# BÀI 3 – PHÂN CỤM

Ở bài này ta sẽ phân cụm các mã chứng khoán lại với nhau theo giá thị trường của chúng bằng phương pháp *k*-means với 3 độ đo khoảng cách: Euclidean, DBA và Soft-DTW. Bài này sẽ sử dụng thư viện *tslearn* và dữ liệu đã được chuẩn hóa ở bài 2.

Trước tiên, ta lấy ra danh sách các mã chứng khoán bằng cách lấy theo danh sách cột.

symbol\_list = df.columns

symbol = []

for i in symbol\_list:

  symbol.append(str(i))

## *3.1. Standard Euclidean k-means*

Ta sẽ dùng k-means với độ đo khoảng cách giữa các điểm dữ liệu là khoảng cách Euclid.

Trước tiên ta sử dụng phương pháp elbow để xác định số k nhóm thích hợp. Ta sẽ chạy vòng lặp từ 1 đến 10, với mỗi vòng lặp ta tạo một model k-means với số k là số lần lặp. Sau đó ta lấy distortions bằng cách gọi thuộc tính *inertia\_*, thuộc tính này trả về tổng khoảng cách giữa tâm cụm đến các điểm. Thông số này sẽ là thước đo để so sánh giữa các model với số k nhóm khác nhau. Ta cũng tiến hành đo thời gian xây dựng model với thư viện *time*.

start = time.time()

distortions = []

euclidean\_model = []

for k in range(1,10):

  euclidean\_model.append(TimeSeriesKMeans(n\_clusters=k, random\_state = 42))

  euclidean\_model[k-1].fit(data)

  distortions.append(euclidean\_model[k-1].inertia\_)

print('Time: ', time.time() - start)

Time: 0.7796363830566406

Xong đoạn mã trên, ta tiến hành vẽ sơ đồ distortion của từng model để trực quan hóa phương pháp elbow.

plt.figure(figsize=(16,8))

plt.plot(range(1, 10), distortions, 'bo-')

plt.xlabel('k')

plt.ylabel('Distortion')

plt.title('Elbow Method')

plt.show()

Chart

Description automatically generated with medium confidence

*Phương pháp elbow với euclidean k-means*

Nhìn hình ta có thể thấy điểm khuỷu tay ở hình chính là ở k = 2.

Với k = 2, ta chọn mô hình *k*-means thứ 2 đã tạo ở vòng lặp trên và tiến hành phân cụm dữ liệu với phương thức *predict()*.

y\_pred\_euclidean = euclidean\_model[1].predict(data)

Ta tạo mảng *cluster\_index[]*, đây là mảng sẽ chứa vị trí của các điểm dữ liệu theo nhóm phân cụm, ta cũng tiến hành vẽ kết quả phân cụm.

Cho chạy vòng lặp k = 2 lần, với mỗi vòng lặp ta lưu lại vị trí của từng nhóm phân cụm vào *cluster\_index[]* và vẽ kết quả phân cụm của nhóm đó. Biến *count* là biến đếm số phần tử của mỗi nhóm.

Với từng phần tử của mỗi nhóm phân cụm, ta dùng hàm *ravel()* để làm phẳng nó và thể hiện lên sơ đồ. Với mỗi nhóm phân cụm, ta dùng *cluster\_centers\_* để lấy đường trung tâm của nhóm đó.

plt.figure(figsize=(20, 5))

for i in range(2):

  plt.subplot(1, 2, i + 1)

  cluster\_index.append(np.where(y\_pred\_euclidean == i))

  count = 0

  for x in data[y\_pred\_euclidean == i]:

    count += 1

    plt.plot(x.ravel(), 'k-', alpha = 0.2)

  plt.plot(euclidean\_model[1].cluster\_centers\_[i].ravel(), 'r-')

  plt.title('Cluster %d: %d stocks' % ((i + 1), count))

plt.show()

A picture containing text

Description automatically generated

Hình trên thể hiện 2 sơ đồ với mỗi sơ đồ là một nhóm phân cụm với số phần tử(mã chứng khoán) của từng nhóm. Đường màu xám là đường biến động giá chứng khoán và đường màu đỏ chính là đường trung tâm của các đường màu xám.

