FINAL REPORT

GÓI PHẦN MỀM ỨNG DỤNG CHO TÀI CHÍNH

Name: Đặng Đức Duy

Student ID: K205030798

Ho Chi Minh 01 - 2023

TOPIC: PHÂN LOẠI CÔNG TY CHỨNG KHOÁN VIỆT NAM BẰNG PCA

Truy cập đường link vào github để xem tất cả về code:
https://github.com/DucDuyDang/G-I-PH-N-M-M-NG-D-NG-CHO-T-I-CH-NH/blob/main/README.md

Table of Contents

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1. Lý do chọn đề tài:

Trong lĩnh vực tài chính, thuật ngữ "đầu tư" (investing) và "kinh doanh" (trading) đều nhằm mục đích tìm kiếm lợi nhuận trên thị trường tài chính nhưng theo các cách thức khác nhau. Danh mục đầu tư (Investment Portfolio) là sự kết hợp các mã cổ phiếu khác nhau nhằm tối ưu hóa lợi nhuận và giảm thiểu rủi ro trong đầu tư. Vì rủi ro là một yếu tố luôn hiện hữu trong kinh doanh và đầu tư. Một nhà đầu tư thành công là người có thể cân bằng giữa hai yếu tố này và đạt mục tiêu đầu tư của mình.

Đa dạng hóa danh mục đầu tư là một trong những biện pháp quan trọng trong quản trị rủi ro khi đầu tư. Sự kết hợp phù hợp của nhiều mã chứng khoán trong ngành hoặc toàn thị trường giúp giảm rủi ro tổng thể so với việc nắm giữ một loại chứng khoán hay một loại tài sản duy nhất trên danh mục đầu tư.

Tùy vào nhu cầu đầu tư và theo biến động thị trường thì việc phân loại các doanh nghiệp có cùng mức độ biến động sẽ giảm thiểu rủi ro khi đầu tư vào 1 nhóm chứng khoán ấy. Một biến động của của nhóm ngành đó sẽ làm cho danh mục đầu tư của chúng ta sụt giảm nghiêm trọng vì thế phân loại dựa vào giá chứng khoán là một nhu cầu thiết thực và cấp thiết giúp nhà đầu tư tối ưu được vốn và năng cao lợi nhuận

2. Mục tiêu chủ đề:

-Tìm được sự liên quan giữa các mã chứng khoán trên sàn chứng khoán Việt Nam dựa vào biến động về giá theo phương pháp PCA.

-Tìm được quy luật (rule)/ nguyên nhân tạo ra sự phân loại không theo nhóm về cơ cấu vốn/ kết quả kinh doanh của các ngân hàng đã lên sàn chứng khoán VN.

-Đánh giá tác động của việc lên sàn chứng khoán đến giá cổ phiếu nhóm ngân hàng.

3. Cấu trúc báo cáo:

Chương 1: Tổng quan đề tài
Chương 2: Cơ sở lý luận
Chương 3: Phân tích bộ dữ liệu
Chương 4: Kết luận
Phụ lục code

CHƯƠNG 2: CƠ SỞ LÝ LUẬN

1. Lý thuyết và phương pháp sử dụng:

1.1. Khái niệm PCA:

Principal Component Analysis (PCA) là Phân tích thành phần chính (PCA). Đây là nghĩa tiếng Việt của thuật ngữ Principal Component Analysis (PCA)
Phép phân tích thành phần chính (PCA) là một kỹ thuật được sử dụng để xác định một số lượng nhỏ của các biến không tương quan được gọi là thành phần chủ yếu từ một tập lớn của dữ liệu. Kỹ thuật này được sử dụng rộng rãi để nhấn mạnh sự thay đổi và nắm bắt được mô hình mạnh mẽ trong một tập dữ liệu. Phát minh bởi Karl Pearson vào năm 1901, phân tích thành phần chính là một công cụ được sử dụng trong mô hình dự báo và phân tích dữ liệu thăm dò. phép phân tích thành phần chính được coi là một phương pháp thống kê hữu ích và được sử dụng trong các lĩnh vực như nén hình ảnh, nhận diện khuôn mặt, khoa học thần kinh và đồ họa máy tính.
Đây là thuật toán sinh ra để giải quyết vấn đề dữ liệu có quá nhiều chiều dữ liệu, cần giảm bớt chiều dữ liệu nhằm tăng tốc độ xử lí, nhưng vẫn giữ lại thông tin nhiều nhất có thể (high variance).

1.2. Đặc điểm:

- PCA chuyển dữ liệu từ linear thành các thuộc tính mới không liên quan lẫn nhau.
- Chúng ta cần tìm ra chiều dữ liệu có độ quan trọng cao, nhằm giảm bớt việc tính toán, cũng như tăng tốc độ xử lý.

Việc làm như trên sẽ giúp cho chúng ta:

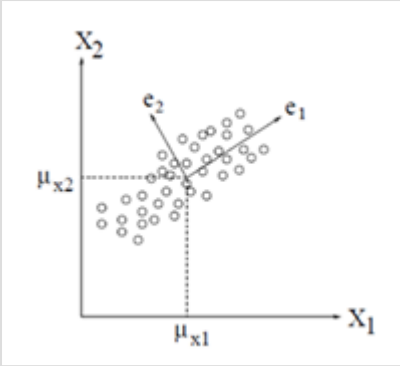
- Giảm chiều dữ liệu mà vẫn giữ được đặc trưng chính, chỉ mất đi "chút ít" đặc trưng.
- Tiết kiệm thời gian, chi phí tính toán
- Dễ dàng visualize dữ liệu hơn để giúp ta có cái nhìn trực quan hơn.

PCA tìm ra mean và principal components.

1.3. Phương pháp thực hiện của PCA:

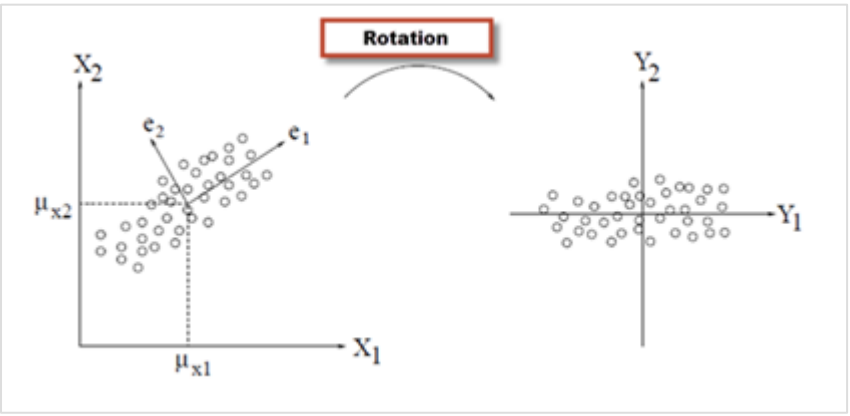
- Biến đổi X về dạng đồng nhất.
- Tính toán covariance matrix Σ
- Tìm eigenvectors của Σ
- Lấy K dimensions có giá trị variance cao nhất

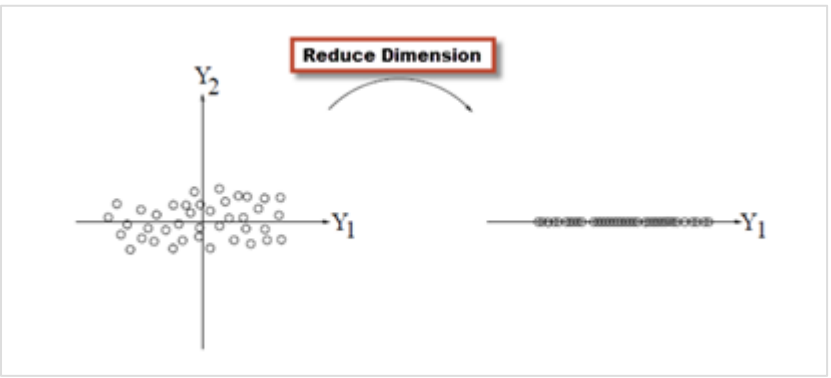
* Chúng ta sẽ bắt đầu bằng một ví dụ mang tính chất minh họa trực quan:
Chúng ta có 2 biến X1 và X2 có tương quan (tuyến tính), được biểu diễn bằng đồ thị sau:



Ta biết rằng khi thực hiện các phân tích đa biến mà trong đó các biến có tương quan với nhau là rất khó chịu!

