75]: comparison_table = pd.DataFrame({
    'Valor Real': gold_rosario_temperature_test,
    'Simple Smoothing Forecast': simple_forecast,
    'Holt Smoothing Forecast': holt_forecast,
    'Holt-Winters Smoothing Forecast': holt_winters_forecast
})

print(tabulate(comparison_table, headers='keys'))
```

ecast	Holt-Winters	Valor Real	Simple Smoothing Forecast	Holt Smoothing For
		Smoothing Forecast		
2024-01-30 00:00:00	5.9845	19.8	24.1726	2
2024-01-31 00:00:00	6.2537	22.5076	24.1726	2
2024-02-01 00:00:00	6.5228	18	24.1726	2
2024-02-02 00:00:00	6.792	21.6042	24.1726	2
2024-02-03 00:00:00	7.0611	22.1188	24.1726	2
2024-02-04 00:00:00	7.3303	20.1	24.1726	2
2024-02-05 00:00:00	7.5994	20.2	24.1726	2
2024-02-06 00:00:00	7.8686	22.0633	24.1726	2
2024-02-07 00:00:00	8.1378	27	24.1726	2
2024-02-08 00:00:00	8.4069	22.7744	24.1726	2
2024-02-09 00:00:00	8.6761	26.1	24.1726	2
2024-02-10 00:00:00	8.9452	22.8782	24.1726	2
2024-02-11 00:00:00	9.2144	25.5	24.1726	2
2024-02-12 00:00:00	9.4835	22.5004	24.1726	2
2024-02-13 00:00:00	9.7527	20.6	24.1726	2
2024-02-14 00:00:00	0.0218	21.6597	24.1726	2
2024-02-15 00:00:00	0.291	27.3	24.1726	2
2024-02-16 00:00:00	0.5601	21.7708	24.1726	2
2024-02-17 00:00:00	0.8293	23.8	24.1726	2
2024-02-18 00:00:00	1.0984	22.5101	24.1726	2
2024-02-19 00:00:00	1.3676	20.2	24.1726	2
2024-02-20 00:00:00	1.6367	22.9351	24.1726	2
2024-02-21 00:00:00	2.0059	22.6	24.1726	2
