

Universidad Complutense de Madrid

MÁSTER EN BIG DATA DATA SCIENCE & INTELIGENCIA
ARTIFICIAL

MINERÍA DE DATOS Y MODELIZACIÓN
PREDICTIVA: SERIES TEMPORALES

Autor:
José D'Orazio

20 de marzo de 2025

Índice

1. Objetivo	2
2. Importación, Exploración y Limpieza de los Datos	2
2.1. División en Conjuntos de Entrenamiento y Prueba	3
3. Análisis de Estacionalidad	3
3.1. Descomposición de la Serie Temporal	3
4. Selección del Modelo de Suavizado	5
4.1. Modelo Holt-Winters	5
4.1.1. Errores del Modelo	5
4.1.2. Transformación Logarítmica	6
4.2. Errores del Modelo de Transformación Logarítmica	7
5. Evaluación y Corrección de Estacionariedad	7
6. Autocorrelación	8
6.1. Serie Diferencia Orden 1	9
6.2. Serie Diferenciada Orden 7	10
6.3. Autocorrelación de la Serie Diferenciada de Orden 7	11
7. Modelo SARIMAX	12
7.1. Validación del Modelo	13
8. Modelo ARIMA Auto	14
8.1. Validación	16
9. Conclusion	17
10. Bibliografía	18

1. Objetivo

El principal objetivo del análisis de este conjunto de datos es explorar los patrones, tendencia y estacionalidad para desarrollar modelos de pronóstico precisos. Para hacer un análisis de series temporales solo necesitamos las variables de tiempo y la variable a predecir, en este caso sería *Date* y *Sales*. La fecha tiene este formato *YYYY – MM – DD*.

1. Comprender los Patrones de Ventas: Identificar tendencias (si las ventas aumentan, disminuyen o se mantienen estables). Examinar fluctuaciones periódicas debido a diferencias entre los fines de semana y los días de semana.
2. Evaluar la Estacionalidad y Comportamiento Cíclico
3. Desarrollar Modelos de Pronóstico: Probar modelos como SARIMA y Suavizamiento Exponencial Comparar diferentes enfoques para determinar cuál ofrece la mayor precisión en las predicciones.

A continuación se irá explicando paso a paso y justificando cada decisión.

2. Importación, Exploración y Limpieza de los Datos

Se realiza la carga de las librerías e importación del conjunto de datos `sales.csv`, obtenido de Kaggle. Se utiliza el siguiente código.

```
train = pd.read_csv("sales.csv")
print(train.head())
if (train.isna().sum().sum() > 0):
    print(train.isna().sum())
train.info()
```

El conjunto de datos contiene 365 registros por columna y no presenta valores faltantes. Esto equivale a 365 días de datos. Sin embargo, cuenta con tres columnas, mientras que para la creación de una serie temporal solo se requieren dos: una para el tiempo y otra como variable predictora.

```
Unnamed: 0      Date      Sales
0      0  2021-10-01  29.109547
1      1  2021-10-02  30.720435
2      2  2021-10-03  30.305173
3      3  2021-10-04  27.390893
4      4  2021-10-05  22.841123
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Unnamed: 0  365 non-null    int64
1   Date        365 non-null    object
2   Sales       365 non-null    float64
dtypes: float64(1), int64(1), object(1)
memory usage: 8.7+ KB
```

Figura 1: Conjunto de Datos Original

El código que se muestra a continuación es el utilizado para eliminar la columna sobrante *Unnamed : 0* y asignar a la variable *Date* convertida en tipo de dato fecha como el nuevo índice.

```
train = train.drop(["Unnamed: 0"], axis = 1)
train["Date"] = pd.to_datetime(train["Date"])
train.set_index("Date", inplace = True)
```

2.1. División en Conjuntos de Entrenamiento y Prueba

Para finalizar esta sección se dividen el conjunto de datos en entrenamiento y prueba. Para ello, se crea una variable *test*, en la cuál se guardaran los último 20 registros de la variable *train* con el fin de comparar nuestras predicciones con los resultados verdaderos.

```
test = train.tail(20)
train = train.iloc[: -20]
```

3. Análisis de Estacionalidad

Estacionariedad significa que el conjunto de datos debería tener una media, varianza y covarianza constante.

Para descubrir si este conjunto es estacionario, se inicia por crear una representación gráfica de los datos de *train* como del *test*, con el fin de observar su comportamiento. Esto se encuentra en la siguiente figura.

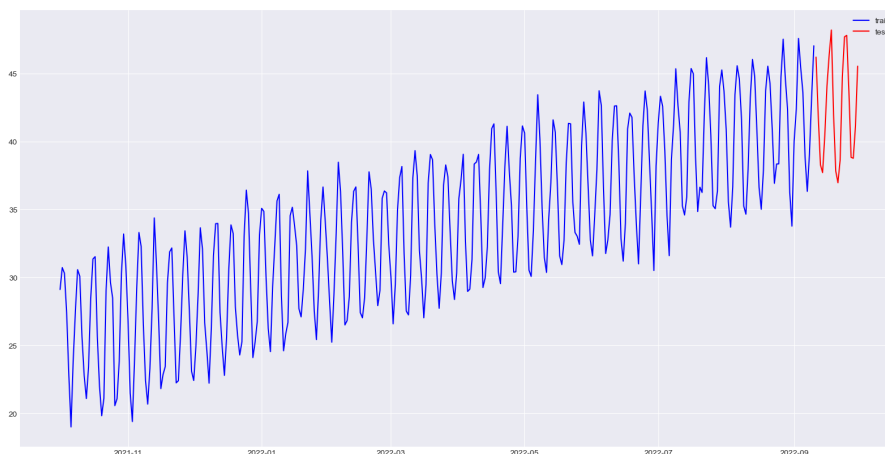


Figura 2: Gráfico de Train y Test

A simple vista se pueden sacar las siguientes conclusiones:

1. Existe una tendencia ascendente a lo largo del tiempo, lo que indica un crecimiento en la serie
2. La varianza es relativamente estable, aunque hay algunas fluctuaciones. La media no es constante debido a la tendencia creciente.
3. La serie parece ser heterocdástica, ya que la variabilidad cambia a lo largo del tiempo aunque de manera gradual
4. La serie no es estacionaria, dado que tanto la media como la variabilidad presentan cambios a lo largo del tiempo
5. La serie presenta estacionalidad, ya que exhibe un patrón recurrente en ciertos periodos, lo que sugiere la existencia de ciclos repetitivos.