Cuối cùng, ta tiến hành lấy danh sách các mã chứng khoán theo từng nhóm phân cụm dựa vào *cluster\_index[]* đã tạo ở trên.

#Danh sách các stock theo nhóm phân cụm

cluster\_euclidean = []

for arr in cluster\_index:

  for i in arr:

    tmp = []

    for j in i:

      tmp.append(symbol[int(j)])

  cluster\_euclidean.append(tmp)

for i, el in zip(range(2), cluster\_euclidean):

  print(i+1, el)

1 ['AAA', 'AAM', 'ABT', 'ACC', 'ACL', 'ADS', 'AGM', 'AGR', 'AMD', 'ANV', 'APC', 'APG', 'ASM', 'ASP', 'BCE', 'BCG', 'BFC', 'BIC', 'BMC', 'BMI', 'BRC', 'BSI', 'BTP', 'BWE', 'C32', 'C47', 'CCI', 'CCL', 'CDC', 'CEE', 'CHP', 'CIG', 'CII', 'CKG', 'CLC', 'CLL', 'CLW', 'CMV', 'CMX', 'CNG', 'CRC', 'CRE', 'CSM', 'CSV', 'CTF', 'CTG', 'CTI', 'CTS', 'CVT', 'DAG', 'DAH', 'DAT', 'DCL', 'DCM', 'DHM', 'DIG', 'DLG', 'DPM', 'DQC', 'DRC', 'DRH', 'DTA', 'DTL', 'DTT', 'DXG', 'DXV', 'EIB', 'ELC', 'EMC', 'EVE', 'FCM', 'FCN', 'FDC', 'FIR', 'FIT', 'FLC']

2 ['AGG', 'AST', 'BBC', 'BHN', 'BID', 'BMP', 'BTT', 'BVH', 'CAV', 'CMG', 'COM', 'CTD', 'D2D', 'DBC', 'DBD', 'DHA', 'DHC', 'DHG', 'DMC', 'DPG', 'DPR', 'DRL', 'DSN', 'DVP']

## *3.2. DBA k-means*

Ta sẽ dùng k-means với độ đo khoảng cách giữa các điểm dữ liệu là Dynamic time warping hay độ tương đồng giữa 2 chuỗi thời gian.

Các bước thực hiện cũng tượng tự như Euclidean k-means. Ta vẫn sẽ dùng phương pháp elbow để xác định số k nhóm phù hợp. Lúc tạo model, ta thêm tham số metric='dtw' để model sử dụng độ đo DTW.

distortions = []

dba\_model = []

start = time.time()

for k in range(1,10):

  dba\_model.append(TimeSeriesKMeans(n\_clusters=k, metric='dtw', random\_state = 42))

  dba\_model[k-1].fit(data)

  distortions.append(dba\_model[k-1].inertia\_)

print('Time: ', time.time() - start)

Time: 1492.416310787201

Với DBA k-means, ta tạo 10 model với thời gian hơn 24 phút, lâu hơn nhiều so với Euclidean k-means.

Tiếp theo, ta trực quan hóa phương pháp elbow.

Chart

Description automatically generated with low confidence

Có thể thấy được k = 2 là phù hợp nhất. Nên ta lấy model thứ 2.

y\_pred\_dba = dba\_model[1].predict(data)

Tiếp theo ta tiến hành vẽ sơ đồ.

cluster\_index = []

plt.figure(figsize=(20, 5))

for i in range(2):

  plt.subplot(1, 2, i + 1)

  cluster\_index.append(np.where(y\_pred\_dba == i))

  count = 0

  for x in data[y\_pred\_dba == i]:

    count +=1

    plt.plot(x.ravel(), 'k-', alpha = 0.2)

  plt.plot(dba\_model[1].cluster\_centers\_[i].ravel(), 'r-')

  plt.title('Cluster %d: %d stocks' % ((i + 1), count))

plt.show()

A picture containing graphical user interface

Description automatically generated

Có thể thấy đường trung tâm của các nhóm DBA k-means biến động nhiều hơn so với Euclidean k-means dù số mã chứng khoán theo nhóm của cả 2 gần tương tự với nhau.