2024-02-22 00:00:00	2.175	22.0425	24.1726	2
2024-02-23 00:00:00	2.4442	28.2	24.1726	2
2024-02-24 00:00:00	2.7133	34	24.1726	2
		21.139	24.1726	2
		21.6537	24.1726	2
		21.5982	24.1726	3
		23.0781	24.1726	3
		22.3093	24.1726	3
		22.413	24.1726	3
		22.0353	24.1726	3
		21.1946	24.1726	3
		21.3057	24.1726	3
		25.9	24.1726	3
		22.045	24.1726	3
		22.47	24.1726	3
		22.4	24.1726	3
		22.6129	24.1726	3
		33.5	24.1726	3
		21.5774	24.1726	3
		32.1	24.1726	3
		20.6739		3

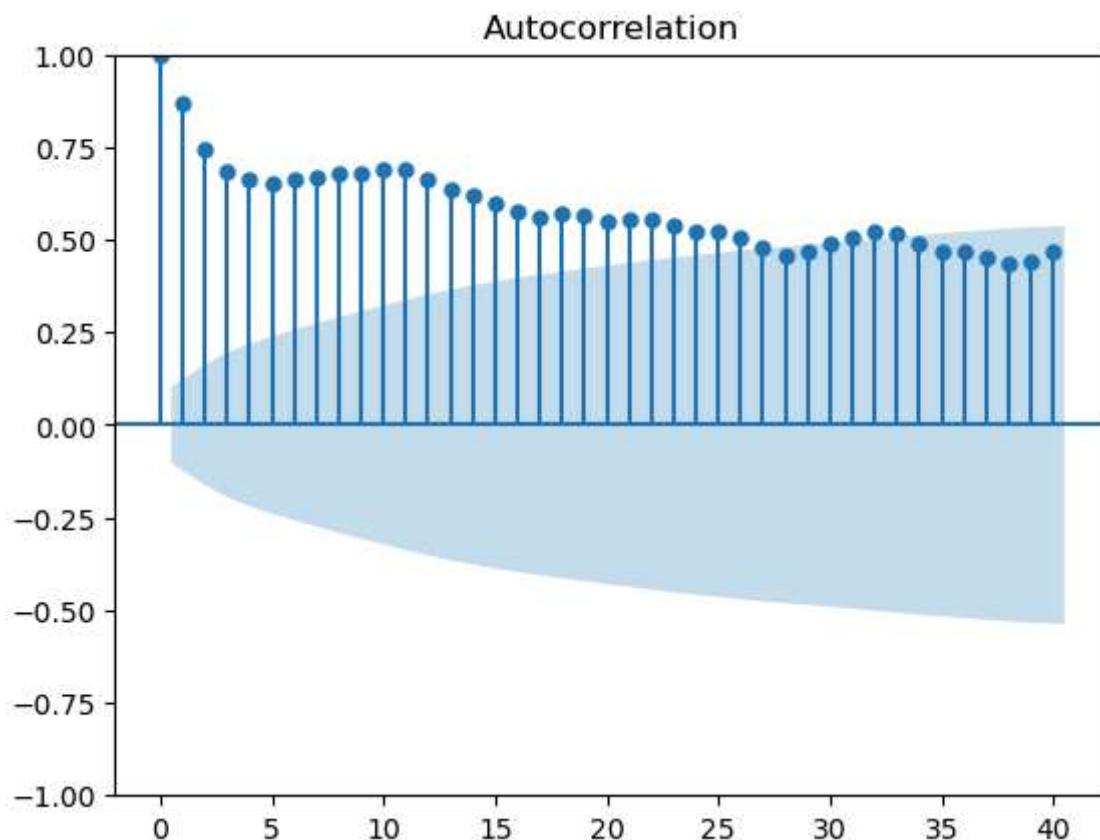
2024-02-25 00:00:00	30.4	24.1726	3
2.9825	21.1886		
2024-02-26 00:00:00	29.1	24.1726	3
3.2517	21.133		
2024-02-27 00:00:00	26	24.1726	3
3.5208	21.8441		
2024-02-28 00:00:00	25.5	24.1726	3
3.79	21.9479		
2024-02-29 00:00:00	28.4	24.1726	3
4.0591	21.5702		
2024-03-01 00:00:00	30.5	24.1726	3
4.3283	20.7295		
2024-03-02 00:00:00	36.4	24.1726	3
4.5974	20.8406		
2024-03-03 00:00:00	28.4	24.1726	3
4.8666	21.5798		
2024-03-04 00:00:00	31.6	24.1726	3
5.1357	22.0048		
2024-03-05 00:00:00	30.9	24.1726	3
5.4049	22.1478		
2024-03-06 00:00:00	32.6	24.1726	3
5.674	21.1123		

En la tabla podemos ver un poco lo que ya se vio en las imagenes. A pesar de aciertos ocasionales, no tenemos ningun modelo que de buenas predicciones de nuestra serie original. El que menos mal se comporto es el metodo de suavizado con tendencia de Holt

## 5 - Correlogramas

### Autocorrelaciones simples