3.1. Descomposición de la Serie Temporal

Un conjunto de datos que fluctue alrededor de la tendencia y además, presenta valores homogéneos, suele tener como preferencia un esquema aditivo. El siguiente código presenta el calculo de descomposición estacional.

```
result = seasonal_decompose(train["Sales"], model = "additive")
```



Figura 3: Descomposición Estacional

La figura anterior muestra la descomposición de la serie temporal en sus componentes principales.

1. Serie Observada

- Presenta un patrón cíclico con variaciones periódicas
- Hay una tendencia alcista

2. Tendencia

- Muestra un incremento gradual y sostenido en el tiempo
- Hay momentos de suavización en el crecimiento, lo que puede indicar cambios en el comportamiento de la variable

3. Estacionalidad

- Hay un patrón repetitivo claro, lo que confirma que la serie es estacional.
- La amplitud parece relativamente constante (-5, +5), lo que indica que la variabilidad estacional no cambia drásticamente

4. Residuo

- Representa las fluctuaciones no explicadas por la tendencia y la estacionalidad
- Hay variabilidad pero no hay un patrón claro, lo que sugiere que la descomposición ha capturado bien los componentes principales
- Algunos picos son más elevados, esto puede indicar valores atípicos

4. Selección del Modelo de Suavizado

En esta sección se aplican diferentes métodos de suavizado para analizar la evolución de la serie temporal. Algunos modelos fueron descartados previamente, ya que no se ajustaban a las características de la serie en estudio.

El **modelo de alisado simple** fue descartado, ya que está diseñado para series sin tendencia ni estacionalidad. Dado que la serie presenta ambos componentes, este método resulta inadecuado.

El **modelo de alisado doble** permite modelar la tendencia, pero no la estacionalidad. Dado que nuestra serie exhibe un comportamiento estacional, este método también fue descartado.

Finalmente, el **modelo de alisado Holt-Winters** fue seleccionado, ya que es el más adecuado para series con con tendencia y estacionalidad.

4.1. Modelo Holt-Winters

Para crear el modelo de Holt-Winters en Python se utiliza el siguiente código:

```
modelo_holt_winters = sm.tsa.ExponentialSmoothing(train_TR["Sales"], trend = "a",
seasonal = "additive", seasonal_periods = 7).fit()
predicciones_hw = modelo_holt_winters.forecast(steps = 20)
modelo_holt_winters.summary()
```

Los parámetros para la función `sm.tsa.ExponentialSmoothing` se escogieron utilizando el siguiente criterio:

- **train_TR["Sales"]**: son los datos de entrenamiento.
- **trend=.add**: indica que la tendencia se ajusta de forma aditiva, es decir, el crecimiento de la serie en el tiempo se modela sumando un valor constante en cada período. La tendencia presenta un movimiento lineal, en vez de parabolico.
- **seasonal=.additive**: modela la estacionalidad de forma aditiva. Se ha elegido dado que la amplitud de la estacionalidad es aproximadamente constante a lo largo del tiempo.
- **seasonal_periods=7**: se observa que los períodos estacionales suelen repetirse aproximadamente cada 7 días, es decir, muestra una estacionalidad semanal.

Para hacer el `forecast()` se escogio un step de 20. Este número representa 20 días, justamente la cantidad reservada en el conjunto de datos de test. La siguiente figura muestra el conjunto observado, el test y el modelo Holt-Winters.

4.1.1. Errores del Modelo

Para evaluar la precisión del modelo de predicción se calcularon el error absoluto y el error absoluto medio (MAE). La siguiente figura muestra la evolución del valor del error absoluto por día.

Se obtiene el error absoluto medio para el modelo: 0,70488. So consideramos que el valor medio de *Sales* en el conjunto *test* es de 45, este error representa aproximadamente 1,58 %, lo que indica un buen desempeño en términos relativos.

