기술적 분석에 대한 성과분석

FR 32기 나종진

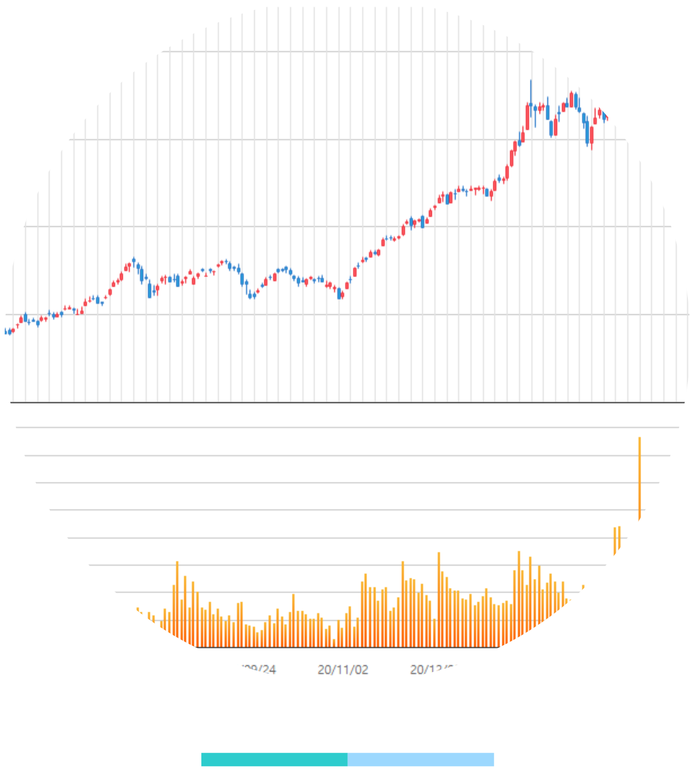
May 2021

성과 분석 배경

- 수익을 낼 방법 고안

- 주식차트 분석

- 차트 분석이 효과가 있는가?



Agenda

01 기술적 분석

02 분석 지표 종류

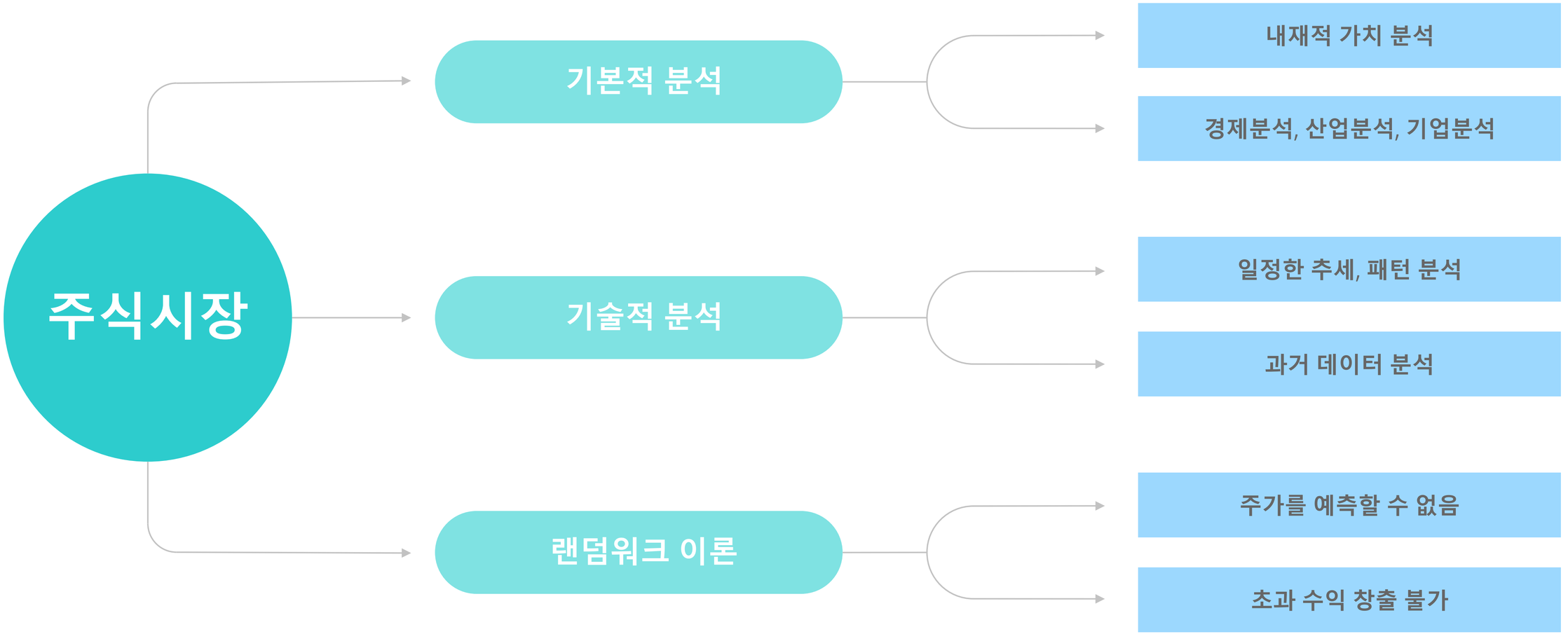
03 코딩

04 결론

01 - 기술적 분석

기술적 분석

주식시장을 접근하는 3가지 방법



기술적 분석

기술적 분석 필요 자료

차트 유형

막대차트, 종가차트, P&F 차트, 봉차트

추세

어느 기간 동안 같은 방향으로 움직이는 경향

일정한 직선 또는 곡선으로 나타내는 선

이동평균선

특정기간동안 일정한 방향성을 수치화 한 것

장기(120일), 중기(60일), 단기(20, 5일) 이동평균선

기술적 분석

기술적 분석 필요 자료

차트패턴

일일패턴: 캡, 스파이크, 반전일, 추력일, 대변동일

지속형 패턴: 삼각형, 깃발형, 페넌트형

천장형과 바닥형: V형과 역 V형, 이중 천장형과 이중 바닥형, 머리어깨형, 둥근 천장형과 둥근 바닥형, 삼각형, 쐐기형, 섬꼴 반전형

오실레이터

횡보장세에서의 전환점을 포착하는 데 적합한 방법

넓은 의미로 가격의 움직임을 나타내는 모든 지수

좁은 의미로 최근의 가격에서 과거 일정시점의 가격을 빼서 산출한 결과

차트 유형

다양한 차트 유형

종가 차트

종가를 연결하여 나타냄.

P&F 차트

시간의 개념을 배제, 사소한 주가변동을 제외

일정 기준 이상으로 주사 상승할 때 X

하락할 때 O

캔들 차트

시가, 종가, 저가, 고가를 하나의 봉에 나타냄

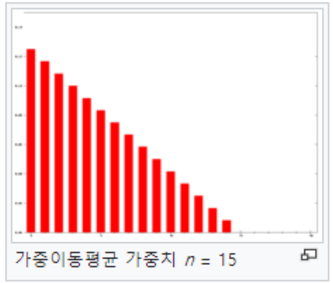
일봉, 주봉, 월봉 등 다양함.