```
In [55]: plot_acf(gold_rosario_temperature_series, lags=40)
plt.show()
```

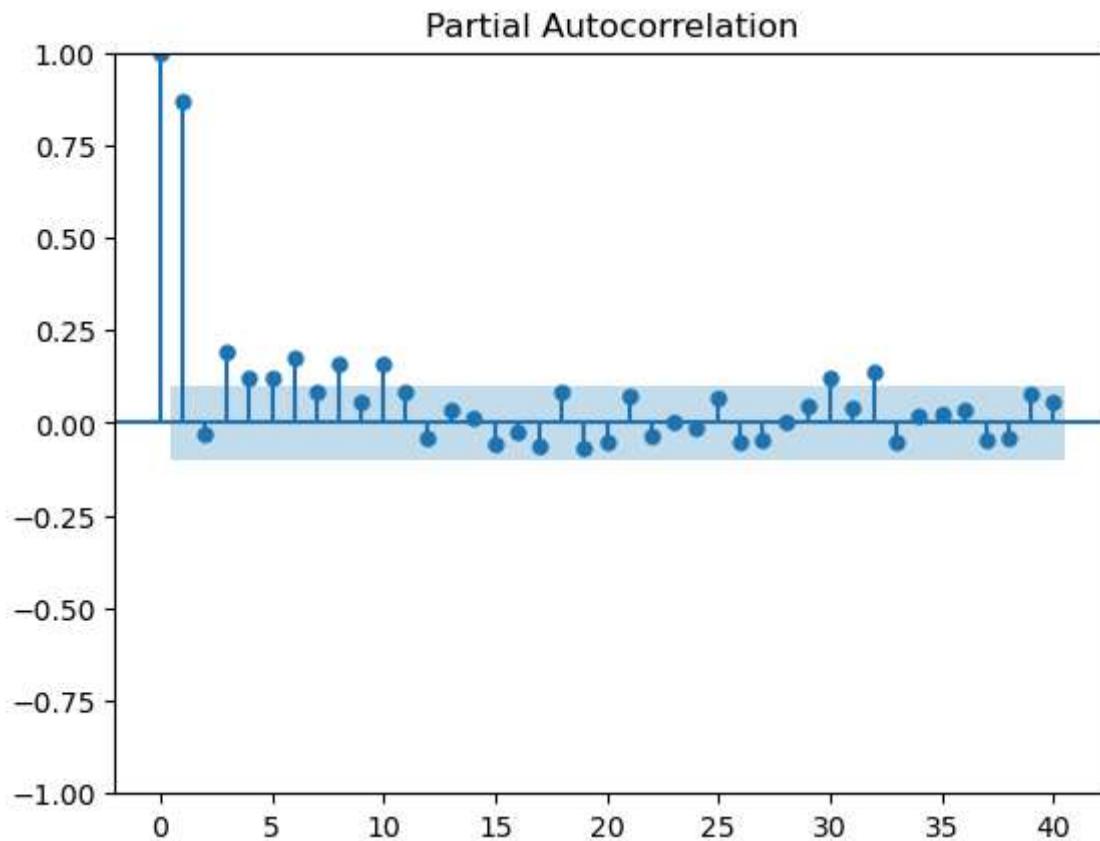


En nuestro grafico de autocorrelacion podemos observar tiene un decaimiento leve, esto sugiere que no es estacionaria o que el componente estacionario tiene una influencia reducida en la serie. Segun lo que podemos ver hay una fuerte dependencia de la serie en diferentes momentos. Esta idea se ve reforzada por rezagos fuera del area subrayada indicando que muestran una dependencia superior

## Autocorrelaciones Parciales

