

국내/해외주식 데이터를 활용한 숨은 투자 기회 찾기

금융 팀

17기 조성윤, 18기 박제재



목차

01

주제 설명

02

데이터 소개

03

전처리

04

유명산업&기업분석

05

감성분석

06

그래프 기반 분석

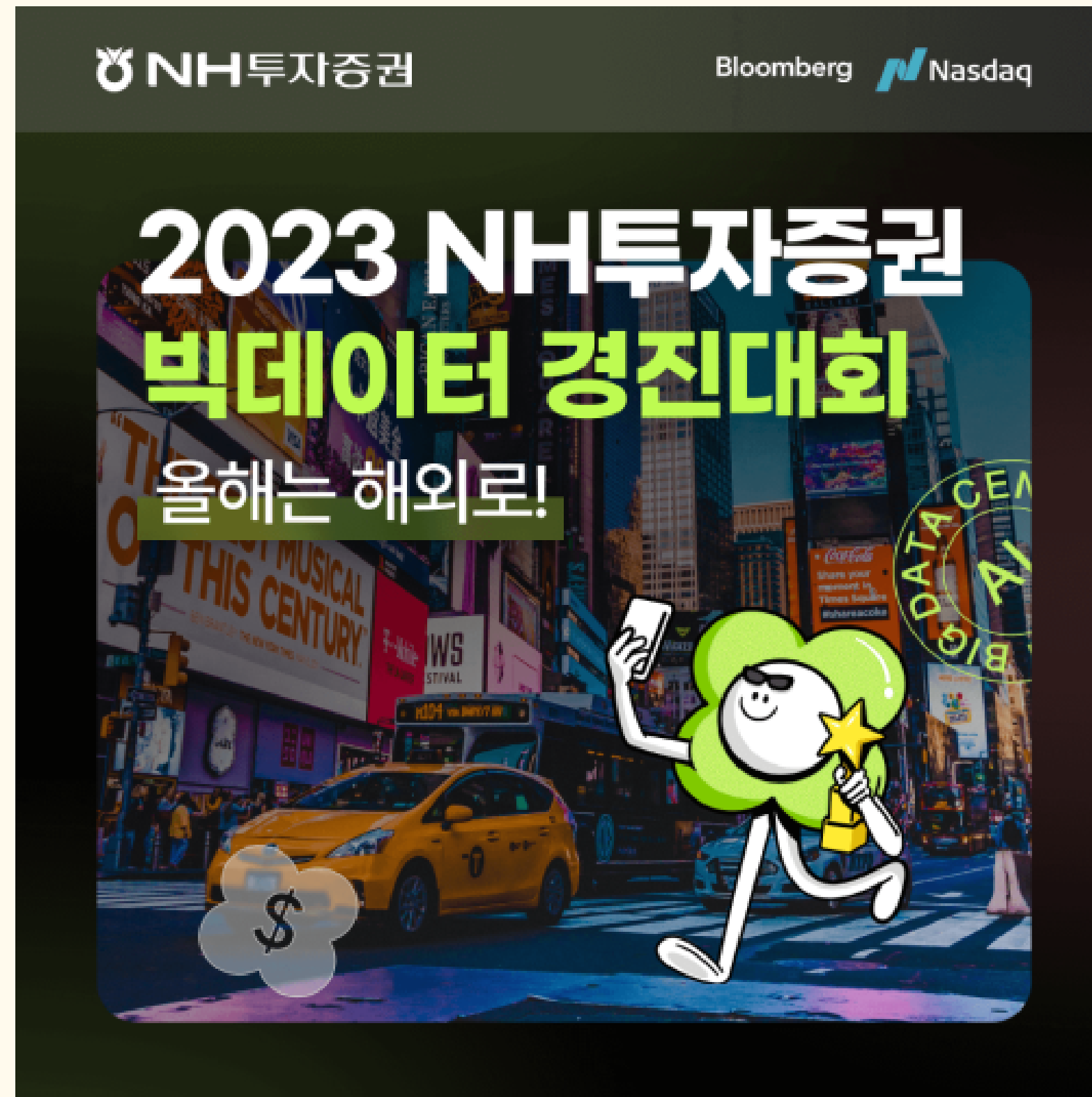
07

한계

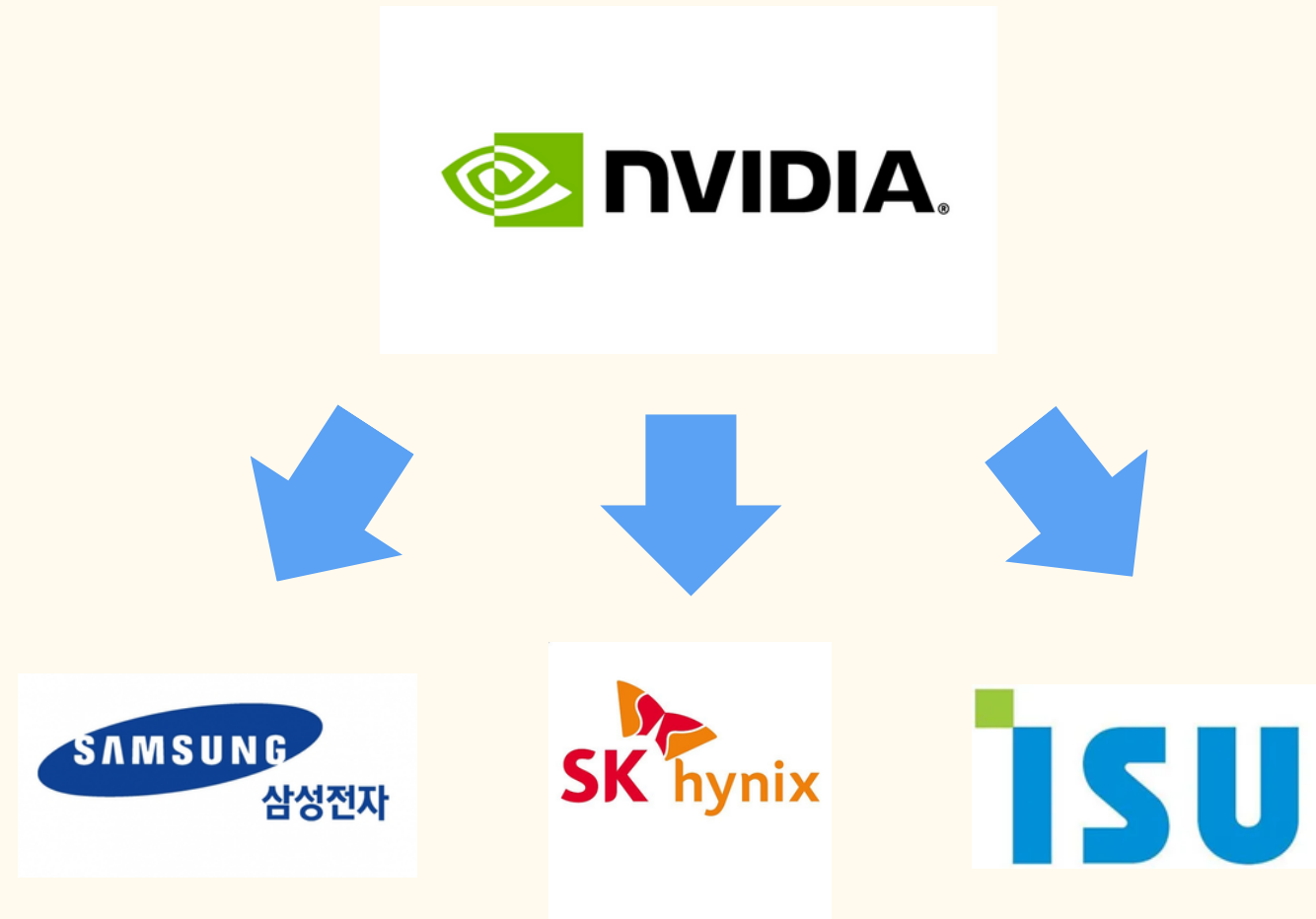


1. 주제 설명

소개



동향



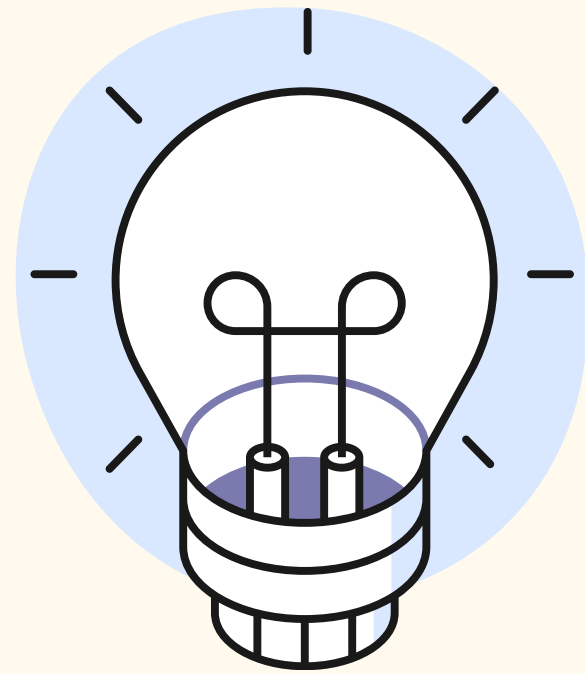
“동조화 현상“

미국 시장 → 국내 시장



“금융권 DT“

비정형 데이터 분석

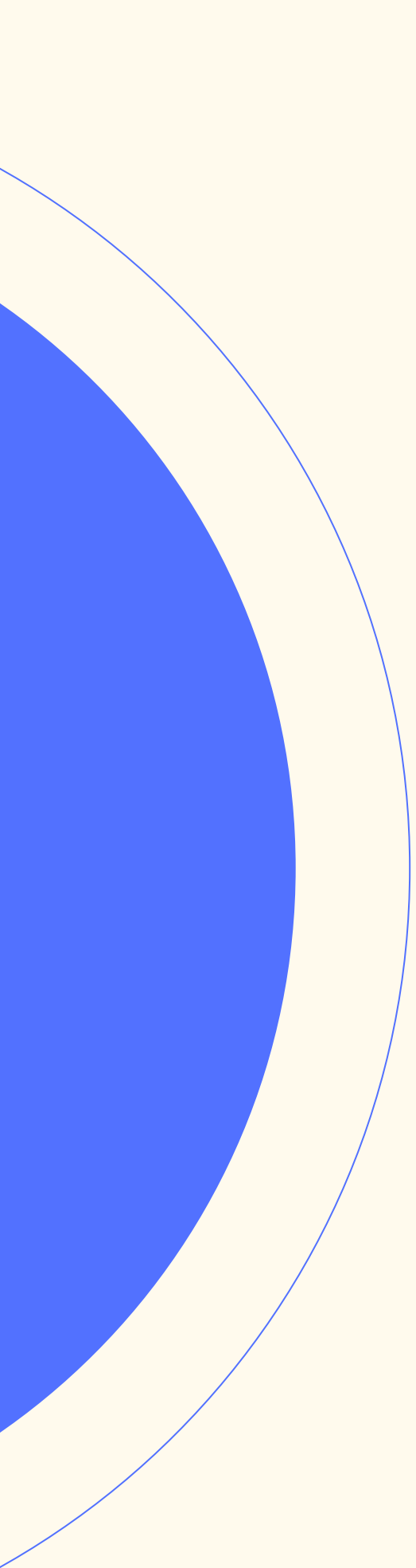


“데이터 속에 숨은 투자 기회”

1. 주식 시세 데이터를 활용한 **국내/해외 종목 관계 분석**

2. 뉴스 데이터를 이용한 **감성분석**

→ **해외 주식 기반 국내 주식 종목 추천 시스템**



2. 데이터 소개

데이터 소개 – NH 제공

NASDAQ_FC_STK_IEM_IFO

2023년 미국 나스닥 거래소에서 시세를 제공하는 주문 가능한 종목 정보

ISIN_IEM_CD : ISIN 코드

TCK_IEM_CD : 종목 티커 코드

FC_SEC_KRL_NM : 해외주식 종목 한글명

FC_SEC_ENG_NM : 해외주식 종목 영문명.