이동평균법

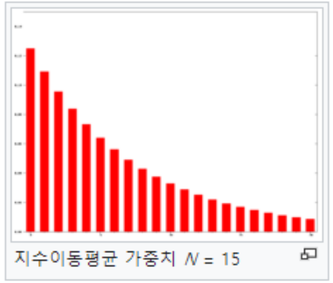
다양한 이동평균 방법

$$\bar{p}_{SM} = \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n} = \frac{1}{n}\sum_{i=0}^{n-1} p_{M-i}.$$

$$\mathrm{WMA}_M = \frac{np_M + (n-1)p_{M-1} + \cdots + 2p_{(M-n+2)} + p_{(M-n+1)}}{n + (n-1) + \cdots + 2 + 1}$$



$$S_t = \begin{cases} Y_1, & t = 1 \\ \alpha \cdot Y_t + (1-\alpha) \cdot S_{t-1}, & t > 1 \end{cases}$$



단순이동평균

이전 n개 데이터의 비가중 평균

동일한 가중치가 적용

가중이동평균

이전 n개 데이터의 가중 평균

최신 날짜의 가중치는 n, 두번째 최신 가중치 n-1

지수이동평균

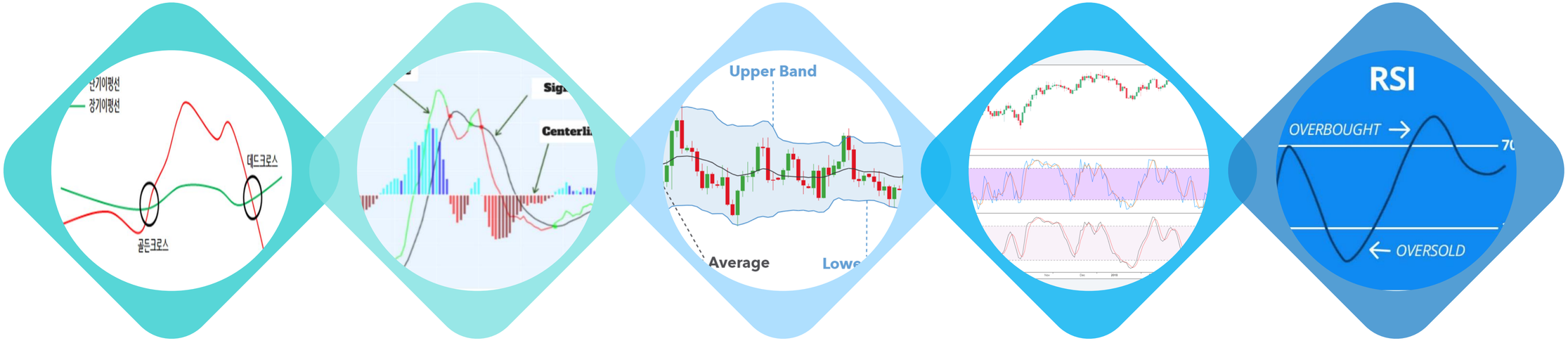
이전 n개 데이터의 가중평균

최근 데이터에 가중치를 높게 둠

02-기술적 분석 종류

기술적 분석 방법