Ta sẽ loại bỏ sự tương quan này bằng cách xoay trục (cơ sở)





Ta thấy rằng dữ liệu trên trục mới đã giảm sự tương quan đáng kể (biến Y1 và Y2 gần như không tương quan), và sự thay đổi của dữ liệu phụ thuộc phần lớn vào biến Y1, ta có thể chỉ dùng một biến Y1 để biểu diễn dữ liệu, điều này giúp ta giảm số chiều dữ liệu mà không làm giảm quá nhiều "phương sai" của dữ liệu. Đây cũng chính là tư tưởng của phương pháp PCA. Sau đây chúng ta sẽ làm rõ hơn về mặt lý thuyết của phương pháp này

1.2. Áp dụng phương pháp PCA vào tài chính:

Phương thức giảm chiều dữ liệu để dự báo giá chứng khoán có thể được sử dụng (số lượng hiện tại còn khá ít) mặc dù hiện nay đã có rất nhiều kỹ thuật dự báo thị trường chứng khoán nói chung và giá cổ phiếu nói riêng.

Dự báo lợi nhuận của thị trường chứng khoán theo ngày bằng cách sử dụng kỹ thuật PCA và 02 kỹ thuật PCA phi tuyến khác là phân tích thành phần chính mờ mạnh (RFPCA) và phân tích thành phần chính hạt nhân (KPCA) để giảm chiều của tập dữ liệu và sử dụng kỹ thuật mạng nơtron nhân tạo (ANN) để phân lớp. Bài báo đó đã chỉ ra rằng PCA+ANN cho kết quả dự báo phân lớp tốt hơn so với RFPCA+ANN và KPCA+ANN.
sử dụng kỹ thuật PCA để giảm tiếp chiều biến của tập dữ liệu sau lần giảm đầu và cuối cùng sử dụng mô hình ADL được ước lượng theo phương pháp hồi quy nhiều biến để dự báo chỉ số VNINDEX theo ngày. Độ chính xác dự báo theo phương pháp này tốt hơn so với phương pháp được đề xuất trong

2. Tổng quan các thư viện sử dụng :

2.1. Khái niệm về Scikit-learn:

Thư viện của Python cho phép sử dụng phương pháp PCA, đó là: sklearn
Scikit-learn (Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.
Thư viện được cấp phép bản quyền chuẩn FreeBSD và chạy được trên nhiều nền tảng Linux. Scikit-learn được sử dụng như một tài liệu để học tập.
Để cài đặt scikit-learn trước tiên phải cài thư viện SciPy (Scientific Python). Những thành phần gồm:

- Numpy: Gói thư viện xử lý dãy số và ma trận nhiều chiều
- SciPy: Gói các hàm tính toán logic khoa học
- Matplotlib: Biểu diễn dữ liệu dưới dạng đồ thị 2 chiều, 3 chiều
- IPython: Notebook dùng để tương tác trực quan với Python
- SymPy: Gói thư viện các kí tự toán học
- Pandas: Xử lý, phân tích dữ liệu dưới dạng bảng

2.2.Mô hình GMM (Gaussian Mixture Model):

Mô hình Gaussian hỗn hợp (Gaussian Mixture Model - GMM) là một hàm tham số mật độ xác suất được biểu diễn như là một tổng trọng số của các mật độ Gaussian thành phần
Để đại diện cho một quần thể con được phân phối chuẩn trong một tổng thể, chúng tôi sử dụng Gaussian Mixture Model. GMM không yêu cầu dữ liệu thuộc về dân số con. Điều này cho phép mô hình tự động tìm hiểu các quần thể con. Vì chúng ta không biết sự phân công của dân số con, nên nó không được giám sát.
Tại sao chúng ta cần Gaussian Mixture Model?

Có hai lĩnh vực phổ biến nhất của Machine learning – Học có giám sát và Học không được giám sát. Chúng ta có thể dễ dàng phân biệt giữa hai loại này dựa trên bản chất của dữ liệu mà chúng sử dụng và các phương pháp tiếp cận để giải quyết vấn đề. Để phân cụm các điểm dựa trên các đặc điểm tương tự, chúng tôi sử dụng các thuật toán phân cụm.
Các mô hình Gaussian (GMM) thường được sử dụng để phân cụm dữ liệu. Bạn có thể sử dụng GMM để thực hiện phân cụm cứng hoặc phân cụm mềm trên dữ liệu truy vấn. Để thực hiện phân cụm cứng, GMM gán các điểm dữ liệu truy vấn cho các thành phần thông thường đa biến để tối đa hóa xác suất sau thành phần, được cung cấp dữ liệu. Nghĩa là, với một GMM được sử dụng, cụm gán dữ liệu truy vấn cho thành phần mang lại xác suất nghiệm cao nhất.

2.3. Phân kỳ Kullback–Leibler

Kullback–Leibler (hay còn gọi là khoảng cách Kullback–Leibler, entropy tương đối) là một phép đo cách một phân phối xác suất khác biệt so với cái còn lại, phân phối xác suất tham chiếu. Các ứng dụng chứa đặc tính entropy thông tin quan hệ trong các hệ thống thông tin, ngẫu nhiên theo chuỗi thời gian liên tục, và thông tin đạt được khi so sánh các mô hình thống kê suy luận. Tương phản với sự thay đổi thông tin, phân kỳ Kullback–Leibler là một phép đo bất đối xứng phân phối thông minh và vì vậy không đủ điều kiện là một metric thống kê có tính lây lan. Trong trường hợp đơn giản, một phân kỳ Kullback–Leibler có giá trị 0 chỉ ra rằng hai phân phối là giống nhau. Nói cách đơn giản, phân kỳ Kullback–Leibler là một phép đo sự ngạc nhiên, với nhiều ứng dụng khác nhau

CHƯƠNG 3: PHÂN TÍCH BỘ DỮ LIỆU

1.Mô tả dữ liệu nguồn:

Ở bài báo cáo này, em lấy dữ liệu từ gói vnstock để đơn giản việc crawl data. Đầu tiên là việc lấy dữ liệu từ các sàn chứng khoán, với phương châm sử dụng tối đa những thứ có sẵn, em dùng tool https://github.com/phamdinhkhanh/vnquant .
Dùng tool này mình viết module lấy dữ liệu từ VNDIRECT. Các mã chứng khoán mình quan tâm mình lưu trữ trên bộ nhớ của máy.

2. Phân tích kết quả:

2.1.Import thư viện liên quan:

```
import pandas as pd
import numpy as np
import datetime as dt
import seaborn as sns
import matplotlib.pyplot as plt
from math import log2
from scipy.special import rel_entr
from operator import itemgetter
from vnstock import *

import cufflinks as cf
cf.go_offline()
import plotly.io as pio

from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler, normalize
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score
from sklearn import metrics
```

Sau khi đã import các thư viện như trên, chúng ta đã có thể liệt kê ra được các công ty đã thực hiện IPO cũng như niêm yết trên cả 3 sàn chứng khoán HOSE, HNX, Upcom.

```
#Liệt kê các mã chứng khoán có trên thị trường
df = listing_companies()
df.head()
```

	ticker	group_code	company_name	company_short_name
0	VVS	UpcomIndex	Công ty Cổ phần Đầu tư Phát triển Máy Việt Nam	Đầu tư Phát triển Máy Việt Nam
1	XDC	UpcomIndex	Công ty TNHH MTV Xây dựng Công trình Tân Cảng	Xây dựng Công trình Tân Cảng
2	HSV	UpcomIndex	Công ty Cổ phần Gang Thép Hà Nội	Gang Thép Hà Nội
3	CST	UpcomIndex	Công ty Cổ phần Than Cao Sơn - TKV	Than Cao Sơn - TKV
4	BVL	UpcomIndex	Công ty Cổ phần BV Land	BV Land

```
df.shape
```

```
(1631, 4)
```

Tạo 1 variable tên là df để lưu trữ tên tất cả các mã chứng khoán. Xem 5 dòng đầu tiên trong dataframe.

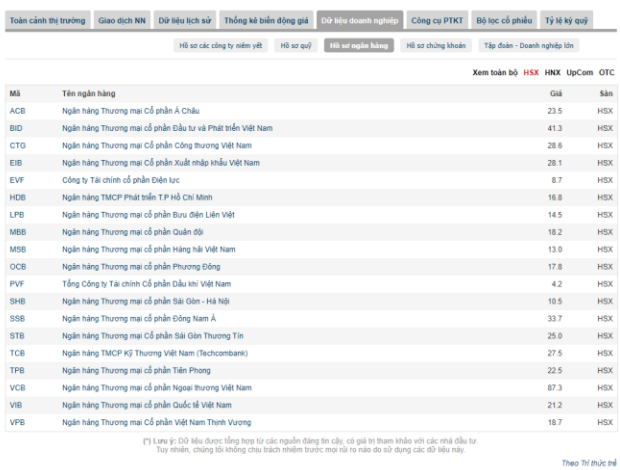
Tiếp theo biểu thị độ dài của mảng - tương ứng với số lượng các phần tử trên mỗi trục (kích thước) hay còn gọi là dòng và cột. Ở đây dữ liệu df của ta có 1641 dòng và 4 cột, tương ứng với 1641 mã/ công ty niêm yết

Giải thích về các columns:

+ ticker: mã chứng khoán
+ group_code: Mã đó được giao dịch tại sàn chứng khoán nào
+ company_name: Mỗi mã tương ứng với tên công ty niêm yết
+ company_short_name: tên thường gọi/ tên viết tắt

Theo thống kê tại cafeF (http://s.cafef.vn/du-lieu-doanh-nghiep.chn), hiện nay số lượng công ty niêm yết trên sàn chứng khoán là 1747 công ty(HOSE: 545 ; HNX: 341; Upcom: 861). Như vậy bộ dữ liệu của chúng ta có khiếm khuyết 106 mã chứng khoán.