NASDAQ_DT_FC_STK_QUT

2023년 종목의 시세 정보

TRD_DT : 거래일자

TCK_IEM_CD : 티커종목코드

IEM_ONG_PR : 종목시가

IEM_HI_PR : 종목고가

IEM_LOW_PI : 종목저가

IEM_END_PR : 종목종가

ACL_TRD_ATY : 누적거래수량

SLL_CNS_SUM_QTY : 매도체결합계수량

BYN_CNS_SUM_QTY : 매수체결합계수량



NH투자증권

NASDAQ_RSS_IFO

RGS_DT : 발행일자

TCK_IEM_CD : TCK_IEM_CD

TIL_IFO : 제목정보

CTGY_CFC_IFO : 카테고리분류정보

MDI_IFO : 미디어정보

NEWS_SMY_IFO : 뉴스요약정보

RLD_OSE_IEM_TCK_CD : 관련해외종목티커코드

URL_IFO : URL정보

데이터 크롤링 - 뉴스 원문



	title	date	category	key_points	text	url
0	'I work just 5 hours a week': A 39-year-old wh...	2023-01-01	Success	N/A	Graham Cochrane, Founder of The Recording Revo...	https://www.cnbc.com/2023/01/01/39-year-old-wh...
1	Chinese state media seek to reassure public ov...	2023-01-01	Asia-Pacific News	Chinese state media sought to reassure the pub...	Revelers prepare to release balloons to celebr...	https://www.cnbc.com/2023/01/01/chinese-state-...
2	Should you get creative with your resume? Expe...	2023-01-01	Land the Job	N/A	Mature businessman congratulating young profes...	https://www.cnbc.com/2023/01/01/cv-will-a-crea...
3	Market misery deals sovereign wealth funds his...	2023-01-01	Markets	Heavy falls in stock and bond markets over the...	A trader works on the floor of the New York St...	https://www.cnbc.com/2023/01/01/market-misery-...
4	More social media regulation is coming in 2023...	2023-01-01	Tech	Days after Congress passed a bipartisan spendi...	The U.K.'s Online Safety Bill, which aims to r...	https://www.cnbc.com/2023/01/01/more-social-me...

NEWSPAPER LIBRARY 활용하여 기사본문 크롤링

+

selenium & bs4를 활용하여 미국 경제 뉴스 “CNBC” 크롤링

데이터 소개 - 외부 크롤링

국내 주식 시세 데이터 - KRX 정보데이터

<http://data.krx.co.kr/contents/MDC/MAIN/main/index.cmd>

	A098120	A009520	A095570	A006840	A282330	A027410	A138930
date							
2023-01-02	5120.0	7710	5720.0	16250	202000.0	4115.0	6300.0
2023-01-03	5160.0	7740	5790.0	16200	199000.0	4100.0	6340.0
2023-01-04	5480.0	7760	5760.0	16250	197500.0	4155.0	6550.0
2023-01-05	5570.0	7540	5760.0	16400	192000.0	4230.0	6720.0
2023-01-06	5670.0	7570	5720.0	16050	191500.0	4225.0	6790.0

NASDAQ 종합지수

```
date
2023-01-03    10386.99
2023-01-04    10458.76
2023-01-05    10305.24
2023-01-06    10569.29
2023-01-09    10635.65
..            ..
```

재무제표 데이터

yahoo!
finance

전자공시시스템
DART

나스닥 외화주식 종목 정보 : [NASDAQ.COM](https://www.nasdaq.com)

<https://www.nasdaq.com/market-activity/stocks/screener>



3. 전처리

주제 설명

데이터 소개

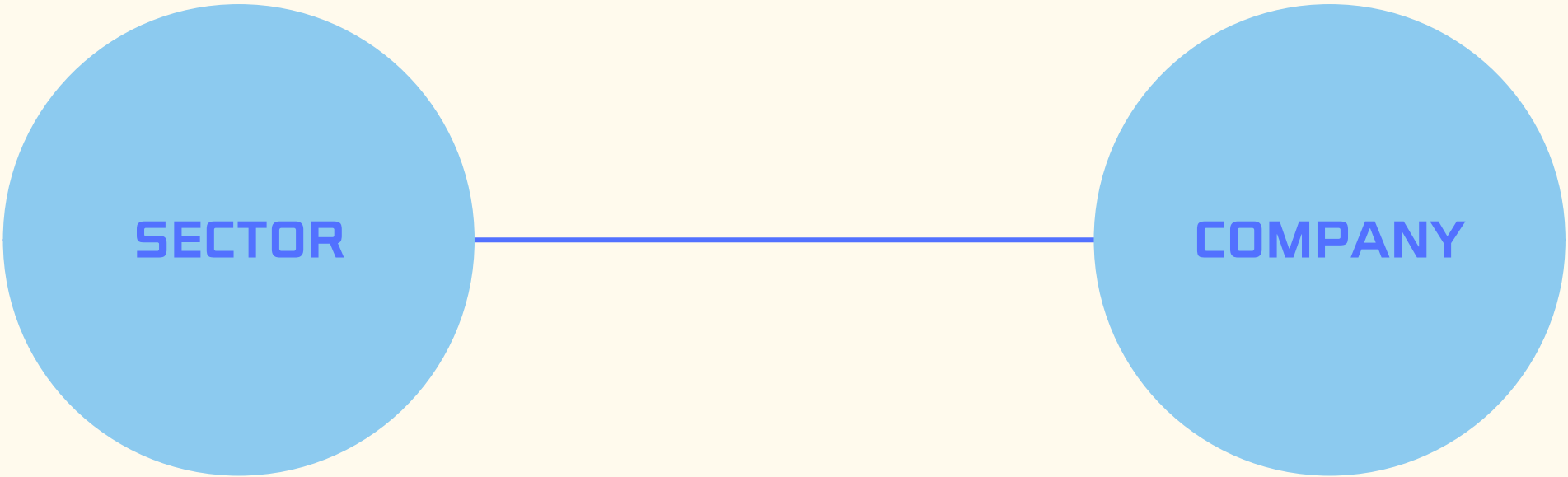
전처리



데이터 분석

한계

전처리 목표



HOT TOPIC 분야 찾기

관련 유망 기업 찾기

뉴스 데이터 자연어처리

CNBC 경제 뉴스 사이트에서 인기 있는 기사 토픽을 먼저 파악하고자 함

*월에 5번도 언급되지 않은 카테고리는 주가 분석에 있어
중요하지 않은 기사라고 판단해 삭제함.

```
# 월 별로 5번 이상 등장한 카테고리 찾기  
valid_categories = monthly_counts[monthly_counts['count'] >= 5]['category'].unique()
```

토큰화 및 불용어 제거

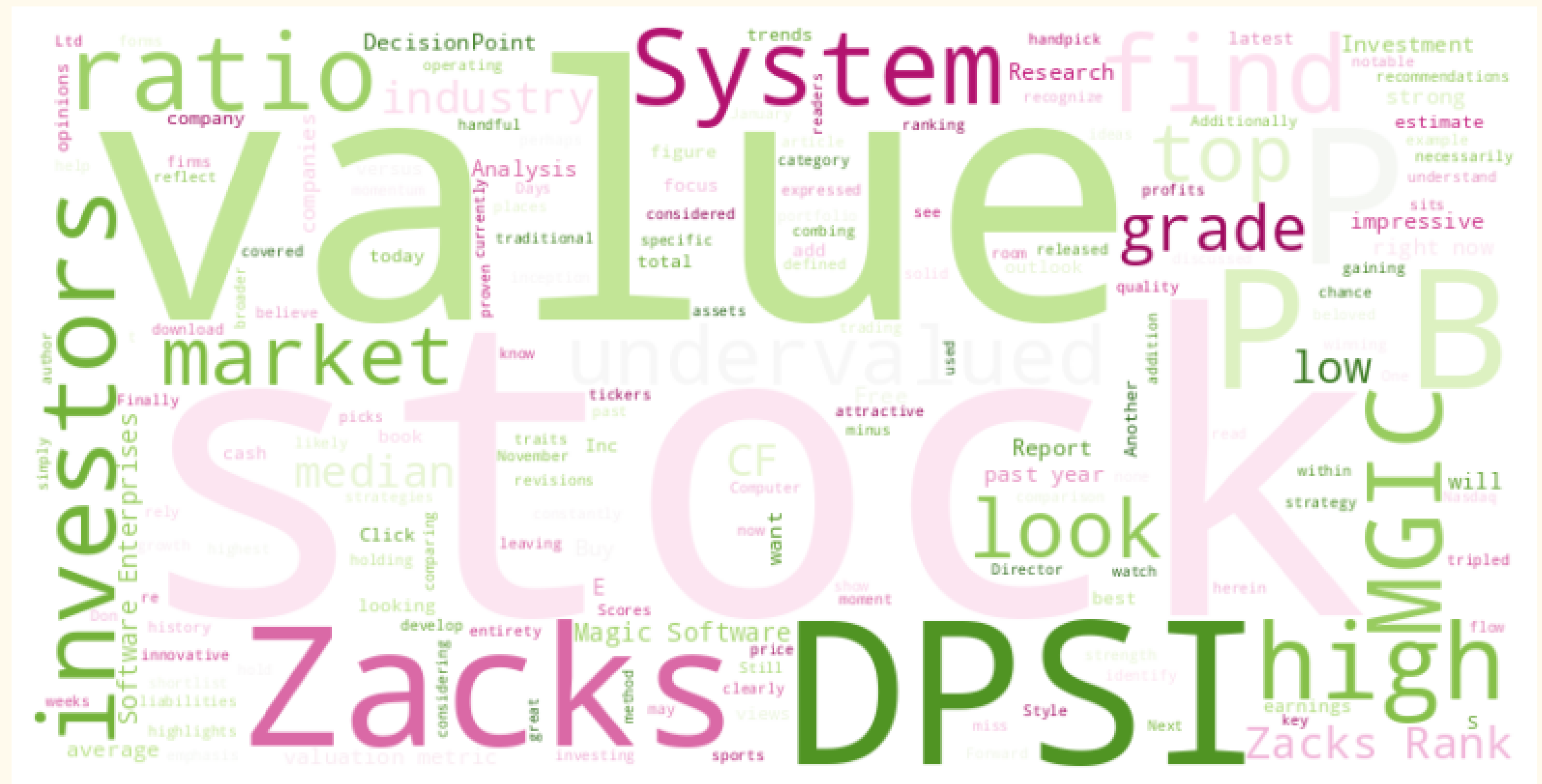
- 크롤링 데이터의 특수문자와 문장부호 제거
- 불용어 제거(n차 토픽모델링 후 후처리 진행)
- 불용어 제거 및 토큰화 : 성능 면에서 nltk보다 **spacy**가 우수
- 긴 텍스트(기사 본문)를 처리해야 하므로 en_core_web_lg 모델을 사용했다.
- 또한 tokenizer 함수를 생성할 때는 명사만 추출하도록 했으며, 개체명은 'TIME','CARDINAL','DATE'을 제외한 모든 entity를 사용했다.

```
token.ent_type_ not in ['TIME','CARDINAL','DATE']:
```

The logo for spaCy, a natural language processing library, is displayed in a light blue color.

워드클라우드 - 키워드 추출

워드 클라우드를 활용한 뉴스별 주요 단어 확인하기 : 감성분석을 적용하기 전 크롤링



주가 데이터 전처리

```
# 기간 중 거래정지 종목 제외
trd_stop = (all_krx_adj_open == 0).sum()
trd_stop_stocks = trd_stop[trd_stop>0].index.values
trd_stop_stocks_cnt = trd_stop_stocks.shape[0]

cond_stocks = np.setdiff1d(period_listed_stocks, trd_stop_stocks)
cond_stocks_cnt = cond_stocks.shape[0]

print(f'현재 상장 종목: {all_stocks_cnt}')
print(f'- 기간 중 상장 및 폐지 종목: {all_stocks_cnt - period_listed_stocks_cnt}')
print(f'- 기간 중 거래정지 종목: {trd_stop_stocks_cnt}')
print(f'제거율: {1 - cond_stocks_cnt/all_stocks_cnt:.2%}')
print()
print(f'분석 대상 종목: {cond_stocks_cnt}')
```

현재 상장 종목: 2754
- 기간 중 상장 및 폐지 종목: 103
- 기간 중 거래정지 종목: 289
제거율: 13.91%

분석 대상 종목: 2371

성과지표(Techinical Indicators)

주식의 수익률을 측정하기 위한 지표가 필요하여 직접 구현

$$Sharpe = \frac{\mu_p - r_f}{\sigma_p}$$

$$Sortino = \frac{\mu_p - r_f}{D\sigma_p}$$

$$Calmar = -\frac{\mu_p - r_f}{MDD_p}$$

$$VaRRatio = -\frac{\mu_p - r_f}{N * VaR_{\delta,p}}$$



4. 유망산업&기업 분석

토픽 모델링

```
# 모델 생성
```

```
topic_model = create_topic_model(embedding_model, umap_model, hdbscan_model, vectorizer_model, representation_model)
```

- 임베딩 모델 선정 기준 : sbert.net의 sentencetransformer 중 가장 Performance가 높은 모델(all-mpnet-base-v2) 선정
- 5배나 빠른 속도에 정확도가 높은 all-MiniLM-L6-v2로도 시도해봤지만 성능이 좋지 않았음.

```
embedding_model = SentenceTransformer("all-mpnet-base-v2") #임베딩 모델
```

```
embeddings = embedding_model.encode(all_texts, show_progress_bar=True) #임베딩 미리 계산(파라미터 수정 용이 위함)
```

- 파라미터 튜닝 결과 아래의 파라미터로 하는 것이 가장 토픽을 잘 찾는다고 판단함.
- UMAP(n_neighbors=8, min_dist=0.1, n_components=2)
- HDBSCAN(min_cluster_size=5)
- TfidfVectorizer 사용 이유 : 단어의 빈도 뿐만 아니라, 그 단어가 전체 문서 집합에서 얼마나 중요한지를 고려하기에 토픽모델링 시에 해당 모델을 사용하는 것이 적합함.
- MaximalMarginalRelevance 사용 이유 : 토픽의 키워드를 통해 관련주를 찾아내야 하므로, 토픽 키워드를 추출하는 것이 정교해야 한다고 판단함. 또한 diversity를 0.2로 설정해 토픽과 관련된 키워드를 다소 다양하게 뽑고자 했음.