5가지 기술적 분석 방법



골든크로스, 데드크로스

MACD

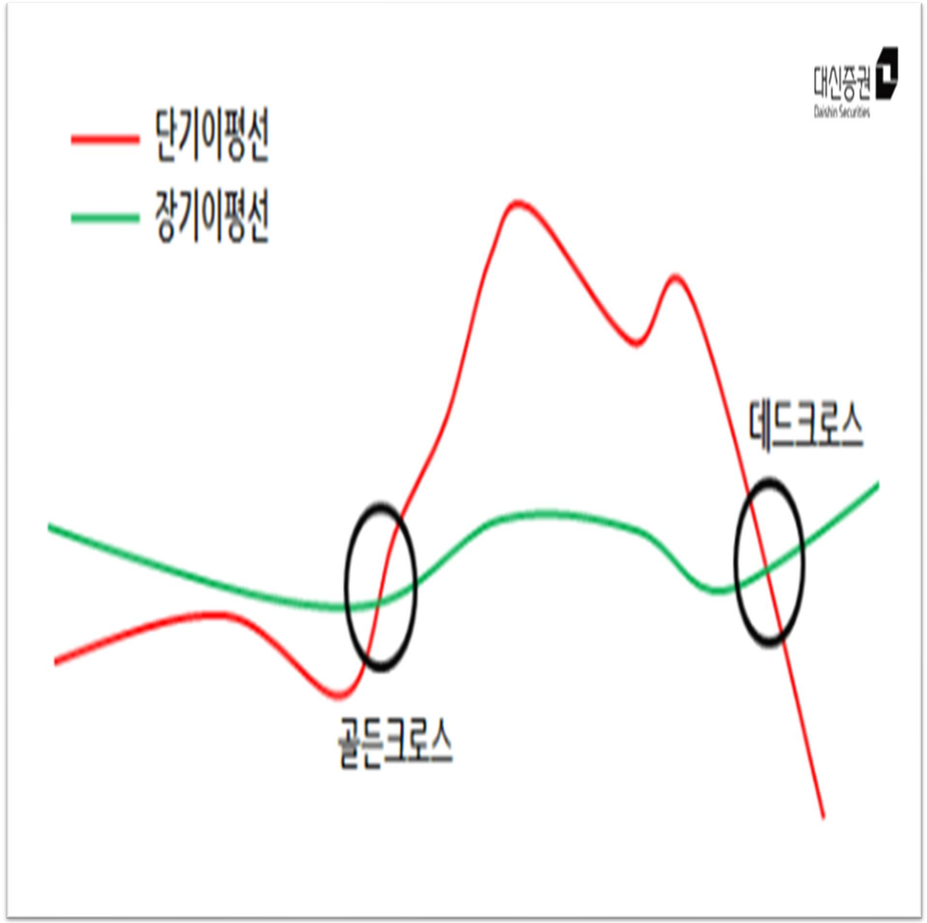
볼린저밴드

스토케스틱

RSI

골든, 데드크로스

골든크로스, 데드크로스의 간단한 설명



단순이동평균선

보통 5일 이동평균선과 20일 이동평균선을 사용

골든크로스

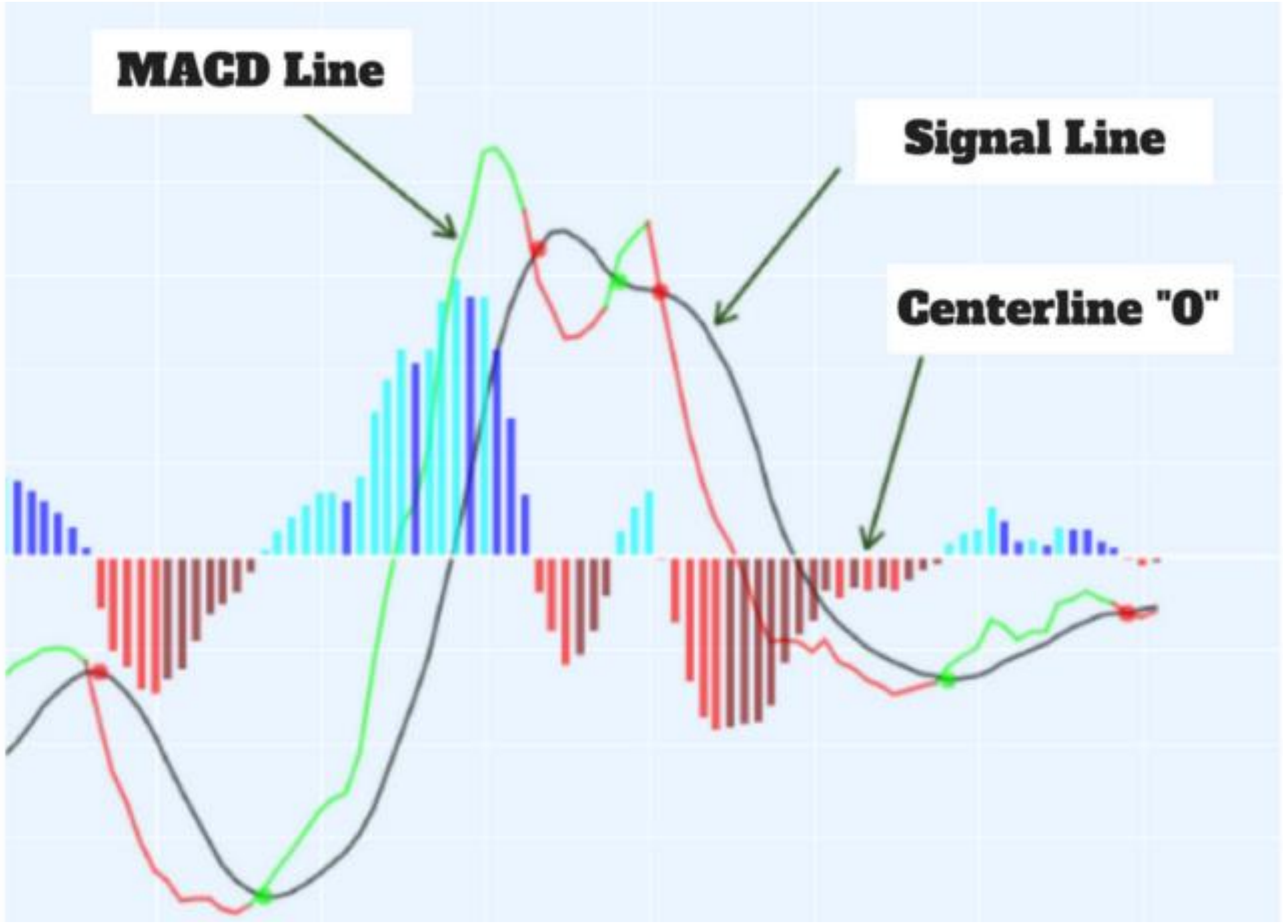
단기이동평균선이 장기이동평균선을 위로 뚫고 올라갈 때 매수

데드크로스

단기이동평균선이 장기이동평균선을 아래로 뚫고 내려갈 때 매도

MACD

MACD의 간단한 설명



지수이동평균선

단순이동평균선은 후행성이 강하여 주가의
추세를 늦게 반영하는 단점있음

MACD 오실레이터, 시그널

MACD 오실레이터: 단기 EMA(12) – 장기 EMA(26)
시그널: MACD 오실레이터의 9일 이평선

매매 전략

1. 교차전략
2. 과매수, 과매도 전략
3. 확산전략

볼린저밴드

볼린저밴드의 간단한 설명

중심선, 상한선, 하한선

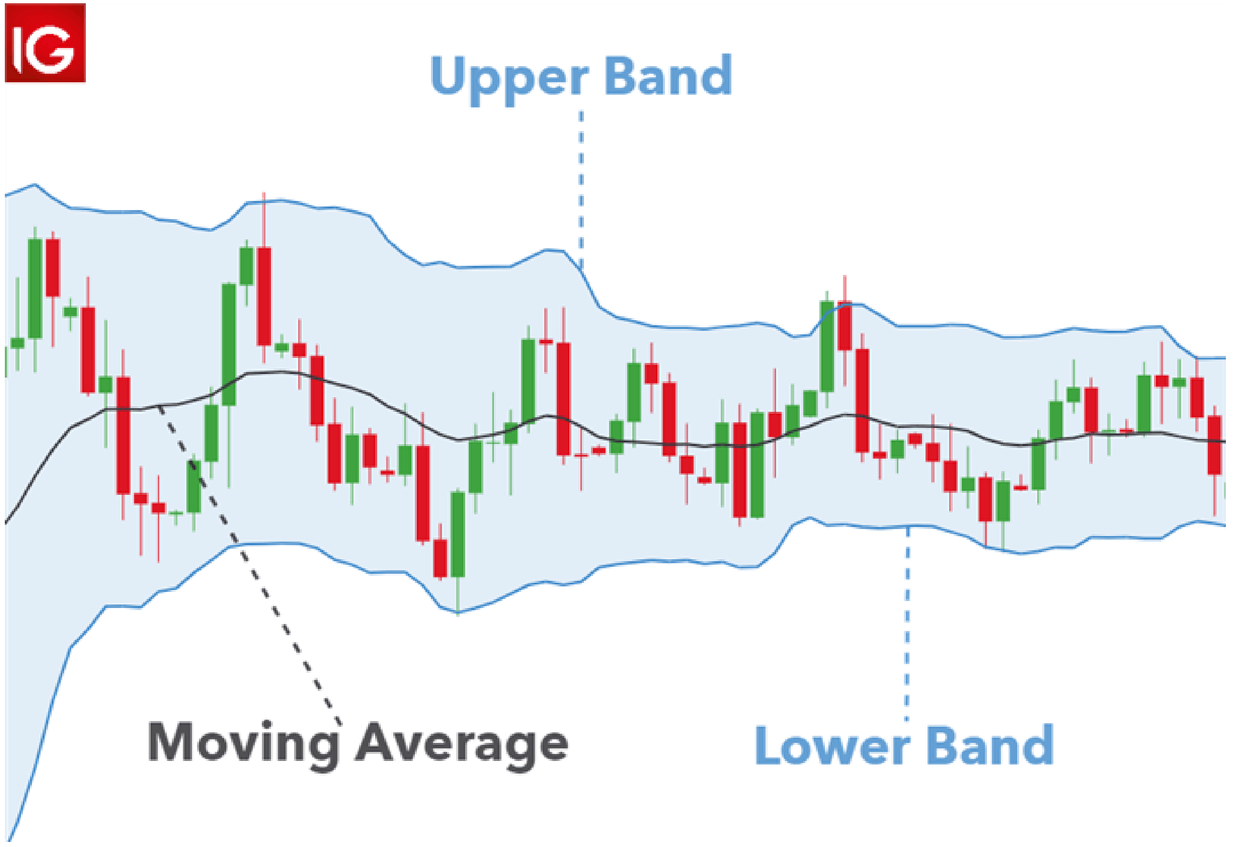
중심선: 20일 이동평균선
상한선: 중심선 + 20일간 표준편차 *2
하한선: 중심선 - 20일간 표준편차 *2

