# Phân tích chuỗi thời gian sử dụng Kalman filter

1. Tổng quan

* Dữ liệu gồm 12 features (id, Podcast\_Name, Episode\_Title, Episode\_Lenghth, Genre, Host\_Popularity, Publication\_Day, Publication\_Time, Guest\_Popularity, Number\_of\_Ads) 750000 bản ghi.
* Dự đoán cho nhãn Listening\_Time\_minutes.
* Áp dụng Kalman filter vào 3 thuật toán chuỗi thời gian AR, MA, ARIMA.

1. Dữ liệu

* Kiểm tra dữ liệu:
* <class 'pandas.core.frame.DataFrame'>
* RangeIndex: 750000 entries, 0 to 749999
* Data columns (total 12 columns):
* # Column Non-Null Count Dtype
* --- ------ -------------- -----
* 0 id 750000 non-null int64
* 1 Podcast\_Name 750000 non-null object
* 2 Episode\_Title 750000 non-null object
* 3 Episode\_Length\_minutes 662907 non-null float64
* 4 Genre 750000 non-null object
* 5 Host\_Popularity\_percentage 750000 non-null float64
* 6 Publication\_Day 750000 non-null object
* 7 Publication\_Time 750000 non-null object
* 8 Guest\_Popularity\_percentage 603970 non-null float64
* 9 Number\_of\_Ads 749999 non-null float64
* 10 Episode\_Sentiment 750000 non-null object
* 11 Listening\_Time\_minutes 750000 non-null float64
* dtypes: float64(5), int64(1), object(6)
* Yêu cầu đề bài: MSV %7 +1
* 🡪 2251262641%7+1 = 6 Đề 6 lọc các dòng có Publication\_Day = Saturday

df\_saturday = df[df['Publication\_Day'] == 'Saturday'].copy()

* Lấy chỉ số làm chuỗi thời gian giả định và Listening\_time\_minutes làm giá trị nhãn
* target\_series = df\_saturday['Listening\_Time\_minutes'].reset\_index(drop=True)
* target\_series = target\_series.dropna()
* *# Kiểm tra một vài giá trị đầu tiên*
* target\_series.head()

1. Áp dụng Kalman filter

def simple\_kalman\_filter(z, Q=1e-5, R=1.0):

n = len(z)

xhat = np.zeros(n) *# a posteri estimate of x*

P = np.zeros(n) *# a posteri error estimate*

xhatminus = np.zeros(n) *# a priori estimate of x*

Pminus = np.zeros(n) *# a priori error estimate*

K = np.zeros(n) *# gain or blending factor*

*# Initial guess*

xhat[0] = z[0]

P[0] = 1.0

for k in range(1, n):

xhatminus[k] = xhat[k-1]

Pminus[k] = P[k-1] + Q

K[k] = Pminus[k] / (Pminus[k] + R)

xhat[k] = xhatminus[k] + K[k] \* (z[k] - xhatminus[k])

P[k] = (1 - K[k]) \* Pminus[k]

return xhat

* Xây dựng hàm theo công thức

+ Dự đoán trạng thái tiếp theo

A math equation on a dark background

AI-generated content may be incorrect.

xhatminus[k] = xhat[k-1]

+ Dự đoán hiệp phương sai lỗi

A math formula on a dark background

AI-generated content may be incorrect.

Pminus[k] = P[k-1] + Q

+ Tính hệ số Kalman Gain

K[k] = Pminus[k] / (Pminus[k] + R)

+ Cập nhật trạng thái

xhat[k] = xhatminus[k] + K[k] \* (z[k] - xhatminus[k])

+ Cập nhật hiệp phương sai lỗi

P[k] = (1 - K[k]) \* Pminus[k]

* Số lượng phép đo được gán bằng số lượng mẫu
* Khởi tạo Q và R

+ việc khởi tạo Q = 1e-5 (quá nhỏ) kết hợp với R = 1.0 (lớn)

Khiến cho bộ lọc không dự báo sát với thực tế dữ liệu.

+ Hiệu chỉnh chỉ số Q = 1e-2 và R = 0.5 sẽ giúp mô hình dự đoán sát với thực tế hơn.

1. Áp dụng vào mô hình

*# 1. Mô hình AR*

ar\_model = AutoReg(train, lags=20).fit()

ar\_preds = ar\_model.predict(start=len(train), end=len(train)+len(test)-1)

*# 2. Mô hình MA (ARIMA(0,0,q))*

ma\_model = ARIMA(train, order=(0, 0, 10)).fit()

ma\_preds = ma\_model.forecast(steps=len(test))

*# 3. Mô hình ARIMA (tùy chọn cấu hình)*

arima\_model = ARIMA(train, order=(5, 0, 5)).fit()

arima\_preds = arima\_model.forecast(steps=len(test))

*# Đánh giá*

print("RMSE AR:", np.sqrt(mean\_squared\_error(test, ar\_preds)))

print("RMSE MA:", np.sqrt(mean\_squared\_error(test, ma\_preds)))

print("RMSE ARIMA:", np.sqrt(mean\_squared\_error(test, arima\_preds)))

* Bộ dữ liệu sau khi áp dụng bộ lọc đưa vào các mô hình timeseres AR, MA và ARIMA

RMSE AR: 1.0183313885951273

RMSE MA: 0.9783629311069378

RMSE ARIMA: 1.8263060994856264

* MA là mô hình có độ chính xác cao nhất trong khi đó ARIMA có độ chính xác thấp nhất.
* Việc hiệu chỉnh các hệ số trong Kalman filter ảnh hưởng khác nhau đến các thuật toán. Vì vậy, cần tính toán kỹ lưỡng để chọn ra hệ số phù hợp giúp giảm sai số để phù hợp với mô hình.