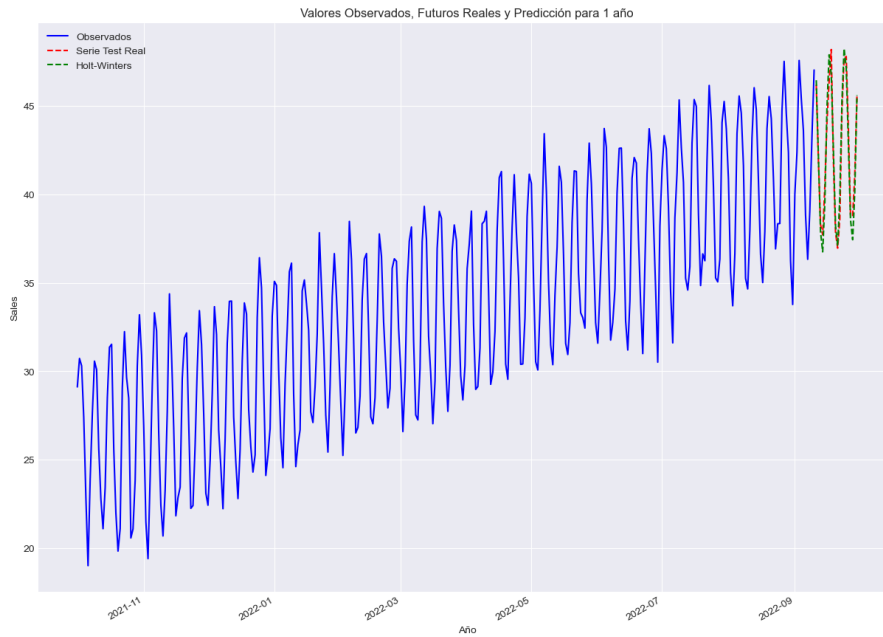


Figura 4: Gráfico Modelo Holt-Winters

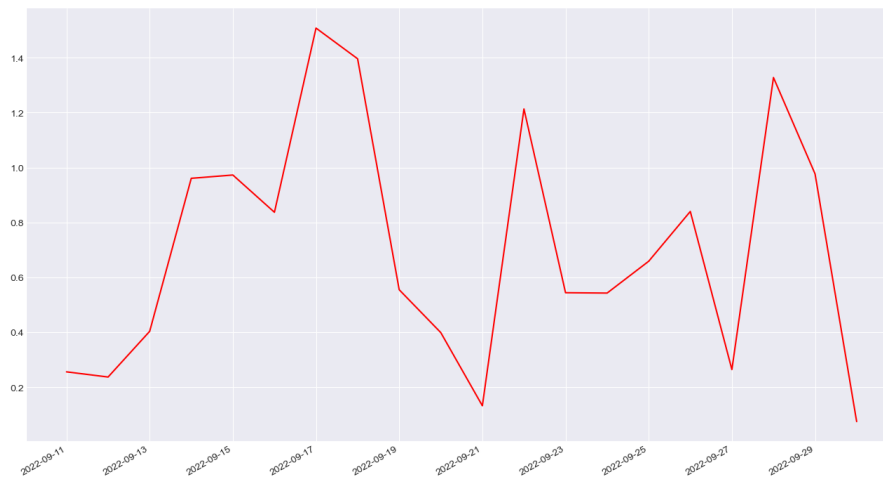


Figura 5: Gráfico Error Valor Absoluto por Día

4.1.2. Transformación Logarítmica

Para mejorar la estabilidad de la serie temporal, se aplicó una transformación logarítmica, cuyo objetivo principal es reducir la heterocedasticidad, la variabilidad no constante, y hacer que la serie sea más adecuada para modelos aditivos. Esta transformación es especialmente útil cuando la amplitud de la estacionalidad crece con el tiempo.

Además, se aplicó una diferenciación a la serie logarítmica, un proceso que permite eliminar tendencias de largo plazo y convertir la serie en estacionaria, una propiedad fundamental para muchos modelos de series temporales, incluidos los métodos de suavizamiento exponencial como Holt-Winters. El siguiente código muestra los pasos realizados. Los parámetros utilizados son los mismo que en el primer modelo de Holt-Winters.

```
train_log = np.log(train_TR[ 'Sales '])
train_log_diff = train_log.diff().dropna()
modelo_hw_diff = sm.tsa.ExponentialSmoothing(train_log_diff, trend='add',
    seasonal='add', seasonal_periods=7).fit()
pred_diff = modelo_hw_diff.forecast(steps=20)
```

Una vez obtenidas las predicciones en la escala logarítmica diferenciada, es necesario reconstruir la serie original. Para ello, se revierte la diferenciación acumulando los valores predichos y se aplica la exponenciación para deshacer la transformación logarítmica.

```
last_log_value = train_log.iloc[-1]
pred_log = last_log_value + pred_diff.cumsum()
pred_final = np.exp(pred_log)
```

Realizar la combinación de transformación logarítmica, diferenciación y el modelo Holt-Winters permitió generar predicciones con una baja tasa de error, asegurando que la serie sea más estable y adecuada para el análisis.

La siguiente figura muestra la comparación entre el modelo logarítmico y el modelo básico frente a los datos del conjunto de prueba.

Ambos modelos presentan un comportamiento muy similar; sin embargo, se observa que el modelo logarítmico se extiende más en la parte inferior, lo que sugiere una mejor cobertura de los márgenes de error.

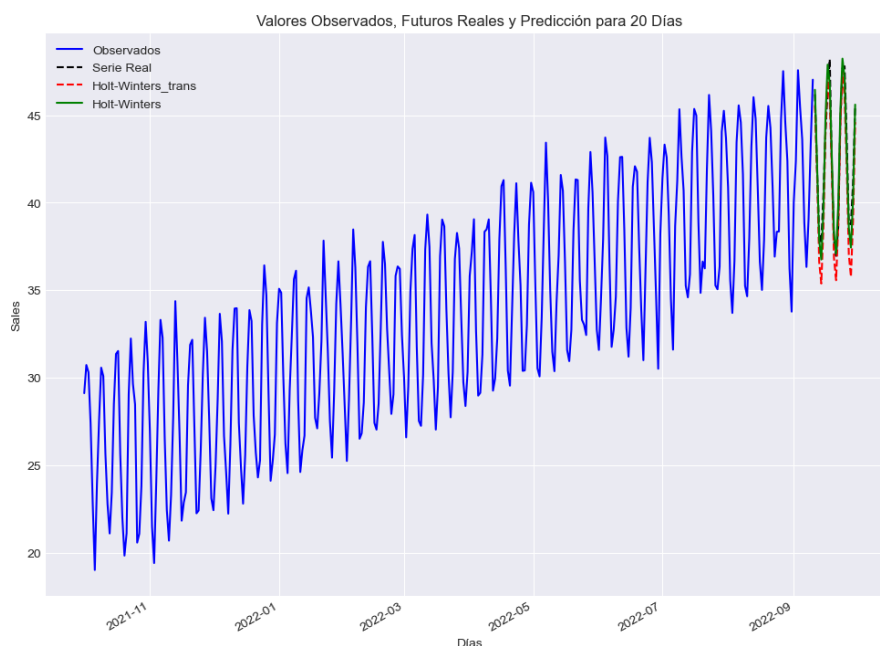


Figura 6: Gráfico de Holt-Winters-Trans

4.2. Errores del Modelo de Transformación Logarítmica

Se evalúa de la misma forma que para el modelo original. Se obtiene un MAE de 1.228. Esto representaría un error del 2,79 %.

5. Evaluación y Corrección de Estacionariedad

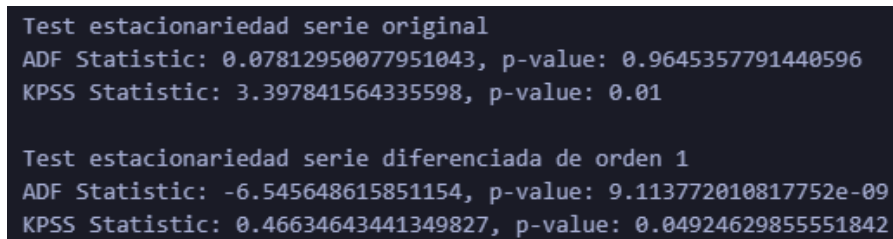
Al igual que el modelo de Holt-Winters, el modelo de Arima también requiere que los datos sean estacionarios, es decir, que su media y varianza sean constantes en el tiempo.