Danh sách các mã chứng khoán theo từng nhóm phân cụm

#Danh sách các stock theo nhóm phân cụm

cluster\_dba = []

for arr in cluster\_index:

  for i in arr:

    tmp = []

    for j in i:

      tmp.append(symbol[int(j)])

  cluster\_dba.append(tmp)

for i, el in zip(range(2), cluster\_dba):

  print(i+1, el)

1 ['AAA', 'AAM', 'ABT', 'ACC', 'ACL', 'ADS', 'AGM', 'AGR', 'AMD', 'ANV', 'APC', 'APG', 'ASM', 'ASP', 'BCE', 'BCG', 'BFC', 'BIC', 'BID', 'BMC', 'BMI', 'BRC', 'BSI', 'BTP', 'BWE', 'C32', 'C47', 'CCI', 'CCL', 'CDC', 'CEE', 'CHP', 'CIG', 'CII', 'CKG', 'CLC', 'CLL', 'CLW', 'CMV', 'CMX', 'CNG', 'CRC', 'CRE', 'CSM', 'CTF', 'CTG', 'CTI', 'CTS', 'CVT', 'DAG', 'DAH', 'DAT', 'DCL', 'DCM', 'DHM', 'DLG', 'DPM', 'DQC', 'DRC', 'DRH', 'DTA', 'DTL', 'DTT', 'DXG', 'DXV', 'EIB', 'ELC', 'EMC', 'EVE', 'FCM', 'FCN', 'FDC', 'FIR', 'FIT', 'FLC']

2 ['AGG', 'AST', 'BBC', 'BHN', 'BMP', 'BTT', 'BVH', 'CAV', 'CMG', 'COM', 'CSV', 'CTD', 'D2D', 'DBC', 'DBD', 'DHA', 'DHC', 'DHG', 'DIG', 'DMC', 'DPG', 'DPR', 'DRL', 'DSN', 'DVP']

## *3.3. Soft-DTW k-means*

Ta sẽ dùng k-means với độ đo khoảng cách giữa các điểm dữ liệu là chỉ số soft-DTW.

Các bước thực hiện cũng tượng tự như Euclidean k-means. Ta vẫn sẽ dùng phương pháp elbow để xác định số k nhóm phù hợp. Lúc tạo model, ta thêm tham số metric='softdtw'. Đối với Soft-DTW k-means, ta thêm tham số metric\_params để tránh bị overfiting.

distortions = []

soft\_dtw\_model = []

start = time.time()

soft\_dtw\_model.append(TimeSeriesKMeans(n\_clusters=1, metric='softdtw', metric\_params={"gamma": .01}, max\_iter = 10, random\_state = 42))

soft\_dtw\_model[0].fit(data)

distortions.append(soft\_dtw\_model[0].inertia\_)

for k in range(2,10):

  idx\_new\_center = np.random.choice(data.shape[0])

  barycenters\_init = np.vstack((soft\_dtw\_model[-1].cluster\_centers\_, data[np.newaxis, idx\_new\_center]))

  soft\_dtw\_model.append(TimeSeriesKMeans(n\_clusters=k, metric="softdtw", metric\_params={"gamma": .01}, max\_iter=10, init=barycenters\_init))

  soft\_dtw\_model[k-1].fit(data)

  distortions.append(soft\_dtw\_model[k-1].inertia\_)

print('Time: ', time.time() - start)

Time: 3126.5598282814026

Có thể thấy Soft-DTW *k*-means chạy lâu nhất trong 3 cách, tốn hơn 50 phút.