주가 상태 확인

주가의 높고, 낮음을 상대적으로 정의

매매 전략

1. 교차전략
2. 패턴 인식
3. 매매 신호



스토캐스틱

스토캐스틱의 간단한 설명

Fast 지표, Slow 지표

Fast %K: 100 * (**현가 – 특정기간 낮은 가격**) / (**특정기간 높은 가격 – 특정기간 낮은 가격**)

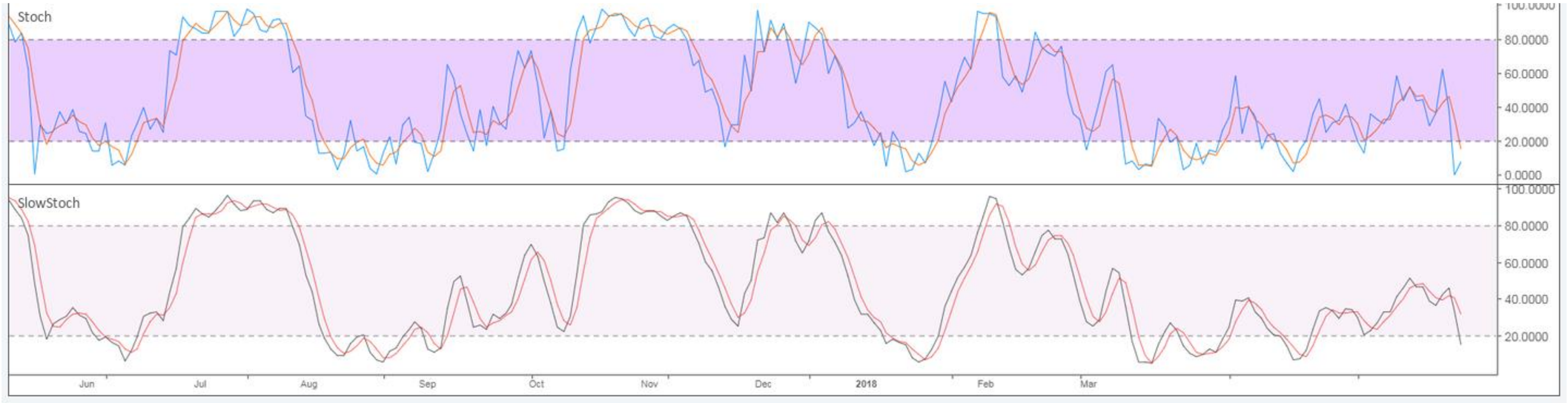
Fast %D: %K**의 이동평균**

Slow %K: Fast %K**의 이동평균**

Slow %D: %K**의 이동평균**

매매 전략

1. **매매 신호**
2. **교차전략**
3. **다이버젼스**



RSI

RSI의 간단한 설명

RSI 지수

AU: **이동평균 기간 중 등락이 +인 경우의 평균**
AD: **이동평균 기간 중 등락이 +인 경우의 평균**
RSI **지수**: 100 * AU / (AU + AD)

매매전략

1. **추세**
2. **패턴**
3. **지지와 저항**
4. **발산**
5. Failure Swing

03-코딩

사용 모듈

사용한 모듈 설명

```
1    from numpy.core.numeric import NaN
2    import requests
3    import csv
4    from bs4 import BeautifulSoup
5    import pandas as pd
6    import matplotlib.pyplot as plt
```

크롤링

Requests
BeautifulSoup
csv

데이터프레임

pandas

그래프

matplotlib

주가 데이터 불러오기

네이버 금융에서 주가 데이터 가져오기

```python
def get_data():
    url = "https://finance.naver.com/item/sise_day.nhn" # 네이버 금융 일별시세 사이트
    headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML

    title = ("날짜  종가    전일비  시가    고가    저가    거래량".split("\t"))
    name_code_dict = {"삼성전자": "005930", "SK하이닉스": "000660", "LG화학": "051910", "삼성전자우

    for code in name_code_dict:
        # 엑셀파일 만들기(.scv)
        filename = "{}.csv".format(code)
        f = open(filename, "w", encoding="utf-8-sig", newline="")
        writer = csv.writer(f)
        writer.writerow(title)

        # 마지막 페이지 찾기
        res = requests.get("{}?code={}".format(url, name_code_dict[code]), headers = headers)
        html = BeautifulSoup(res.text, 'lxml')
        pgrr = html.find('td', class_='pgRR')
        s = str(pgrr.a['href']).split('=')
        last_page = s[-1]

        # 각각의 페이지 BeautifulSoup 실행
        for page in range(1, int(last_page)+1):
            page_url = "{}?code={}&page={}".format(url, name_code_dict[code], page)
            res = requests.get(page_url, headers = headers)
            res.raise_for_status()
            soup = BeautifulSoup(res.text, "lxml")

            date_rows = soup.find("table", attrs={"class":"type2"}).find_all("tr")

            # 정보를 저장
            for row in date_rows:
                columns = row.find_all("td")
                if len(columns)<=1:
                    continue
                data = [column.get_text().strip() for column in columns]
                writer.writerow(data)

        print("{} 생성완료".format(code))

```

네이버 금융 사이트

네이버 금융 일일 시세 사이트에서 데이터 가져오기

엑셀파일로 저장

미리 저장하여 한번만 불러올 수 있게 만듦

시가총액기준 10순위 기업

삼성전자, SK하이닉스, LG화학, 삼성전자우, NAVER,
삼성바이오로직스, 카카오, 현대차, 삼성SDI, 셀트리온

데이터 변환

계산 가능한 숫자형태로 변환

```
48  def make_df(name):
49
50      df = pd.read_csv("{}.csv".format(name), index_col=0)
51      df = df[:252]
52      df = df.loc[::-1]
53
54      lst = ["종가", "전일비", "시가", "고가", "저가", "거래량"]
55      for i in lst:
56          smt = []
57          for j in df[i]:
58              b = j[-11:-8] + j[-7:-4] + j[-3:] # , 앞부분과 ,뒷부분으로 나누어 합쳐줌
59              b = int(b) # 숫자로 변환
60              smt.append(b) # 리스트에 저장
61          smt = pd.Series(smt)
62          smt.index = df.index
63          df[i] = smt
64
65      return df
```

날짜 순서 변경

데이터 가져올 때 최근 데이터가 먼저 오게 됨.
그래프 및 계산을 위해 최근 데이터를 뒤로 보냄.