CONTENTS

주제 설명

데이터 소개

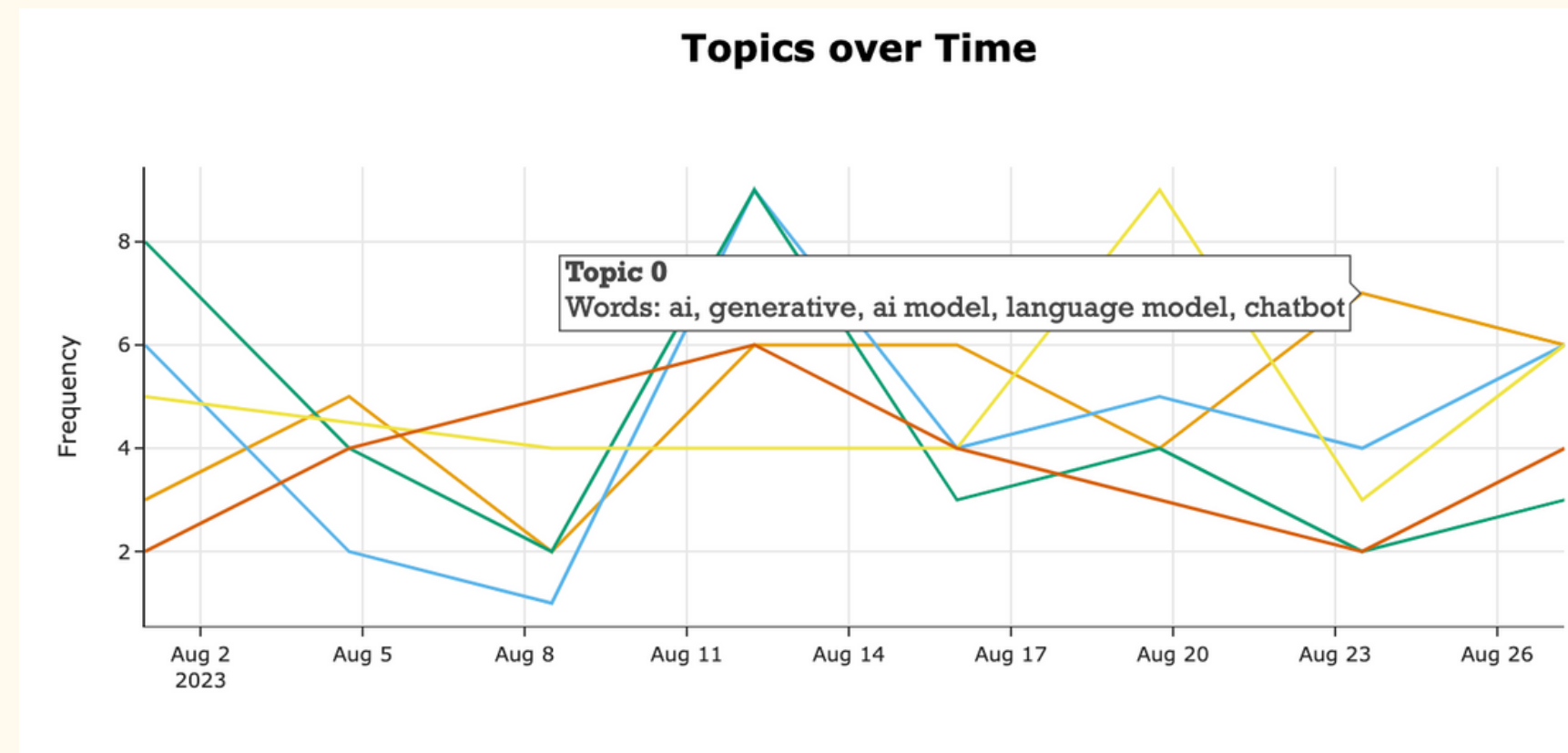
전처리

데이터 분석

한계

모델링 시각화 & 키워드 제시

ex) AI 테마 키워드



Representation

[bitcoin, resume, oracle, market, sec, etf, ap...]
[analyst refinitiv, analyst, cramer, price tar...]
[ai, chatgpt, google, ai model, openai, chatbo...]
[retailer, walmart, foot locker, merchandise, ...]
[pfizer, vaccine, pharmacy, medication, obesit...]
[cnn, disney, microsoft, activision, espn, sal...]
[election, president donald, indictment, case,...]
[iphone, apple, ipad, smartphone, apple iphone...]
[playlist, schwartz, taylor, feedback, billion...]
[china, beijing, chinas, economist, yuan, peop...]
[twitter, musk, app, meta, elon musk, fda, bot...]
[happiness, gate, harvard, brain, lifestyle, p...]
[oil, vessel, port, sailing, coast, gulf, ocea...]
[inflation, ecb, european central, rate hike, ...]
[climate, carbon, emission, gigawatt, climate ...]
[debt, expense, finance, spending, survey, car...]
[xpeng, yuan, malaysia, toyota, tesla, volkswa...]
[rent, city, new york, housing, cost living, m...]
[hayes, menu, saudi, restaurant, breakfast, or...]
[treasury, treasury yield, fed, inflation, bas...

연관성 높은 기업 필터링

```
# 토픽 관련 기업 찾기
def retrieve_companies_by_keywords(keywords):
    keywords_set = set([word.lower() for word in keywords])

    # 기업 리스트
    cp = []

    for i, row in stock_info_df.iterrows():
        description = row['description']

        # 단어 추출
        if isinstance(description, str):
            description_words = set(description.replace(",", " ").lower().replace('.', ' ').split(' '))

        # 토픽 키워드와 description이 겹치는 기업 찾기
        if description_words & keywords_set:
            cp.append(row['tck_iem_cd'])

    return list(set(cp))
```

```
# 키워드 언급횟수 count
def count_companies_by_keywords(keywords):
    keywords_set = set([word.lower() for word in keywords])

    # 키워드 언급횟수 딕셔너리 생성
    keyword_counts = {word: 0 for word in keywords_set}

    for i, row in stock_info_df.iterrows():
        description = row['description']

        if isinstance(description, str):
            description_words = set(description.replace(",", " ").lower().replace('.', ' ').split(" "))

        # 키워드 횟수 카운트
        for word in keywords_set:
            if word in description_words:
                keyword_counts[word] += 1

    for keyword, count in keyword_counts.items():
        print(f"{keyword}: {count} companies")
```

```
computings: 0 companies
chatgpts: 0 companies
advanced micro: 0 companies
ai model: 0 companies
generative ai: 0 companies
micro device: 0 companies
chatbots: 0 companies
processing units: 0 companies
googles: 0 companies
openais: 0 companies
gpu: 1 companies
gpus: 1 companies
ais: 0 companies
chatgpt: 1 companies
ai models: 0 companies
aw: 0 companies
vmwares: 0 companies
chatbot: 0 companies
graphic processings: 0 companies
computing: 52 companies
czech: 4 companies
amds: 0 companies
advanced micros: 0 companies
language model: 0 companies
micro devices: 0 companies
graphic processing: 0 companies
processing unit: 0 companies
vmware: 0 companies
amd: 3 companies
openai: 0 companies
language models: 0 companies
google: 9 companies
generative ais: 0 companies
czechs: 0 companies
ai: 40 companies
aws: 2 companies
```

XGBOOST 기반 기업 Selection

dmlc
XGBoost

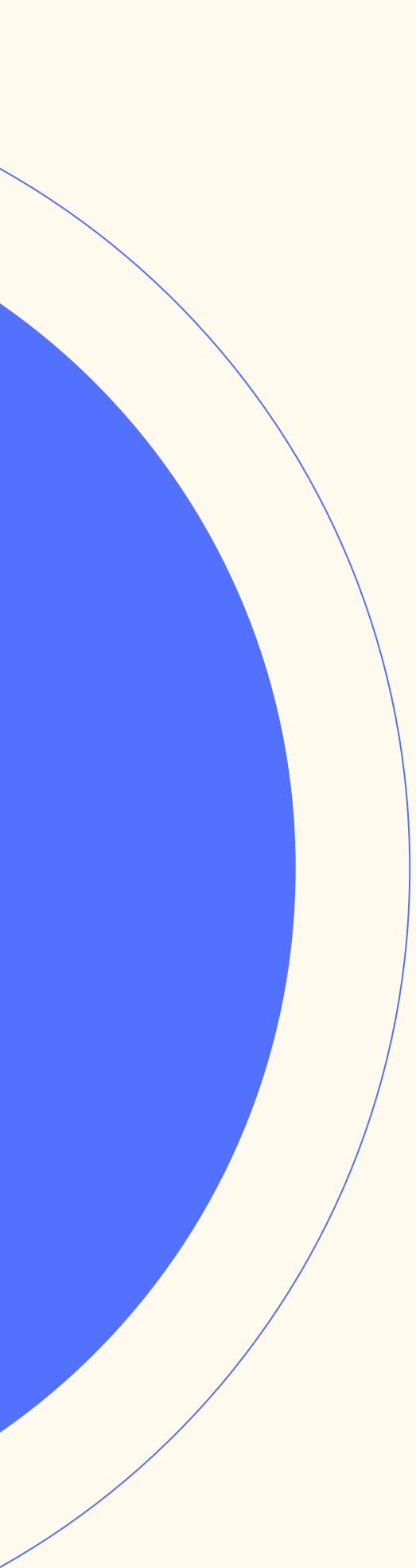
- XGBoost를 사용한 이유 : 해당 데이터에는 결측치가 많아 결측치를 효과적으로 처리하는 모델이 필요함. 또한 분류 모델 중 가장 성능이 높다고 알려진 XGBoost로 해당 분류를 진행함.
- 재무제표 데이터, 성과지표 등 정량 데이터를 활용해 8월 대비 9월의 평균 주가가 오를 것으로 예상되는 기업을 추출함.

```
# 현금 창출, 매출 관련 (ttm)
'totalRevenue': '총매출액',
'grossProfits': '매출총이익', # 매출이익(매출액 - 매출원가)
'revenuePerShare': '주당매출액',
'ebitda': 'EBITDA', # 감가상각 등의 부가비용을 차감하기 전의 금액, 영업 활동을 통한 현금 창출 능력, 유형자산의 가치까지 포함하는 지표
'ebitdaMargins': 'EBITDA마진', # 유형자산의 유지비용을 고려한 기업의 현금 창출 능력

# 재무 상태 관련 (mrq)
'debtToEquity': '부채자본비율',
'operatingCashflow': '영업현금흐름', # 영업현금흐름 : 영업이익 - 법인세 - 이자비용 + 감가상각비
'freeCashflow': '잉여현금흐름', # 기업의 본원적 영업활동을 위해 현금을 창출하고, 영업자산에 투자하고도 남은 현금
'totalCashPerShare': '주당현금흐름',
'currentRatio': '유동비율', # 회사가 가지고 있는 단기 부채 상환 능력
'quickRatio': '당좌비율', # 회사가 가지고 있는 단기 부채 상환 능력
'overallRisk': '위험 점수',

# 경영 효율 관련
'returnOnAssets': '자기자본이익률', # mrq : 간단히 말해, 얼마를 투자해서 얼마를 벌었냐
'returnOnEquity': '총자산순이익률', # mrq : ROE와 비교하여 기업이 가지고 있는 부채의 비중을 볼 때
'grossMargins': '매출총이익률', # ttm : 매출이익(매출액 - 매출원가) / 매출액 : 매출이익률, Gross Profit Margin (GPM)
'operatingMargins': '영업이익률', # ttm : 매출총이익 - 판관비 - 감가상각비
'profitMargins': '순이익률', # ttm : Net Income(순이익) / Revenue(총수익) : 순이익률, Net Profit Margin (NPM)
```

	Company	xgboost_prob
52	VOD	0.897710
22	INTC	0.847029
33	NICE	0.794178
17	DRS	0.750374
39	PERI	0.692716
27	LNTH	0.692540
14	CSCO	0.686896
20	GOOG	0.664316
47	SOUN	0.661067
25	KTOS	0.639091
6	AOSL	0.635895
32	NEWT	0.598093
9	CCCS	0.573763
7	APLD	0.568441
31	MSFT	0.566283
15	CTSH	0.555496
43	RGTI	0.547021
40	PLTK	0.542991
55	WDC	0.518878
45	SCSC	0.513791
36	NTAP	0.511056
29	MCHP	0.494042
54	VUZI	0.493545
16	DIOD	0.492292
53	VRNT	0.487612
3	AMD	0.487412
12	CEVA	0.483946
56	XRX	0.470001
0	AEHR	0.463278