Tiếp theo, ta vẽ sơ đồ elbow

A picture containing shape

Description automatically generated

Nhìn hình ta thấy được rằng k = 2 là phù hợp nhất. Ta sẽ chọn mô hình thứ 2 để phân cụm dữ liệu và vẽ sơ đồ.

y\_pred\_soft = soft\_dtw\_model[1].predict(data)

cluster\_index = []

plt.figure(figsize=(20, 5))

for i in range(2):

  plt.subplot(1, 2, i + 1)

  cluster\_index.append(np.where(y\_pred\_soft == i))

  count = 0

  for x in data[y\_pred\_soft == i]:

    count +=1

    plt.plot(x.ravel(), 'k-', alpha = 0.2)

  plt.plot(soft\_dtw\_model[1].cluster\_centers\_[i].ravel(), 'r-')

  plt.title('Cluster %d: %d stocks' % ((i + 1), count))

plt.show()

A picture containing text

Description automatically generated

#Danh sách các stock theo nhóm phân cụm

cluster\_soft\_dtw = []

for arr in cluster\_index:

  for i in arr:

    tmp = []

    for j in i:

      tmp.append(symbol[int(j)])

  cluster\_soft\_dtw.append(tmp)

for i, el in zip(range(2), cluster\_soft\_dtw):

  print(i+1, el)

1 ['AAA', 'AAM', 'ABT', 'ACC', 'ACL', 'ADS', 'AGM', 'AGR', 'AMD', 'ANV', 'APC', 'APG', 'ASM', 'ASP', 'BCE', 'BCG', 'BFC', 'BIC', 'BMC', 'BMI', 'BRC', 'BSI', 'BTP', 'BWE', 'C32', 'C47', 'CCI', 'CCL', 'CDC', 'CEE', 'CHP', 'CIG', 'CII', 'CKG', 'CLC', 'CLL', 'CLW', 'CMV', 'CMX', 'CNG', 'CRC', 'CRE', 'CSM', 'CSV', 'CTF', 'CTG', 'CTI', 'CTS', 'CVT', 'DAG', 'DAH', 'DAT', 'DCL', 'DCM', 'DHM', 'DLG', 'DPM', 'DQC', 'DRC', 'DRH', 'DTA', 'DTL', 'DTT', 'DXG', 'DXV', 'EIB', 'ELC', 'EMC', 'EVE', 'FCM', 'FCN', 'FDC', 'FIR', 'FIT', 'FLC']

2 ['AGG', 'AST', 'BBC', 'BHN', 'BID', 'BMP', 'BTT', 'BVH', 'CAV', 'CMG', 'COM', 'CTD', 'D2D', 'DBC', 'DBD', 'DHA', 'DHC', 'DHG', 'DIG', 'DMC', 'DPG', 'DPR', 'DRL', 'DSN', 'DVP']

Ta thấy rõ được rằng cả 3 cách đều cho ra 2 nhóm phân cụm với số phần tử mỗi nhóm tượng tự nhau (một nhóm hơn 70 phần tử và một nhóm hơn 20 phần tử).

# BÀI 4 - DỰ ĐOÁN GIÁ CHỨNG KHOÁN AAA

## *4.1. Xử lý dữ liệu với MinMaxScaler*

Mã chứng khoán ta sẽ chọn để thực hiện bài toán dự đoán ở đây là AAA thuộc Công ty cổ phần Nhựa An Phát Xanh.

Bởi vì *TimeSeriesScalerMeanVariance* không có hàm scale ngược từ giá trị scaled về giá trị ban đầu nên ở bài này ta sẽ chuẩn hóa dữ liệu theo *MinMaxScaler*.

Đầu tiên, ta đọc file *stock.csv* để lấy dữ liệu.

#Đọc file data

df = pd.read\_csv('/content/drive/My Drive/Data/Final/stock.csv')

Table

Description automatically generated

Tiếp theo, ta scale dữ liệu về 0 -> 1 bằng *MinMaxScaler()*

#Scale dữ liệu

scaler = MinMaxScaler()

df\_sc = df.copy()

df\_sc.iloc[:, 1:] = scaler.fit\_transform(df\_sc.iloc[:, 1:])

Application, table

Description automatically generated

Ở cột ngày, ta chọn những dòng dữ liệu năm 2022 để làm dữ liệu test

#Tìm index của 2022 để test -> index từ 431

df\_test = df\_sc[df\_sc['Unnamed: 0'].str.contains("2022")]['AAA']

431 0.843400

432 0.862043

433 0.970172

434 1.000000

435 0.884415

...