Para verificar esto se pueden hacer dos pruebas estadísticas: Augmented Dickey-Füller Test (ADF) y Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS).

La primera tiene la hipótesis Nula, que la serie no es estacionaria, y por lo tanto si el p -value es menor a 0,05 es rechazada y la serie es estacionaria.

La segunda tiene como hipótesis Nula, que la serie es estacionaria. Si aplicamos el siguiente código, obtenemos el resultado de la siguiente figura que resume los resultados del código.

```
datos_diff_1 = datos.diff(1).dropna()
print('Test estacionariedad serie original')
print(f'ADF Statistic: {adfuller(datos)[0]}, p-value: {adfuller(datos)[1]}')
print(f'KPSS Statistic: {kpss(datos)[0]}, p-value: {kpss(datos)[1]}')
print('\nTest estacionariedad serie diferenciada de orden 1')
print(f'ADF Statistic: {adfuller(datos_diff_1)[0]}, p-value: {adfuller(datos_diff_1)[1]}')
print(f'KPSS Statistic: {kpss(datos_diff_1)[0]}, p-value: {kpss(datos_diff_1)[1]}')
```



```
Test estacionariedad serie original
ADF Statistic: 0.07812950077951043, p-value: 0.9645357791440596
KPSS Statistic: 3.397841564335598, p-value: 0.01

Test estacionariedad serie diferenciada de orden 1
ADF Statistic: -6.545648615851154, p-value: 9.113772010817752e-09
KPSS Statistic: 0.46634643441349827, p-value: 0.04924629855551842
```

Figura 7: ADF y KPSS orden 1

Conclusiones:

1. Serie Original:

- ADF Statistic = 0.0781, p-value = 0.9645. No se rechaza Hipótesis Nula, lo que indica que la serie no es estacionaria.
- KPSS Statistic = 3.3978, p-value = 0.01. Se rechaza Hipótesis Nula, lo que confirma que la serie no es estacionaria.

2. Serie Diferenciada:

- ADF Statistic = -6.5456, p-value = 9.11e-09. Se rechaza Hipótesis Nula, lo que indica que la serie ahora es estacionaria.
- KPSS Statistic = 0.4663, p-value = 0.0492. No se rechaza Hipótesis Nula, lo que confirma que la serie es estacionaria.

Una vez transformada la serie a estacionaria se puede empezar el análisis de autocorrelación.

6. Autocorrelación

Se distingue entre la autocorrelación simple (ACF) y parcial (PACF).

La autocorrelación ACF ayuda a identificar si una serie es o no estacionaria, mientras que la parcial PACF ayuda a identificar el modelo autorregresivo a utilizar.

Análisis del gráfico superior:

- Presenta una disminución lenta de la autocorrelación, pero aún se observan algunos valores altos.
- Esto sugiere que la serie aún puede tener cierta estructura de dependencia en el tiempo, lo que indica que la diferenciación puede no haber sido suficiente o que la serie tiene una componente de media móvil.

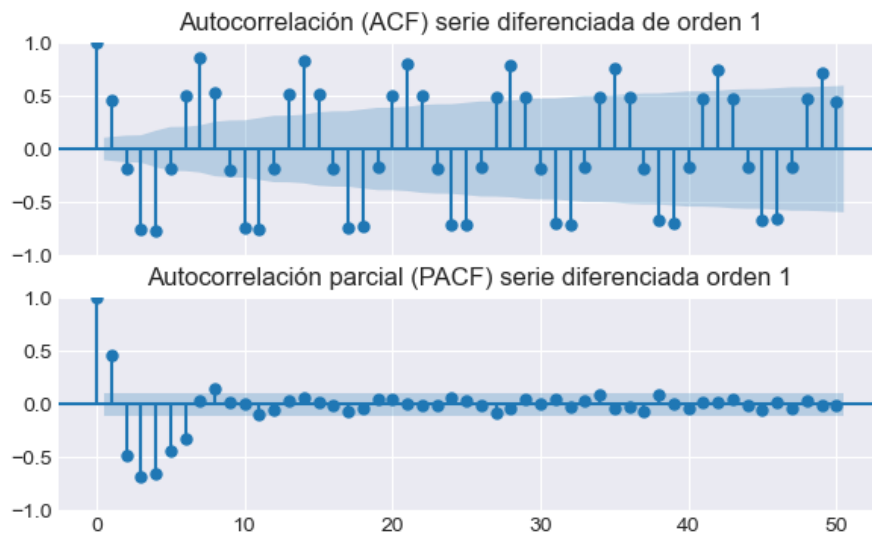


Figura 8: Autocorrelación ACF y PACF orden 1

Análisis del gráfico inferior:

- Muestra un corte brusco después de los primeros rezagos, lo cual es característico de un proceso $AR(p)$.
- La presencia de valores significativos en los primeros rezagos indica que la serie podría ser modelada con un proceso ARMA o ARIMA con un componente autorregresivo.

Se obtienen las siguientes conclusiones de las autocorrelaciones:

- La ACF muestra una disminución lenta, lo que puede indicar que un modelo $MA(q)$ o un modelo mixto ARMA sería adecuado.
- La PACF apunta a un modelo ARIMA con un componente AR de bajo orden.

Conclusiones preliminares: La serie ya ha sido diferenciada una vez, lo que sugiere que podría haber sido originalmente no estacionaria. La ACF muestra una disminución lenta, lo que podría indicar que un modelo $MA(q)$ o un modelo mixto ARMA sería adecuado.

6.1. Serie Diferencia Orden 1

Para poder seleccionar el modelo, primero se integrará la descomposición con el análisis de ACF y PACF previamente hecho. Esto permitirá hacer una descripción integral que ayude a comprender la estructura subyacente de los datos y a determinar los valores óptimos de los parámetros ARIMA.