5. 감성분석

FINBERT

- Unsupervised pretraining
- generic BERT 모델보다 강력
- corporate report / conference call transcript / analysis report 사용

- 금융 뉴스데이터에서 4,840 개의 문장 포함
- 16명의 전문지식을 갖춘 연구자들에 의해 수동 라벨링 하여 만들었음.
- 감정 라벨 : positive, neutral, negative

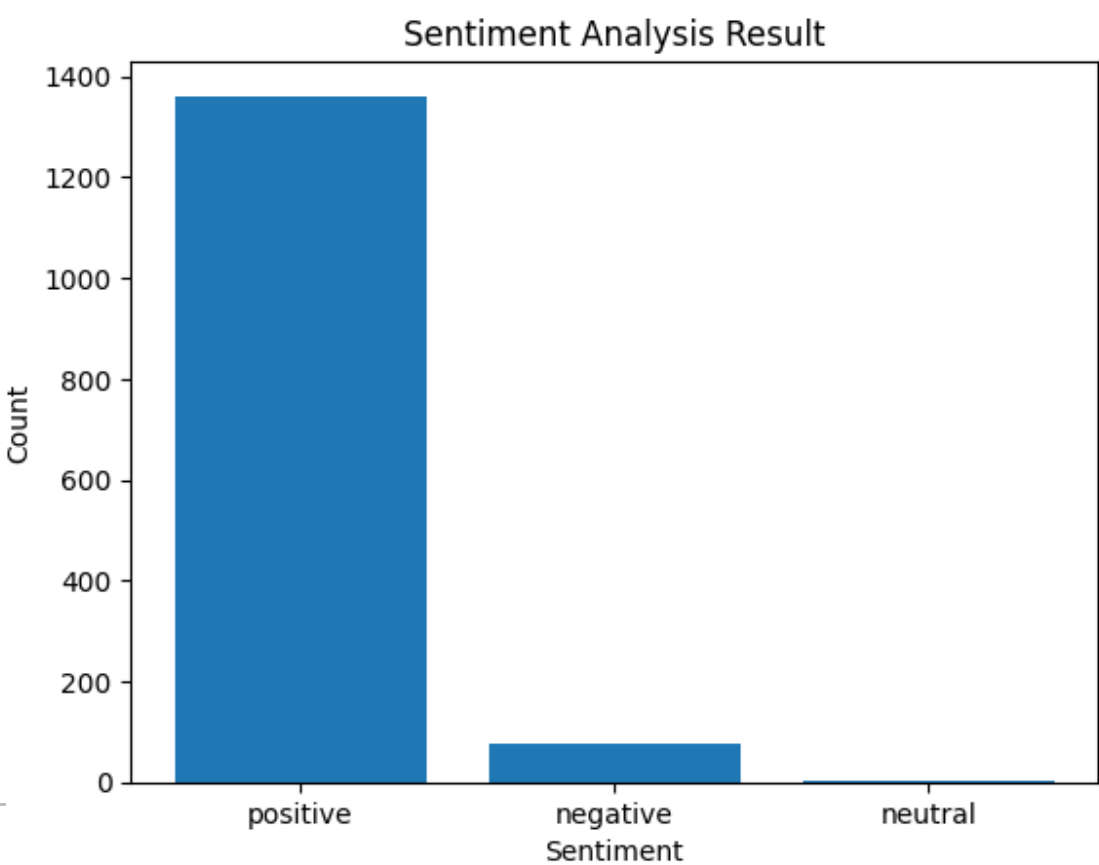
	BERT		FinBERT-BaseVocab		FinBERT-FinVocab	
	cased	uncased	cased	uncased	cased	uncased
PhraseBank	0.755	0.835	0.856	0.870	0.864	0.872
FiQA	0.653	0.730	0.767	0.796	0.814	0.844
AnalystTone	0.840	0.850	0.872	0.880	0.876	0.887

감성지수 도출

- XGBoost에서 선택된 기업들에, 제공된 NASDAQ 뉴스의 8월 데이터에서 FinBERT 감성분석을 실시함
- 이때, all_tck_iem_cd 열에 여러 기업이 있는 경우, 뉴스 기사에 여러 기업에 대한 언급이 포함됨
- 특정 기업에 대한 감성분석을 실시하려면 해당 기업코드 또는 기업명이 들어간 문단을 추출하는 게 적절하다고 판단함
- FinBERT를 사용한 이유는, 금융 도메인에 특화하여 pre-trained된 모델이기 때문임
- 그 중 ProsusAI의 모델을 사용한 이유는, 해당 모델이 금융 뉴스 문장들로 이루어진 Financial PhraseBank 데이터로 fine-tuning되어, 뉴스 데이터를 분석하기 적합하다고 판단했기 때문

Company	Count	Score_Pos_Mean	Score_Neg_Mean	Score_Neu_Mean	Score_Pos_Count	Score_Neg_Count	Score_Neu_Count
MSFT	434	0.392048	0.147402	0.460549	0.396175	0.128415	0.475410
AMD	226	0.369931	0.331374	0.298695	0.398058	0.325243	0.276699
GOOG	184	0.315841	0.160963	0.523195	0.289855	0.152174	0.557971
INTC	176	0.300308	0.277963	0.421729	0.301887	0.270440	0.427673
CSCO	111	0.351525	0.175900	0.472576	0.323810	0.161905	0.514286
PERI	15	0.287211	0.121754	0.591035	0.214286	0.071429	0.714286
SOUN	14	0.433710	0.146551	0.419739	0.500000	0.142857	0.357143
RGTI	10	0.512812	0.308145	0.179043	0.500000	0.400000	0.100000
NEWT	8	0.593522	0.106689	0.299788	0.625000	0.125000	0.250000
AEHR	7	0.404243	0.148987	0.446769	0.285714	0.142857	0.571429

기업 VOD의 기사는 8개입니다.
Mean: 0.5204 0.0188 0.4608
Count: 0.5000 0.0000 0.5000
기업 INTC의 기사는 176개입니다.
Mean: 0.3008 0.2780 0.4217
Count: 0.3019 0.2704 0.4277
기업 NIOE의 기사는 6개입니다.
Mean: 0.6678 0.0944 0.2360
Count: 0.6000 0.0000 0.2000





6. 그래프 기반 분석 (Similarity)

주제 설명

데이터 소개

전처리

데이터 분석

한계

참고 데이터

	A	B	C	D	E	F	G	H	I	J	K	L	M				
1	종목코드	종목명	매매거래장	정리매매	관리종목	투자주의	불성실공	단일가매	상장주식	수단기과	열종	투자주의	종	투자경고	종	투자위험	종목
2	60310	3S	X	X	X	X	X	X	X	X	X	X	X				
3	95570	AJ네트웍스	X	X	X	X	O	X	X	X	X	X	X				
4	6840	AK홀딩스	X	X	X	X	X	X	X	X	X	X	X				
5	54620	APS	X	X	X	X	X	X	X	X	X	X	X				
6	265520	AP시스템	X	X	X	X	X	X	X	X	X	X	X				
7	211270	AP위성	X	X	X	X	X	X	X	X	X	X	X				
8	27410	BGF	X	X	X	X	X	X	X	X	X	X	X				
9	282330	BGF리테일	X	X	X	X	X	X	X	X	X	X	X				
10	126600	BGF에코머	X	X	X	X	X	X	X	X	X	X	X				
11	138930	BNK금융지	X	X	X	X	X	X	X	X	X	X	X				
12	1460	BYC	X	X	X	X	X	X	X	X	X	X	X				
13	1465	BYC우	X	X	X	X	X	X	O	X	X	X	X				
14	13720	CRI	X	X	X	X	X	X	X	X	X	X	X				

	1	표준코드	단축코드	한글 종목명	한글 종목약명	영문 종목명	상장일	시장구분	증권구분	소속부	주식종류	액면가	상장주식수
2	KR7095570	95570	AJ네트웍스보	AJ네트웍스	AJ Networks Co.,Ltd.	2015-08-21	KOSPI	주권			보통주	1000	46822295
3	KR7006840	6840	AK홀딩스보통	AK홀딩스	AK Holdings, Inc.	1999-08-11	KOSPI	주권			보통주	5000	13247561
4	KR7282330	282330	BGF리테일보	BGF리테일	BGF Retail	2017-12-08	KOSPI	주권			보통주	1000	17283906
5	KR7027410	27410	BGF보통주	BGF	BGF	2014-05-19	KOSPI	주권			보통주	1000	95716791
6	KR7138930	138930	BNK금융지주	BNK금융지주	BNK Financial Group Inc.	2011-03-30	KOSPI	주권			보통주	5000	3.22E+08
7	KR7001460	1465	BYC1우선주	BYC우	BYC(1P)	1989-09-05	KOSPI	주권			구형우선주	5000	215385
8	KR7001460	1460	BYC보통주	BYC	BYC	1975-06-02	KOSPI	주권			보통주	5000	624615
9	KR7001040	1045	CJ1우선주	CJ우	CJ(1P)	1989-08-18	KOSPI	주권			구형우선주	5000	2260223
10	KR7001040	00104K	CJ4우선주(전	CJ4우(전환)	CJ(4PC)	2019-08-09	KOSPI	주권			신형우선주	5000	4226512
11	KR7001040	1040	CJ보통주	CJ	CJ Corp.	1973-06-29	KOSPI	주권			보통주	5000	29176998
12	KR7011150	11155	CJ씨푸드1우	CJ씨푸드1우	CJSEAFOODCORPORATIO	1990-01-13	KOSPI	주권			구형우선주	500	200000
13	KR7011150	11150	CJ씨푸드보통	CJ씨푸드	CJSEAFOODCORPORATIO	1990-11-26	KOSPI	주권			H투주	500	35030772

KRX 크롤링

- **앞선 데이터를 활용해 현재 거래가능한 주식 코드들을 리스트에 저장.**
- **pykrx를 통해 국내 주식 종목코드, 시가, 고가, 저가, 종가, 거래량을 크롤링.**

```
# Crawling stock info from KRX 정보데이터시스템
total = []
tk_ori = {}
ms = {}
md = {}

df1 = pd.read_csv('./data_5155_20230916.csv', encoding='cp949')
df2 = pd.read_csv('./data_5206_20230916.csv', encoding='cp949')

df1['상장일'] = pd.to_datetime(df1['상장일']).dt.strftime('%Y%m%d').astype('int32')
df2['상장일'] = pd.to_datetime(df2['상장일']).dt.strftime('%Y%m%d').astype('int32')

df1 = df1[df1['상장일'] <= 20230101]
df2 = df2[df2['상장일'] <= 20230101]

tickers = df1['단축코드'].values.tolist()
tickers += df2['단축코드'].values.tolist()

red = pd.read_csv('./data_.csv', encoding='cp949')
stopped = red[red['매매거래정지']=='0']['종목코드'].values.tolist()

tickers = list(set(tickers) - set(stopped))

for t in tqdm(tickers):
    df = stock.get_market_ohlcv('20230103', '20230831', t)

    df['tck_iem_cd'] = t
    df.reset_index(inplace=True)
    df = df.rename(columns={"날짜": 'trd_dt', "시가": "gts_iem_ong_pr", '고가': 'gts_iem_hi_pr', '저가': 'gts_iem_low_pr', '종가': 'gts_iem_end_pr', '거래량': 'gts_acl_trd_qty'})
    df = df.drop(['등락률'], axis=1)

    try:
        ms[t] = int(stock.get_market_trading_value_by_date('20230103', '20230831', t, on='매수')['전체'].values)
        md[t] = int(stock.get_market_trading_value_by_date('20230103', '20230831', t, on='매도')['전체'].values)
    except:
        ms[t] = 0
        md[t] = 0

df['gts_sll_cns_sum_qty'] = df['tck_iem_cd'].map(md)
df['gts_byn_cns_sum_qty'] = df['tck_iem_cd'].map(ms)
cols = ['trd_dt', 'tck_iem_cd', 'gts_iem_ong_pr', 'gts_iem_hi_pr', 'gts_iem_low_pr', 'gts_iem_end_pr', 'gts_acl_trd_qty', 'gts_sll_cns_sum_qty', 'gts_byn_cns_sum_qty']
df = df[cols]
```

