Análisis de Series de Tiempo

Carrera de Especialización en Inteligencia Artificial

Clase 3

Ing. Magdalena Bouza, Ing. Carlos German Carreño Romano

Random Walk

Random walks

Un modelo muy sencillo es el modelo del caminante aleatorio. Se define como

$$\left.\begin{aligned} Y_1 &= e_1 \\ Y_2 &= e_1 + e_2 \\ &\vdots \\ Y_t &= e_1 + e_2 + \cdots + e_t \end{aligned}\right\}$$

Si *et* es un proceso blanco de media nula y varianza σ_e^2, es fácil ver que

$$\mathbb{E}[Y_t] = 0 \quad y \quad var(Y_t) = t\sigma_e^2$$

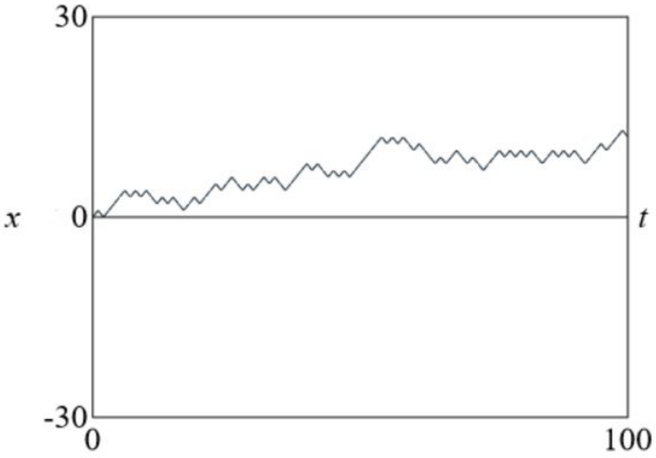
de forma que el **proc.** no es **estacionario**. Además $C_{t,s} = cov(Y_t, Y_s) = t\sigma_e^2,\ t \leq s$ y

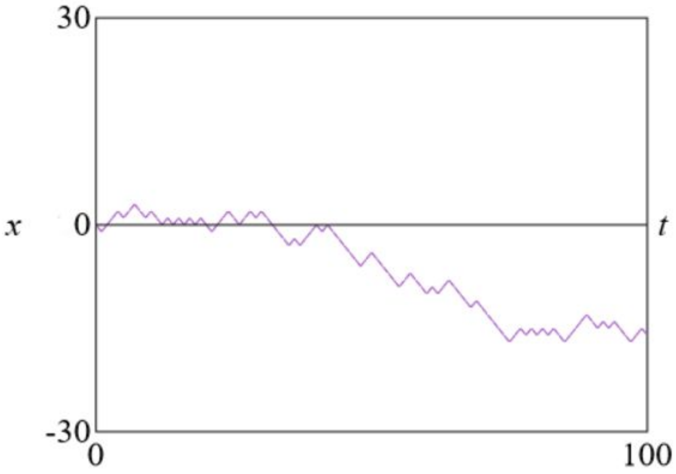
$$R_{t,s} = \frac{cov(Y_t,Y_s)}{\sqrt{var(Y_t)var(Y_s)}} = \sqrt{\frac{t}{s}},\ t \leq s$$

Análisis de tendencia

Tendencia estocástica vs. tendencia determinística

Si bien no hay una definición unificada de qué son las **tendencias estocásticas**, se puede decir que son aquellas que un observador podría hallar al analizar una realización de una serie de tiempo, pero que si se tiene una realización distinta esa tendencia cambia.





Tendencia estocástica vs. tendencia determinística

La **tendencia determinística** es aquella que viene dada por el modelo, y es fija para toda la serie de tiempo, sin importar que realización se tenga. *Por ejemplo las variaciones cíclicas a los largo de las distintas estaciones del año.*

En el caso de la tendencia determinística, podemos estimarla y descontarla de la serie de tiempo. Esta situación se puede modelar como

$$Y_t = X_t + \mu_t$$

donde μ_t es la tendencia determinística y X_t es una serie de tiempo de media cero alrededor de μ_t.

Algunos modelos comunes para tendencia

- Constante: $\mu_t = \mu \quad \forall t$
- Lineal: $\mu_t = \beta_0 + \beta_1 t \quad \forall t$
- Cuadrática: $\mu_t = \beta_0 + \beta_1 t + \beta_2 t^2 \quad \forall t$
- Cíclicas: $\mu_t = \mu_{t-T} \quad \forall t$, donde T es el período del ciclo. *Ejemplo: temperatura a lo largo del año tiene un período de T=12 meses*
- Coseno: $\mu_t = \beta \cos(2\pi f t + \phi) \quad \forall t$

¿Cómo estimar estas tendencias?

Suele emplearse el método de cuadrados mínimos para estimar los valores de los parámetros que describen la tendencia.

Si llamamos $f(t;\theta)$ a los modelos presentados anteriormente, buscamos hallar los parámetros θ que minimicen el ECM. Es decir, buscamos θ que minimice

$$Q(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_t - f(t;\theta))^2$$

Caso constante:

$$\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \mu + \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_1 \end{bmatrix}$$

Luego, la solución por c.m. es

$$\mu = \underbrace{\left(\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right)}_{n}^{-1} \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_1 \end{bmatrix} = \frac{1}{n} \sum_{t=1}^{n} y_t$$

Caso lineal

$$\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 & n \\ 1 & n-1 \\ \vdots & \vdots \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_1 \end{bmatrix}$$

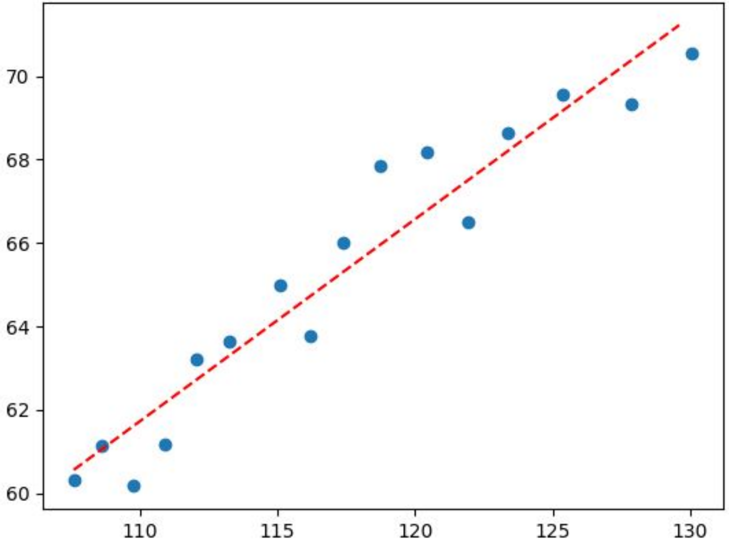
Y la solución por c.m resulta

$$\begin{bmatrix} \hat\beta_0 \\ \hat\beta_1 \end{bmatrix} = \left(\begin{bmatrix} 1 & n \\ 1 & n-1 \\ \vdots & \vdots \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} 1 & n \\ 1 & n-1 \\ \vdots & \vdots \\ 1 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & n \\ 1 & n-1 \\ \vdots & \vdots \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_1 \end{bmatrix} =$$

$$\hat\beta_1 = \frac{\sum_{t=1}^{n} (y_t - \bar y)(t - \frac{n+1}{2})}{\sum_{t=1}^{n} (t - \frac{n+1}{2})^2} \qquad \hat\beta_0 = \bar y - \hat\beta_1 \frac{n+1}{2}$$

Caso lineal

```python
from scipy.optimize import curve_fit
# define the true objective function
def objective(x, a, b):
    return a * x + b
```




```python
dataframe = read_csv(url, header=None)
data = dataframe.values
# choose the input and output variables
x, y = data[:, 4], data[:, -1]
# curve fit
popt, _ = curve_fit(objective, x, y)
# summarize the parameter values
a, b = popt
print('y = %.5f * x + %.5f' % (a, b))
# plot input vs output
pyplot.scatter(x, y)
# define a sequence of inputs between the smalle
x_line = arange(min(x), max(x), 1)
# calculate the output for the range
y_line = objective(x_line, a, b)
# create a line plot for the mapping function
pyplot.plot(x_line, y_line, '--', color='red')
pyplot.show()
```