525 0.250559

526 0.243102

527 0.250559

528 0.239374

529 0.231916

Name: AAA, Length: 99, dtype: float64

Từ kết quả, ta có thể thấy dữ liệu test gồm có 99 dòng sẽ bắt đầu từ dòng thứ 431 nên những dòng trước đó ta sẽ lấy làm dữ liệu train.

#Lấy price của 2020, 2021 để train, Lấy price của 2022 để test

data\_AAA = df\_sc['AAA'].values

data\_train = data\_AAA[:431]

data\_test = data\_AAA[431:]

## *4.2. Simple Exponential Smoothing*

Phương pháp ta sẽ dùng để dự đoán cho bài toán này là phương pháp Simple Exponential Smoothing. Simple Exponential Smoothing (SES) là một phương pháp dự đoán dữ liệu theo thời gian, nó sử dụng tổng có trọng lượng của dữ liệu trước trong dãy dữ liệu theo thời gian để đoán các dữ liệu trong tương lai.

Và vì dữ liệu này không có trend cũng như không có seasonality, nên phương pháp SES sẽ rất phù hợp.

Chart, line chart

Description automatically generated

Ta sử dụng *SimpleExpSmoothing()* để tạo model và cho chạy vòng lặp với từng phần tử trong tập test. Với lần chạy đầu tiên, model sẽ sử dụng dữ liệu train để dự đoán giá chứng khoán của ngày thứ 431 và sau đó thêm dữ liệu giá chứng khoán ngày thứ 431 từ dữ liệu test vào dữ liệu train, và cứ tiếp tục như vậy cho đến hết vòng lặp. Model sẽ dự đoán dữ liệu của ngày tiếp theo dựa vào việc học tất cả dữ liệu trước đó, kể cả dữ liệu trong dữ liệu test.

from statsmodels.tsa.holtwinters import SimpleExpSmoothing

history = data\_train

predictions = []

for new\_data in data\_test:

  model = SimpleExpSmoothing(history)

  model\_fit = model.fit()

  prediction = model\_fit.forecast(steps=1)

  predictions.append(prediction[0])

  history = np.append(history, new\_data)

Sau khi train model và dự đoán xong, ta tiến hành scale ngược lại từ giá trị normalized về giá trị ban đầu. Ta ghép mảng dữ liệu train và dữ liệu dự đoán vào với nhau và thay vào cột dữ liệu của mã chứng khoán AAA trong dataframe.

df\_tmp = df\_sc.copy()

line = np.append(data\_train, predictions)

df\_tmp['AAA'] = line

Sau đó ta scale ngược về giá trị ban đầu bằng *inverse\_transform()* và lấy ra những giá trị thuộc AAA tức cột đầu tiên. Giá trị dự đoán là những giá trị bắt đầu từ vị trí 431 trong mảng.

lines = scaler.inverse\_transform(df\_tmp.iloc[:, 1:])

lines\_AAA = lines[:, 0]

prediction\_line = lines\_AAA[431:]

Cuối cùng, ta hiển thị kết quả so sánh giữa kết quả thực tế và kết quả dự đoán

plt.figure(figsize=(15, 5))

plt.plot(range(0, 530), df['AAA'].values, 'b-', label = 'Actual data')

plt.plot(range(431, 530), prediction\_line, 'orange', label = 'Predicted data')

plt.title('Simple Exponential Smoothing Prediction')

plt.xlabel('Dates')

plt.ylabel('Stock price')

plt.legend(loc='upper left')

plt.grid()

plt.show()

Graphical user interface, chart, line chart

Description automatically generated

Đường màu xanh chính là giá chứng khoán thực tế từ năm 2020 hết tháng 5 năm 2022, đường màu cam là giá chứng khoán dự đoán 5 tháng đầu năm 2022. Từ hình có thể thấy đường màu cam gần như trùng với phần cuối của đường màu xanh, điều này chứng tỏ mô hình SES đã dự đoán tốt giá chứng khoán của 5 tháng đầu năm 2022.