주제 설명

데이터 소개

전처리

데이터 분석

한계

KRX 크롤링 결과물

< 해외 주식 >

1	trd_dt	tck_iem_cd	gts_iem_ong_pr	gts_iem_hi_pr	gts_iem_low_pr	gts_iem_end_pr	gts_acl_trd_qty	gts_sll_cns_sum_qty	gts_byn_cns_sum_qty
2	20230103	NVDA	148.51	149.96	140.96	143.15	4.01E+07	0	0
3	20230103	APLT	0.7535	0.79	0.7308	0.7522	63714	19658	44056
4	20230103	ANY	1.89	2.0293	1.8354	1.96	24896.4	0	0
5	20230103	CLRB	1.66	1.72	1.63	1.63	21212	16444	4768
6	20230103	NYMTM	18.43	18.74	18.43	18.7	8061	0	0
7	20230103	GT	10.39	10.54	10.18	10.19	3353929	1804076	1549853
8	20230103	XWEL	0.35	0.41	0.346499	0.3901	525300	0	0
9	20230103	SBGI	15.34	15.92	15.28	15.32	708182	0	0
10	20230103	SGC	10.11	10.4	10.1056	10.36	14886	0	0
11	20230103	SILCO	2.97	3.43	2.73	3.07	1062079	504684	557395

< 국내 주식 >

trd_dt	tck_iem_cd	gts_iem_ong_pr	gts_iem_hi_pr	gts_iem_low_pr	gts_iem_end_pr	gts_acl_trd_qty
2023-01-03	302430	12450	13050	12050	13050	9953
2023-01-03	121440	4410	4500	4360	4435	108558
2023-01-03	71320	27650	27750	26700	27400	1970
2023-01-03	2410	3300	3350	3190	3300	379430
2023-01-03	52860	2410	2430	2335	2375	91640
2023-01-03	18290	5160	5210	5080	5170	127929
2023-01-03	365900	8160	8240	7730	8160	307591
2023-01-03	70	66500	68100	65900	67100	16623
2023-01-03	33830	1125	1135	1065	1105	513285
2023-01-03	317400	5880	5910	5610	5710	128364
2023-01-03	348950	4260	4280	4225	4255	253967
2023-01-03	377030	11150	11650	10850	11550	275145
2023-01-03	182360	15550	17000	15500	16750	323564
2023-01-03	00104K	73700	74100	71900	73500	2483

주제 설명

데이터 소개

전처리

데이터 분석

한계

Featuring Engineering

<기본지표>

```
# make basic indicators
def get_basic_indicators(df, week=5, month=21):
    df_copy = df.copy()
    price = df_copy['gts_iem_end_pr']
    volume = df_copy['gts_acl_trd_qty']
    avg_price_week = price.rolling(window=week).mean()
    avg_price_month = price.rolling(window=month).mean()
    std_price_week = price.rolling(window=week).std()
    std_price_month = price.rolling(window=month).std()

    avg_volume_week = volume.rolling(window=week).mean()
    avg_volume_month = volume.rolling(window=month).mean()
    std_volume_week = volume.rolling(window=week).std()
    std_volume_month = volume.rolling(window=month).std()

    return_1 = ((price - price.shift(1)) / price.shift(1)).shift(1).fillna(0)
    return_week = ((price - price.shift(week)) / price.shift(week)).shift(1).fillna(0)
    return_month = ((price - price.shift(month)) / price.shift(month)).shift(1).fillna(0)

    return_ma_week = return_1.rolling(window=week).mean()
    return_ma_month = return_1.rolling(window=month).mean()

    df_copy['avg_price_week'] = avg_price_week
    df_copy['avg_price_month'] = avg_price_month
    df_copy['std_price_week'] = std_price_week
    df_copy['std_price_month'] = std_price_month

    df_copy['avg_volume_week'] = avg_volume_week
    df_copy['avg_volume_month'] = avg_volume_month
    df_copy['std_volume_week'] = std_volume_week
    df_copy['std_volume_month'] = std_volume_month

    df_copy['return_1'] = return_1
    df_copy['return_week'] = return_week
    df_copy['return_month'] = return_month

    df_copy['return_ma_week'] = return_ma_week
    df_copy['return_ma_month'] = return_ma_month

    return df_copy
```

“Correlation에 쓸 평가지표 생성”

<볼린저 밴드>

```
[ ] # make techincaI indicators
def get_boll(price,n=20):
    ...

    parameters:
        price: 가격 (종가, dtype=list)
        n: 종가 단순이동평균 계산 날짜수
    returns:
        볼린저밴드 하한선, 볼린저밴드 중심선, 볼린저밴드 상한선
        ...

    sma=price.rolling(window=n).mean()
    std=price.rolling(window=n).std()
    boll_mid=sma #볼린저밴드 중심선
    boll_high=sma+2*std #볼린저밴드 상한선
    boll_low=sma-2*std #볼린저밴드 하한선
    return boll_low, boll_mid, boll_high
```

주제 설명

데이터 소개

전처리

데이터 분석

한계

Featuring Engineering

< MACD 이동평균 수렴확산지수 >

```
def get_macd(price,n=9):
    ...

    parameters:
        price: 가격 (종가, dtype=list)
        n: 이동평균 계산 날짜수 (dtype=int)
    returns:
        MACD, MACD 시그널, MACD 오실레이터
        ...

    # 매는 지수 이동 평균 Exponential Moving Average (EMA) 을 사용함
    ewm12=price.ewm(span=12, adjust=False).mean()
    ewm26=price.ewm(span=26, adjust=False).mean()
    macd=ewm12-ewm26 #MACD
    macd_signal=macd.ewm(span=n,min_periods=n-1,adjust=False).mean() #MACD Signal
    macd_oscil=macd-macd_signal #MACD Oscillator
    return macd, macd_signal, macd_oscil
```

< 스토캐스틱 지표 >

```
def get_stochastic(high,low,close,n=15, m=5, t=3):
    ...

    parmameters:
        high: 고가 (dtype=list)
        low: 저가 (dtype=list)
        close: 종가 (dtype=list)
        n: fast K stochastic 도출 시 이동평균 구간
        m: slow K stochastic 도출 시 이동평균 구간
        t: slow D stochastic 도출 시 이동평균 구간
    returns:
        Fast%K, Slow%K, Slow%D
        ...

    min_low=low.rolling(n).min()
    max_high=high.rolling(n).max()
    fast_k=100*(close-min_low)/(max_high-min_low)
    slow_k=fast_k.rolling(m).mean()
    slow_d=slow_k.rolling(t).mean()
    return fast_k, slow_k, slow_d
```

Featuring Engineering

< RSI 상대강도지수 >

```
def get_rsi(price,n1=14,n2=9):
    ...

    parameters:
        price: 가격 (종가, dtype=list)
        n1: 평균상승률/평균하락률 지수이동평균 계산 날짜수
        n2: 단순이동평균 계산 날짜수
    returns: RSI, RSI 시그널
    ...

    delta=price.diff()
    up=delta.clip(lower=0)
    down=delta.clip(upper=0)
    ema_up=up.ewm(com=n1, adjust=False).mean()
    ema_down=abs(down.ewm(com=n1, adjust=False).mean())
    rs=ema_up/ema_down
    rsi=100-np.floor((100/(1+rs))) #RSI
    rsi_signal=rsi.rolling(window=n2).mean() #RSI Signal
    return rsi, rsi_signal
```

< turbulence 지표 >

```
def calculate_turbulence(data):
    """calculate turbulence index based on dow 30"""

    df = data.copy()
    df_price_pivot = df.pivot(index='trd_dt', columns='tck_iem_cd', values='gts_iem_end_pr')
    df_price_pivot = df_price_pivot.pct_change()

    unique_date = df.trd_dt.unique()
    # start after a month
    start = 21
    turbulence_index = [0] * start
    count = 0
    for i in tqdm(range(start, len(unique_date))):
        current_price = df_price_pivot[df_price_pivot.index == unique_date[i]]
        # use one month rolling window to calculate covariance
        hist_price = df_price_pivot[
            (df_price_pivot.index < unique_date[i])
            & (df_price_pivot.index >= unique_date[i - start])]
        # Drop tickers which has number missing values more than the "oldest" ticker
        filtered_hist_price = hist_price.iloc[hist_price.isna().sum().min():].dropna(axis=1)

        cov_temp = filtered_hist_price.cov()
        current_temp = current_price[[x for x in filtered_hist_price]] - np.mean(filtered_hist_price, axis=0)

        temp = current_temp.values.dot(np.linalg.pinv(cov_temp)).dot(current_temp.values.T)
        if temp > 0:
            count += 1
            if count > 2:
                turbulence_temp = temp[0][0]
            else:
                # avoid large outlier because of the calculation just begins
                turbulence_temp = 0
        else:
            turbulence_temp = 0
        turbulence_index.append(turbulence_temp)
    turbulence_index = pd.DataFrame({'trd_dt': df_price_pivot.index, 'turbulence': turbulence_index})
    return turbulence_index
```