De la siguiente figura se puede observar lo siguiente:

1. Serie Observada

- Se ve una fuerte estacionalidad y fluctuaciones regulares

2. Tendencia

- No se aprecia una tendencia clara, es bastante errática, esto indica que la diferenciación ya eliminó cualquier tendencia significativa.

3. Estacionalidad

- Presenta patrones repetitivos en intervalos regulares.
- La serie sigue teniendo una fuerte estacionalidad, lo que indica que la diferenciación de primer orden no eliminó por completo la componente estacional. Para modelar con ARIMA es necesario aplicar otra diferenciación.

4. Residuo

- Se ven ciertas fluctuaciones y correlaciones, lo que sugiere que aún hay información en la serie que podría ser modelada con ARMA o ARIMA.

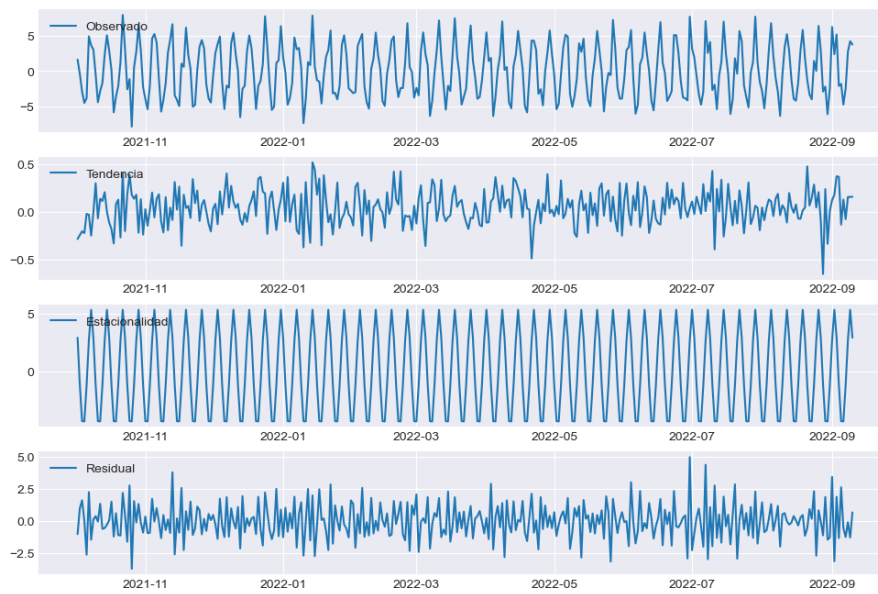


Figura 9: Descomposicion Estacional orden 1

6.2. Serie Diferenciada Orden 7

Se decide aplicar la función de autocorrelación (ACF) y autocorrelación parcial (PACF) sobre una serie diferenciada de orden 7, debido a que la estacionalidad observada parece repetirse semanalmente.

```
datos_diff_1_7 = datos_diff_1.diff(7).dropna()
print('Test estacionariedad serie de orden 7')
print('\nTest estacionariedad serie diferenciada de orden 7')
print(f'ADF Statistic: {adfuller(datos_diff_1_7)[0]}, p-value: {adfuller(datos_diff_1_7)[1]}')
print(f'KPSS Statistic: {kpss(datos_diff_1_7)[0]}, p-value: {kpss(datos_diff_1_7)[1]}')
```

Los resultados de los tests de estacionariedad muestran que la serie es estacionaria.

```
Test estacionariedad serie de orden 7
Test estacionariedad serie diferenciada de orden 7
ADF Statistic: -9.58159604621037, p-value: 2.1511532423012928e-16
KPSS Statistic: 0.09138752664055161, p-value: 0.1
```

Figura 10: Resultados ADF y KPSS orden 7

En la siguiente figura se observa que la estacionalidad se ha reducido, la tendencia es más estable y los residuos muestran un comportamiento más aleatorio que antes. Dado que la componente estacional aún persiste, un modelo SARIMA con $D = 1$ podría ser una opción adecuada.

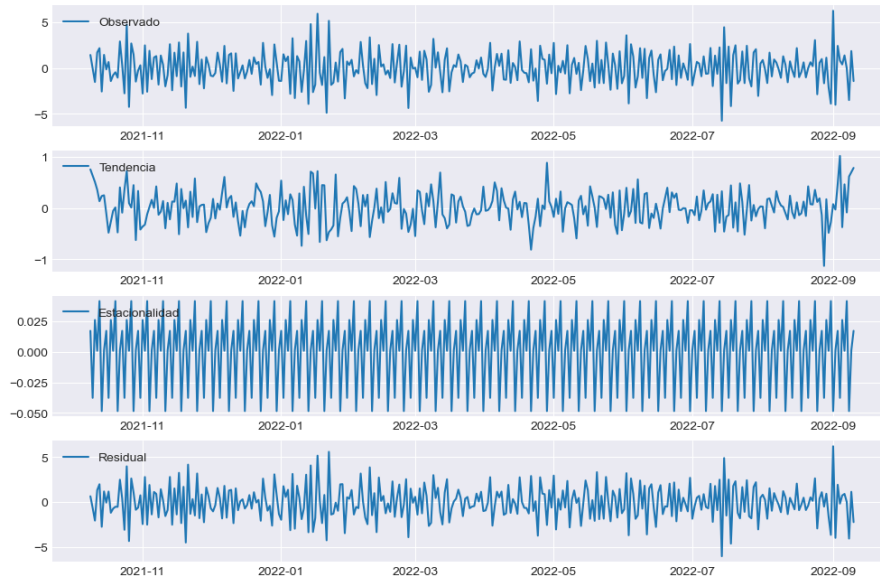


Figura 11: Descomposición Estacional orden 7

6.3. Autocorrelación de la Serie Diferenciada de Orden 7

La siguiente figura muestra la función de autocorrelación y autocorrelación parcial aplicadas a la serie diferenciada de orden 7.