```
In [56]: plot_pacf(gold_rosario_temperature_series, lags=40)
plt.show()
```

```
c:\Users\pablo\anaconda3\envs\mineria_de_datos_ucm\lib\site-packages\statsmodels\graphics\tsaplots.py:348: FutureWarning: The default method 'yw' can produce PACF values outside of the [-1,1] interval. After 0.13, the default will change to unadjusted Yule-Walker ('ywm'). You can use this method now by setting method='ywm'.
warnings.warn(
```



Se ve con claridad un corte abrupto despues del primer rezago esto nos dice que hay una fuerte correlacion entre el primer valor y el valor inmediatamente anterior a la serie. Los demas rezago caen en significancia estando casi todos dentro del area sombreada que representa el area de significancia para demostrar si la serie tiene coeficiente de autocorrelacion 0. Por consecuencia no podemos decir que se prueba la hipotesis nula pero ciertamente el coficiente de autocorrelacion probablemente se encuentre en algun lugar cerca del 0

## ¿Qué modelo elegir?

### Analisis del ACF y el PACF

El ACF tiene un decaimiento lento lo que nos dice que tenemos una posible tendencia y una estacionariedad nula o reducida. El PACF debido a su corte abrupto despues de la segunda obsevacion nos sugiere un componente autorregresivo de orden 1 AR(1)

### Eleccion del modelo ARIMA

Debido a la escasa estacionalidad que nos indica el ACF necesitamos hacerla estacionaria diferenciandola. Aplicamos la primera diferenciacion.  $I = 1$  El corte abrupto nos indica que la dependencia fuerte es con el primer elemento por lo tanto, como ya lo dijimos, nos sugiere un AR con  $p = 1$  Debido a que el corte en el PACF es abrupto solo en el primer elemento y que luego decae para las demas observaciones nos sugiere un MA con coeficiente 0.

En conclusión el modelo ARIMA que considero mas apropiado, luego de analizar los correlogramos, es un ARIMA( $p=1, l=1, q=0$ ) o ARIMA(1,1,0)

## 6 - Ajuste del Modelo ARIMA Automatico y Manual

### ARIMA Ajuste Manual

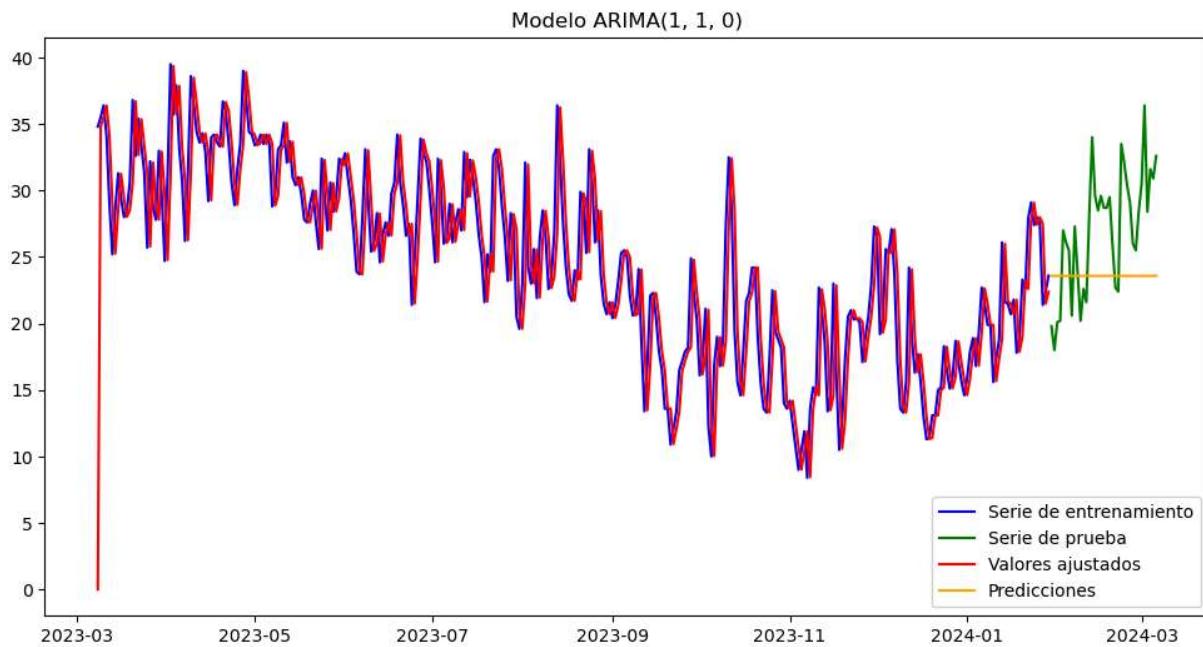
El ajuste manual va a corresponder a lo seleccionado en el punto anterior: ARIMA(1,1,0)

```
In [72]: model = ARIMA(gold_rosario_temperature_train, order=(1, 1, 0))
model_fit = model.fit()

arima_predictions = model_fit.predict(start=len(gold_rosario_temperature_train), end=len(gold_rosario_temperature_test))

plt.figure(figsize=(12, 6))
plt.plot(gold_rosario_temperature_train, label='Serie de entrenamiento', color='blue')
plt.plot(gold_rosario_temperature_test.index, gold_rosario_temperature_test, label='Serie de prueba', color='green')
plt.plot(gold_rosario_temperature_train.index, model_fit.fittedvalues, color='red')
plt.plot(gold_rosario_temperature_test.index, arima_predictions, color='orange', label='Valores ajustados')
plt.title('Modelo ARIMA(1, 1, 0)')
plt.legend()
```

Out[72]: <matplotlib.legend.Legend at 0x261f3f9e8b0>



### ARIMA Ajuste Automático

```
In [65]: auto_model = pm.auto_arima(gold_rosario_temperature_train, start_p=0, start_q=0,
                                 max_p=5, max_q=5, m=12,
                                 d=None, seasonal=False,
                                 start_P=0, D=0, start_Q=0,
                                 error_action='ignore',
                                 suppress_warnings=True,
```