Caso cuadrático

$$\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & n & n^2 \\ 1 & n-1 & (n-1)^2 \\ \vdots & \vdots & \\ 1 & 1 & 1 \end{bmatrix}}_{A} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_1 \end{bmatrix}$$

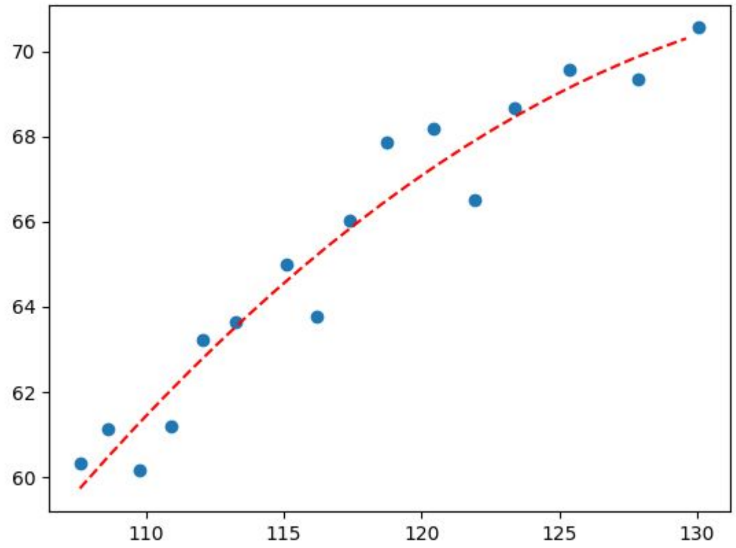
Y la solución por c.m. queda

$$\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = (A^T A)^{-1} A^T \begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_1 \end{bmatrix}$$

Caso cuadrático

```
# define the true objective function
def objective(x, a, b, c):
    return a * x + b * x**2 + c

# choose the input and output variables
x, y = data[:, 4], data[:, -1]
# curve fit
popt, _ = curve_fit(objective, x, y)
# summarize the parameter values
a, b, c = popt
print('y = %.5f * x + %.5f * x^2 + %.5f' % (a, b, c))
# plot input vs output
pyplot.scatter(x, y)
# define a sequence of inputs between the smallest anc
x_line = arange(min(x), max(x), 1)
# calculate the output for the range
y_line = objective(x_line, a, b, c)
```



Caso cíclico

$$\mu_t = \begin{cases} \beta_1 & t = 1, 1+T, 1+2T, \ldots \\ \beta_2 & t = 2, 2+T, 2+2T, \ldots \\ \vdots & \\ \beta_T & t = T, 2T, 3T, \ldots \end{cases}$$

$$\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_1 \end{bmatrix} = A \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_T \end{bmatrix} + \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_1 \end{bmatrix}, \quad a_{i,j} = \begin{cases} 1 & (i \mod j) = 0 \\ 0 & (i \mod j) \neq 0 \end{cases}$$

Caso cíclico

- ejecutar ejemplo con serie de tiempo periódica en la práctica

Caso coseno

Reescribiendo la tendencia podemos llevarla a una expresión lineal:

$$\mu_t = \beta \cos(2\pi f t + \phi) = \beta_1 \cos(2\pi f t) + \beta_2 \cos(2\pi f t),$$
$$\beta_1 = \beta \cos(\phi),\ \beta_1 = \beta \sin(\phi)$$

$$\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \cos(2\pi f n) & \sin(2\pi f n) \\ 1 & \cos(2\pi f (n-1)) & \sin(2\pi f (n-1)) \\ \vdots & \vdots & \\ 1 & \cos(2\pi f 1) & \sin(2\pi f 1) \end{bmatrix}}_{A} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_1 \end{bmatrix}$$

¡Cuidado!

Los resultados por cuadrados mínimos son válidos estrictamente sólo cuando Xt es un proceso blanco (i.e. las muestras temporales son independientes entre sí).

Sin embargo, se puede demostrar que si el proceso es estacionario los resultados son asintóticamente válidos, y sus varianzas coinciden con la de los estimadores de mínima varianza para los parámetros.

Estimando X_t

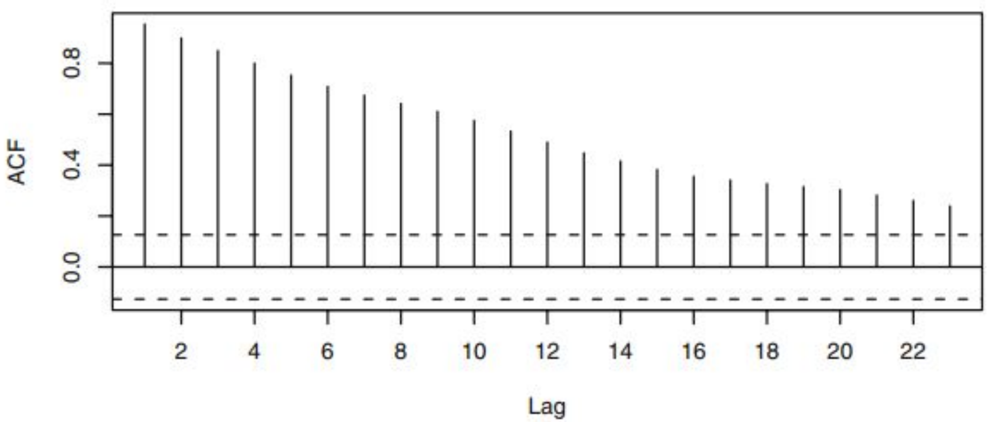
Los valores de X_t pueden ser estimados a partir de los residuos de la estimación de los parámetros:

$$\hat{x}_t = y_t - \hat{\mu}_t$$

Análisis de estacionariedad

Autocorrelación

Un método relativamente fácil, aunque bastante a ojo, para verificar que una serie **no** es **estacionaria** es a través de su función de autocorrelación muestral.



Si la gráfica no alcanza valores nulos para lags grandes es porque posiblemente no sea estacionaria

Test de Dickey-Fuller

El test de Dickey Fuller, es un test de hipótesis que analiza la hipótesis nula que un modelo ARMA tiene algún coeficiente tal que $a_i = 1$

Se lo conoce también como test de raíz unitaria.

Por qué tiene sentido analizar estacionariedad bajo la suposición de un modelo ARMA?

El teorema de Wold dice que toda serie de covarianza estacionaria Yt puede escribirse como

$$Y_t = \sum_{k=1}^{\infty} b_k e_{t-k} + \eta_t$$

ARMA se puede descomponer de esta forma (PLG)

donde et es una secuencia descorrelacionada de media cero (innovaciones), η_t es una señal determinística

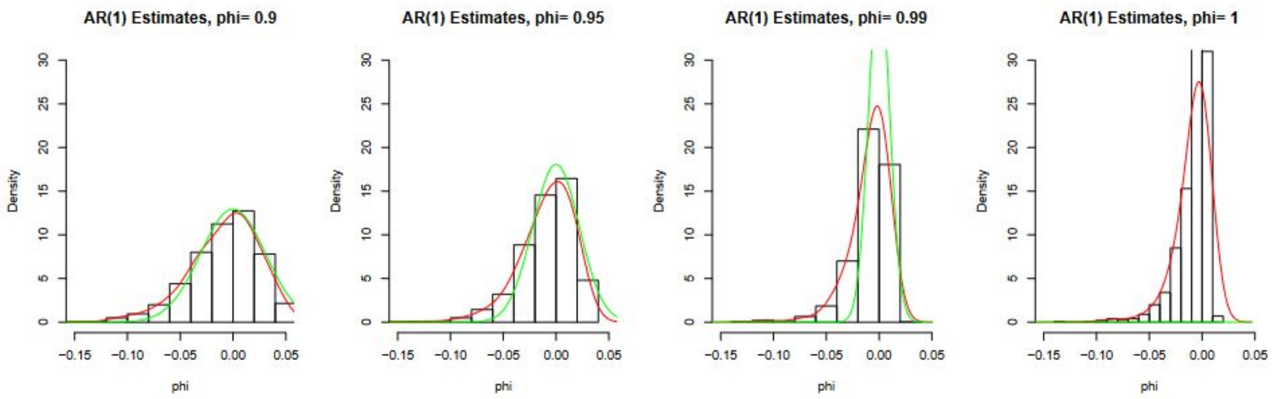
Test de Dickey-Fuller

Tengo un modelo AR(1) de la forma $Y_t = a_1 Y_{t-1} + e_t$ y quiero saber si $a_1 = 1$.