CONTENTS

주제 설명

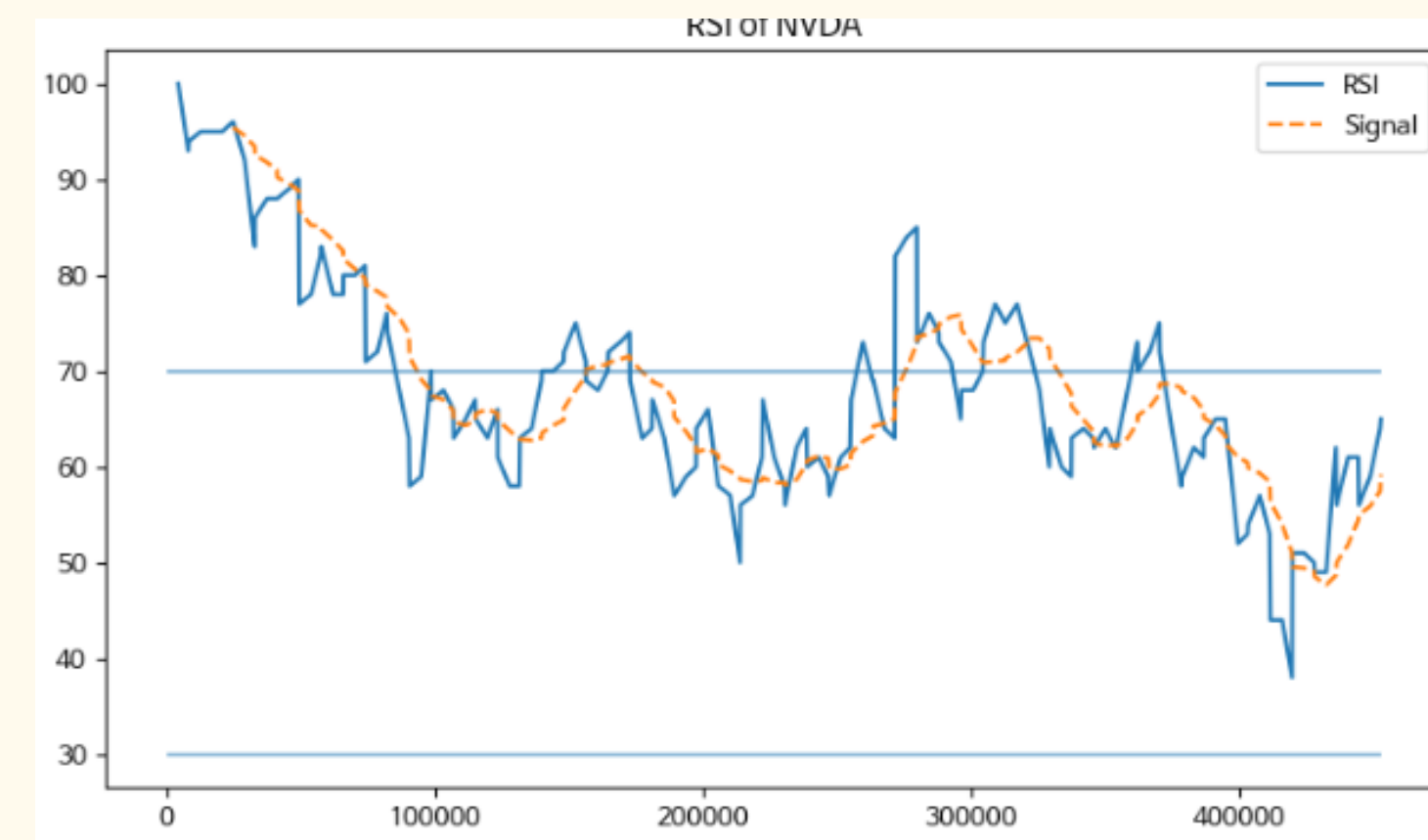
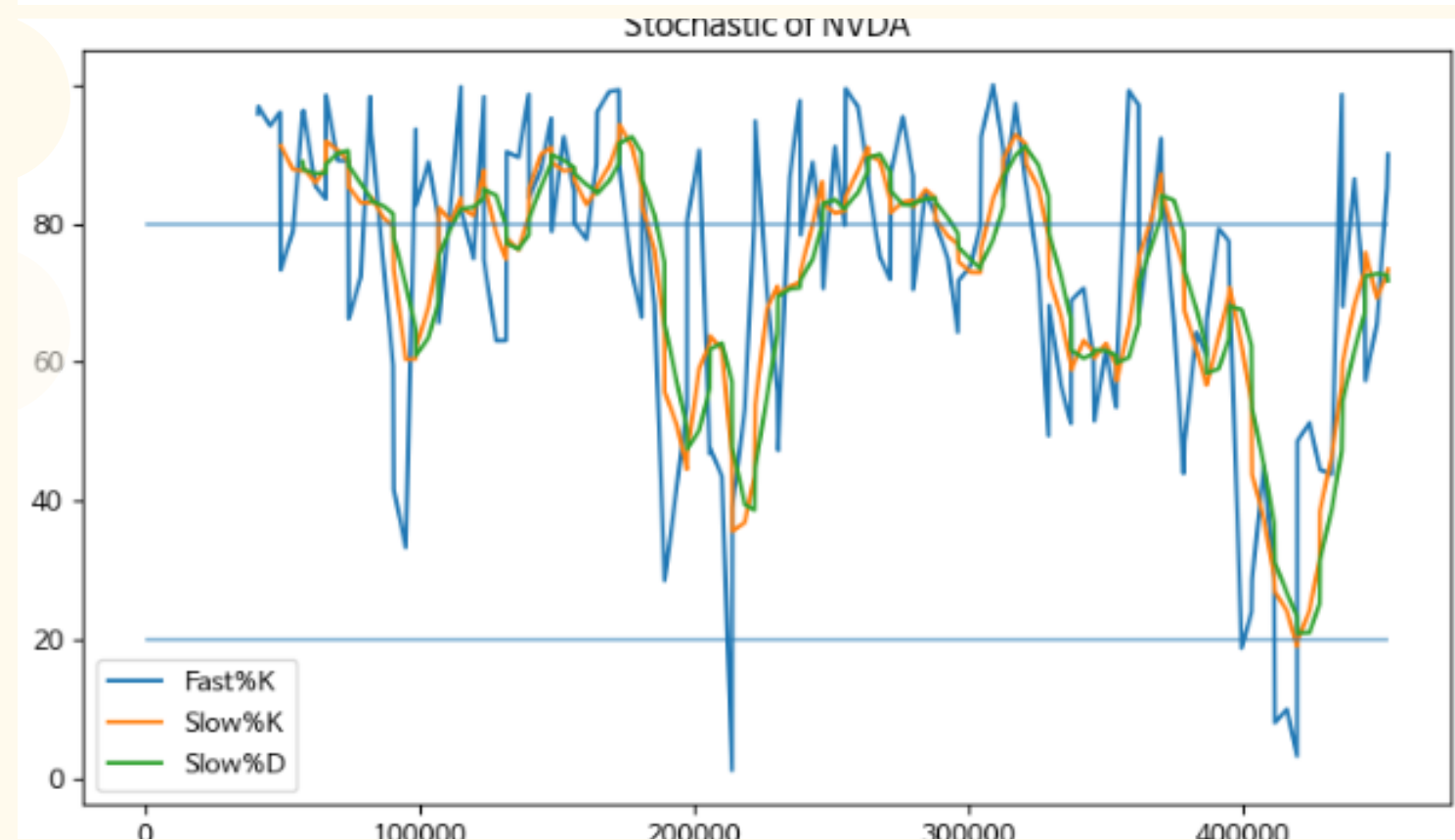
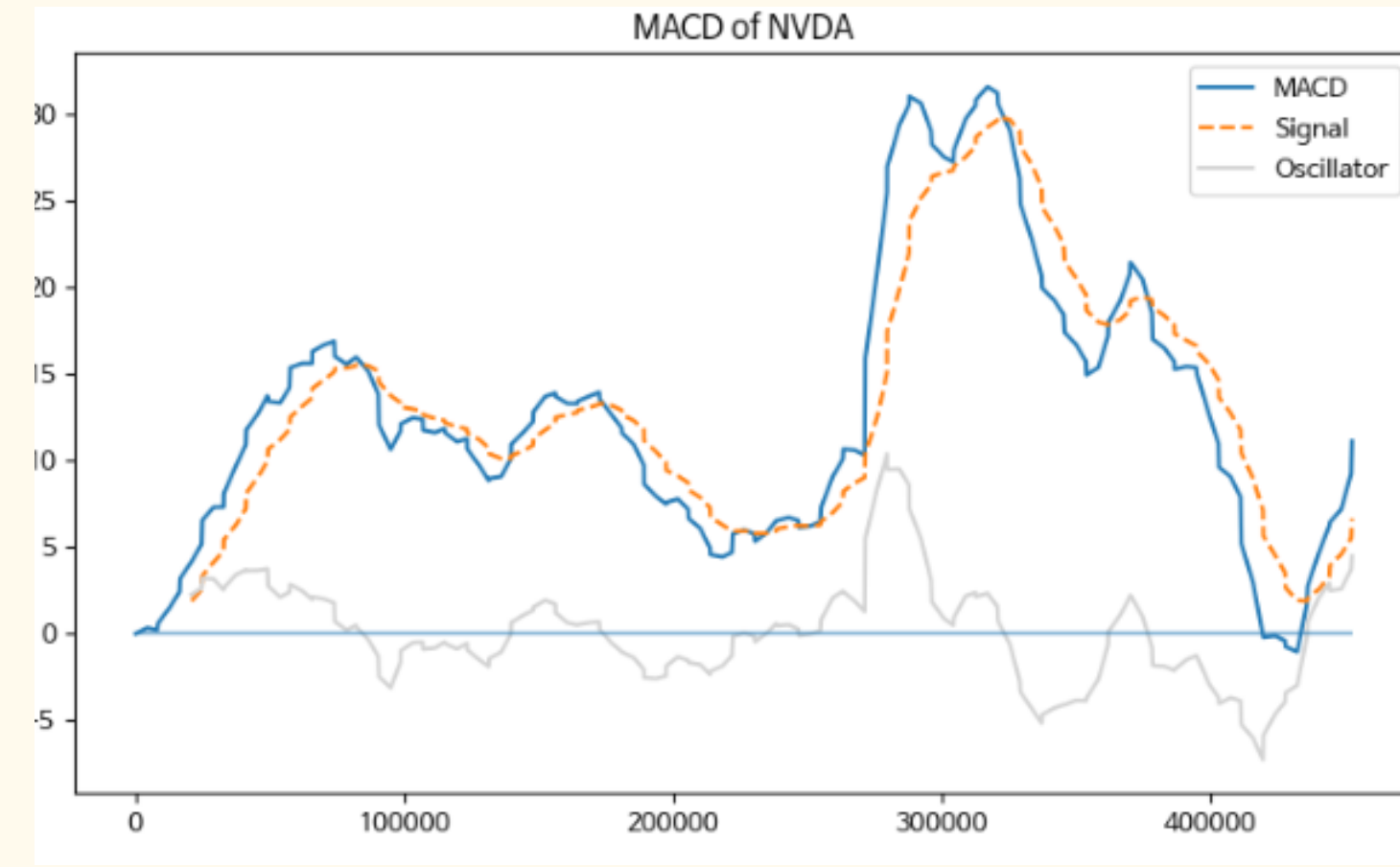
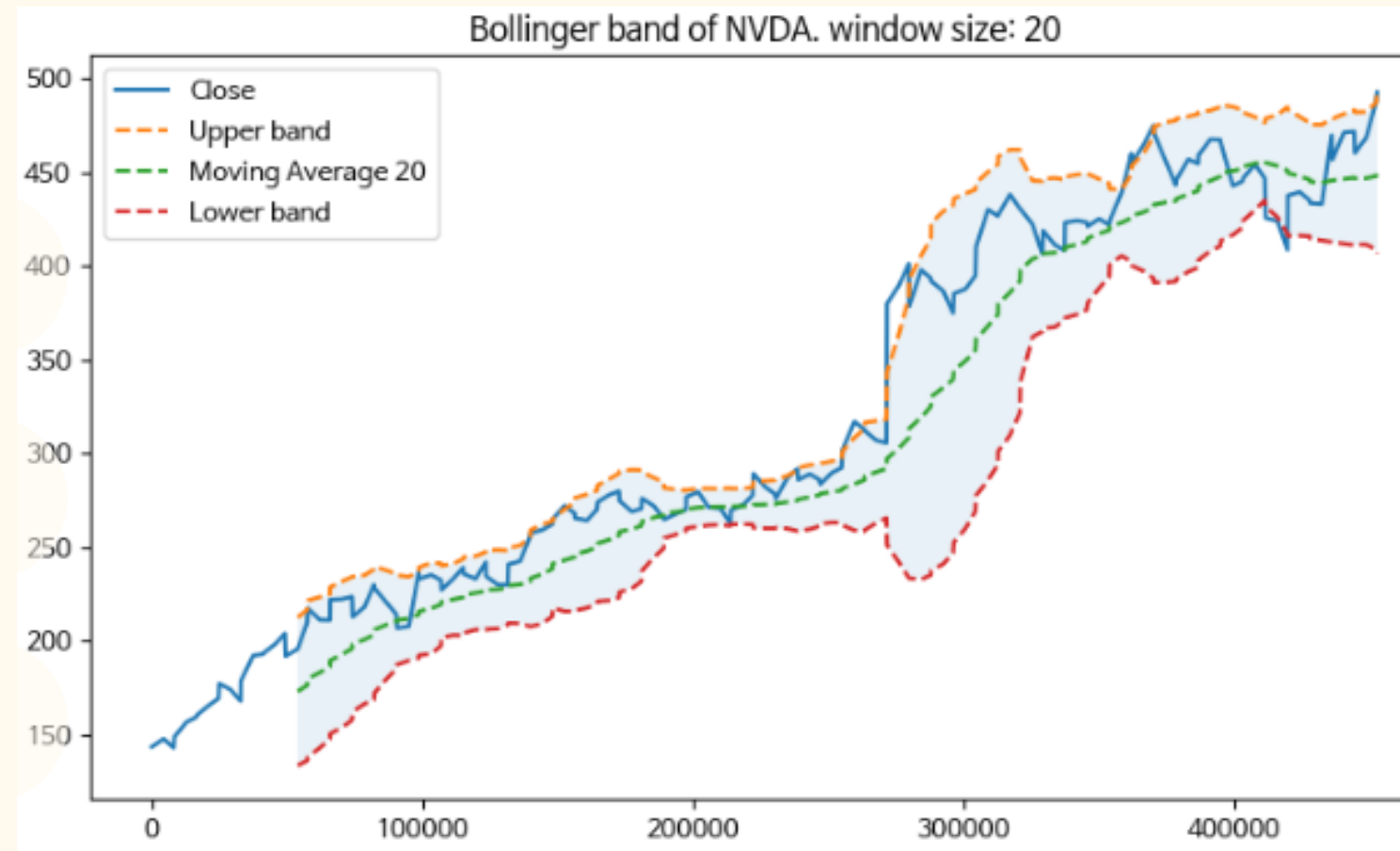
데이터 소개

전처리

데이터 분석

한계

Featuring Engineering



주제 설명

데이터 소개

전처리

데이터 분석

한계

Featuring Engineering

	trd_dt	tck_iem_cd	gts_iem_ong_pr	gts_iem_hi_pr	gts_iem_low_pr	gts_iem_end_pr	gts_acl_trd_qty	gts_sll_cns_sum_qty	gts_byn_cns_sum_qty	turbulence	...	std_price_month	avg_volume_week	avg_volume_month	std_volume
0	20230103	AACG	1.22	1.3000	1.2200	1.2600	6972.0	420.0	6552.0	0.000000e+00	...	NaN	NaN	NaN	
1	20230104	AACG	1.28	1.2800	1.2500	1.2700	1350.0	0.0	1350.0	0.000000e+00	...	NaN	NaN	NaN	
2	20230105	AACG	1.24	1.3665	1.2400	1.3210	2307.0	980.0	1327.0	0.000000e+00	...	NaN	NaN	NaN	
3	20230106	AACG	1.36	1.4400	1.3543	1.4123	1314.0	657.0	657.0	0.000000e+00	...	NaN	NaN	NaN	
4	20230109	AACG	1.42	1.4767	1.4000	1.4000	7422.0	1212.0	6210.0	0.000000e+00	...	NaN	3873.0	NaN	3064.0
...
455333	20230824	ZYXI	8.02	8.0200	7.6500	7.7500	268578.0	170630.0	97948.0	8.339505e+00	...	0.732384	275789.6	333038.857143	36862.0
455334	20230825	ZYXI	7.74	7.7700	7.5100	7.7000	179197.0	81549.0	97648.0	2.240631e+01	...	0.646207	255064.4	327993.571429	56054.0
455335	20230828	ZYXI	7.70	7.7700	7.5700	7.6600	200784.0	104724.0	96060.0	0.000000e+00	...	0.504455	230575.0	321928.952381	44355.0
455336	20230829	ZYXI	7.65	7.7400	7.5500	7.7100	247955.0	124040.0	123915.0	2.454570e+11	...	0.303054	223463.2	323634.476190	35763.0
455337	20230830	ZYXI	7.72	7.8700	7.5250	7.7500	339119.0	142987.0	196132.0	1.319626e+00	...	0.254210	247126.6	299311.904762	62620.0

455338 rows × 34 columns

피쳐로 쓸 여러 지표를 만들고 값의 결과를 데이터프레임에 합하여 저장

국내 주식 데이터 또한 동일하게 진행하여 저장.