Cũng theo cafeF, hiện nay có 17 ngân hàng đang niêm yết trên sàn chứng khoán HOSE bao gồm:



Toàn cảnh thị trường | Giao dịch NN | Dữ liệu lịch sử | Thống kê biến động giá | Dữ liệu doanh nghiệp | Công cụ PTKT | Bộ lọc cổ phiếu | Tỷ lệ ký quỹ

Hồ sơ các công ty niêm yết | Hồ sơ quỹ | Hồ sơ ngân hàng | Hồ sơ chứng khoán | Tập đoàn - Doanh nghiệp lớn

Xem toàn bộ HSX HNX UpCom OTC

Mã	Tên ngân hàng	Giá	Sàn
ACB	Ngân hàng Thương mại Cổ phần Á Châu	23.5	HSX
BID	Ngân hàng Thương mại cổ phần Đầu tư và Phát triển Việt Nam	41.3	HSX
CTG	Ngân hàng Thương mại Cổ phần Công thương Việt Nam	28.6	HSX
EIB	Ngân hàng Thương mại Cổ phần Xuất nhập khẩu Việt Nam	28.1	HSX
EVF	Công ty Tài chính cổ phần Điện lực	8.7	HSX
HDB	Ngân hàng TMCP Phát triển T.P Hồ Chí Minh	16.8	HSX
LPB	Ngân hàng Thương mại cổ phần Bưu điện Liên Việt	14.5	HSX
MBB	Ngân hàng Thương mại cổ phần Quân đội	18.2	HSX
MSB	Ngân hàng Thương mại cổ phần Hàng hải Việt Nam	13.0	HSX
OCB	Ngân hàng Thương mại cổ phần Phương Đông	17.8	HSX
PVF	Tổng Công ty Tài chính Cổ phần Dầu khí Việt Nam	4.2	HSX
SHB	Ngân hàng Thương mại cổ phần Sài Gòn - Hà Nội	10.5	HSX
SSB	Ngân hàng Thương mại cổ phần Đông Nam Á	33.7	HSX
STB	Ngân hàng Thương mại Cổ phần Sài Gòn Thương Tín	25.0	HSX
TCB	Ngân hàng TMCP Kỹ Thương Việt Nam (Techcombank)	27.5	HSX
TPB	Ngân hàng Thương mại cổ phần Tiên Phong	22.5	HSX
VCB	Ngân hàng Thương mại cổ phần Ngoại thương Việt Nam	87.3	HSX
VIB	Ngân hàng Thương mại cổ phần Quốc tế Việt Nam	21.2	HSX
VPB	Ngân hàng Thương mại Cổ phần Việt Nam Thịnh Vượng	18.7	HSX

(*) Lưu ý: Dữ liệu được tổng hợp từ các nguồn đáng tin cậy, có giá trị tham khảo với các nhà đầu tư. Tuy nhiên, chúng tôi không chịu trách nhiệm trước mọi rủi ro nào do sử dụng các dữ liệu này.

Theo Trí thức trẻ

Tiếp theo, tạo 1 list chứa tên các ngân hàng đã niêm yết trên sàn chứng khoán HOSE để phân tích.

```
#Tạo variable bao gồm các stock codes for analysis ở nhóm ngành Ngân hàng
a = ['BID', 'CTG', 'VCB', 'VIB', 'EIB', 'HDB', 'MBB', 'STB', 'TCB', 'TPB', 'ACB', 'LPB', 'MSB', 'OCB', 'SHB', 'SSB', 'VPB']
```

```
# Retrieve historical price data of any n stocks
start = "2017-01-01"
end  = dt.datetime.now().strftime("%Y-%m-%d")
empt_lists = []
for i in a:
    empt_lists.append(i)
    globals()[i] =  stock_historical_data(symbol=i, start_date=start, end_date=end).set_index('TradingDate')
empt_lists
```

```
['BID',
 'CTG',
 'VCB',
 'VIB',
 'EIB',
 'HDB',
 'MBB',
 'STB',
 'TCB',
 'TPB',
 'ACB',
 'LPB',
 'MSB',
 'OCB',
 'SHB',
 'SSB',
 'VPB']
```

Sử dụng gói vnstock để lấy dữ liệu từ ngày 01.01.2017 đến ngày hôm nay (ngày chạy code) thông qua câu lệnh xác định ngày end now().
Tiếp tục sử dụng globals để cho phép thực hiện sửa đổi bên ngoài dataframe ban đầu

```
#Chuyển từ list sang globals
empt_lists_new = empt_lists
empt_lists_new = globals()
bank_stocks = pd.concat(empt_lists_new, axis = 1, keys = empt_lists)
bank_stocks.head()
```

	BID					CTG					...	SSB					VPB		
	Open	High	Low	Close	Volume	Open	High	Low	Close	Volume	...	Open	High	Low	Close	Volume	Open	High	Low
TradingDate																			
2017-01-03	10463.0	11046.0	10463.0	11046.0	4954000	10517.0	11104.0	10517.0	11104.0	1008370	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2017-01-04	11155.0	11447.0	11155.0	11265.0	4674470	11311.0	11448.0	11104.0	11311.0	1006570	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2017-01-05	11192.0	11447.0	11082.0	11228.0	1988580	11242.0	11379.0	11035.0	11379.0	892010	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2017-01-06	11082.0	11702.0	11082.0	11483.0	6338280	11104.0	11793.0	11104.0	11483.0	1151370	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2017-01-09	11520.0	11957.0	11520.0	11811.0	6160700	11724.0	11931.0	11586.0	11793.0	1335810	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 85 columns

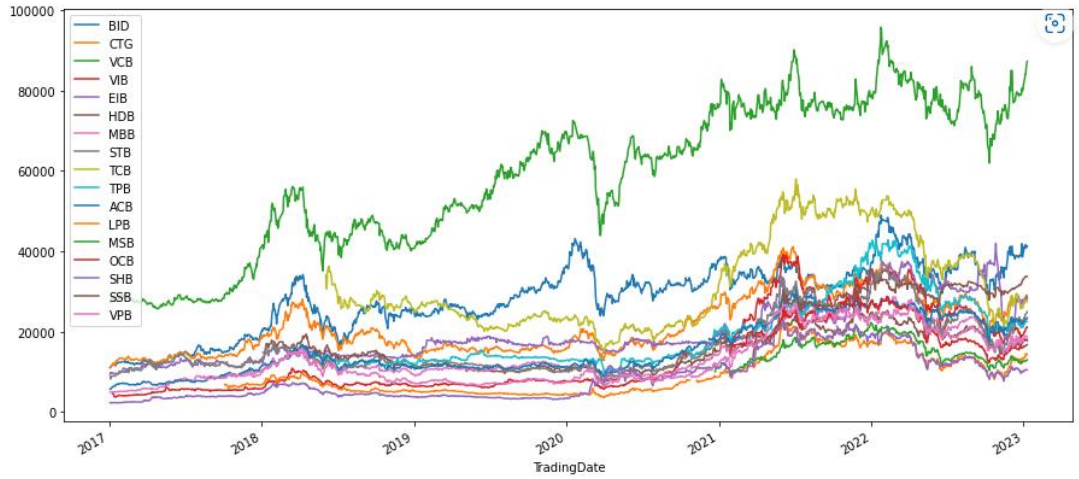
Mỗi mã chứng khoán khi giao dịch đều sẽ có OHLCV tương ứng với Open, High, Low, Close, Volume

```
value_banks = pd.DataFrame()
for name in empt_lists:
    value_banks[name] = bank_stocks[name]['Close']
value_banks.dropna(inplace= True)

value_banks.head()
```