Si desconocemos la estacionariedad del modelo, sólo podemos estimar el parámetro por OLS:

$$\hat{a}_1 = \frac{\sum_{t=1}^{n} Y_t Y_{t-1}}{\sum_{t=1}^{n} Y_{t-1}^2}$$

Se puede demostrar que $\sqrt{n}(\hat{a}_1 - a_1) \sim \mathcal{N}(0, 1 - a_1^2)$ si $|a_1| < 1$, mientras que si $a_1 = 1$ el estimador tiene una distribución asintótica de la forma



Test de Dickey-Fuller

Dickey y Fuller descubrieron la distribución asintótica de $n(\hat{a}_1 - 1)$, cuando $a_1 = 1$, y proponen un test para $H_0 : a_1 = 1$, donde en lugar de estimar por OLS el modelo AR(1), lo hace sobre el modelo diferenciado, restando m.a.m. Y_{t-1}.

Se estima $a_1 - 1$ y se analiza

$$H_0 : (a_1 - 1) = 0, \quad vs. \quad H_1 : (a_1 - 1) < 0$$

Vamos a rechazar el test cuando $\widehat{a_1 - 1} < k_\alpha$.

Observación: bajo H0, los datos seguirán la distribución hallada por D-F, y se debe usar algún software apropiado para realizar el análisis.

Test de Dickey-Fuller aumentado

Incorpora un término de ruido dependiente (pero estacionario).

Consideremos un modelo de la forma, $Y_t = \alpha Y_{t-1} + X_t, \; t = 1, 2, \ldots$ donde X_t es un proceso estacionario. Se puede ver que si $\alpha = 1$ Y_t es no estacionaria, pero es estacionaria si $|\alpha| < 1$.

Supongamos que $\{X_t\}$ es un proceso AR(p):

$$X_t = a_1 X_{t-1} + a_2 X_{t-2} + \ldots + a_p X_{t-p} + e_t$$

Bajo H0, $X_t = Y_t - Y_{t-1}$, con lo cual

$$\begin{aligned} Y_t - Y_{t-1} &= (\alpha - 1) Y_{t-1} + X_t \\ &= (\alpha - 1) Y_{t-1} + a_1 X_{t-1} + \ldots + a_p X_{t-p} + e_t \\ &= (\alpha - 1) Y_{t-1} + a_1 (Y_{t-1} - Y_{t-2}) + a_2 (Y_{t-2} - Y_{t-3}) + \ldots + a_p (Y_{t-p} - Y_{t-p-1}) + e_t \end{aligned}$$

Bibliografia extra para D-F

- Testing for unit roots
- Augmented Dickey-Fuller root tests

- Tendencia determinística (cap2 del libre)
- Darles alguna serie estacionaria salvo tendencia y que vean como modelar con AR/MA/ARMA
- Dicky-Fuller pare verificar estacionariedad
- CASO de uso: paper de charlie

Aplicaciones (parte I)

Stationary Time Series	Non-Stationary Time Series
Statistical properties of a stationary time series are independent of the point in time where it is observed.	Statistical properties of a non-stationary time series is a function of time where it is observed.
Mean, variance and other statistics of a stationary time series remains constant. Hence, the conclusions from the analysis of stationary series is reliable.	Mean, variance and other statistics of a non-stationary time series changes with time. Hence, the conclusions from the analysis of a non-stationary series might be misleading.
A stationary time series always reverts to the long-term mean.	A non-stationary time series does not revert to the long term mean.
A stationary time series will not have trends, seasonality, etc.	Presence of trends, seasonality makes a series non-stationary.

https://blog.quantinsti.com/stationarity/

IEEE ARGENCON 2018

Sizing Techniques applied to Network Capacity Planning

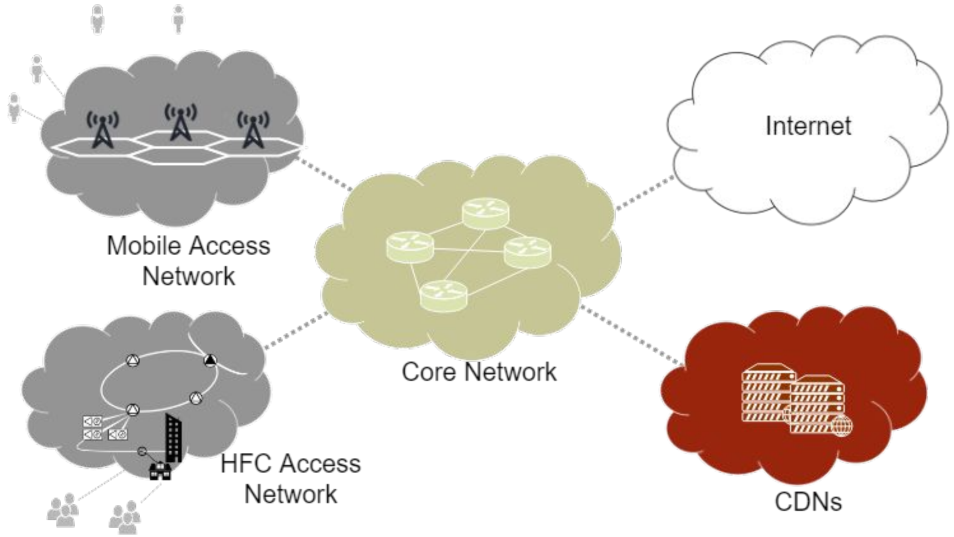
Técnicas de dimensionamiento aplicadas al planeamiento de capacidad de red

- Carlos G. Carreño Romano
- Natalia A. Clivio Velilla

https://ieeexplore.ieee.org/document/8646077

Aplicaciones en dimensionamiento de redes

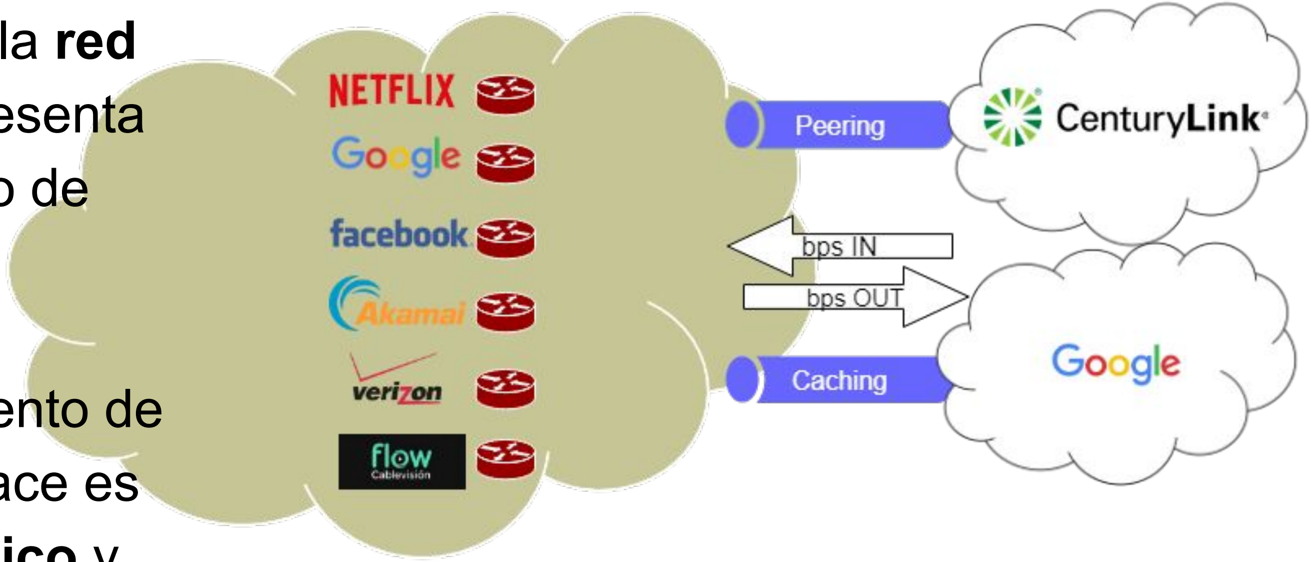
- Proveedores de Servicios de Comunicaciones (CSP)
- Redes de acceso
- Redes Core
- Redes de Distribución de Contenidos (CDN)
- Internet