주제 설명

데이터 소개

전처리

데이터 분석 ●

한계

Graph Building

```
def get_monthly_return(df):
    df_copy = df.copy()
    df_copy['trd_dt'] = df_copy['trd_dt'].astype(str)
    df_copy['date'] = pd.to_datetime(df_copy['trd_dt'])
    df_copy = df_copy.set_index('date')
    df_copy = df_copy.resample('M').last('D')
    df_copy['monthly_return'] = df_copy['gts_iem_end_pr'].pct_change()
    # df_copy = df_copy.fillna(0)
    df_copy = df_copy.dropna()
    return df_copy['monthly_return'].values

# from target paper
def pearson_correlation(x, y):
    return np.dot((x - np.mean(x)), (y - np.mean(y))) / ((np.linalg.norm(x - np.mean(x))) * (np.linalg.norm(y - np.mean(y))))

def adjusted_correlation(x, y):
    return np.dot(x, y) / ((np.linalg.norm(x)) * (np.linalg.norm(y)))

def calculate_cumulative_return(c_i, c_j):
    return np.abs((c_i+1).cumprod()[-1] - (c_j+1).cumprod()[-1])

def final_similarity(c_i, c_j, w=0.5):
    e = calculate_cumulative_return(c_i, c_j)
    tau = adjusted_correlation(c_i, c_j)
    return w / (1+e) + (1-w) * tau
```

Graph Building

```
n, m = len(df_per_name_nas), len(df_per_name_kor)
threshold = 0.7
adjacency_matrix = sp.lil_matrix((n, m))

monthly_returns_nas = [get_monthly_return(i) for i in df_per_name_nas]
monthly_returns_kor = [get_monthly_return(i) for i in df_per_name_kor]

for i in tqdm(range(n)):
    for j in range(m):
        x = monthly_returns_nas[i]
        y = monthly_returns_kor[j]
        if len(x) != 7 or len(y) != 7:
            continue
        similarity = final_similarity(x, y)
        if similarity > threshold:
            adjacency_matrix[i, j] = similarity

adjacency_matrix = adjacency_matrix.tocsr()
sp.save_npz('Stock_Graph_Both.npz', adjacency_matrix)
```

100%|██████████| 2743/2743 [01:40<00:00, 27.28it/s]

**국내/해외 새로운 피처를 추가한 데이터셋으로 하여금
Correlation과 Cumulative return을 고려한
stock-specific metric을 통해 유사한 종목계산**

```
# Load our stock graph
adj = sp.load_npz('./Stock_Graph_Both.npz')

# 전체 엣지 개수
print('Total:', np.count_nonzero(adj.toarray()))
print(f'Ratio: {np.count_nonzero(adj.toarray()) / (n * m) * 100:.2f}%')
```

Total: 351901
Ratio: 5.24%

Graph Building

- 타겟으로 하는 주식코드 (000560)와 가장 유사한 코드를 제시.

```
target_node = 560
adj_np = adj.toarray()
similar_node = adj_np[target_node].argmax()

print('Target:', target_node)
print('Most similar stock:', similar_node)
print('Similarity:', adj[target_node, similar_node])
```

```
Target: 560
Most similar stock: 1071
Similarity: 0.9832331936173919
```

CONTENTS

주제 설명

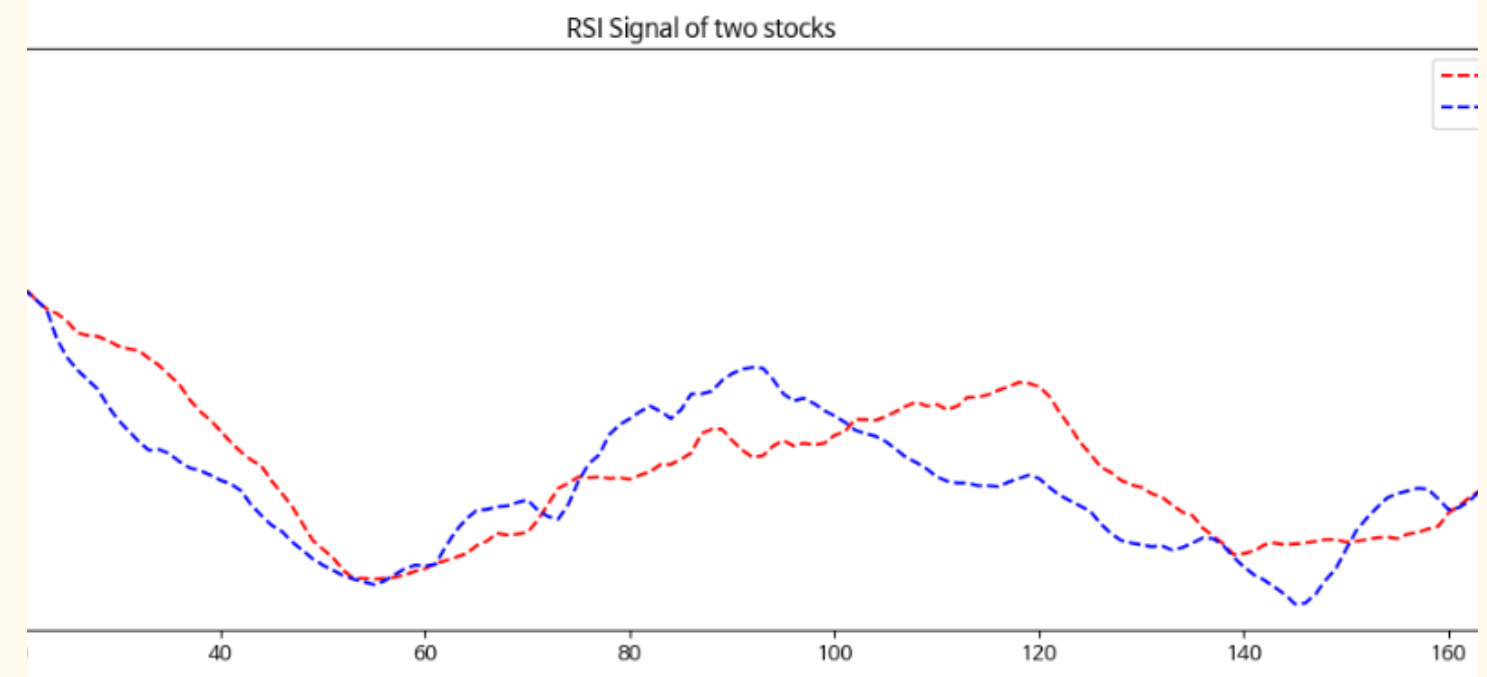
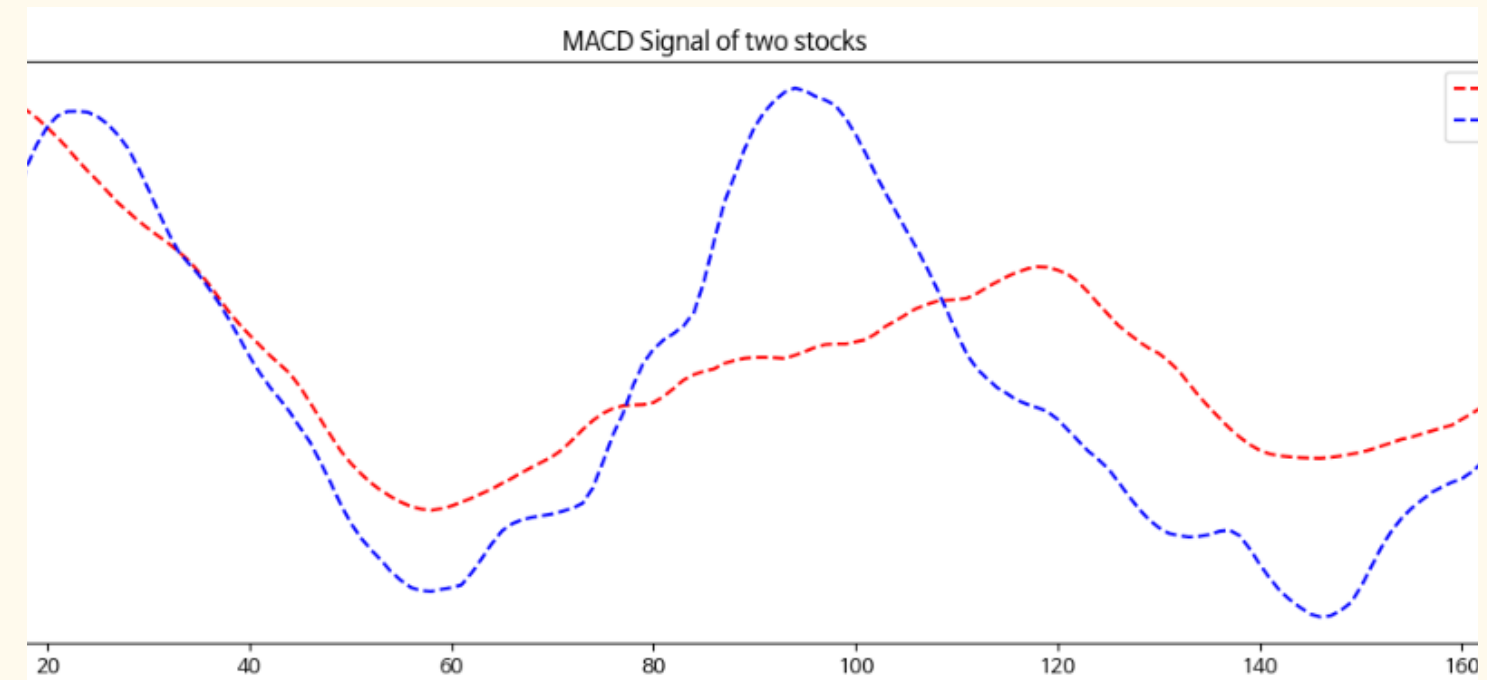
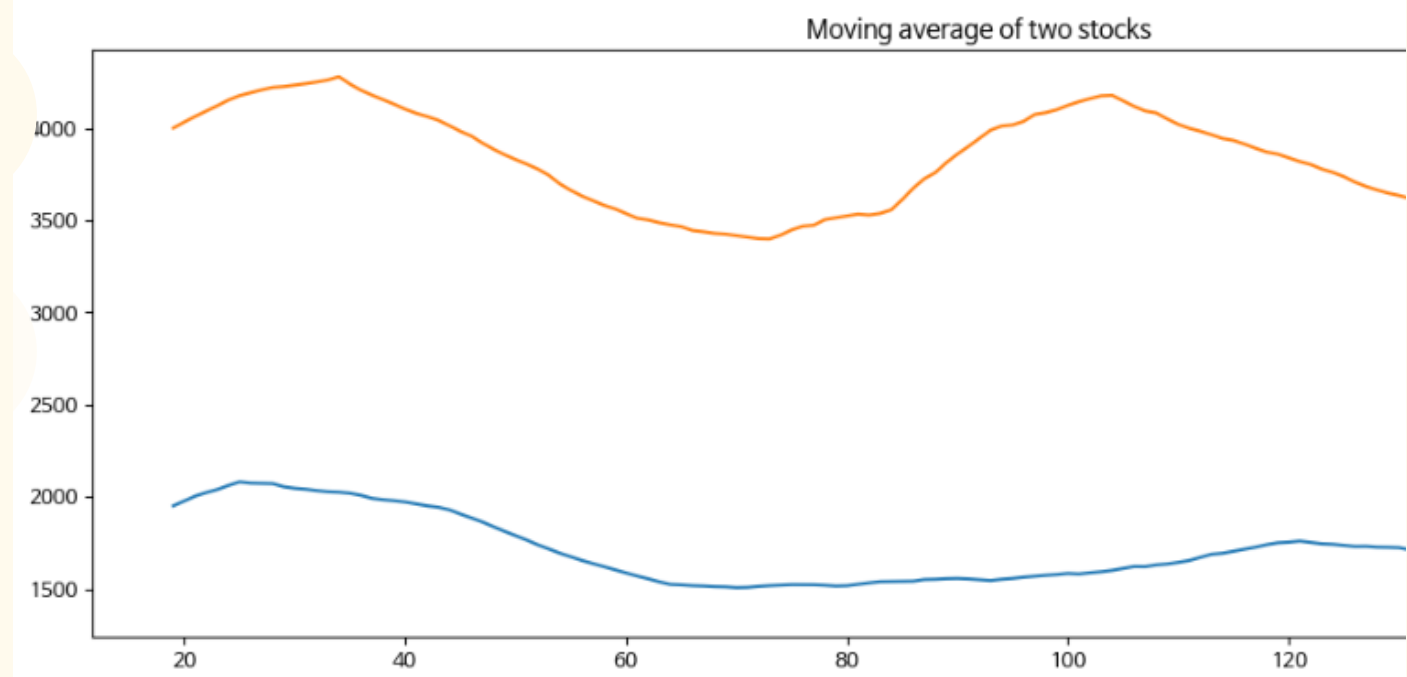
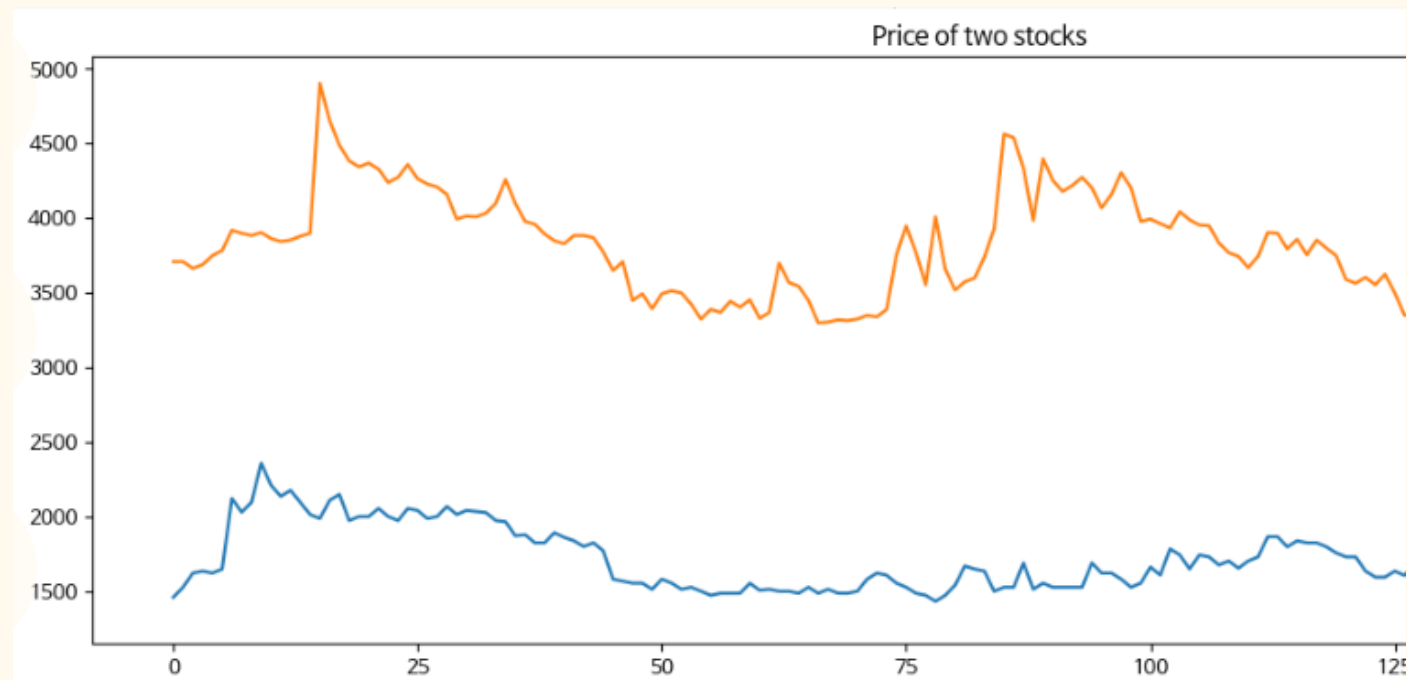
데이터 소개