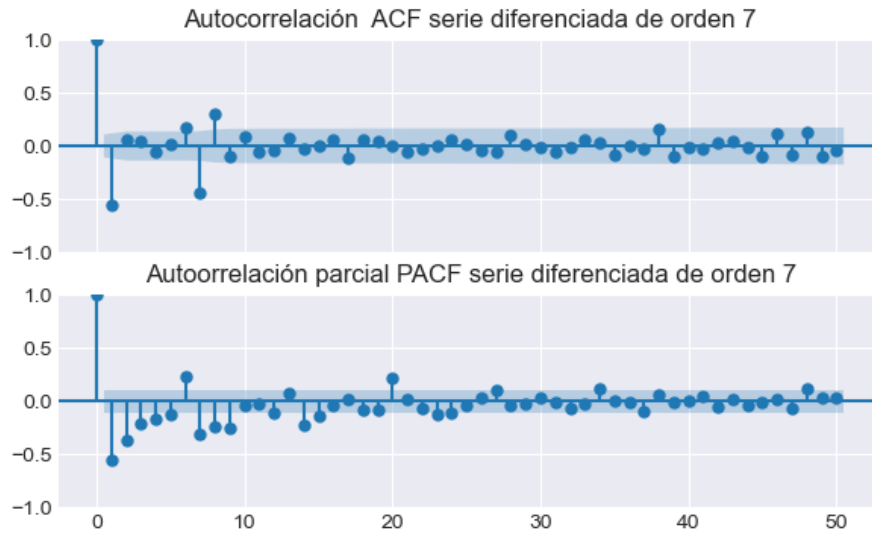


Figura 12: Autocorrelación ACF y PACF orden 7

- La serie es estacionaria tras aplicar una diferenciación de orden 7, por lo que se establece $d = 1$ y $D = 1$ para eliminar la tendencia y la estacionalidad semanal.
- El PACF muestra un corte significativo en el lag 1, lo que sugiere un orden autoregresivo de $p = 1$. También se observa un pico en el lag 7, lo que justifica la elección de $P = 1$.
- El ACF decrece rápidamente después del primer lag y oscila cerca de 0, lo que sugiere $q = 1$. Al igual que en PACF, se detecta un pico en el lag 7, por lo que se establece $Q = 1$.

7. Modelo SARIMAX

Para la creación del modelo los valores utilizados serán los siguientes:

- $d = 1, D = 1$. Para eliminar tendencia y estacionalidad.
- $p = 1, P = 1$. Basado en la ACF y PACF.
- $q = 1, Q = 1$ Basado en la ACF y PACF en los lags estacionales ($s = 7$).

Se insertan en el código. Los parámetros en la función SARIMAX, son los antes mencionados. En la función `acorr_ljungbox` se utilizan lags 7, 14 y 50. El primero representa la estacionalidad en 7 días, el segundo el doble del período estacional y el último la cantidad de lags utilizados para la autocorrelación.

La expresión algebraica para representar SARIMAX(1,1,1)(1,1,1,7) sería:

$$[h](1 - \phi_1 L) \cdot (1 - \Phi_1 L^7) \cdot (1 - L) Y_t = (1 - \theta_1 L) \cdot (1 - \Theta_1 L^7) \cdot \epsilon_t \quad (1)$$

De tu salida de SARIMAX Results:

- L equivale al operador.
- ϵ_t es un error aleatorio.
- $\phi_1 = -0,1192$. Coeficiente del término autorregresivo (AR)
- $\Phi_1 = 0,0890$. Coeficiente de la parte estacional autorregresiva (SAR).
- $\theta_1 = -0,9317$. Coeficiente del término de media móvil (MA).
- $\Theta = (-) 0.997$ Coeficiente de la parte estacional de media móvil (SMA).

```
warnings.filterwarnings("ignore", category=UserWarning,
message='Non-invertible | Non-stationary ')
modelo = SARIMAX(endog = train, order =
(1, 1, 1), seasonal_order = (1, 1, 1, 7))
modelo_res = modelo.fit(dispatch=0)
warnings.filterwarnings("default")
modelo_res.summary()
from statsmodels.stats.diagnostic import acorr_ljungbox
estadistico_lb, p_valor_lb = acorr_ljungbox
(modelo_res.resid, lags=[7, 14, 50])

predicciones = modelo_res.get_forecast(steps=20)
predicciones_statsmodels = predicciones.predicted_mean
intervalo_conf = predicciones.conf_int(alpha=0.05)
print(f"Intervalo Confianza = {intervalo_conf.head()}")
predicciones_statsmodels.name = 'predicciones_statsmodels'
display(predicciones_statsmodels.head(4))
```

El código anterior permite calcular el intervalo de confianza y muestra las predicciones. Como podemos ver las predicciones se encuentran del intervalo de confianza para la mayoría de las predicciones. Esto significa que, con un 95% de confianza, las ventas reales para ese día caerán dentro de este rango.

La figura muestra las predicciones del modelo SARIMAX junto con las muestras de *test* reales. El modelo parece ajustarse perfectamente.

SARIMAX Results						
Dep. Variable:		Sales		No. Observations:		345
Model:		SARIMAX(1, 1, 1)x(1, 1, 1, 7)			Log Likelihood	-484.731
Date:		Thu, 20 Mar 2025			AIC	979.463
Time:		21:56:24			BIC	998.563
Sample:		10-01-2021			HQIC	987.076
- 09-10-2022						
Covariance Type:				opg		
	coef	std err	z	P> z	[0.025	0.975]
ar.L1	-0.1192	0.056	-2.121	0.034	-0.229	-0.009
ma.L1	-0.9317	0.024	-38.321	0.000	-0.979	-0.884
ar.S.L7	0.0890	0.064	1.386	0.166	-0.037	0.215
ma.S.L7	-0.9997	6.366	-0.157	0.875	-13.476	11.477
sigma2	0.9512	6.031	0.158	0.875	-10.869	12.772
Ljung-Box (L1) (Q):		0.00	Jarque-Bera (JB):		1.61	
Prob(Q):		0.94	Prob(JB):		0.45	
Heteroskedasticity (H):		0.92	Skew:		0.14	
Prob(H) (two-sided):		0.65	Kurtosis:		2.81	