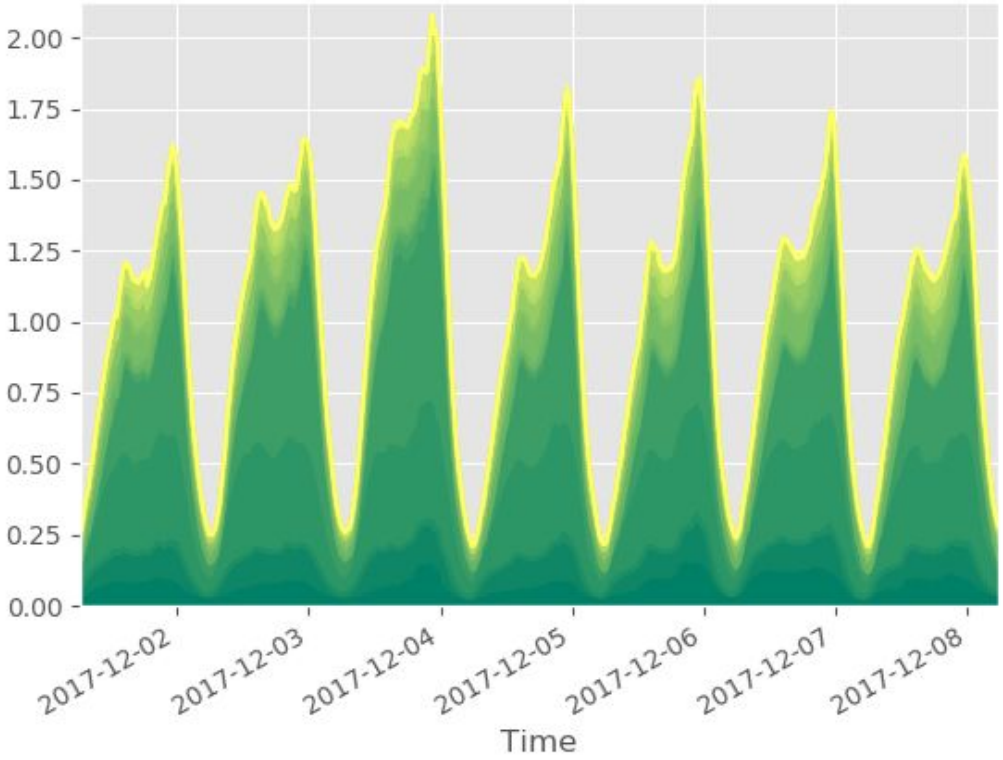
Contexto

Cada enlace entre la **red Core** y las **CDN** presenta una serie de tiempo de tráfico.

Predecir el crecimiento de tráfico en cada enlace es de interés **económico** y **técnico**.



Dataset



Datos de 1 año con una granularidad de 5 minutos.

Objetivo: Comparar predicciones generadas con ARIMA y LSTM RNN para las trazas de tráfico real de la empresa Telecom Argentina

Modelado usando ARIMA

ARIMA viene de **A**utorregresión **I**ntegrados de **M**edia móvil

$$\text{ARIMA}\ (\overbrace{\mathbf{p},\mathbf{d},\mathbf{q}}^{\text{Orden de cada componente}})$$

$$\phi_0 + \sum_{i=1}^{P} \phi_i Z_{m-i}$$

$$D_i - D_{i-d}$$

$$\sum_{i=1}^{Q} \theta_i \varepsilon_{m-i}$$

Autoregresivo de los últimos **‘p’** valores

Diferenciación de ‘**d**’ períodos anteriores

Promedio móvil de los últimos **‘q’** errores

Modelado usando ARIMA

Para aplicar modelos ARIMA se suele descomponer la serie, analizando en primer lugar la **tendencia** de la serie, luego la **estacionalidad** y concentrándose en identificar estas componentes filtradas. Hay otras dos componentes que hacen el modelo completo y son las **componentes cíclicas** y las **componentes aleatorias**. El proceso consiste en la descomposición de la serie en forma aditiva o multiplicativa. Usamos en este trabajo la forma aditiva y definimos entonces una serie de tiempo Y(t) como:

$$Y(t) = T(t) + S(t) + C(t) + e(t)$$

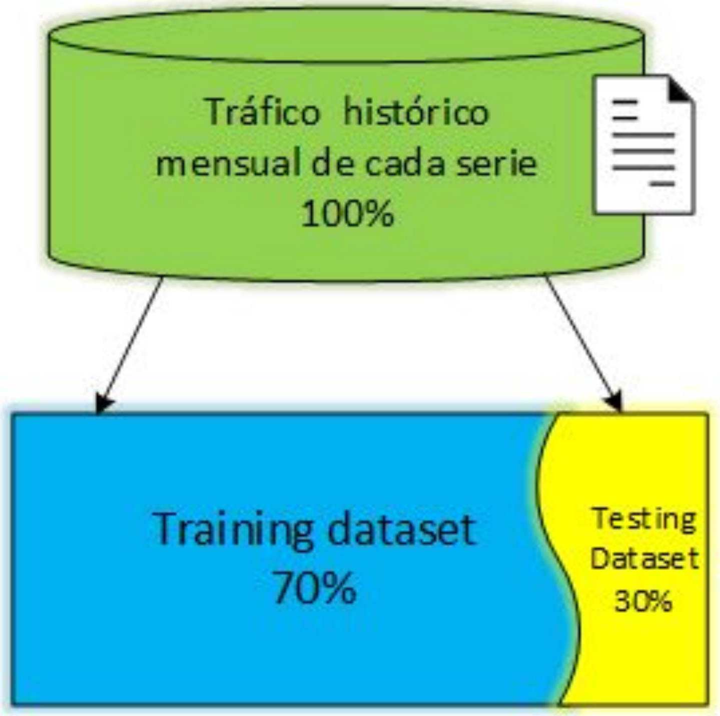
para tiempo contínuo y una serie de tiempo discreto Yt como:

$$Y_t = T_t + S_t + C_t + e_t$$

donde:

- T(t): Tendencia
- S(t): Variación Estacional
- C(t): Componente Cíclica
- e(t): Componente aleatoria

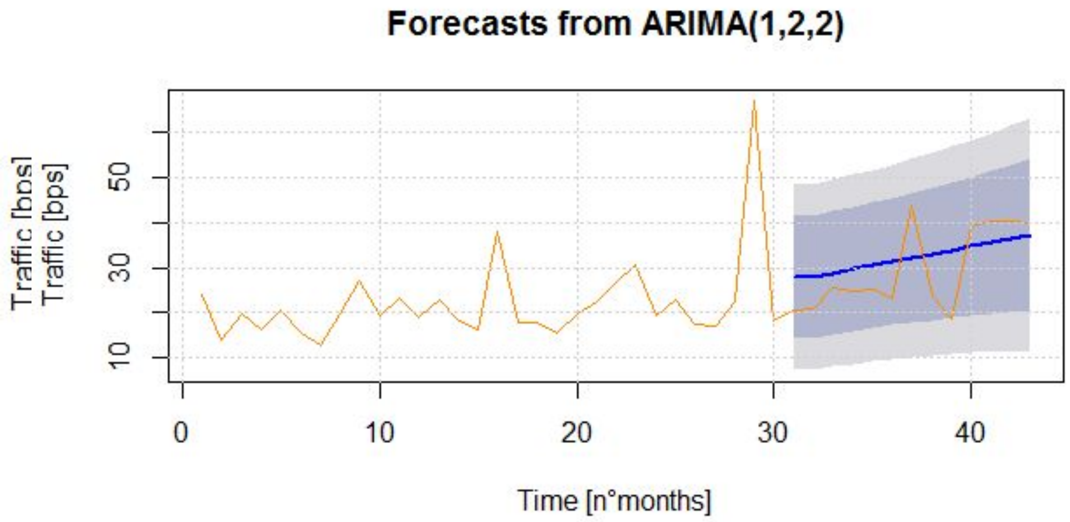
Training and Testing



- Hace falta fraccionar el dataset en dos partes: **training y test**
- En general es útil que las proporciones sean representativas.
- La cantidad y calidad de los datos es un factor siempre presente.
- No todos los algoritmos estadísticos admiten paralelismo.
- Otro factor importante es el nivel de ajuste (**sub fitting** vs. **overfitting**)