	BID	CTG	VCB	VIB	EIB	HDB	MBB	STB	TCB	TPB	ACB	LPB	MSB	OCB	SHB	SSB	VP
TradingDate																	
2021-03-24	33357.0	29495.0	73317.0	23174.0	18500.0	16320.0	17035.0	18400.0	39300.0	19962.0	20768.0	11233.0	13017.0	19080.0	11509.0	13588.0	16
2021-03-25	33437.0	29798.0	73626.0	22803.0	19000.0	16384.0	17005.0	18750.0	39350.0	19925.0	20512.0	11198.0	12869.0	18840.0	11381.0	14532.0	16
2021-03-26	33318.0	29533.0	73472.0	22883.0	19000.0	16480.0	16974.0	18950.0	39400.0	19777.0	20768.0	11058.0	12869.0	18840.0	12468.0	15543.0	16
2021-03-29	33674.0	30251.0	73781.0	23253.0	18900.0	16640.0	17437.0	19200.0	39550.0	20295.0	21120.0	11514.0	12810.0	19160.0	13683.0	16622.0	16
2021-03-30	33911.0	30554.0	73394.0	24549.0	19500.0	16768.0	17467.0	20500.0	40050.0	20591.0	21312.0	12146.0	12810.0	19280.0	15026.0	17769.0	16

Tạo 1 variable chỉ chứa giá Close (giá đóng cửa) của 17 mã trên để thực hiện phân tích

```
#Lấy giá cổ phiếu theo tuần
value_banks = value_banks.resample('W').last()
for tick in empt_lists:
    bank_stocks[tick]['Close'].plot(figsize=(15,7),label=tick)
plt.legend();
```



Tạo 1 value_bank để chứa giá cổ phiếu theo tuần bằng hàm resample('W'). Sau đó biểu diễn bằng đồ thị line.

Ở đây, giá cổ phiếu của VCB là cao nhất.

```
#Calculate rate of return (% change over time)
stoc_returns = value_banks.pct_change().fillna(method='bfill')
#pct_change(): (current row value - previous row value)/ (previous row value).
stoc_returns.head()
```

TradingDate	BID	CTG	VCB	VIB	EIB	HDB	MBB	STB	TCB	TPB	ACB	LPB	MSB	OCB
2021-03-28	0.068882	0.057597	0.030556	0.172224	0.073684	0.058252	0.076352	0.192612	0.046954	0.059918	0.067797	0.117472	0.039086	0.048832
2021-04-04	0.068882	0.057597	0.030556	0.172224	0.073684	0.058252	0.076352	0.192612	0.046954	0.059918	0.067797	0.117472	0.039086	0.048832
2021-04-11	-0.015556	0.029071	-0.003064	0.000000	0.100490	0.027523	0.059113	0.004425	0.008485	0.017651	-0.007215	0.031237	-0.024379	0.000000
2021-04-18	-0.051912	-0.011760	-0.012320	0.005928	0.167038	-0.048214	-0.036693	-0.026432	-0.027644	-0.038206	-0.037791	0.044024	0.004522	-0.028340
2021-04-25	0.000000	-0.016686	0.079968	0.045102	0.028626	0.009381	0.003326	0.015837	0.001236	0.010869	0.009063	0.060734	-0.011217	-0.041667

Tiếp theo chúng ta tính tỷ lệ % thay đổi của giá Close bằng hàm pct_change() - so sánh mọi phần tử với phần tử trước của nó. Và thực hiện fillna() theo phương pháp method='bfill' - sử dụng giá trị bên dưới giá trị bị thiếu để bổ sung. Và chúng ta cùng xem 5 row đầu tiên sau khi tính toán. Vì ngày đầu tiên sẽ không có %thay đổi về giá nên nó được bổ sung từ ngày thứ 2 - tức là ngày sau nó bổ sung.

2.2. Clustering:

Đầu tiên chúng ta sử dụng phương pháp elbow method để tìm số cụm hợp lí - xác định giá trị tối ưu của k sẽ là bao nhiêu. Trước khi thực hiện elbow method thì chúng ta phải kiểm tra xem bộ dữ liệu có giá trị null nào hay không và bắt buộc các variable trong dataframe phải là giá trị int/float không phải là string hay object. Thông qua các câu lệnh dưới đây, chúng ta xác định biến stoc_returns không có giá trị nào bị trống và kiểm tra dtypes cũng đều thỏa.

```
stoc_returns.isnull().sum()
```

```
BID    0
CTG    0
VCB    0
VIB    0
EIB    0
HDB    0
MBB    0
STB    0
TCB    0
TPB    0
ACB    0
LPB    0
MSB    0
OCB    0
SHB    0
SSB    0
VPB    0
dtype: int64
```

```
np.any(np.isnan(value_banks))
np.all(np.isfinite(value_banks))
```

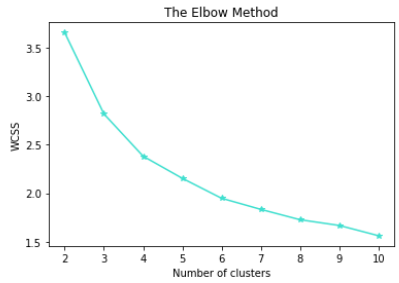
```
False
```

```
stoc_returns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 95 entries, 2021-03-28 to 2023-01-15
Freq: W-SUN
Data columns (total 17 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   BID     95 non-null     float64
 1   CTG     95 non-null     float64
 2   VCB     95 non-null     float64
 3   VIB     95 non-null     float64
 4   EIB     95 non-null     float64
 5   HDB     95 non-null     float64
 6   MBB     95 non-null     float64
 7   STB     95 non-null     float64
 8   TCB     95 non-null     float64
 9   TPB     95 non-null     float64
 10  ACB     95 non-null     float64
 11  LPB     95 non-null     float64
 12  MSB     95 non-null     float64
 13  OCB     95 non-null     float64
 14  SHB     95 non-null     float64
 15  SSB     95 non-null     float64
 16  VPB     95 non-null     float64
dtypes: float64(17)
memory usage: 13.4 KB
```

Tiếp theo là thực hiện phương pháp elbow method. Nhìn chart ta thấy điểm khuỷu tay sẽ là 3 tức bộ dữ liệu phân chia với mức k = 3 sẽ hợp lý nhất.

```
# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
K = range(2, 11)
for i in K:
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 5)
    kmeans.fit(stoc_returns)
    wcss.append(kmeans.inertia_)

plt.plot(K, wcss,color = "turquoise",marker = '*')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



Thực hiện phương pháp phân cụm dữ liệu với số cụm là 3 ta được kết quả sau:

```
def get_name(symbol):
    return df['company_short_name'][df['ticker']==symbol].values

def plot_stock(symbol, stocks=value_banks):
    stocks[symbol].plot(title=symbol, label=symbol, alpha=0.9);
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(stoc_returns.T);

clusters = {}
for l in np.unique(kmeans.labels_):
    clusters[l] = []

for i,l in enumerate(kmeans.predict(stoc_returns.T)):
    clusters[l].append(stoc_returns.columns[i])

for c in sorted(clusters):
    print('Cluster ' + str(c) + ': ', end='')
    for symbol in clusters[c]:
        print(get_name(symbol), '(' + symbol + ')', end='; ')
    print()
    print()
```

```
Cluster 0: ['BIDV'] (BID); ['VietinBank'] (CTG); ['Vietcombank'] (VCB); ['VIBBank'] (VIB); ['HDBank'] (HDB); ['MBBank'] (MBB);
['Sacombank'] (STB); ['Techcombank'] (TCB); ['Ngân hàng Tiên Phong'] (TPB); ['Ngân hàng Á Châu'] (ACB); ['LienViet Post Bank']
(LPB); ['MSB Bank'] (MSB); ['Ngân hàng Phương Đông'] (OCB); ['VPBank'] (VPB);

Cluster 1: ['SHB'] (SHB);