문자형, 숫자형 변경

데이터를 가져올 때 문자형으로 가지고 옴.
계산을 위해 숫자형으로 변경
Split함수로도 가능

데이터프레임에 저장

숫자형으로 바꾼 데이터를 데이터프레임에 다시 저장

함수 설정

지표를 구하는 함수

```
67  def get_moving_average(x, w):
68      x = pd.Series(x)
69      ma = x.rolling(w).mean()
70      return ma
71
72  def get_exponential_moving_average(x, w):
73      x = pd.Series(x)
74      ema = x.ewm(span=w).mean()
75      return ema
76
77  def get_bollinger_band(x, w=20, k=2): # 2 표준편차 안에 있을 확률 95%
78      x = pd.Series(x)
79      mbb = x.rolling(w).mean()
80      ubb = mbb + k * x.rolling(w).std()
81      lbb = mbb - k * x.rolling(w).std()
82      return mbb, ubb, lbb
83
```

단순 이동평균선

Pandas의 rolling을 이용하여 계산

지수 이동평균선

Pandas의 ewm을 이용하여 계산

볼린저밴드

Pandas의 rolling을 이용하여 계산

함수 설정

지표를 구하는 함수

```
84  def get_stochastic(x, n=15, m=5, t=3):
85      x = pd.Series(x)
86
87      ndays_high = x.rolling(window=n, min_periods=1).max() # n일 중 최고가
88      ndays_low = x.rolling(window=n, min_periods=1).min() # n일 중 최저가
89
90      Fast_k = ((x - ndays_low) / (ndays_high - ndays_low))*100 # Fast%K 구하기
91      Fast_D = Fast_k.ewm(span=m).mean() # Fast%D 구하기 = Slow%K와 같음 지수이동평균으로 계산함.
92      Slow_D = Fast_D.ewm(span=t).mean() # Slow%D 구하기
93
94      # dataframe에 컬럼 추가
95      # df = df.assign(Fast_k=Fast_k, Fast_D=Fast_D, Slow_D=Slow_D).dropna()
96
97      return Fast_k, Fast_D, Slow_D
98
99  def get_RSI(x, w=14):
100
101     fluct = [x["종가"][i] - x["종가"][i-1] for i in range(len(x)) ]
102     fluct = pd.Series(fluct)
103     fluct[0] = NaN # 첫날은 전날이 없기 때문에
104     au = fluct.apply(lambda x: x if x>0 else 0)
105     ad = fluct.apply(lambda x: x*(-1) if x<0 else 0)
106     AU = get_moving_average(au, w)
107     AD = get_moving_average(ad, w)
108     RSI = 100*(AU/ (AU+AD))
109     RSI.index = x.index
110     return RSI
111
```

스토캐스틱

지수이동평균으로 계산.

RSI

등락을 +, -로 구분하여 평균 계싼

RSI 계싼

골든, 데드크로스 실행 함수

골든크로스, 데드크로스 실행

```python
def Gold_Dead_Cross(name):
    df = make_df(name)
    returns = (df["종가"][-1]-df["종가"][0])/df["종가"][0] # 수익률

    # 이동평균선 제작
    ma5 = get_moving_average(df["종가"],5) # 5평선
    ma20 = get_moving_average(df["종가"],20) # 20평선
    ma60 = get_moving_average(df["종가"],60) # 60평선

    # 데이터프레임에 이동평균선 추가
    df["5평선"] = ma5
    df["20평선"] = ma20
    df["60평선"] = ma60
    sub = ma5 - ma20
    df["sub"]=sub

    pres, yest, sums, num_sts = 0, 0, 0, 0

    for i in df.index:
        pres = df["sub"][i] # 오늘자
        if (pres > 0) and (yest < 0): # 매수타이밍
            if num_sts == 0:
                sums -=df["종가"][i]
                num_sts +=1
        if  (pres < 0) and (yest > 0): # 매도타이밍
            if num_sts == 1:
                sums +=df["종가"][i]
                num_sts -=1
        yest = df["sub"][i] # 이제 어제자

    print("{}의 시장수익률은 {}% 입니다.".format(name, round(returns, 4) * 100))
    print("{}의 골든, 데드 크로스 수익률은 {}% 입니다.".format(name, round(sums/df["종가"][0], 4) * 100))

    df["종가"].plot(x=df, y="PRICE", c="red")
    ma5.plot(x=df, y="PRICE",c="blue")
    ma20.plot(x=df, y="PRICE",c="green")
    ma60.plot(x=df, y="PRICE",c="black")
    plt.show()

```

이동평균선

단순이동평균선 함수를 이용하여 5일, 20일, 60일 이동평균선 제작

교차 전략

단기 이동평균선과 장기 이동평균선이 교차하는 지점을 찾기 위해 오실레이터를 구함
금일 오실레이터와 작일 오실레이터를 비교.

MACD 실행 함수

MACD 실행

```python
def MACD(name):
    df = make_df(name)
    returns = (df["종가"][-1]-df["종가"][0])/df["종가"][0] # 수익률

    # 지수이동평균선 제작 1970년대 MACD 만들때 미국 주 6일이라 12일은 2주
    ema12 = get_exponential_moving_average(df["종가"], 12) # 12 지수이평선 2주
    ema26 = get_exponential_moving_average(df["종가"], 26) # 26 지수이평선 1달

    # 데이터프레임에 이동평균선 추가
    df["12 지수이평선"] = ema12
    df["26 지수이평선"] = ema26
    macd = ema12 - ema26
    df["sub"]=macd
    signal = get_exponential_moving_average(df["sub"], 9)
    df["Signal"] = signal
    oscillator = macd - signal
    df["Oscillator"] = oscillator

    pres, yest, sums, num_sts = 0, 0, 0, 0

    for i in df.index:
        pres = df["Oscillator"][i] # 오늘자
        if (pres > 0) and (yest < 0): # 매수타이밍 MACD가 시그널 선을 아래서 위로 올라갈 때
            if num_sts == 0:
                sums -=df["종가"][i]
                num_sts +=1
        if  (pres < 0) and (yest > 0): # 매도타이밍
            if num_sts == 1:
                sums +=df["종가"][i]
                num_sts -=1
        yest = df["Oscillator"][i] # 이제 어제자

    print("{}의 시장수익률은 {}% 입니다.".format(name, round(returns, 4) * 100))
    print("{}의 MACD 수익률은 {}% 입니다.".format(name, round(sums/df["종가"][0], 4) * 100))

    macd.plot(x=df, y="MACD",c="red")
    signal.plot(x=df, y="Signal",c="blue")
    oscillator.plot(x=df, y="Oscillator",c="green")
    plt.show()

```

이동평균선

지수이동평균선 함수를 이용하여 12일, 26일
지수이동평균선 제작
장기, 단기 이평선 차이의 9일 지수이동평균선 제작

오실레이터

MACD와 Signal의 차로 오실레이터 계산

교차전략

오실레이터를 이용하여 MACD와 Signal의
교차를 이용하여 계산.

볼린저밴드 실행 함수

볼린저밴드 실행

```
191  def Bollinger(name): # 매수 매도 타이밍 잡기 어려움.
192      df = make_df(name)
193      returns = (df["종가"][-1]-df["종가"][0])/df["종가"][0] # 수익률
194
195
196      # 볼린저벤드 제작
197      mbb, ubb, lbb = get_bollinger_band(df["종가"])
198
199      # 데이터프레임에 이동평균선 추가
200      df["MBB"] = mbb
201      df["UBB"] = ubb
202      df["LBB"] = lbb
203
204      sums, num_sts = 0, 0
205
206      for i in df.index:
207          if (df["종가"][i] < df["LBB"][i]): # 매수타이밍 가격이 lbb 아래로 내려갔을 때
208              if num_sts == 0:
209                  sums -=df["종가"][i]
210                  num_sts +=1
211          if  (df["종가"][i] > df["UBB"][i]): # 매도타이밍
212              if num_sts == 1:
213                  sums +=df["종가"][i]
214                  num_sts -=1
215
216      print("{}의 시장수익률은 {}% 입니다.".format(name, round(returns, 4) * 100))
217      print("{}의 볼린저밴드 수익률은 {}% 입니다.".format(name, round(sums/df["종가"][0], 4) * 100))
218
219      df["price"].plot(x=df, y="PRICE",c="red")
220      mbb.plot(x=df, y="MBB",c="blue")
221      ubb.plot(x=df, y="UBB",c="green")
222      lbb.plot(x=df, y="LBB",c="black")
223      plt.show()
224
```

볼린저밴드

볼린저밴드 함수를 이용하여 중심선, 상한선, 하한선을 계산.