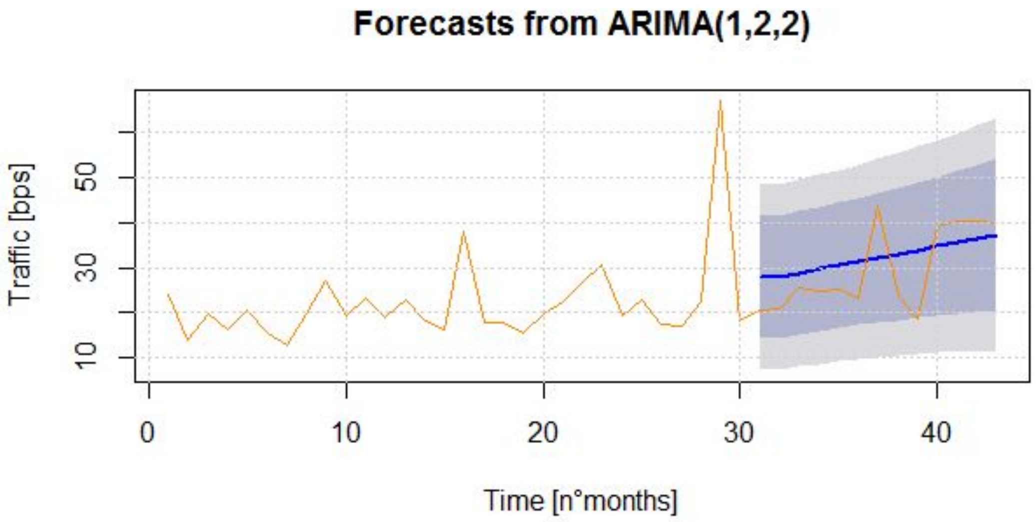
Resultados usando ARIMA

- En azul se grafica la **tendencia**
- En gris oscuro el intervalo de confianza del 95%
- En gris claro el intervalo e 90%
- En naranja el fragmento de testing de la serie



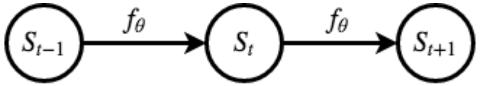
Compromisos

- Los intervalos de confianza pueden ser muy amplios en términos absolutos.
- Si es una variable económica el desvío puede ser demasiado significativo.

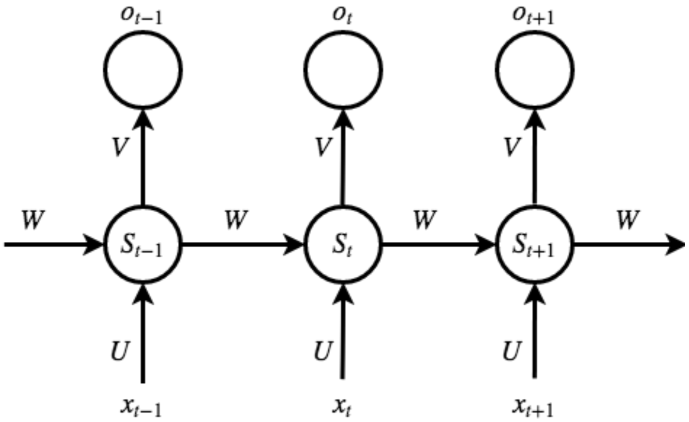


Alternativa usando Redes Neuronales

- Usamos redes basadas en estados
- este tipo de redes se llaman Redes Neuronales Recurrentes (RNN)



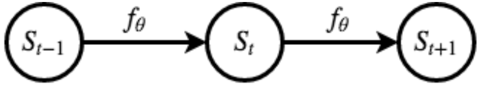
$$s_t = f_\theta(s_{t-1})$$



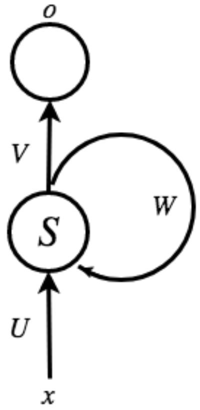
Vanilla Recurrent Neural Network

- se puede usar la notación de estados
- se suele usar la topología desplegada

Redes Neuronales LSTM



$$s_t = f_\theta(s_{t-1})$$

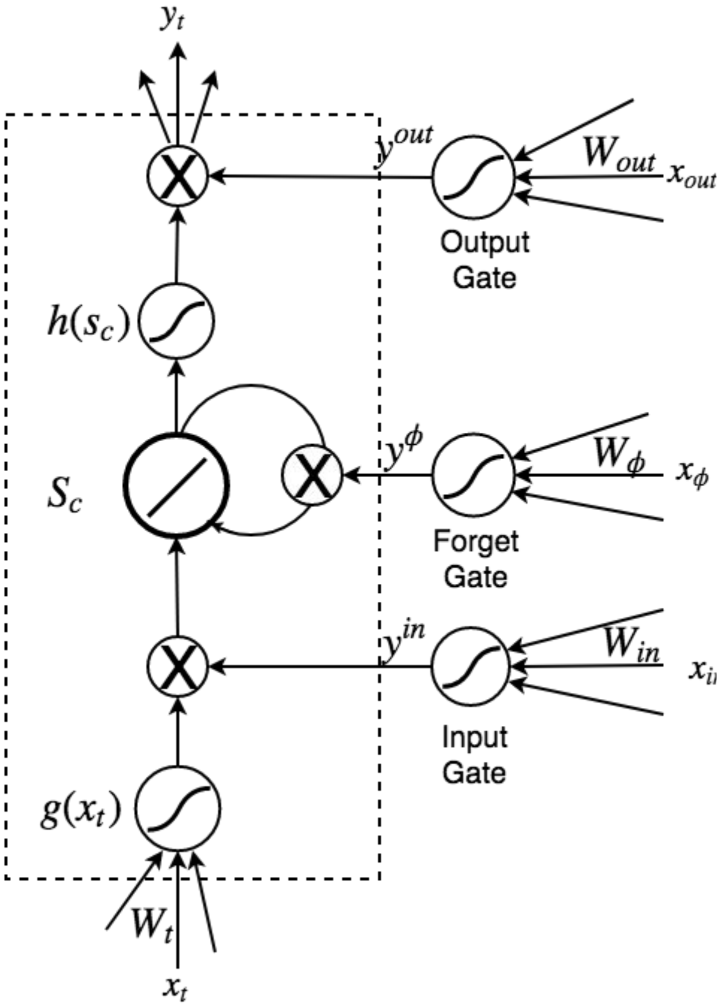


celda LSTM

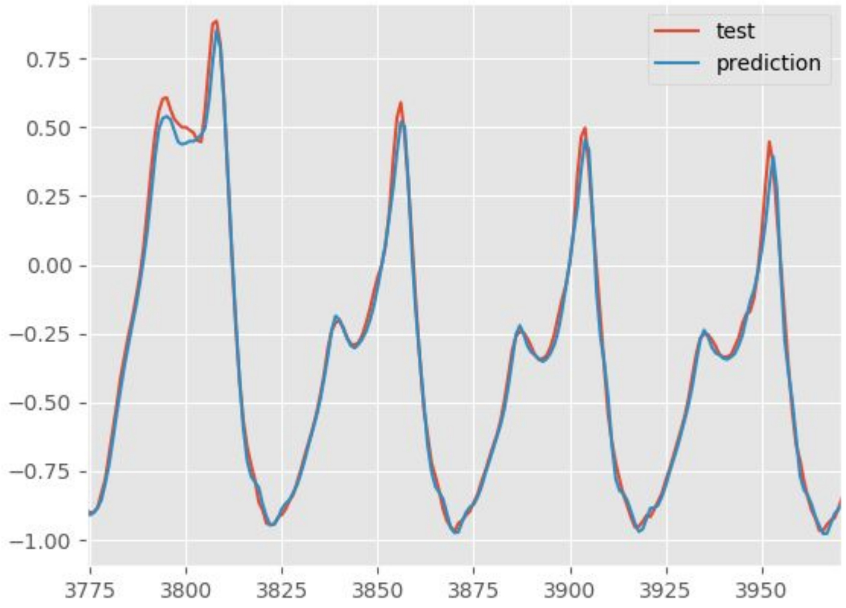
Contiene una entrada y una salida mas tres entradas de control:

- Input gate
- Forget gate
- Output gate

La función core es lineal

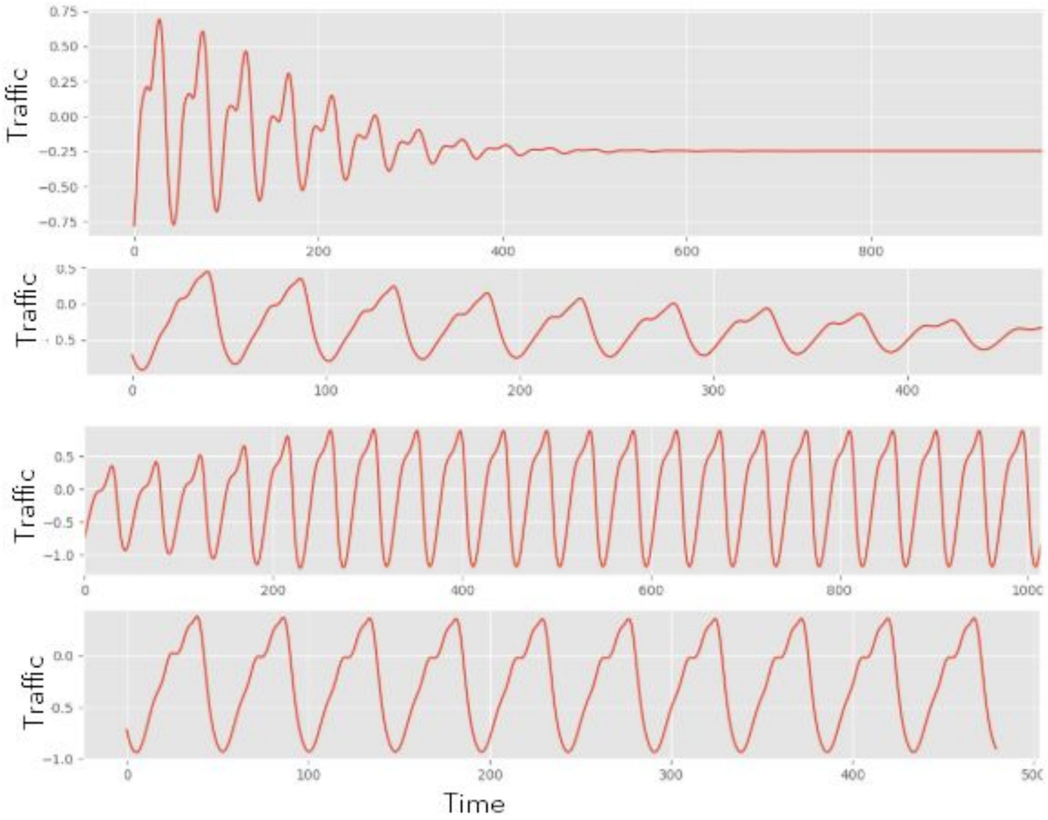


LSTM RNN: entrenamiento y test



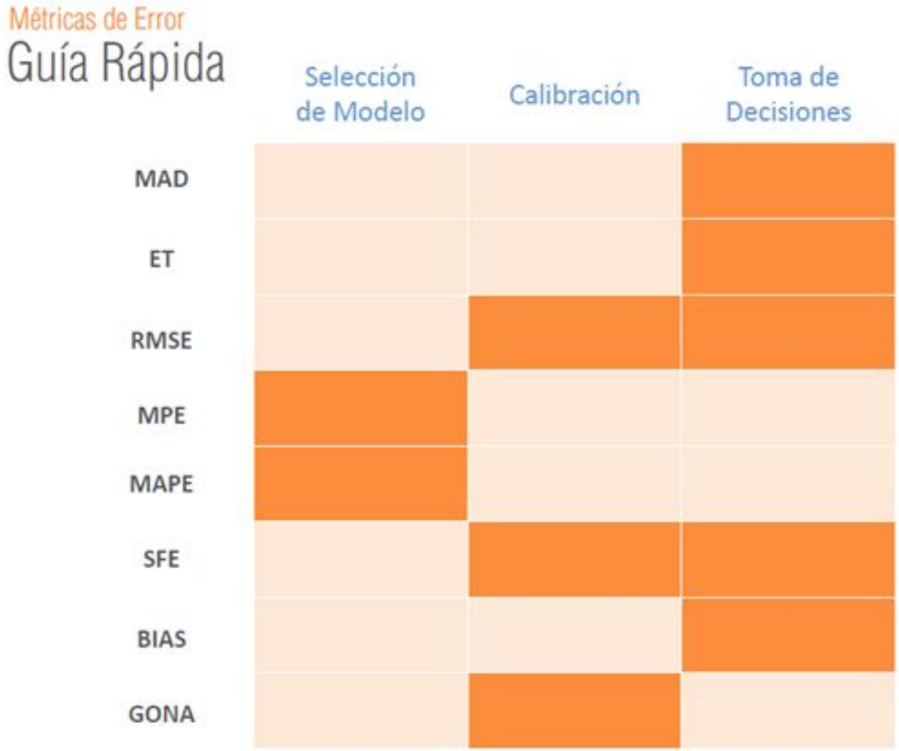
Predicciones usando ventana deslizante

- La técnica de ventana deslizante consiste en ir realimentando la serie predecida intervalo a intervalo.
- El largo de la ventana, la densidad de datos, la cantidad de períodos muestreados son factores de diseño.



Figuras de mérito

- Como figuras de mérito de los modelos elegimos algunas métricas que sirvan para hacer los modelos comparables.
- El error cuadrático medio, el máximo error absoluto, mínimo error porcentual (absoluto), entre otros.



Resultados

Training set	ARIMA (p,d,q)	RMSE	MAE	MPE (%)	MAPE (%)	LSTM (inputs,batch, epochs)	RMSE	MAE	MAPE
CDN Google	1,2,1	13,08	10,35	1,16	2,40	**144**, 144, 5	$8.7x10^{-4}$	$9.2x10^{-3}$	< 0.00%
CND Netflix	1,2,1	27,70	19,76	0,22	2,86	144, 144, 5	$3.3x10^{-3}$	$3.1x10^{-3}$	< 0.00%
CDN Verizon	1,2,2	9,29	5,28	-1,72	20,63	**336(7d)**, 336,5	$2.5x10^{-4}$	$2.3x10^{-2}$	< 0.0%
CDN Akamai	2,2,2	15,25	10,79	-1,54	10,79	336, 336,5	$5.7x10^{-4}$	$6.2x10^{-2}$	< 0.0%

Conclusiones

- Para este modelo resultaron algunas observaciones particulares.
- Las componentes estacionales requieren datos interanuales
- Pronósticos de corto plazo: LSTM
- Pronósticos a largo plazo: ARIMA
- Compromiso entre cantidad de datos y método utilizado
- Posibles mejoras extendiendo el tamaño de la red LSTM
- Trabajo futuro: monitoreo, descomposición y combinación de métodos

¿preguntas?

Trabajo práctico

Trabajo Práctico

1: Graficar una serie a partir de un dataset relevante. Explicar observaciones

2: Descomponer una serie de tiempo usando el modelo aditivo y el modelo multiplicativo.

3: Aplicar los modelos vistos en clase:

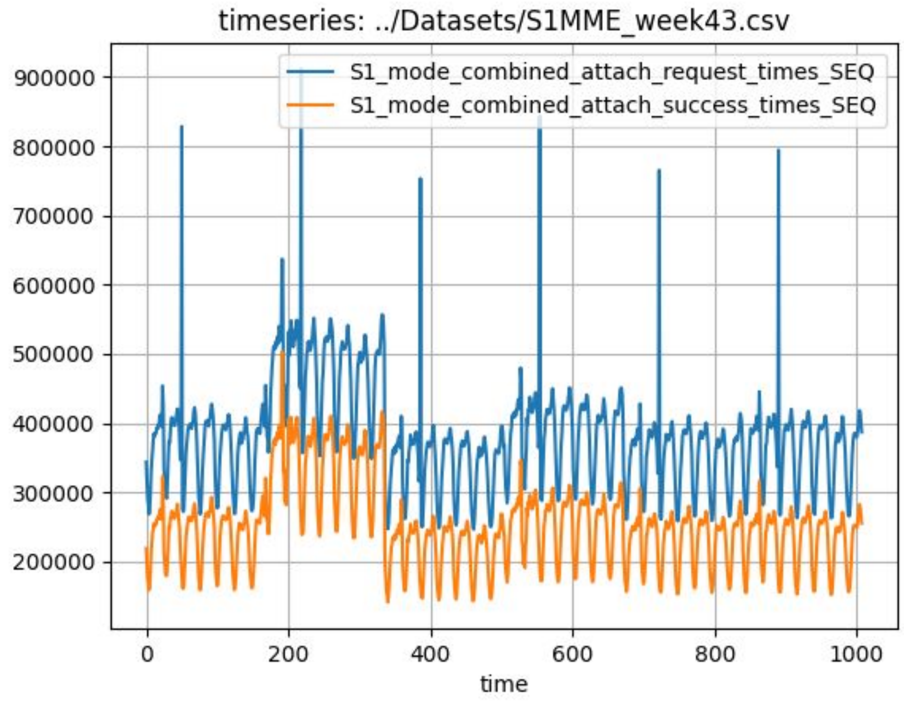
- para la tendencia usar cuadrados mínimos y expresar los coeficientes. Sacar conclusiones acerca de la validez del modelo
- componente cíclica: usar análisis espectral y hallar las frecuencias principales
- para la componente estacional usar ARIMA
- para la componente de error obtener R_k, C_k

