Series de Tiempo

Data Science (CC3084) - 2026

Series de Tiempo

Semestre 02, 2026

Definición

Una serie de tiempo es una secuencia de observaciones de una variable registradas en orden cronológico, generalmente a intervalos regulares (cada hora, día, mes, año).

Lo que la hace especial

- El orden importa: no podemos reordenar las observaciones como en un dataset tabular normal.
- Las observaciones cercanas en el tiempo suelen estar correlacionadas (autocorrelación).
- El objetivo típico es pronosticar (forecasting): estimar valores futuros a partir del pasado.

Ejemplos comunes

- Precio de una acción día a día.
- Ventas mensuales de una tienda.

- Temperatura horaria de una ciudad.
- Consumo eléctrico de un país.
- Número de casos de una enfermedad por semana.

Notación

Denotamos la serie como $y_1, y_2, \dots, y_t$, donde y_t es el valor en el instante t. Pronosticar significa estimar $y_{t+1}, y_{t+2}, \dots$ usando la información disponible hasta t.

Componentes de una Serie de Tiempo

Casi cualquier serie puede descomponerse en cuatro componentes que la explican.

Los cuatro componentes

- Tendencia (trend): dirección de largo plazo, sube o baja de forma sostenida.
- Estacionalidad (seasonality): patrones que se repiten en un período fijo y conocido (cada 12 meses, cada 7 días).
- Ciclo (cycle): oscilaciones de largo plazo sin período fijo (ciclos económicos).
- Ruido / irregular (residual): variación aleatoria que no se explica con lo anterior.

Descomposición

- Aditiva: $y_t = T_t + S_t + R_t$. Se usa cuando la amplitud estacional es constante.
- Multiplicativa: $y_t = T_t \times S_t \times R_t$. Se usa cuando la amplitud estacional crece con el nivel de la serie.

Descomposición con Python

```
from statsmodels.tsa.seasonal import seasonal_decompose

resultado = seasonal_decompose(serie, model="additive", period=12)
resultado.plot()
```

Estacionariedad

Una serie es estacionaria cuando sus propiedades estadísticas no cambian con el tiempo. Es el supuesto central de los modelos clásicos (AR, MA, ARMA, ARIMA).

Condiciones (estacionariedad débil)

- Media constante en el tiempo.
- Varianza constante en el tiempo.
- La autocovarianza depende solo de la distancia entre observaciones (el lag), no del instante.

Por qué importa

Una serie estacionaria es predecible en su estructura: lo que aprendemos del pasado sigue siendo válido en el futuro. Si hay tendencia o estacionalidad, esos patrones cambian la media o la varianza y rompen el supuesto.

Cómo detectarla

- Visualmente: graficar la serie y buscar tendencia o cambios de varianza.
- Prueba de Dickey-Fuller Aumentada (ADF): la hipótesis nula es que la serie NO es estacionaria; si el p-valor es menor a 0.05, se rechaza y consideramos la serie estacionaria.

- Prueba KPSS: complementaria a la ADF, con hipótesis nula opuesta (la serie SÍ es estacionaria).

```
from statsmodels.tsa.stattools import adfuller

resultado = adfuller(serie)
print("Estadístico ADF:", resultado[0])
print("p-valor:", resultado[1])
```

Cómo volverla estacionaria

- Diferenciación: trabajar con $y_t - y_{t-1}$ en lugar del valor original; elimina tendencia.
- Diferenciación estacional: $y_t - y_{t-s}$ para quitar estacionalidad de período s.
- Transformación logarítmica o de Box-Cox: estabiliza la varianza.

Autocorrelación: ACF y PACF

La autocorrelación mide cuánto se parece la serie a una versión de sí misma desplazada k pasos (lag k). Es la herramienta para elegir los parámetros de los modelos.

ACF (función de autocorrelación)

Mide la correlación entre y_t y y_{t-k} incluyendo los efectos intermedios. Ayuda a identificar el orden q de la parte de medias móviles (MA).

PACF (función de autocorrelación parcial)

Mide la correlación entre y_t y y_{t-k} eliminando el efecto de los lags intermedios. Ayuda a identificar el orden p de la parte autorregresiva (AR).