Figura 13: Resultados Arima Manual

Intervalo	Confianza =	lower Sales	upper Sales
2022-09-11	44.433943	48.295664	
2022-09-12	40.312918	44.179522	
2022-09-13	35.976196	39.853450	
2022-09-14	34.712487	38.596521	
2022-09-15	37.444558	41.335707	
2022-09-11	46.364803		
2022-09-12	42.246220		
2022-09-13	37.914823		
2022-09-14	36.654504		
Freq: D, Name: predicciones_statsmodels, dtype: float64			

Figura 14: Intervalo de Confianza y Predicción Arima Manual

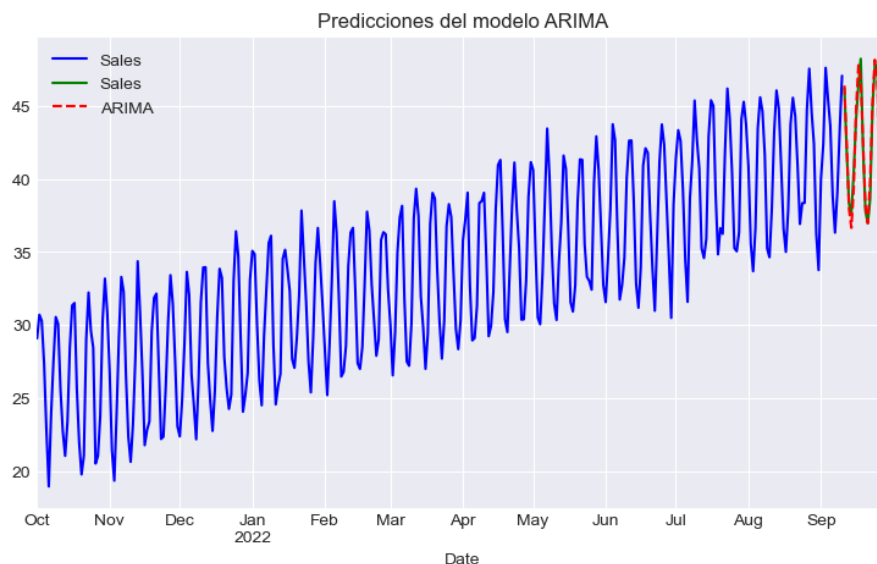


Figura 15: Grafica Arima Manual

7.1. Validación del Modelo

Al igual que se hizo con el modelo Holt-Winters, con este también se observan los errores producidos entre la predicción y los datos reales.

Se observa que los valores obtenidos como errores son muy bajos. Tomando 45 como el promedio de *Sales* para la predicción, el MSE y MAE dde 0.69 y 0.69 respectivamente in-

```
Mean Squared Error (MSE): 0.6904600132812322
Root Mean Squared Error (RMSE): 0.8309392356130695
Mean Absolute Error (MAE): 0.6946959263284562
```

Figura 16: Errores Arima Manual

dican un error de alrededor del 1.5 %, mientras que el RMSE de 0,83 indica un error de aproximadamente 1.8 %, lo cual es muy cercano al valor obtenido en Holt-Winters.

8. Modelo ARIMA Auto

Finalmente, se realiza un modelo ARIMA utilizando los valores obtenidos por el siguiente código. Este encontrará los valores más óptimos para la creación del modelo.

```
modelo = auto_arima(
    y                = train ,
    start_p          = 0 ,
    start_q          = 0 ,
    max_p            = 3 ,
    max_q            = 3 ,
    seasonal         = True ,
    test             = 'adf' ,
    m                = 7, # periodicidad de la estacionalidad
    d                = None, # El algoritmo determina 'd'
    D                = None, # El algoritmo determina 'D'
    trace            = True ,
    error_action      = 'ignore' ,
    suppress_warnings = True ,
    stepwise         = True
)
```

Se obtiene el siguiente resultado: La expresión algebraica para representar SARIMAX(1,1,1)(1,1,1,7)

```
Best model: ARIMA(3,0,0)(1,0,2)[7] intercept
Total fit time: 101.328 seconds
```

Figura 17: Valores Óptimos Arima Auto

sería:

$$[h](1 - \phi_1 L) \cdot (1 - \Phi_1 L^7) \cdot (1 - L) Y_t = (1 - \theta_1 L) \cdot (1 - \Theta_1 L^7) \cdot \epsilon_t \quad (2)$$

Estos se insertan en el código y obtenemos la siguiente figura. Como Se puede apreciar tanto el ARIMA que se ha deducido como el ARIMA auto se asemejan mucho a los valores reales de TEST.

El código anterior permite calcular el intervalo de confianza y muestra las predicciones. Como podemos ver las predicciones se encuentran del intervalo de confianza para la mayoría de las predicciones. sto significa que, con un 95 % de confianza, las ventas reales para ese día caerán dentro de este rango.

La figura muestra las predicciones del modelo SARIMAX auto y manual junto con las muestras de test reales. Ambos modelos se ajustan muy bien al verdadero.

SARIMAX Results						
Dep. Variable:	Sales		No. Observations:		345	
Model:	SARIMAX(3, 0, 0)x(1, 0, [1, 2], 7)			Log Likelihood	-547.301	
Date:	Thu, 20 Mar 2025			AIC	1108.601	
Time:	21:58:20			BIC	1135.506	
Sample:	10-01-2021			HQIC	1119.316	
- 09-10-2022						
Covariance Type:		opg				
	coef	std err	z	P> z	[0.025	0.975]
ar.L1	0.2434	0.048	5.093	0.000	0.150	0.337
ar.L2	0.3919	0.050	7.842	0.000	0.294	0.490
ar.L3	0.3625	0.049	7.452	0.000	0.267	0.458
ar.S.L7	1.0000	2.88e-05	3.47e+04	0.000	1.000	1.000
ma.S.L7	-0.8991	0.036	-25.261	0.000	-0.969	-0.829
ma.S.L14	-0.0758	0.015	-5.056	0.000	-0.105	-0.046
sigma2	1.2080	0.098	12.347	0.000	1.016	1.400
Ljung-Box (L1) (Q):	2.00	Jarque-Bera (JB):	1.22			
Prob(Q):	0.16	Prob(JB):	0.54			
Heteroskedasticity (H):	0.94	Skew:	0.07			
Prob(H) (two-sided):	0.76	Kurtosis:	2.75			