3: Predicciones:

- realizar predicciones usando (S)ARIMA
- realizar predicciones usando redes neuronales LSTM
- extraer conclusiones

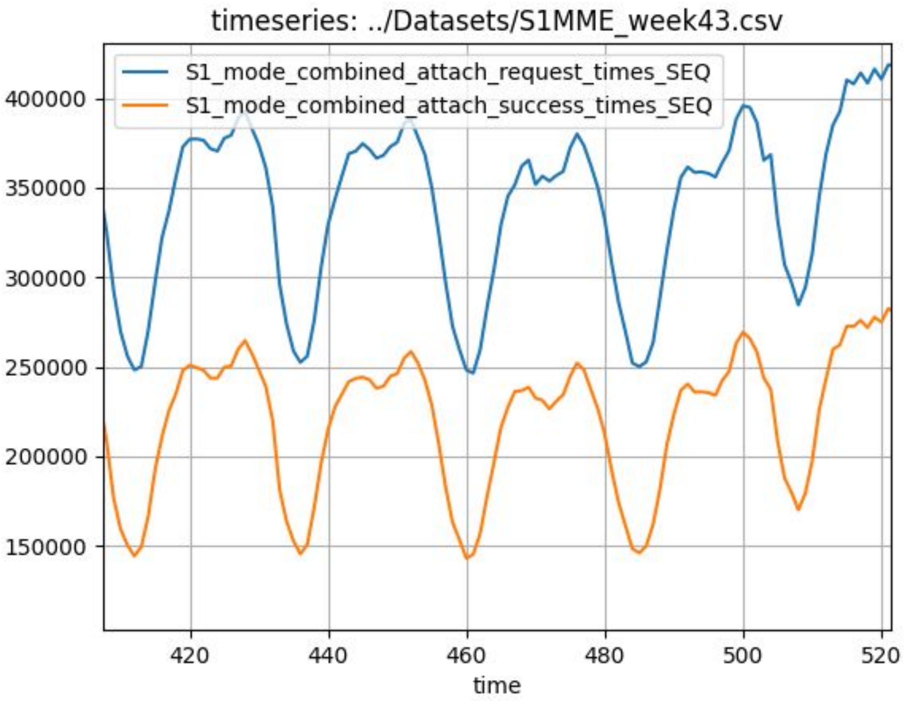
Ejemplo

Análisis de series de tiempo



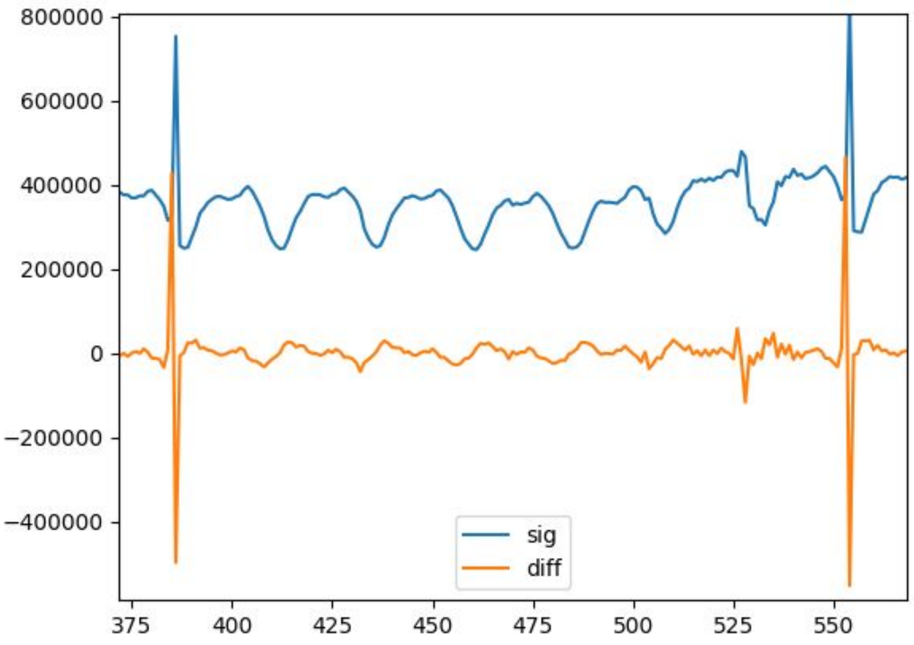
En estas dos series se observa un comportamiento periódico similar. El período observado comprende 24 puntos, posiblemente un punto por hora, y se observa una serie de picos que se extienden fuera del rango de comportamiento normal de la serie.

Análisis de series de tiempo



Cada período presenta picos y valles bien marcados. La serie cae en sus valles a valores mínimos que suelen mantener un valor absoluto regular. En cada período se pueden observar dos picos a una distancia regular, donde el valor del segundo pico supera siempre al primero. Ambos picos tienen variaciones en su valor absoluto, a diferencia de los valles.

Análisis de estacionariedad

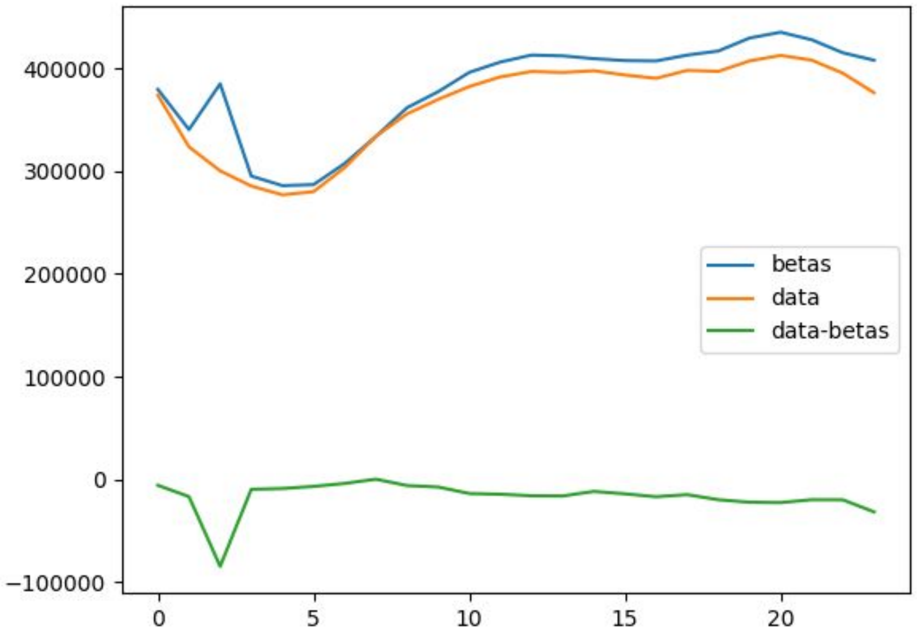


La diferenciación es una transformación muy útil para evaluar estacionariedad.

Observar que la serie original (en azul) es diferenciada y luego oscila alrededor del valor nulo (en naranja).

Hay que testear si la curva naranja es estacionaria.