매매신호

하한선 아래로 내려가면 매수
상한선 위로 올라가면 매도

스토캐스틱 실행 함수

스토캐스틱 실행

```python
def Stochastic(name): # 오실레이터가 20이하로 매수, 80이상 매도, 또는 k선이 d위로 오르면 매수 하향이면 매도
    df = make_df(name)
    returns = (df["종가"][-1]-df["종가"][0])/df["종가"][0] # 수익률


    # 스토캐스틱 제작
    Fast_k, Fast_D, Slow_D = get_stochastic(df["종가"])

    # 데이터프레임에 스토캐스틱 추가
    df = df.assign(Fast_k=Fast_k, Fast_D=Fast_D, Slow_D=Slow_D).dropna()
    Fast_sub = Fast_k - Fast_D
    Slow_sub = Fast_D - Slow_D
    df["Fast_Sub"]= Fast_sub
    df["Slow_Sub"]= Slow_sub

    pres, yest, sums, num_sts = 0, 0, 0, 0

    for i in df.index:
        pres = df["Slow_Sub"][i] # 오늘자
        if (pres > 0) and (yest < 0): # 매수타이밍
            if num_sts == 0:
                sums -=df["종가"][i]
                num_sts +=1
        if  (pres < 0) and (yest > 0): # 매도타이밍
            if num_sts == 1:
                sums +=df["종가"][i]
                num_sts -=1
        yest = df["Slow_Sub"][i] # 이제 어제자

    print("{}의 시장수익률은 {}% 입니다.".format(name, round(returns, 4) * 100))
    print("{}의 스토캐스틱 수익률은 {}% 입니다.".format(name, round(sums/df["종가"][0], 4) * 100))

    Fast_k.plot(x=df, y="PRICE",c="red")
    Fast_D.plot(x=df, y="MBB",c="blue")
    Slow_D.plot(x=df, y="UBB",c="green")
    plt.axhline(80)
    plt.axhline(20)

    # Fast_sub.plot(x=df, y="PRICE",c="Black")
    # Slow_sub.plot(x=df, y="PRICE",c="Black")


    plt.show()
```

스토캐스틱

스토캐스틱 함수를 이용하여 스토캐스틱을 제작

교차전략

%K와 %D의 교차하는 지점을 찾기 위해
오실레이터를 구함
금일 오실레이터와 작일 오실레이터를 비교.

RSI 실행 함수

RSI 실행

```
269  def RSI(name):
270      df = make_df(name)
271      returns = (df["종가"][-1]-df["종가"][0])/df["종가"][0] # 수익률
272
273      RSI = get_RSI(df)
274      df["RSI"] = RSI
275      pres, yest, sums, num_sts = 0, 0, 0, 0
276
277      for i in df.index:
278          pres = df["RSI"][i]
279          if (yest < 30 and pres > 30): # 매수타이밍
280              if num_sts == 0:
281                  sums -=df["종가"][i]
282                  num_sts +=1
283          if  (yest>70 and pres<70): # 매도타이밍
284              if num_sts == 1:
285                  sums +=df["종가"][i]
286                  num_sts -=1
287          yest = df["RSI"][i]
288
289      print("{}의 시장수익률은 {}% 입니다.".format(name, round(returns, 4) * 100))
290      print("{}의 RSI 수익률은 {}% 입니다.".format(name, round(sums/df["종가"][0], 4) * 100))
291
292      # RSI.plot(x=df, y="PRICE",c="red")
293
294      # plt.show()
295
```