Graficarlas

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

plot_acf(serie, lags=30)
plot_pacf(serie, lags=30)
```

Bloques de Construcción: AR y MA

Antes de ARMA y ARIMA necesitamos entender sus dos piezas y el ruido blanco.

Ruido blanco

Una serie de valores independientes, con media cero y varianza constante. No tiene estructura predecible: es el ideal al que deben parecerse los residuos de un buen modelo.

Modelo Autorregresivo AR(p)

El valor actual se explica como una combinación lineal de sus p valores pasados más un error.

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \varepsilon_t$$

Idea: "el futuro se parece al pasado reciente".

Modelo de Medias Móviles MA(q)

El valor actual se explica como una combinación lineal de los q errores pasados.

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q}$$

Idea: "el futuro se corrige según los errores recientes del modelo".

Modelo ARMA

ARMA(p, q) combina las dos ideas anteriores en un solo modelo: parte autorregresiva más parte de medias móviles.

La ecuación

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \varepsilon_t + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}$$

Cuándo usarlo

- Solo para series estacionarias (sin tendencia ni estacionalidad).
- p es el orden autorregresivo (lo sugiere la PACF).
- q es el orden de medias móviles (lo sugiere la ACF).

Limitación

ARMA no puede modelar directamente series con tendencia o estacionalidad. Para eso extendemos el modelo a ARIMA.

Modelo ARIMA

ARIMA(p, d, q) es la generalización más usada. La "I" viene de Integrated: incorpora la diferenciación para manejar series no estacionarias.

Los tres parámetros

| Parámetro | Significado | Cómo se elige | |----------|----------|----------| | p | Orden autorregresivo (AR) | PACF | | d | Número de diferenciaciones | Pruebas de estacionariedad | | q | Orden de medias móviles (MA) | ACF |

La idea

1. Diferenciar la serie d veces hasta volverla estacionaria.
2. Ajustar un modelo ARMA(p, q) sobre la serie diferenciada.
3. Integrar (deshacer la diferenciación) para producir el pronóstico en la escala original.

Metodología Box-Jenkins

Es el flujo estándar para construir un ARIMA:

1. Identificación: volver la serie estacionaria y examinar ACF/PACF para proponer p, d, q.
2. Estimación: ajustar el modelo y estimar sus coeficientes.
3. Diagnóstico: verificar que los residuos parezcan ruido blanco.
4. Pronóstico: si los residuos son adecuados, generar predicciones.

SARIMA: la variante estacional

SARIMA(p, d, q)(P, D, Q)m agrega términos estacionales, donde m es la longitud del período (12 para datos mensuales, 7 para diarios). Se usa cuando la

serie tiene estacionalidad clara.

ARIMA con statsmodels

```
from statsmodels.tsa.arima.model import ARIMA

modelo = ARIMA(serie, order=(1, 1, 1))
ajuste = modelo.fit()
print(ajuste.summary())

pronostico = ajuste.forecast(steps=12)
```

Selección automática con pmdarima

`auto_arima` prueba muchas combinaciones de (p, d, q) y elige la mejor según el criterio de información AIC.

```
import pmdarima as pm

modelo = pm.auto_arima(serie, seasonal=True, m=12,
                       trace=True, suppress_warnings=True)
print(modelo.summary())
```

Prophet

Prophet es una librería de código abierto creada por Meta (Facebook) para pronóstico. Está pensada para que analistas sin experiencia profunda en series de tiempo obtengan buenos resultados rápido.

El modelo

Prophet es un modelo aditivo que descompone la serie en componentes interpretables:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t$$

- $g(t)$: tendencia (lineal o de saturación logística).
- $s(t)$: estacionalidad (anual, semanal, diaria) modelada con series de Fourier.
- $h(t)$: efecto de días festivos y eventos especiales.
- ε_t: error.

Ventajas

- Robusto ante datos faltantes y valores atípicos.
- Detecta automáticamente cambios de tendencia (changepoints).
- Permite agregar festivos y eventos con facilidad.
- Requiere poca configuración y sus componentes son interpretables.

Cuándo brilla

- Series con estacionalidad fuerte de negocio (ventas, tráfico web).
- Datos diarios con efectos de fin de semana y festivos.
- Historiales de varios períodos estacionales.

Uso con Python

Prophet exige un DataFrame con dos columnas: `ds` (fecha) y `y` (valor).

```
from prophet import Prophet

df = df.rename(columns={"fecha": "ds", "valor": "y"})

modelo = Prophet(yearly_seasonality=True, weekly_seasonality=True)
modelo.fit(df)

futuro = modelo.make_future_dataframe(periods=90)
```

```
pronostico = modelo.predict(futuro)

modelo.plot(pronostico)
modelo.plot_components(pronostico)
```

Redes Neuronales para Series de Tiempo

Cuando las relaciones son no lineales o hay muchas variables, las redes neuronales pueden superar a los modelos clásicos, a costa de más datos y menos interpretabilidad.

Reformular como aprendizaje supervisado

La técnica clave es la ventana deslizante (sliding window): usamos los últimos n valores como entrada (X) para predecir el siguiente (y).