Figura 18: Arima Auto

Intervalo	Confianza =	lower Sales	upper Sales
2022-09-11	43.504901	47.823005	
2022-09-12	39.278594	43.722086	
2022-09-13	34.838030	39.688596	
2022-09-14	33.230525	38.663489	
2022-09-15	35.810012	41.513362	
2022-09-11	46.364803		
2022-09-12	42.246220		
2022-09-13	37.914823		
2022-09-14	36.654504		
Freq: D, Name: predicciones_statsmodels, dtype: float64			

Figura 19: Intervalo de Confianza y Predicción Arima Auto

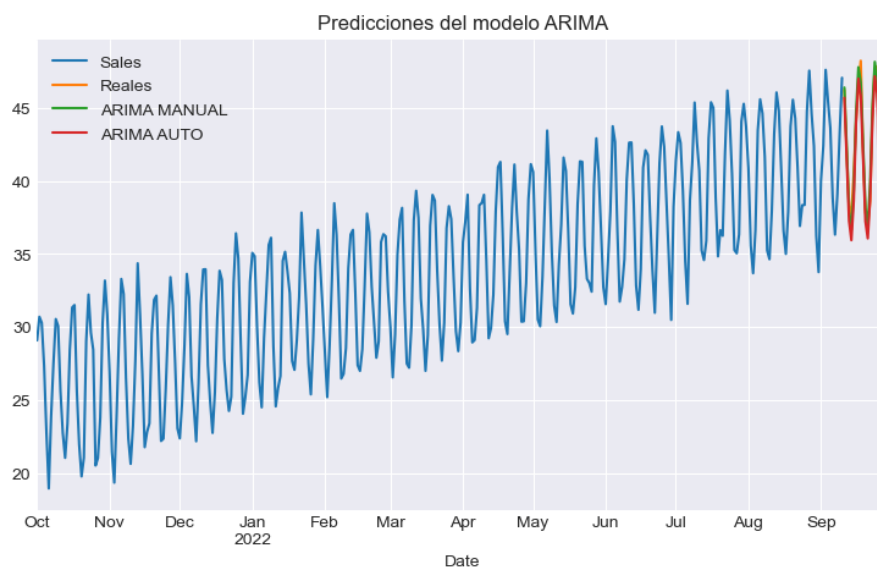


Figura 20: Gráfico Arima Auto

8.1. Validación

Se observa que los valores obtenidos como errores son bajos. Tomando 45 como el promedio de *Sales* para la predicción, el MSE y MAE de 1.871 y 1.141 respectivamente indican un error de alrededor del 4.16 % y 2.53 %, mientras que el RMSE de 1.37 indica un error de aproximadamente 3.04 %, estos son mucho más elevados que los dos modelos anteriores.

```
Mean Squared Error (MSE): 1.8714711979392848  
Root Mean Squared Error (RMSE): 1.3680172506000372  
Mean Absolute Error (MAE): 1.1413157687996514
```

Figura 21: Errores de Arima Auto

9. Conclusion

De acuerdo con las métricas de error (MAE, MSE, RMSE) y la figura 22, el modelo **ARIMA manual** es el más preciso, con los valores más bajos en todas las métricas disponibles. El modelo Holt-Winters también muestra buen rendimiento en MAE, aunque carece de métricas completas para una comparación más exhaustiva. Por otro lado, el modelo ARIMA Auto presenta los peores resultados, especialmente en MAE, MSE y RMSE, lo que sugiere un rendimiento inferior. En resumen, **ARIMA manual** es el modelo más adecuado para las predicciones de esta serie temporal.

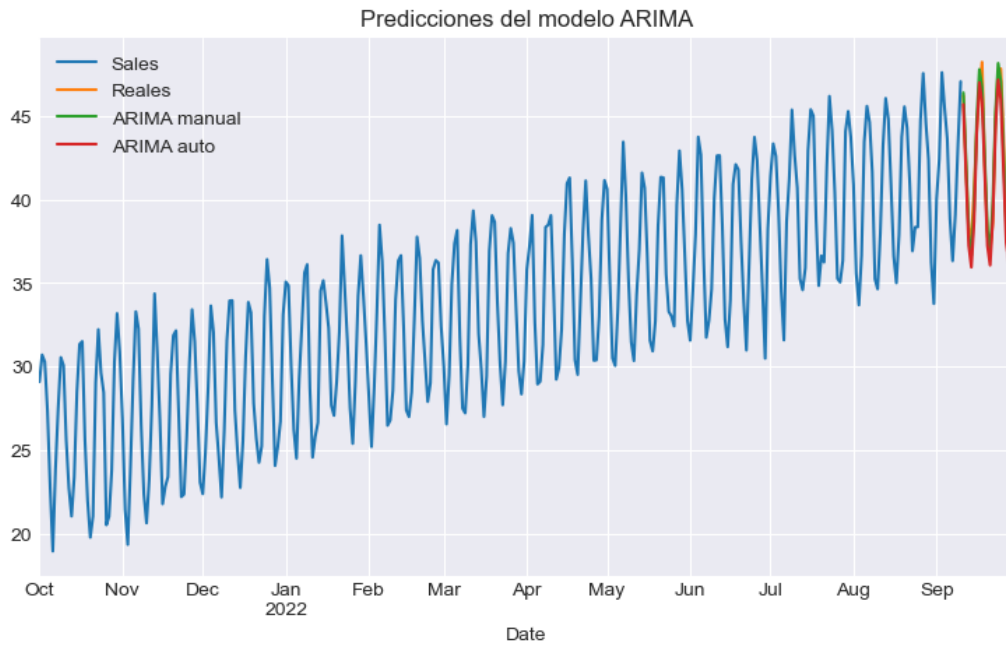


Figura 22: Gráfico de elección del modelo

MODELO	MAE	MSE	RMSE
Holt-Winters	0.704	-	-
Holt-Winters-Trans	1.228	-	-
ARIMA Manual	0.690	0.694	0.830
ARIMA Auto	1.871	1.141	1.368

Cuadro 1: Comparación de modelos

10. Bibliografia

<https://www.kaggle.com/datasets/sudipmanchare/simulated-sales-data-with-timeseries-features>