전처리

데이터 분석

한계

시각화 결과



주제 설명

데이터 소개

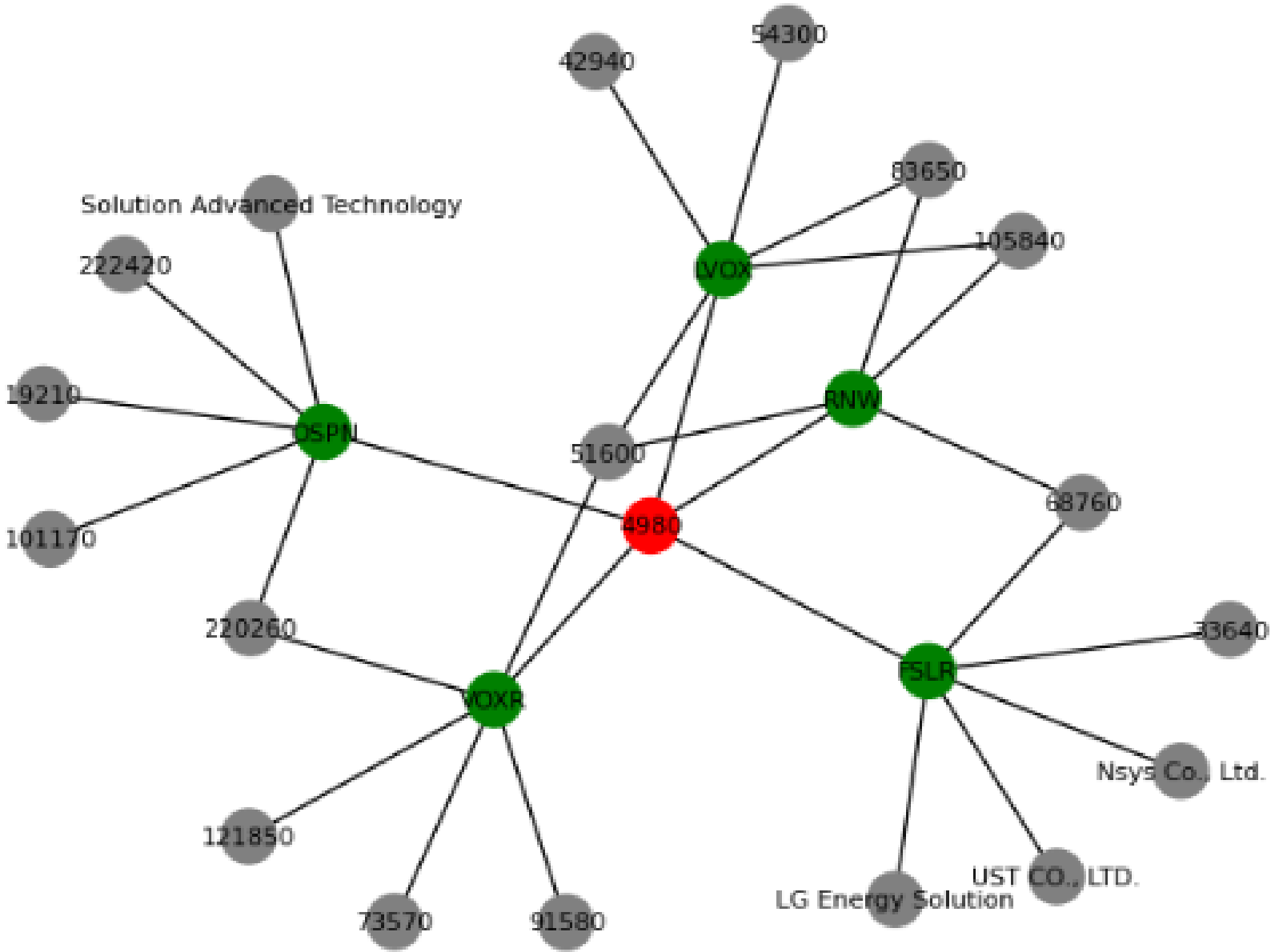
전처리

데이터 분석

한계

최종 시각화 결과물

Case of : 4980



- 004980 (국내) 와 유사한 종목들을 나열
- 이 경우 OSPN, VOXR, FSLR, RNW, LVOX 티커코드 주식의 유사성이 높음.

주제 설명

데이터 소개

전처리

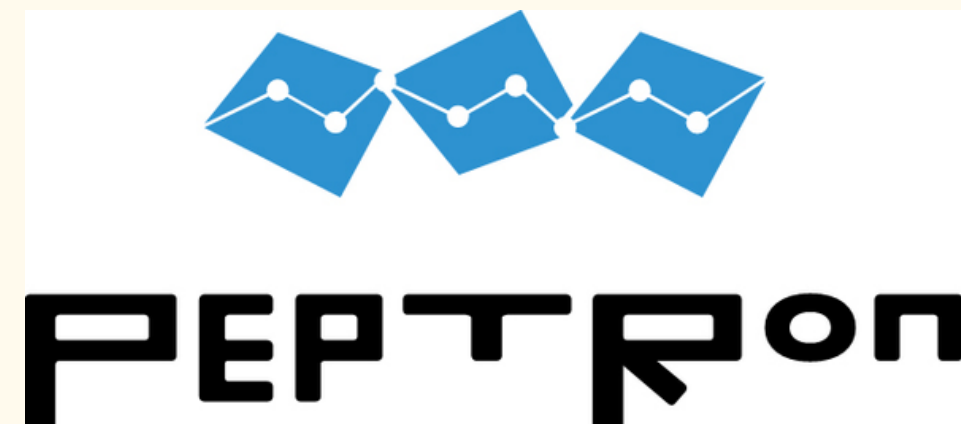
데이터 분석

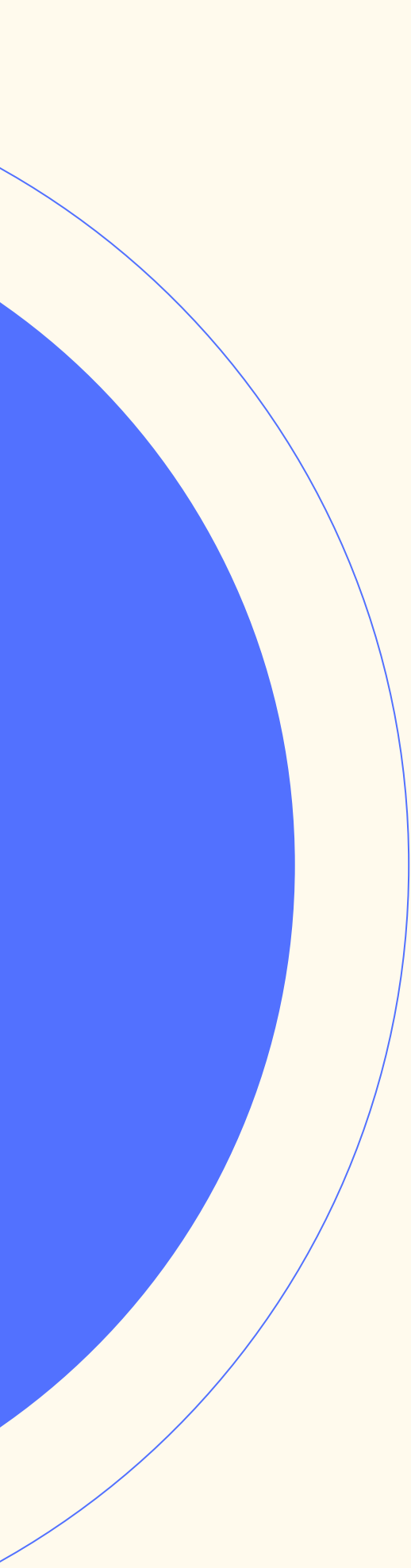
한계

최종 추천 결과물

2023-07-26:	['A010640' 'A032560' 'A042940' 'A045300' 'A060310' 'A127120' 'A317530']
2023-07-27:	['A002710' 'A007770' 'A256610' 'A266490' 'A317770' 'A355150' 'A357550']
2023-07-28:	['A007770' 'A039200' 'A064440' 'A256610' 'A317770' 'A355150' 'A357550']
2023-07-31:	['A010640' 'A032560' 'A042940' 'A045300' 'A070590' 'A162120' 'A236490']
2023-08-01:	['A010640' 'A032560' 'A045300' 'A070590' 'A093520' 'A162120' 'A236490']
2023-08-02:	['A002710' 'A007770' 'A039200' 'A169670' 'A256610' 'A317770' 'A355150']
2023-08-03:	['A002710' 'A007770' 'A039200' 'A169670' 'A256610' 'A317770' 'A355150']

ex) 8월 1일자 포트폴리오:





7. 한계



한계

- 분석 초기에 **Sentiment** 정보를 간과한 점
- 거시경제환경에 대한 미흡한 분석 (ex. 미중 무역전쟁, Fed의 통화정책, 기준금리 등)
- **2023.01** 이전 데이터들의 부재 (장기적 관점에서 분석하기엔 데이터가 부족)
- **Similarity**의 측면에서 평가 후 더 심화적으로 분석하지 못한 점
- 종목들의 구성 비율 (**portion**을 정하지 않음.)
- 포트폴리오의 분산화에 대한 작용이 미비함.
- 크롤링 과정에서 결측이 예상보다 빈번하게 일어나 정밀한 분석에 악영향.



감사합니다!