```

stepwise=True)

print(auto_model.summary())

fitted_values = auto_model.predict_in_sample()
auto_arima_predictions = auto_model.predict(n_periods=len(gold_rosario_temperature_)

plt.figure(figsize=(12, 6))
plt.plot(gold_rosario_temperature_train, label='Serie de entrenamiento', color='blue')
plt.plot(gold_rosario_temperature_test.index, gold_rosario_temperature_test, label='Serie de prueba')
plt.plot(gold_rosario_temperature_train.index, fitted_values, color='red', label='Ajuste')
plt.plot(gold_rosario_temperature_test.index, auto_arima_predictions, color='orange', label='Predicción')
plt.title('Modelo ARIMA de ajuste automático')
plt.legend()
plt.show()

```

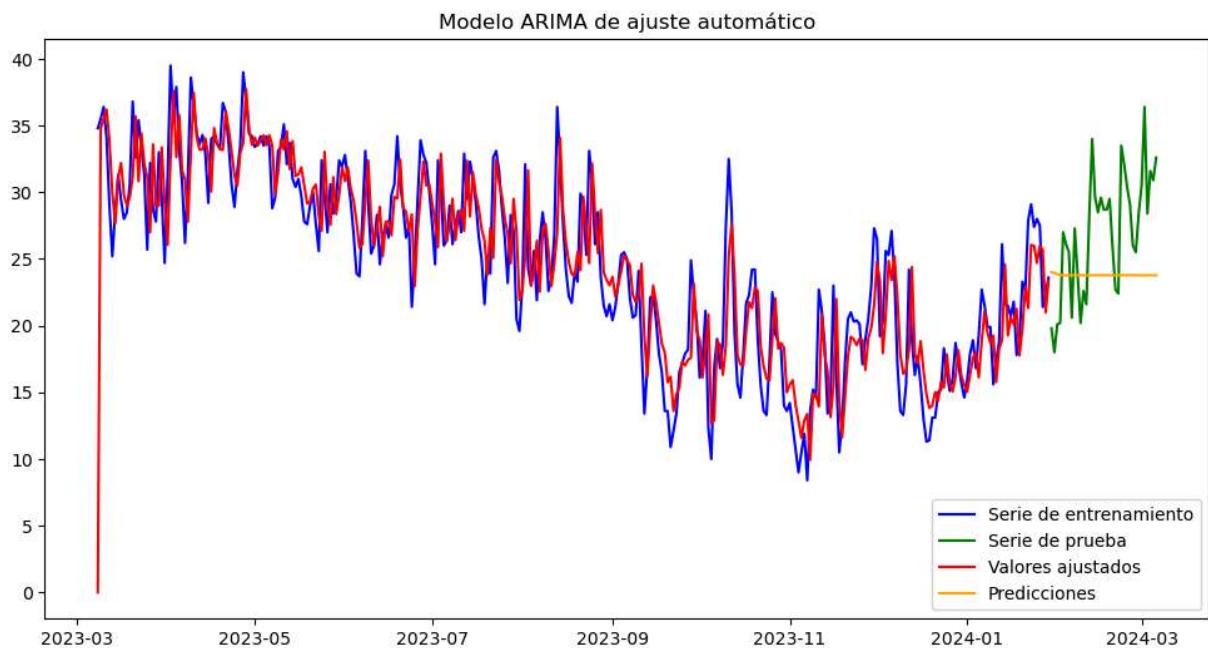
c:\Users\pablo\anaconda3\envs\mineria\_de\_datos\_ucm\lib\site-packages\pmdarima\arima\\_validation.py:62: UserWarning: m (12) set for non-seasonal fit. Setting to 0  
warnings.warn("m (%i) set for non-seasonal fit. Setting to 0" % m)

SARIMAX Results

Dep. Variable:	y	No. Observations:	328			
Model:	SARIMAX(2, 1, 1)	Log Likelihood	-832.867			
Date:	jue, 13 mar 2025	AIC	1673.733			
Time:	14:37:41	BIC	1688.893			
Sample:	03-08-2023 - 01-29-2024	HQIC	1679.782			
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
ar.L1	0.6951	0.063	10.974	0.000	0.571	0.819
ar.L2	-0.2455	0.055	-4.426	0.000	-0.354	-0.137
ma.L1	-0.8672	0.044	-19.735	0.000	-0.953	-0.781
sigma2	9.5223	0.711	13.385	0.000	8.128	10.917
Ljung-Box (L1) (Q):			0.00	Jarque-Bera (JB):		13.66
Prob(Q):			0.98	Prob(JB):		0.00
Heteroskedasticity (H):			1.13	Skew:		0.47
Prob(H) (two-sided):			0.51	Kurtosis:		3.36

#### Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).