Cluster 2: ['Eximbank'] (EIB); ['SeABank'] (SSB);
```

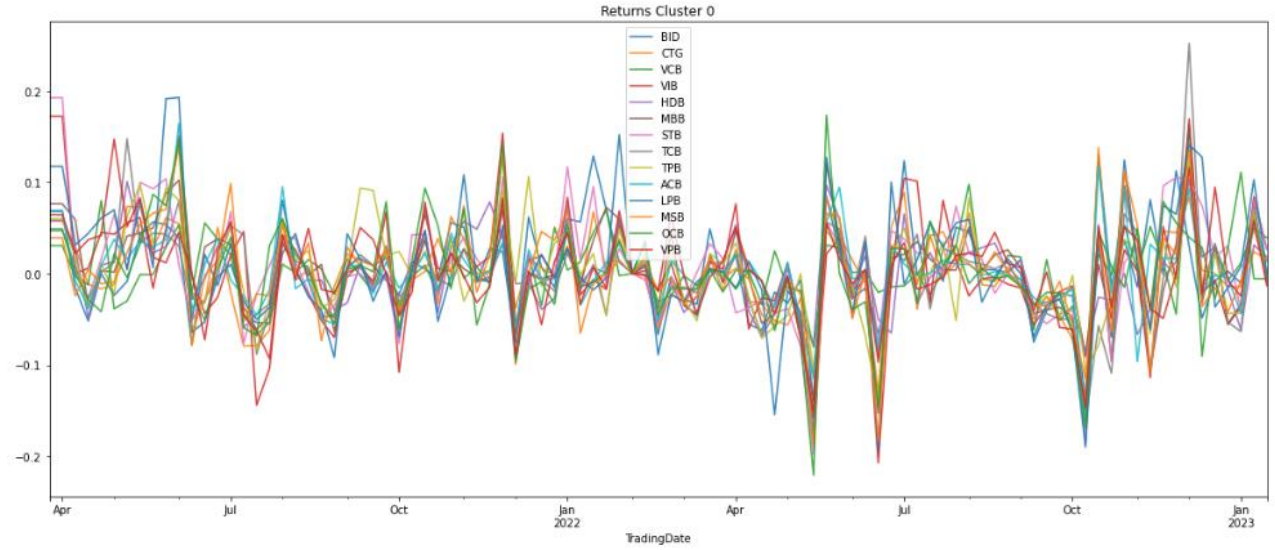
Cụm 1 bao gồm: ['BIDV'] (BID); ['VietinBank'] (CTG); ['Vietcombank'] (VCB); ['VIBBank'] (VIB); ['HDBank'] (HDB); ['MBBank'] (MBB); ['Sacombank'] (STB); ['Techcombank'] (TCB); ['Ngân hàng Tiên Phong'] (TPB); ['Ngân hàng Á Châu'] (ACB); ['LienViet Post Bank'] (LPB); ['MSB Bank'] (MSB); ['Ngân hàng Phương Đông'] (OCB); ['VPBank'] (VPB)

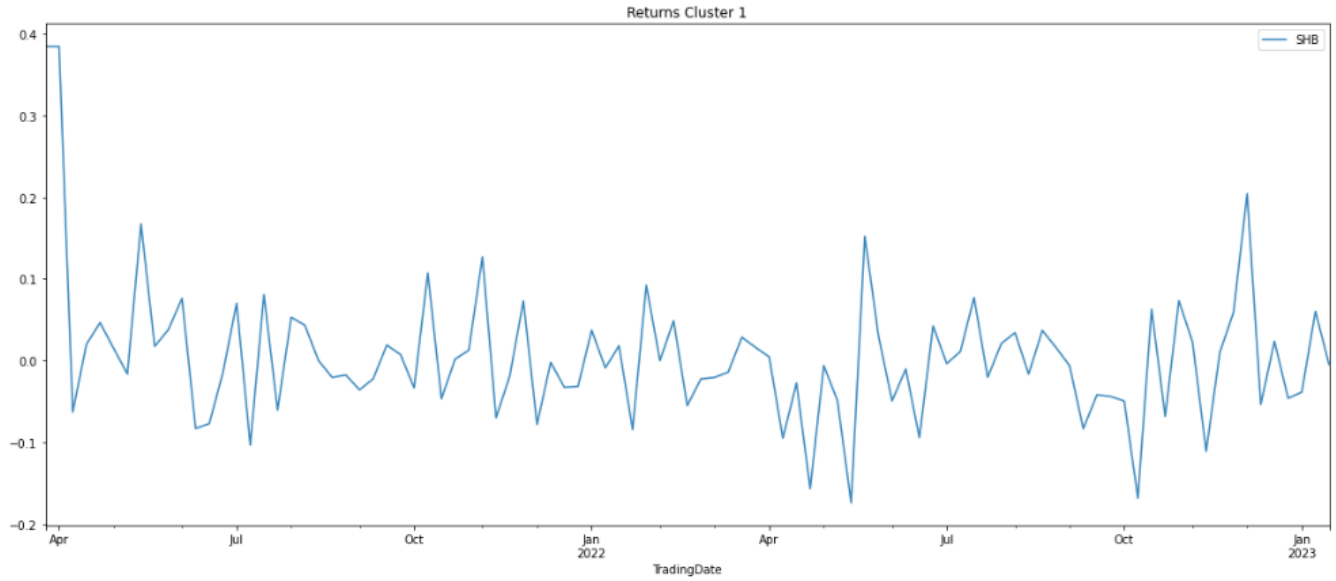
Cụm 2 bao gồm: ['SHB'] (SHB)

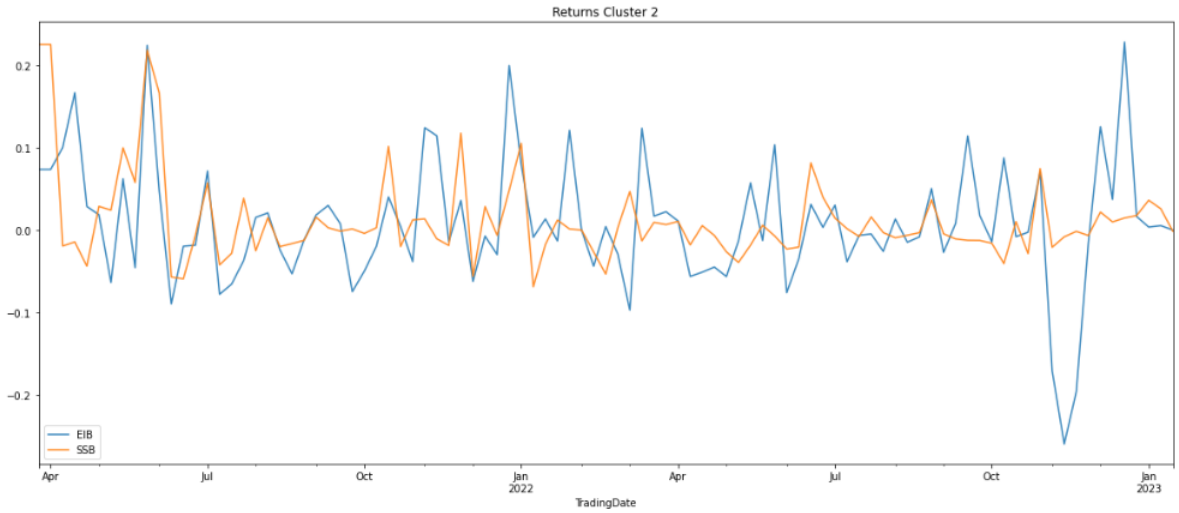
Cụm 3 gồm: ['Eximbank'] (EIB); ['SeABank'] (SSB)

Sau đó, chúng ta bắt đầu visualization theo từng cụm đã chia

```
for c in sorted(clusters):
    plt.figure(figsize = (20,8))
    for symbol in clusters[c]:
        plot_stock(symbol, stocks=stoc_returns)
    plt.title('Returns Cluster ' + str(c))
    plt.legend()
    plt.show()
```







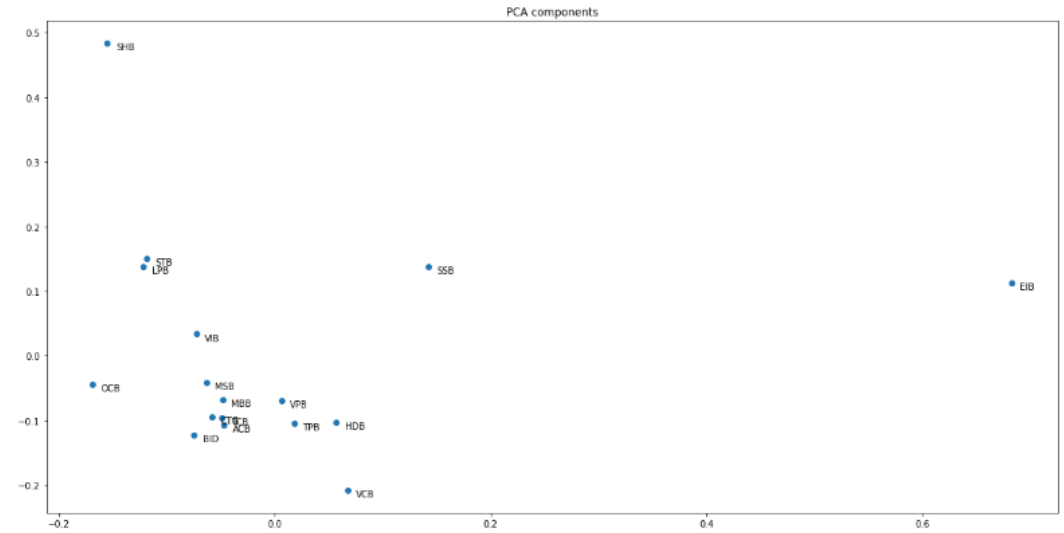
2.3.Principal Components Analysis (PCA) và Clustering:

Biểu diễn phân cụm để thấy rõ hơn

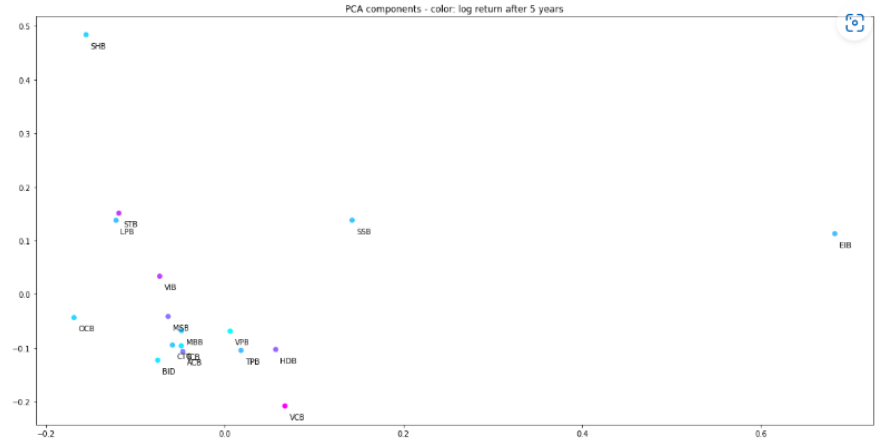
```
pca = PCA(n_components=.5, random_state=10)
components = pca.fit_transform(stoc_returns.T)
print('Number of PCA components', components.shape[1])
print('PCA explained variance ratio:', pca.explained_variance_ratio_)
print('PCA total explained variance: ', np.sum(pca.explained_variance_ratio_))
```