```
Entrada (X)             ->  Salida (y)
[y1, y2, y3]            ->  y4
[y2, y3, y4]            ->  y5
[y3, y4, y5]            ->  y6
```

RNN (Redes Neuronales Recurrentes)

Procesan la secuencia paso a paso y mantienen un estado oculto que resume lo visto. Sufren el problema del desvanecimiento del gradiente en secuencias largas.

LSTM (Long Short-Term Memory)

Un tipo de RNN con compuertas (gates) que deciden qué recordar y qué olvidar. Manejan dependencias de largo plazo y son el estándar clásico para series con redes neuronales.

GRU (Gated Recurrent Unit)

Una versión más simple y ligera que la LSTM, con menos parámetros y desempeño a menudo comparable.

Otras arquitecturas

- CNN 1D: detectan patrones locales en la secuencia; rápidas de entrenar.
- Transformers: usan atención para capturar dependencias largas; base de los modelos modernos de pronóstico.

Ejemplo de LSTM con Keras

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

modelo = Sequential([
    LSTM(50, activation="relu", input_shape=(n_pasos, 1)),
    Dense(1)
])
modelo.compile(optimizer="adam", loss="mse")
modelo.fit(X, y, epochs=100, verbose=0)
```

Costo de usarlas

- Necesitan bastantes datos para entrenar bien.
- Requieren escalar los datos (por ejemplo, a un rango 0–1).
- Son cajas negras: difíciles de interpretar.
- Más lentas de entrenar y ajustar que un ARIMA.

Evaluación de Modelos de Pronóstico

Evaluar series de tiempo es distinto: no podemos mezclar el pasado con el futuro.

Partición temporal (nunca aleatoria)

El conjunto de prueba debe ser siempre posterior en el tiempo al de entrenamiento. Barajar (shuffle) los datos filtra información del futuro hacia el pasado y arruina la evaluación.

```
|------------ entrenamiento ------------|--- prueba ---|
                                        tiempo -->
```

Validación walk-forward

En lugar de una sola partición, se avanza en el tiempo: se entrena, se predice el siguiente paso, se incorpora el valor real y se repite. Simula el uso real del modelo.

Métricas de error

| Métrica | Qué mide | Nota | |---------|----------|------| | MAE | Error absoluto medio | En las unidades de la serie | | RMSE | Raíz del error cuadrático medio | Penaliza más los errores grandes | | MAPE | Error porcentual absoluto medio | Comparable entre series |

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

mae = mean_absolute_error(y_real, y_pred)
rmse = np.sqrt(mean_squared_error(y_real, y_pred))
```

¿Qué Modelo Elegir?

No hay un modelo universalmente mejor; depende de los datos y del objetivo.

Guía rápida

| Situación | Modelo sugerido | |---|---| | Serie estacionaria, pocos datos | ARMA / ARIMA | | Tendencia y estacionalidad clásicas | SARIMA | | Estacionalidad de negocio, festivos, faltantes | Prophet | | Relaciones no lineales, muchos datos | LSTM / GRU / Transformers | | Referencia base para comparar | Modelo naive (último valor) |

Recomendación práctica

- Empieza siempre con un modelo simple (naive o ARIMA) como línea base.
- Sube en complejidad solo si mejora la métrica de forma clara.
- Un modelo más complejo no siempre es mejor: valida contra la línea base.

Resumen

- Una serie de tiempo es una secuencia ordenada en el tiempo; el orden y la autocorrelación son lo que la distingue.
- Se descompone en tendencia, estacionalidad, ciclo y ruido.
- La estacionariedad es el supuesto central de los modelos clásicos; se verifica con ADF/KPSS y se logra diferenciando.
- AR y MA son los bloques; ARMA los combina para series estacionarias.
- ARIMA agrega diferenciación (d) para manejar tendencia; SARIMA agrega estacionalidad.
- Prophet ofrece un modelo aditivo robusto e interpretable, ideal para datos de negocio.

- Las redes neuronales (LSTM, GRU, Transformers) capturan relaciones no lineales a cambio de más datos y menos interpretabilidad.
- Evalúa con partición temporal, validación walk-forward y métricas como MAE, RMSE y MAPE.