El estadístico de Ljung-Box nos indica que no hay evidencia suficiente para considerar a los residuos correlados por lo tanto podemos decir que los residuos están incorrelados para el modelo de ajuste automático.

Ambos modelos son malos, casi tan malos como el del suavizado exponencial simple por lo que casi no habría diferencias entre elegir uno u otro. Dicho esto el modelo con arima de ajuste automático es ligeramente mejor porque arranca con valores algo más altos aunque después decaen. Entre los dos ARIMA el ARIMA de ajuste automático es mejor aunque de manera casi imperceptible

## 7 - Expresión Algebraica del modelo ARIMA seleccionado

Ya que en nuestro modelo ARIMA de ajuste automático tiene el parámetro  $m=0$ , el modelo resultante puede quedar entonces como un ARIMA( $p=2, d=1, q=1$ ).

La expresión algebraica para un ARIMA(2,1,1) con  $\varphi_1 = 0.6951$ ,  $\varphi_2 = 0.2455$  y  $\vartheta_1 = 0.8672$  (organizada para expresar  $Y(t)$ ) sería la siguiente:

$$Y_t = Y_{t-1} + 0.6951(Y_{t-1} - Y_{t-2}) - 0.2455(Y_{t-2} - Y_{t-3}) + \varepsilon_t - 0.8672\varepsilon_{t-1}$$

Donde:

- $Y_t$  es el valor de la serie temporal en el momento  $t$ .
- $B$  es el operador de retardo (lag), tal que  $B^k Y_t = Y_{t-k}$ .
- $\varphi_i$  son los parámetros del componente autorregresivo (AR).
- $\vartheta_i$  son los parámetros del componente de media móvil (MA).
- $d$  es el orden de diferenciación.

- $\varepsilon_t$  es el término de error (ruido blanco).

## 8,9 - Predicciones y Comparacion

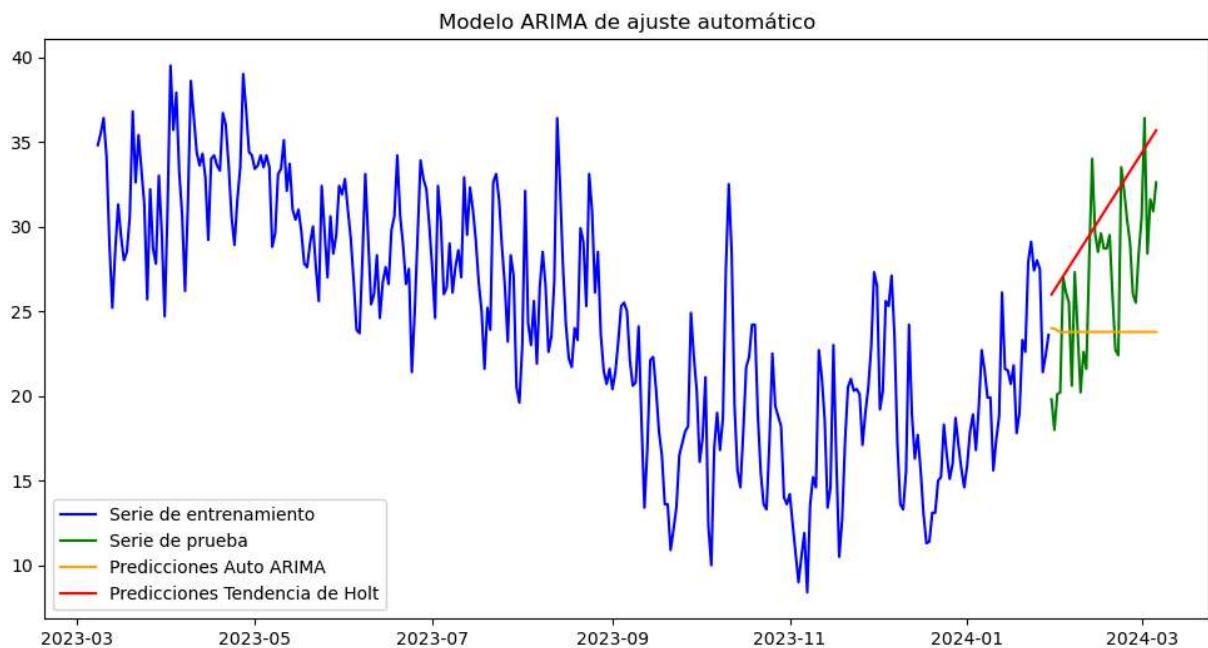
```
In [76]: comparison_table = pd.DataFrame({
    'Valor Real': gold_rosario_temperature_test,
    'Auto ARIMA': auto_arima_predictions,
    'Exponencial con tendencia de Holt': holt_forecast
})