RSI

RSI 함수를 이용하여 RSI제작

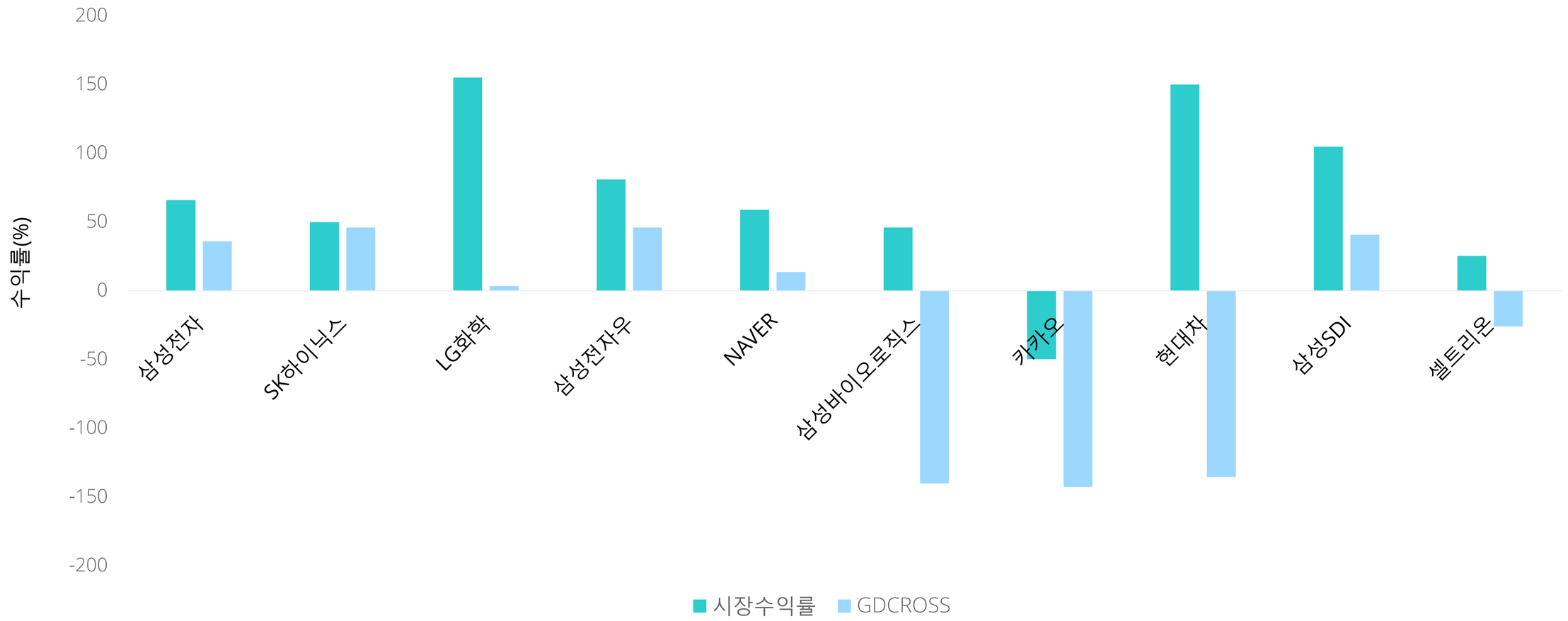
매매 신호

RSI가 70이상에서 이하로 떨어질 경우 매도
RSI가 30이하에서 이상으로 오를 경우 매수

04-결론

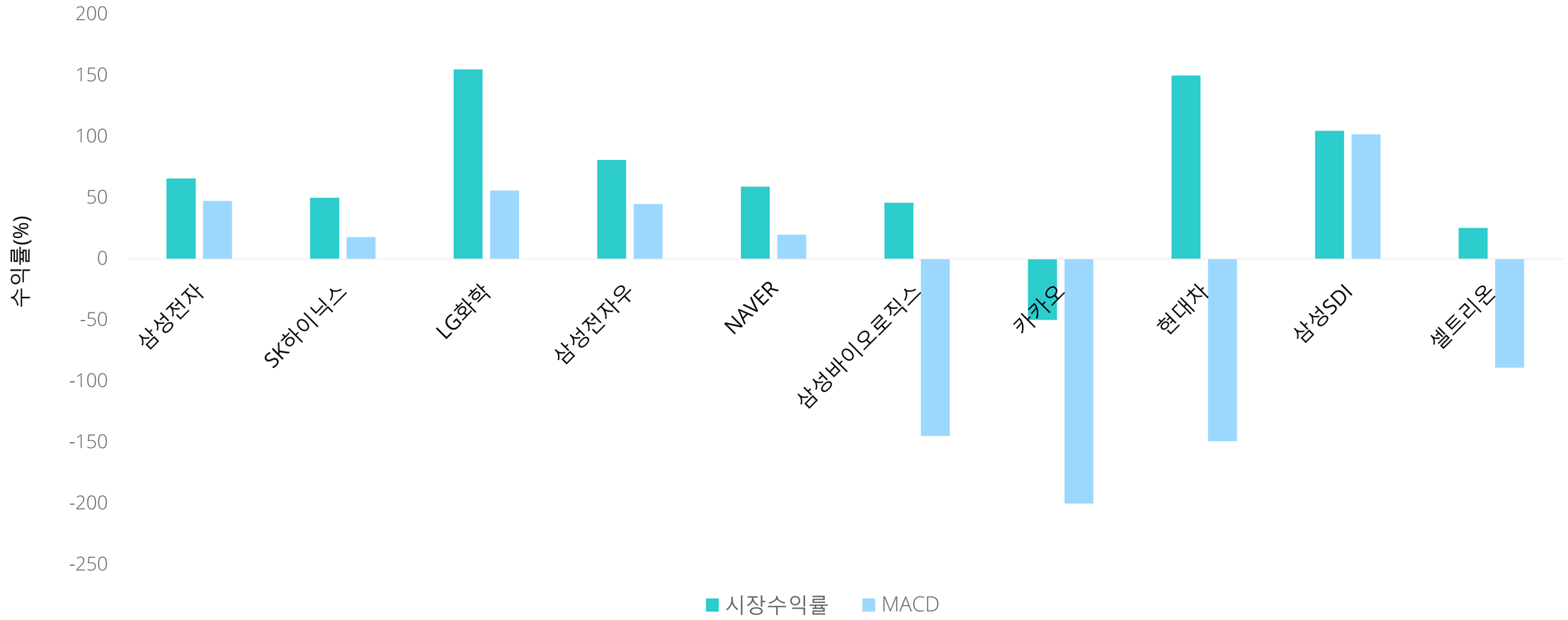
골든, 데드 크로스 수익률

골든크로스, 데드크로스 수익률 비교



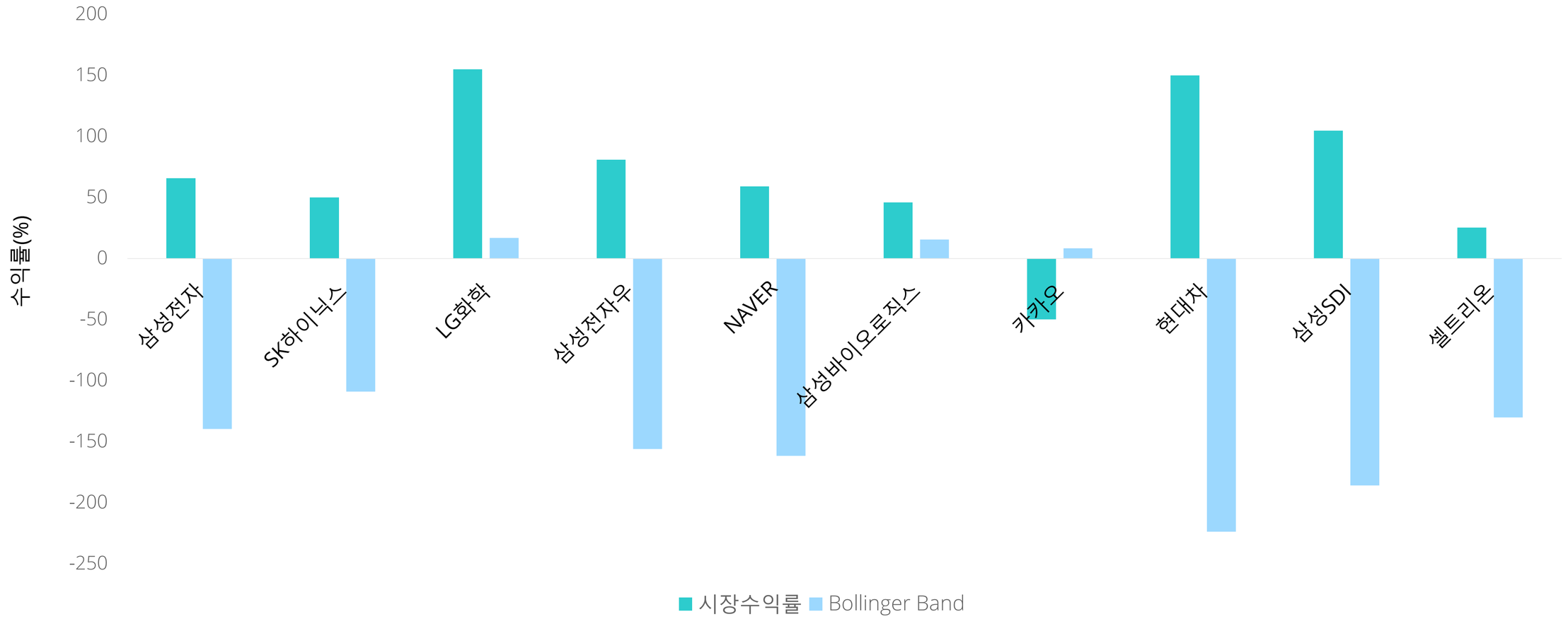
MACD 수익률

MACD 수익률 비교



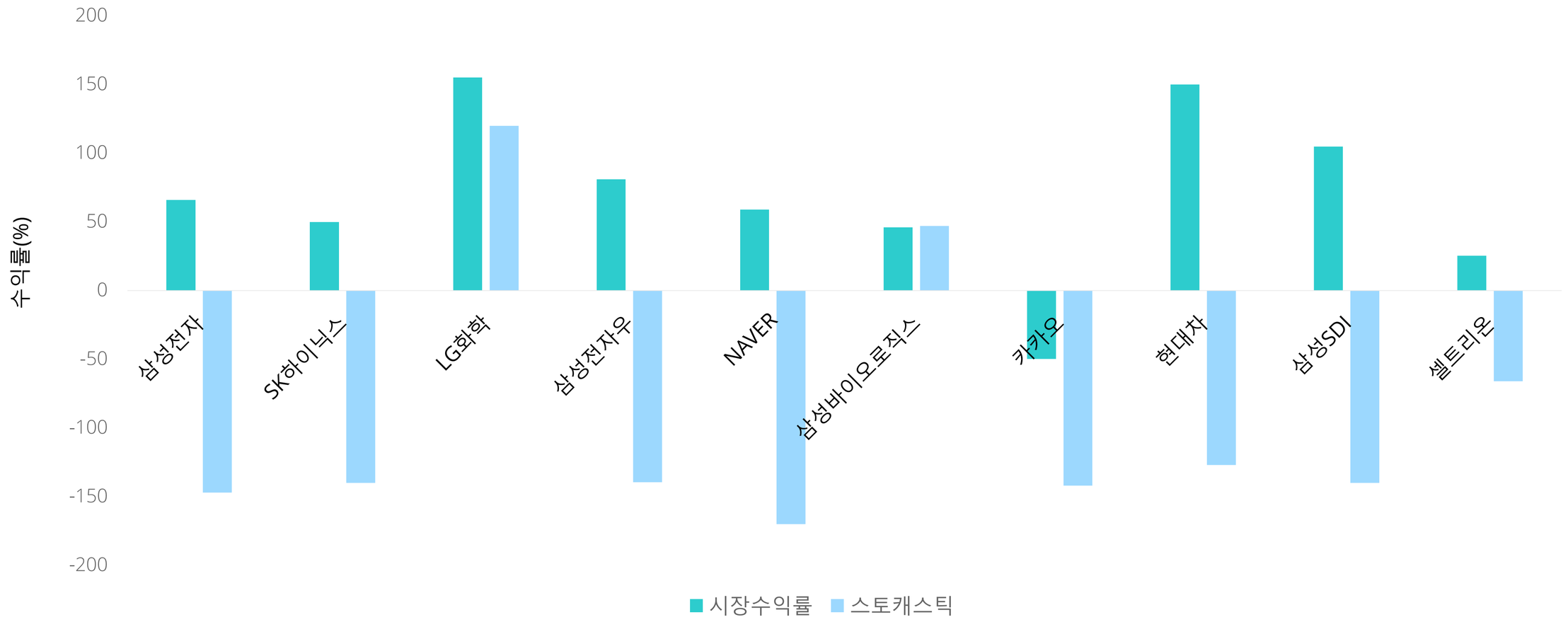
볼린저밴드 수익률

볼린저밴드 수익률 비교



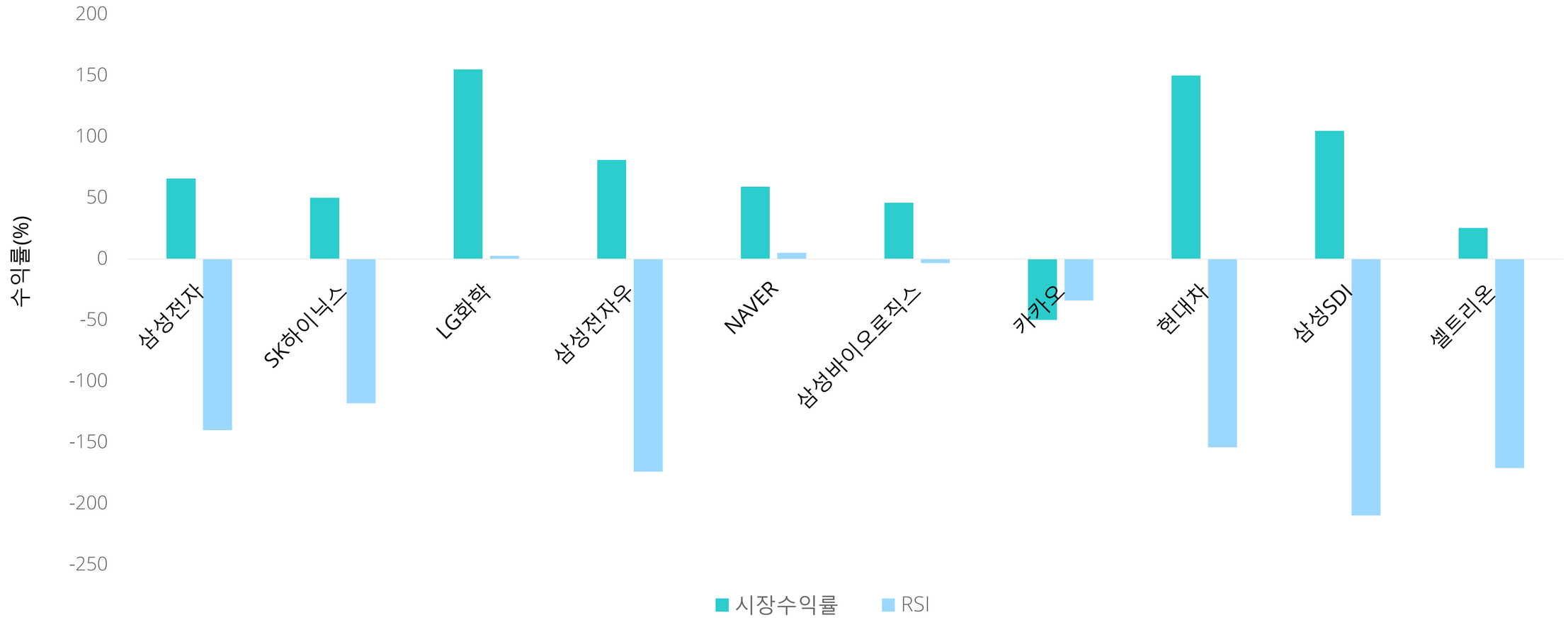
스토캐스틱 수익률

스토캐스틱 수익률 비교



RSI 수익률

RSI 수익률 비교



결과 분석

결과 도출 후 분석

시장수익률과 비교

전체적으로 시장 수익률 대비 낮은 수익률을 보임.

효율적 시장 가설

기술적 분석으로 수익을 낼 수 없기 때문에 약형 효율적 시장으로 판단 가능.

기술적 분석 지표

기술적 분석 지표를 맹목적으로 신뢰해서는 안됨.

지표를 보조지표로써 이용하는 것을 추천.

한계점

문제점과 한계점

매매전략의 한계

다양한 매매 전략이 있지만, 한가지의 매매 전략을 사용하여 수익률을 비교함.

다양한 기술적 분석을 사용하지 않음

5가지 기술적 분석 방법 외 추세를 분석하는 방법, 패턴을 보는 방법 등 다양한 방법이 있지만, 확인하지 못함.

기업의 종류 및 개수

시가총액 기준 10개의 주식만을 이용하여 비교해 봄.

더 많은 기업을 확인해 볼 필요가 있음.

코로나 19의 특수성

최근 1년의 자료를 이용하여 계산을 하여 코로나 19로 인한 특수성이 포함되어 있음.

코로나 19 이전의 데이터를 이용하여 비교해 볼 필요가 있음.

Thank you!

FR 32기 나종진