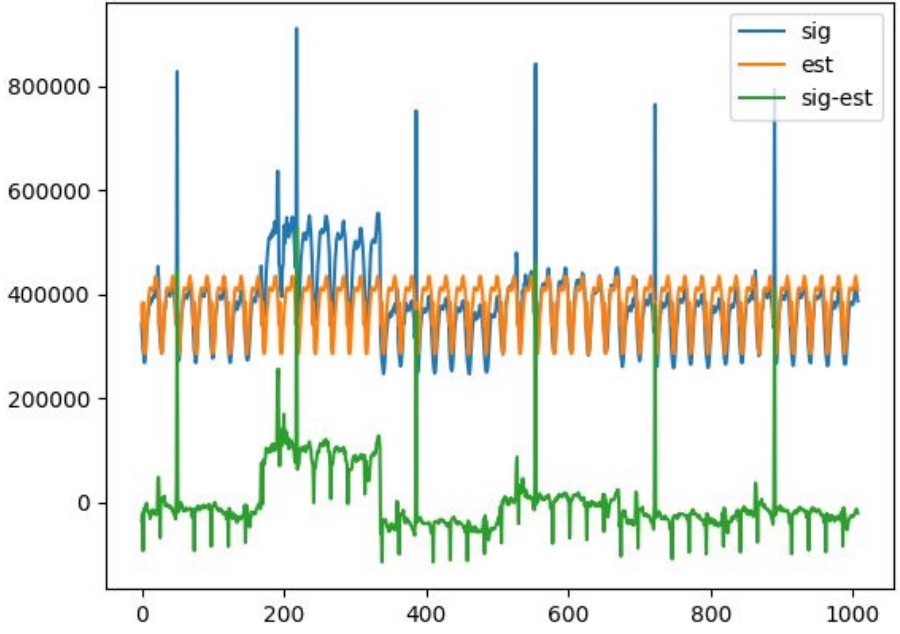
Ajuste por cuadrados mínimos: caso cíclico



Usando la técnica de cuadrados mínimos se pueden estimar los valores promedio de los períodos.

Luego si se resta a la serie original la serie estimada por cuadrados mínimos se obtiene una serie que puede servir para analizar estacionariedad.

Ajuste por cuadrados mínimos: caso cíclico




```
dataset=sig
interval=1
diff = list()
for i in range(interval, len(dataset)):
```

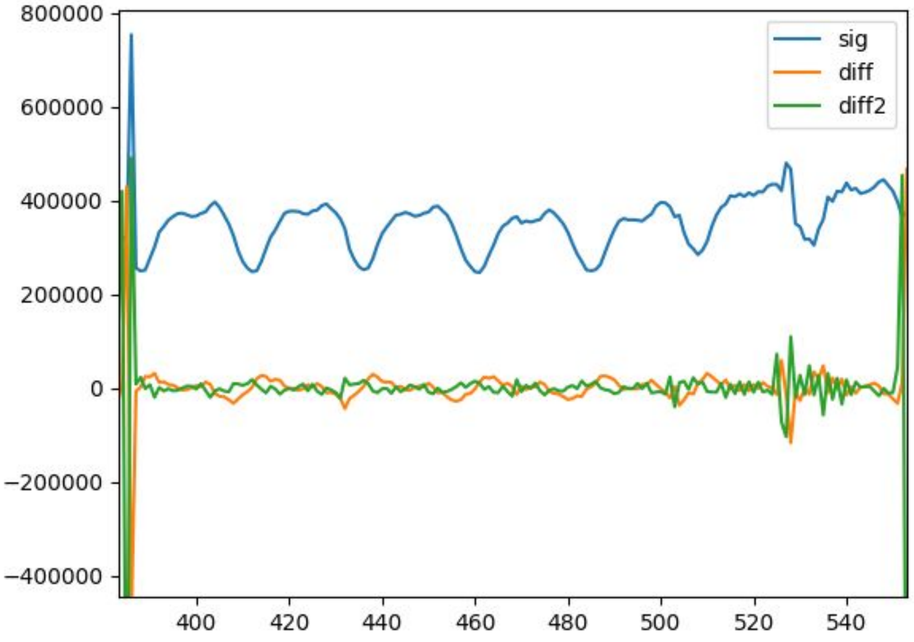
```
    value = dataset[i] - dataset[i - interval]
    diff.append(value)

plt.plot(sig)
plt.plot(diff)
plt.legend(['sig','diff'])
plt.show()

#repito la estimacion pero para la diferenciada
N=24 #hours
dataframe = pd.Series(pd.concat([pd.Series(diff[0]),p
ts=pd.DataFrame(dataframe.values)
rows=int(len(ts)/N)
data = ts.values.reshape(rows,N)

betas=data.mean(axis=0)
```

Ajuste por cuadrados mínimos: caso cíclico

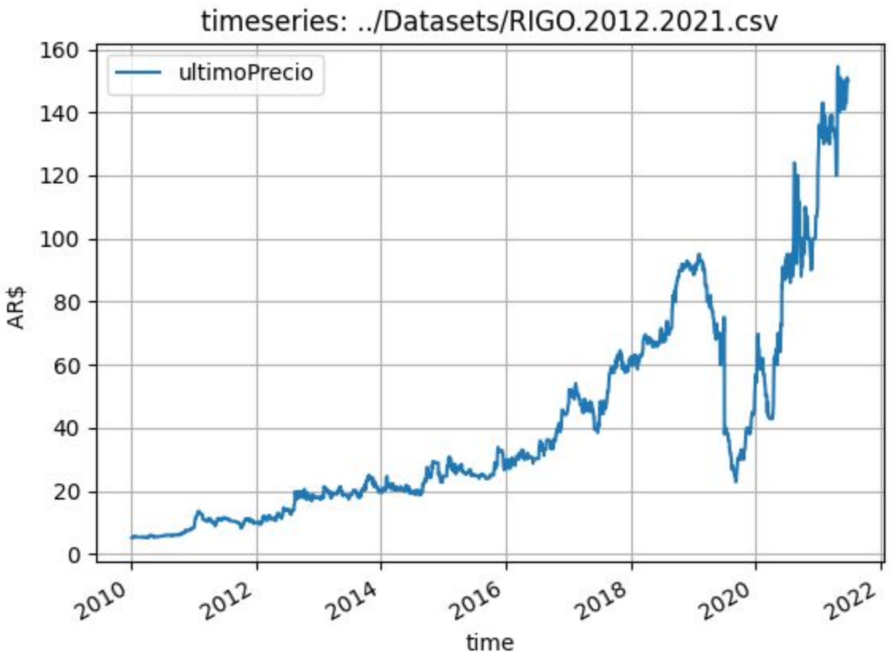


La diferenciación es una transformación muy útil para evaluar estacionariedad.

Observar que la serie original (en azul) es diferenciada y luego oscila alrededor del valor nulo (en naranja).

Hay que testear si la curva naranja es estacionaria.

Análisis de series de tiempo



En esta serie se puede apreciar una evolución creciente a lo largo del tiempo y una caída marcada en el año 2020. Sin embargo, luego de la caída parece mantenerse una tendencia creciente. Se puede observar también que al avanzar los años, aumentan también las variaciones punto a punto o diarias.

Análisis de series de tiempo: TECO.2000.2021

Ejemplo de preprocesamiento

```
inputfile = "../Datasets/TECO2.2000.2021.csv"


ts = pd.read_csv(inputfile, header=0, index_col=0, squeeze=True)
ts.fechaHora = pd.to_datetime(ts.fechaHora)
ts.fechaHora=pd.to_datetime(ts.fechaHora).dt.date
ts.fechaHora=pd.DatetimeIndex(ts.fechaHora)
ts=ts.sort_index(ascending=False)
```

Ejemplo de preprocesamiento

El eje de abscisas es ahora la base de tiempo, en formato *datetime64.*

Con este formato se pueden hacer transformaciones temporales, por ejemplo el promedio semanal, el máximo mensual, etcétera.

El formato con base temporal permite hacer otras operaciones como decimación.

Ejemplo de ajuste usando ARIMA

```
# fit model
model = ARIMA(ts.ultimoPrecio.values, order=(5,1,0))
model_fit = model.fit()
# summary of fit model
print(model_fit.summary())
# line plot of residuals
residuals = DataFrame(model_fit.resid)
residuals.plot()
pyplot.show()
# density plot of residuals
residuals.plot(kind='kde')
pyplot.show()
# summary stats of residuals
print(residuals.describe())
```

Ejemplo de ajuste usando ARIMA

```
# evaluate an ARIMA model using a walk-forward validation
X = ts.ultimoPrecio.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
# walk-forward validation
for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    #print('predicted=%f, expected=%f' % (yhat, obs))
```

Ejemplo de ajuste usando ARIMA