print(tabulate(comparison_table, headers='keys'))

fitted_values = auto_model.predict_in_sample()
auto_arima_predictions = auto_model.predict(n_periods=len(gold_rosario_temperature_)

plt.figure(figsize=(12, 6))
plt.plot(gold_rosario_temperature_train, label='Serie de entrenamiento', color='blue')
plt.plot(gold_rosario_temperature_test.index, gold_rosario_temperature_test, label='Real')
plt.plot(gold_rosario_temperature_test.index, auto_arima_predictions, color='orange')
plt.plot(gold_rosario_temperature_test.index, holt_forecast, color='red', label='Predicción')
plt.title('Modelo ARIMA de ajuste automático')
plt.legend()
plt.show()
```

	Valor Real	Auto ARIMA	Exponencial con tendencia de Holt
2024-01-30 00:00:00	19.8	23.9932	25.9845
2024-01-31 00:00:00	18	23.9719	26.2537
2024-02-01 00:00:00	20.1	23.8605	26.5228
2024-02-02 00:00:00	20.2	23.7884	26.792
2024-02-03 00:00:00	27	23.7656	27.0611
2024-02-04 00:00:00	26.1	23.7674	27.3303
2024-02-05 00:00:00	25.5	23.7743	27.5994
2024-02-06 00:00:00	20.6	23.7786	27.8686
2024-02-07 00:00:00	27.3	23.78	28.1378
2024-02-08 00:00:00	23.8	23.7798	28.4069
2024-02-09 00:00:00	20.2	23.7794	28.6761
2024-02-10 00:00:00	22.6	23.7791	28.9452
2024-02-11 00:00:00	21.6	23.7791	29.2144
2024-02-12 00:00:00	28.2	23.7791	29.4835
2024-02-13 00:00:00	34	23.7791	29.7527
2024-02-14 00:00:00	29.6	23.7791	30.0218
2024-02-15 00:00:00	28.5	23.7791	30.291
2024-02-16 00:00:00	29.6	23.7791	30.5601
2024-02-17 00:00:00	28.7	23.7791	30.8293
2024-02-18 00:00:00	28.7	23.7791	31.0984
2024-02-19 00:00:00	29.5	23.7791	31.3676
2024-02-20 00:00:00	25.9	23.7791	31.6367
2024-02-21 00:00:00	22.7	23.7791	31.9059
2024-02-22 00:00:00	22.4	23.7791	32.175
2024-02-23 00:00:00	33.5	23.7791	32.4442
2024-02-24 00:00:00	32.1	23.7791	32.7133
2024-02-25 00:00:00	30.4	23.7791	32.9825
2024-02-26 00:00:00	29.1	23.7791	33.2517
2024-02-27 00:00:00	26	23.7791	33.5208
2024-02-28 00:00:00	25.5	23.7791	33.79
2024-02-29 00:00:00	28.4	23.7791	34.0591
2024-03-01 00:00:00	30.5	23.7791	34.3283
2024-03-02 00:00:00	36.4	23.7791	34.5974
2024-03-03 00:00:00	28.4	23.7791	34.8666
2024-03-04 00:00:00	31.6	23.7791	35.1357
2024-03-05 00:00:00	30.9	23.7791	35.4049
2024-03-06 00:00:00	32.6	23.7791	35.674



Los modelos no hicieron un buen trabajo prediciendo el comportamiento de la serie, en este sentido creo que en gran parte se debe a la serie en si, el comportamiento estacional es muy reducido y la tendencia es muy variable así como los residuos son significativos. Creo que, si el conjunto de datos hubiese representado el promedio mensual de temperatura para un rango de años hubiese sido ideal para poder captar un componente estacional significativo y esto hubiese mejorado mucho el desempeño de los modelos, sobre todo el modelo ARIMA.

En conclusion el conjunto de entrenamiento no capto la suficiente informacion como para poder ser de utilidad para predecir el comportamiento del conjunto de prueba de test. El que mejor se comporta es el Metodo de Suavizado Exponencial con Tendencia de Holt que lo hace muy bien prediciendo la tendencia pero que es incapaz de predecir la estacionalidad y ajustarse bien al conjunto de datos de test.