```
Number of PCA components 3
PCA explained variance ratio: [0.2496603  0.17912791 0.10410894]
PCA total explained variance:  0.5328971457955183
```

```
plt.figure(figsize=(20,10))
plt.scatter(components[:,0], components[:,1])
for i in range(components.shape[0]):
    plt.text(x=components[i,0]+0.008, y=components[i,1]-0.01, s=stoc_returns.columns[i])
plt.title('PCA components');
```



```
plt.figure(figsize=(20,10))
plt.scatter(components[:,0], components[:,1],  c = stoc_returns.iloc[-1].T.apply(np.log1p), cmap='cool')
for i in range(components.shape[0]):
    plt.text(x=components[i,0]+0.005, y=components[i,1]-0.025, s=stoc_returns.columns[i])
plt.title('PCA components - color: log return after 5 years');
```




```
print('Number of dimensions for clustering with PCA: ', len(components.T))
```

```
Number of dimensions for clustering with PCA:  3
```

Sau khi chạy thực tế, chúng ta xác định và thử sai xem có bao nhiêu thành phần (components) là hợp lý và mang kết quả tốt nhất. Components là 3 sẽ mang lại hiệu quả nhất.

```
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(components);

clusters = {}
for l in np.unique(kmeans.labels_):
    clusters[l] = []

for i,l in enumerate(kmeans.predict(components)):
    clusters[l].append(value_banks.columns[i])

for c in clusters:
    print('Cluster ' + str(c) + ': ', end='')
    for symbol in clusters[c]:
        print(get_name(symbol) + ' (' + symbol + ')', end='; ')
    print()
    print()
```

```
Cluster 0: ['Sacombank (STB)']; ['LienViet Post Bank (LPB)']; ['SHB (SHB)']; ['SeABank (SSB)'];

Cluster 1: ['BIDV (BID)']; ['VietinBank (CTG)']; ['Vietcombank (VCB)']; ['VIBBank (VIB)']; ['HDBank
(HDB)']; ['MBBank (MBB)']; ['Techcombank (TCB)']; ['Ngân hàng Tiên Phong (TPB)']; ['Ngân hàng Á Châu
(ACB)']; ['MSB Bank (MSB)']; ['Ngân hàng Phương Đông (OCB)']; ['VPBank (VPB)'];

Cluster 2: ['Eximbank (EIB)'];
```

Sau khi xác định component = 3 thì chúng ta PCA dữ liệu và nhận về kết quả sau và khác với kết quả clustering.

Có sự khác biệt giữa PCA và K-means (Clustering) là do PCA được sử dụng để giảm kích thước/lựa chọn tính năng/học đại diện, ví dụ: khi không gian tính năng chứa quá nhiều tính năng không liên quan hoặc dư thừa. Mục đích là để tìm ra chiều nội tại của dữ liệu.

Về mặt lý thuyết, phân tích thứ nguyên PCA (việc giữ lại thứ nguyên K đầu tiên cho biết 90% phương sai...không cần phải có mối quan hệ trực tiếp với cụm K Mean), tuy nhiên, giá trị của việc sử dụng PCA đến từ

-Bản chất của các đối tượng phân tích có xu hướng tập hợp xung quanh/phát triển một cách tự nhiên từ (một phân khúc nhất định).
-PCA loại bỏ các thứ nguyên có phương sai thấp (noise value), do đó, bản thân nó tăng thêm giá trị (và hình thành cảm giác tương tự như phân cụm) bằng cách tập trung vào các chiều chính đó
Các cụm mới bao gồm:
Cụm 1: ['Sacombank (STB)']; ['LienViet Post Bank (LPB)']; ['SHB (SHB)']; ['SeABank (SSB)'];
Cụm 2: ['BIDV (BID)']; ['VietinBank (CTG)']; ['Vietcombank (VCB)']; ['VIBBank (VIB)']; ['HDBank (HDB)']; ['MBBank (MBB)']; ['Techcombank (TCB)']; ['Ngân hàng Tiên Phong (TPB)']; ['Ngân hàng Á Châu (ACB)']; ['MSB Bank (MSB)']; ['Ngân hàng Phương Đông (OCB)']; ['VPBank (VPB)'];
Cụm 3: ['Eximbank (EIB)'];
Sau khi giảm chiều dữ liệu thì có sự phân chia lại về số cụm do PCA cải thiện độ nhiễu (noise value)

2.4.Gaussian Mixture model:

Đầu tiên ta lấy các mã chứng khoán có trong nhóm vn index. Ta đếm được có 417 công ty trong VNindex

```
#Lấy các mã cổ phiếu của VNINDEX group
lst_vnindex = df['ticker'][df['group_code']=='VNINDEX']
lst_vnindex.unique()
```

```
array(['FUCTVGF3', 'FUEIP100', 'GMH', 'FUEKIV30', 'NO1', 'FUCTVGF4',
       'FUEDCMID', 'FUEKIVFS', 'AAA', 'AAM', 'AAT', 'ABR', 'ABS', 'ABT',
       'ACB', 'ACC', 'ACL', 'AGM', 'AGR', 'AGG', 'AMD', 'ACG', 'ANV',
       'APC', 'APG', 'APH', 'HII', 'ASG', 'ASM', 'ASP', 'BAF', 'BBC',
       'BCE', 'BCG', 'BFC', 'BHN', 'BIC', 'BID', 'BCM', 'DBD', 'BWE',
       'BKG', 'BMC', 'BMI', 'BMP', 'BRC', 'BSI', 'BTP', 'BTT', 'BVH',
       'TNH', 'C32', 'C47', 'CAV', 'CCI', 'CCL', 'CDC', 'CRE', 'STK',
       'CHP', 'CIG', 'CII', 'CKG', 'CLC', 'ADG', 'CLL', 'CLW', 'CMG',
       'CMV', 'CMX', 'CNG', 'COM', 'CRC', 'CSM', 'CSV', 'CTD', 'CTF',
       'CTG', 'CTI', 'ICT', 'CTR', 'CTS', 'CVT', 'D2D', 'DAG', 'DAH',
       'ADS', 'DPG', 'DBC', 'DBT', 'DC4', 'DCL', 'DCM', 'DGC', 'DGW',
       'DHA', 'DHC', 'DHG', 'DHM', 'TTE', 'DIG', 'DLG', 'DMC', 'DPM',
       'DPR', 'DQC', 'DRC', 'DRH', 'DRL', 'DSN', 'DTA', 'DTL', 'DTT',
       'DVP', 'DXG', 'DXS', 'DXV', 'FUESSV50', 'E1VFVN30', 'EIB', 'ELC',
       'EMC', 'FUESSVFL', 'EVE', 'EVG', 'EVF', 'LEC', 'FCM', 'FCN', 'FDC',
```

```
len(lst_vnindex)
```

```
417
```

Bỏ qua các bước xử lý dữ liệu (Xem đầy đủ tại script code).
Sau đó chúng ta sử dụng GMM giảm chiều của các columns

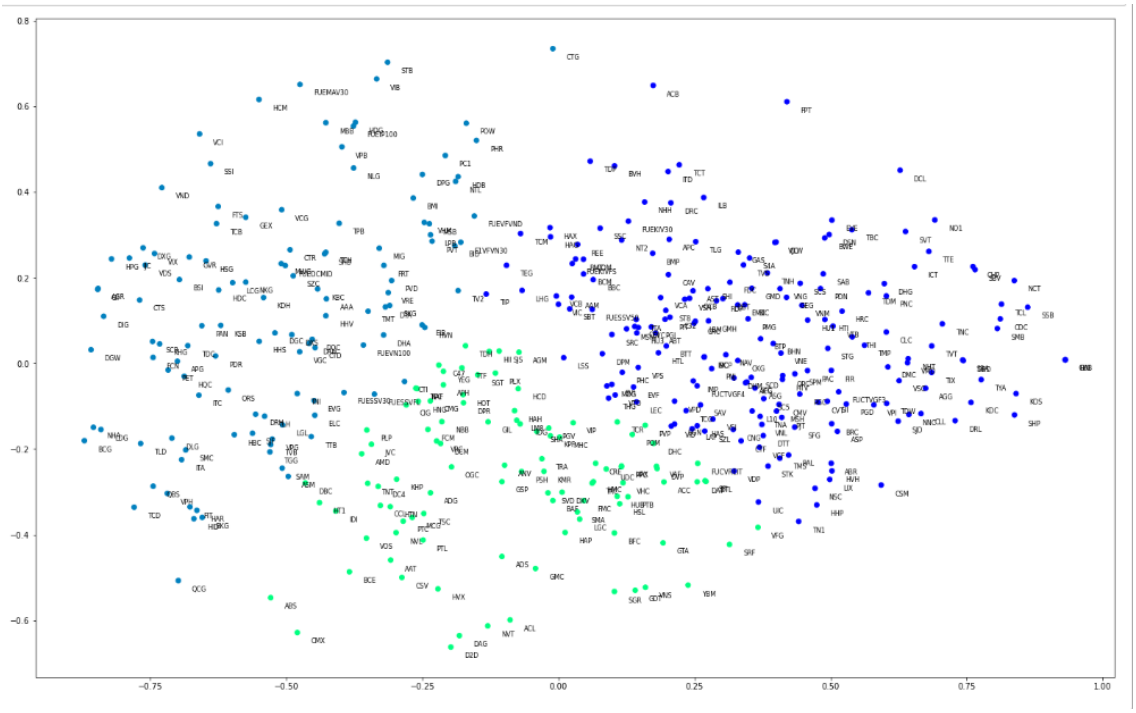
```
scaler = StandardScaler();
scaled_df = scaler.fit_transform(vnindex_returns_b.T);
normalized_df = normalize(scaled_df);

# Converting the numpy array into a pandas DataFrame
normalized_df = pd.DataFrame(normalized_df)

# Reducing the dimensions of the data
pca = PCA(n_components = 2)
X_principal = pca.fit_transform(normalized_df)
X_principal = pd.DataFrame(X_principal)
X_principal.columns = ['P1', 'P2']

X_principal.head(2)
```

	P1	P2
0	0.514732	-0.066347
1	-0.376768	0.554004

```
gmm = GaussianMixture(n_components = 3)
gmm.fit(X_principal)
```

```
GaussianMixture(n_components=3)
```

Tạo vòng lặp thể hiện từng mã cổ phiếu và biểu diễn chúng trên cùng 1 đồ thị



Do ban đầu ta lấy số lượng components = 3, mô hình cho ta thấy biến động về giá của các mã chứng khoán sẽ được phân chia theo 3 nhóm trên. Chiến lược đầu tư có thể linh hoạt theo mức độ biến động giá này.

Đa dạng danh mục đầu tư tránh gom vào tất cả cổ phiếu thuộc cùng 1 nhóm trên đây để tránh sụt giảm về vốn.

Gợi mở: Có thể phân cụm thành nhiều cụm khác nhau tương tự với nhóm cổ phiếu ngân hàng trên sàn HOSE bằng elbow method.

2.5. Phân kỳ Kullback–Leibler

Chúng ta sử dụng KullBack Leibler để tìm danh mục VNINDEX gần đúng
Ý tưởng: Coi dữ liệu là chứng khoán nhóm VNINDEX. Giả sử giá trị của mỗi cột là một phân phối xác suất theo KL, 2 phân phối xác suất gần như giống nhau khi giá trị của KL dần về 0. Vậy tính KL tương ứng cho mỗi cột, K cột có giá trị nhỏ nhất là K chứng minh . Chứng khoán có thể xấp xỉ VNINDEX

```
#Cổ phiếu #K gần bằng(xấp xỉ) VNINDEX
K=30
dict(sorted(dic_KL.items(), key = itemgetter(1))[:K])
```

```
{'FUEIP100': 0.17906069316954695,
 'E1VFVN30': 0.18512822953816213,
 'BCM': 0.1906439654529453,
 'FUEVFVND': 0.19070561186863075,
 'FUEMAV30': 0.19070736912904562,
 'SJD': 0.1953966179448809,
 'DRC': 0.19616631198428716,
 'KOS': 0.19669784720404215,
 'CDC': 0.19745820859391403,
 'RAL': 0.19802629639348296,
 'DPR': 0.19822943995087022,
 'SSB': 0.19877991687414145,
 'DMC': 0.20024153659150226,
 'HVH': 0.20263069657535218,
 'FUESSVFL': 0.2035999241505572,
 'FPT': 0.20401616496701913,
 'ASP': 0.20494238950744056,
 'HTI': 0.2056076249011129,
 'FIR': 0.2072284336046629,
 'REE': 0.20783949770446697,
 'MSB': 0.20874441088997664,
 'VJC': 0.2089918664587746,
 'FLC': 0.2092245556113489,
 'GAB': 0.2092245556113489,
 'HAI': 0.2092245556113489,
 'DRL': 0.20990916976718563,
 'CLL': 0.21119103374131926,
 'SMB': 0.21261294099223063,
 'FUEKIVFS': 0.21284809230289606,
 'HAX': 0.2146396039932018}
```

Sau khi chạy code ta được kết quả K ban đầu chúng ta chọn là 30 mã chứng khoán có biến động về giá gần với các mã chứng khoán trong VN index nhất.
Vì vậy, nếu chúng ta đầu tư danh mục theo độ biến động thì nhóm 30 mã cổ phiếu này sẽ có biến động giá tương tự nhau nhất - tức nghĩa gần nhau về biến động giá nhất (Tùy vào số lượng K ban đầu là 30 hay 3 hay 10) để đưa ra kết luận.

CHƯƠNG 4: PHẦN KẾT LUẬN

Dựa vào việc phân loại công ty dựa vào giá cổ phiếu giúp Nhà đầu tư đưa ra được một danh mục đầu tư có thể tối đa được vốn và bảo vệ nó trước những biến động về giá của các cổ phiếu.

Sử dụng PCA này giúp giảm chiều dữ liệu giúp trực quan hóa các nhóm công ty có cùng biến động về giá với nhau mà ở trước những mô hình khác không thể nào trực quan hóa được.

Tuy nhiên bài báo này vẫn còn 2 nhược điểm chính. Thứ nhất là chưa thực hiện kiểm định để biết phần dư có phương sai thay đổi điều kiện hay không? Nếu có thì khi có những cú sốc tác động đến thị trường chứng khoán (như tình hình thị trường tài chính thế giới thay đổi, chính sách tiền tệ, lãi suất của chính phủ thay đổi,…) phần dư của mô hình sẽ thay đổi đột ngột trong khi mô hình dự báo trung bình không nắm bắt được, dẫn đến hạn chế độ chính xác dự báo. Nhược điểm thứ 2 là: trong số các biến gốc có hệ số tương quan cao với biến đích được lựa chọn lần đầu để sau đó áp dụng kỹ thuật PCA có thể có một số biến có tương quan cao với nhau, khi đó xảy ra hiện tượng một số biến gốc có thể được xác định thông qua một số biến gốc khác. Điều này có nghĩa là có sự dư thừa các biến được lựa chọn lần đầu và có thể đã bỏ sót một số biến thích đáng khác cung cấp thông tin có ích cho dự báo biến đích mặc dù hệ số tương quan của nó với biến đích là không lớn lắm.

TRÍCH DẪN

https://www.researchgate.net/profile/Thanh-Do-Van/publication/332193485_MO_HINH_HOA_DU_BAO_GIA_CO_PHIEU_TRONG_NGU_CANH_DU_LIEU_SO_CHIEU_CAO/links/5cdc4026458515712ead7a54/MO-HINH-HOA-DU-BAO-GIA-CO-PHIEU-TRONG-NGU-CANH-DU-LIEU-SO-CHIEU-CAO.pdf

https://vi.wikipedia.org/wiki/M%C3%B4_h%C3%ACnh_Gaussian_h%E1%BB%97n_h%E1%BB%A3p

https://www.kaggle.com/code/demirkaya/clustering-companies-based-on-stock-price-movement

https://phamdinhkhanh.github.io/2019/01/07/Ky_thuat_feature_engineering.html

https://github.com/baolongnguyenmac/DataScienceProject/blob/main/predict1Day.ipynb

https://nganhangaz.com/cac-ngan-hang-da-niem-yet-tren-san-chung-khoan/

https://codelearn.io/sharing/scikit-learn-trong-python-la-gi

https://tapchinganhang.gov.vn/nghiep-vu-dau-tu-kinh-doanh-chung-khoan-va-phan-loai-chung-khoan-trong-qua-trinh-hach-toan-tai-cac-n.htm

https://vneconomy.vn/nganh-ngan-hang-giai-xong-3-bai-toan-kho-trong-nam-2022.htm

PHỤ CHÚ:

```
PCA phân cụm danh mục đầu tư
import pandas as pd
import numpy as np
import datetime as dt
import seaborn as sns
import matplotlib.pyplot as plt
from math import log2
from scipy.special import rel_entr
from operator import itemgetter
from vnstock import *

import cufflinks as cf
cf.go_offline()
import plotly.io as pio

from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler, normalize
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score
from sklearn import metrics


#Liệt kê các mã chứng khoán có trên thị trường
df = listing_companies()
df.head()
df.shape
#Tạo variable bao gồm các stock codes for analysis ở nhóm ngành Ngân hàng
a = ['BID', 'CTG', 'VCB', 'VIB', 'EIB', 'HDB', 'MBB', 'STB', 'TCB', 'TPB', 'ACB', 'LPB',
'MSB', 'OCB', 'SHB', 'SSB', 'VPB']
# Retrieve historical price data of any n stocks
start = "2017-01-01"
end  = dt.datetime.now().strftime("%Y-%m-%d")
empt_lists = []
for i in a:
   empt_lists.append(i)
   globals()[i]      =      stock_historical_data(symbol=i,      start_date=start,
end_date=end).set_index('TradingDate')
empt_lists

#Chuyển từ list sang globals
```

```
empt_lists_new = empt_lists
empt_lists_new = globals()
bank_stocks = pd.concat(empt_lists_new, axis = 1, keys = empt_lists)
bank_stocks.head()

value_banks = pd.DataFrame()
for name in empt_lists:
    value_banks[name] = bank_stocks[name]['Close']
value_banks.dropna(inplace= True)

value_banks.head()

#Lấy giá cổ phiếu theo tuần
value_banks = value_banks.resample('W').last()
for tick in empt_lists:
    bank_stocks[tick]['Close'].plot(figsize=(15,7),label=tick)
plt.legend();

#Calculate rate of return (% change over time)
stoc_returns = value_banks.pct_change().fillna(method='bfill')
#pct_change(): (current row value – previous row value)/ (previous row value).
stoc_returns.head()

np.any(np.isnan(value_banks))
np.all(np.isfinite(value_banks))

# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
K = range(2, 11)
for i in K:
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 5)
    kmeans.fit(stoc_returns)
    wcss.append(kmeans.inertia_)

plt.plot(K, wcss,color = "turquoise",marker = '*')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

Clustering

In [16]:

```
def get_name(symbol):
    return df['company_short_name'][df['ticker']==symbol].values

def plot_stock(symbol, stocks=value_banks):
    stocks[symbol].plot(title=symbol, label=symbol, alpha=0.9);
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(stoc_returns.T);

clusters = {}
for l in np.unique(kmeans.labels_):
    clusters[l] = []

for i,l in enumerate(kmeans.predict(stoc_returns.T)):
    clusters[l].append(stoc_returns.columns[i])

for c in sorted(clusters):
    print('Cluster ' + str(c) + ': ', end='')
    for symbol in clusters[c]:
        print(get_name(symbol), '(' + symbol + ')', end='; ')
    print()
    print()
for c in sorted(clusters):
     plt.figure(figsize = (20,8))
     for symbol in clusters[c]:
         plot_stock(symbol, stocks=stoc_returns)
     plt.title('Returns Cluster ' + str(c))
     plt.legend()
     plt.show()
```

Principal Components Analysis and Clustering

In [18]:

```
pca = PCA(n_components=.5, random_state=10)
components = pca.fit_transform(stoc_returns.T)
print('Number of PCA components', components.shape[1])
print('PCA explained variance ratio:', pca.explained_variance_ratio_)
print('PCA total explained variance: ', np.sum(pca.explained_variance_ratio_))
                                                                        :
plt.figure(figsize=(20,10))
plt.scatter(components[:,0], components[:,1])
for i in range(components.shape[0]):
    plt.text(x=components[i,0]+0.008,                    y=components[i,1]-0.01,
s=stoc_returns.columns[i])
plt.title('PCA components');
```

```
plt.figure(figsize=(20,10))
plt.scatter(components[:,0],  components[:,1],     c  =  stoc_returns.iloc[-
1].T.apply(np.log1p), cmap='cool')
for i in range(components.shape[0]):
    plt.text(x=components[i,0]+0.005,                y=components[i,1]-0.025,
s=stoc_returns.columns[i])
plt.title('PCA components - color: log return after 5 years');

print('Number of dimensions for clustering with PCA: ', len(components.T))

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(components);

clusters = {}
for l in np.unique(kmeans.labels_):
    clusters[l] = []

for i,l in enumerate(kmeans.predict(components)):
    clusters[l].append(value_banks.columns[i])

for c in clusters:
    print('Cluster ' + str(c) + ': ', end='')
    for symbol in clusters[c]:
        print(get_name(symbol) + ' (' + symbol + ')', end='; ')
    print()
    print()

for c in sorted(clusters):
    plt.figure(figsize = (20,8))
    for symbol in clusters[c]:
        plot_stock(symbol, stocks=stoc_returns)
    plt.title('Returns (clusters from PCA components) cluster ' + str(c))
    plt.legend()
plt.show()
```

#Lấy các mã cổ phiếu của VNINDEX group

```
lst_vnindex = df['ticker'][df['group_code']=='VNINDEX']
lst_vnindex.unique()


len(lst_vnindex)


start = "2017-01-01"
end  = dt.datetime.now().strftime("%Y-%m-%d")
empt_lists_b = []
```

```
for i in lst_vnindex:
    empt_lists_b.append(i)
    globals()[i]   =      stock_historical_data(symbol=i,   start_date=start,
end_date=end).set_index('TradingDate')


empt_lists_new_b = empt_lists_b
empt_lists_new_b = globals()
bank_stocks_b = pd.concat(empt_lists_new_b, axis = 1, keys = empt_lists_b)
bank_stocks_b.tail()


value_banks_b = pd.DataFrame()
for name in empt_lists_b:
    value_banks_b[name] = bank_stocks_b[name]['Close']
value_banks_b.dropna(inplace= True)

#Calculate profit margin
vnindex_returns_b = value_banks_b.pct_change().fillna(method='bfill')
vnindex_returns_b.head()


scaler = StandardScaler();
scaled_df = scaler.fit_transform(vnindex_returns_b.T);
normalized_df = normalize(scaled_df);

# Converting the numpy array into a pandas DataFrame
normalized_df = pd.DataFrame(normalized_df)

# Reducing the dimensions of the data
pca = PCA(n_components = 2)
X_principal = pca.fit_transform(normalized_df)
X_principal = pd.DataFrame(X_principal)
X_principal.columns = ['P1', 'P2']

X_principal.head(2)

gmm = GaussianMixture(n_components = 3)
gmm.fit(X_principal)


plt.figure(figsize=(24,15))
plt.scatter(X_principal['P1'], X_principal['P2'],
            c  =  GaussianMixture(n_components  =  3).fit_predict(X_principal),
alpha = 1, cmap='winter')
for i in range(X_principal.shape[0]):
    plt.text(x=X_principal['P1'][i]+0.025,      y=X_principal['P2'][i]-0.025,
s=vnindex_returns_b.columns[i], fontsize=8)
```

```
plt.show()
```

KullBack Leibler để tìm danh mục gần bằng với VNINDEX

In [47]:

```
#Chuyển về dưới dạng phân phối sác xuất ([0,1])
def softmax(Z):
   e_Z = np.exp(Z)
   A = e_Z / e_Z.sum(axis = 0)
   return A
```

In [48]:

```
vnindex_returns_b = softmax(vnindex_returns_b)
```

In [49]:

```
len(vnindex_returns_b)


list_sum, idx = [], []
for i in vnindex_returns_b.columns:
    tong = 0
    for j in vnindex_returns_b.columns:
        kl_pq        =        sum(rel_entr(list(vnindex_returns_b[i].values),
list(vnindex_returns_b[j].values)))
        tong += kl_pq
    idx.append(i)
    list_sum.append(tong)
dic_KL = dict(zip(idx, list_sum))


#Cổ phiếu #K gần bằng(xấp xỉ) VNINDEX
K=30
dict(sorted(dic_KL.items(), key = itemgetter(1))[:K])
```