배당과 매매차익을 고려한 주식상품의 재발견

SeeStock
강동화 김보배 임형진 정희도

SeeStock

팔로잉 ▾ ...

팀원 **4** 평균 수익률 **18.98%** 홀딩 기간 **2년**

꺼진 주식도 다시 보자

#필터링 포트폴리오 백테스팅
#정기예금만큼의 안정성
#배당과 매매차익을 포함한 그 이상의 수익성
#총배당금 예측을 통한 실제적인 포트폴리오 수익률
#주식에 대한 새로운 가치를 추구

KRX,DACON 님 외 8증권사가 팔로우합니다.

NEW 배당 필터링

기존
고배당50,배당성장50지수에
배당 안정성을 추가한 필터링

기대수익률>요구수익률

DDM모형을 통한 기대수익률,
WACC 요구수익률 필터링

BACKTESTING

BUY&HOLD전략,
적정 기간을 홀딩한
홀딩 수익률 백테스팅

INTERACTIVE PLOT

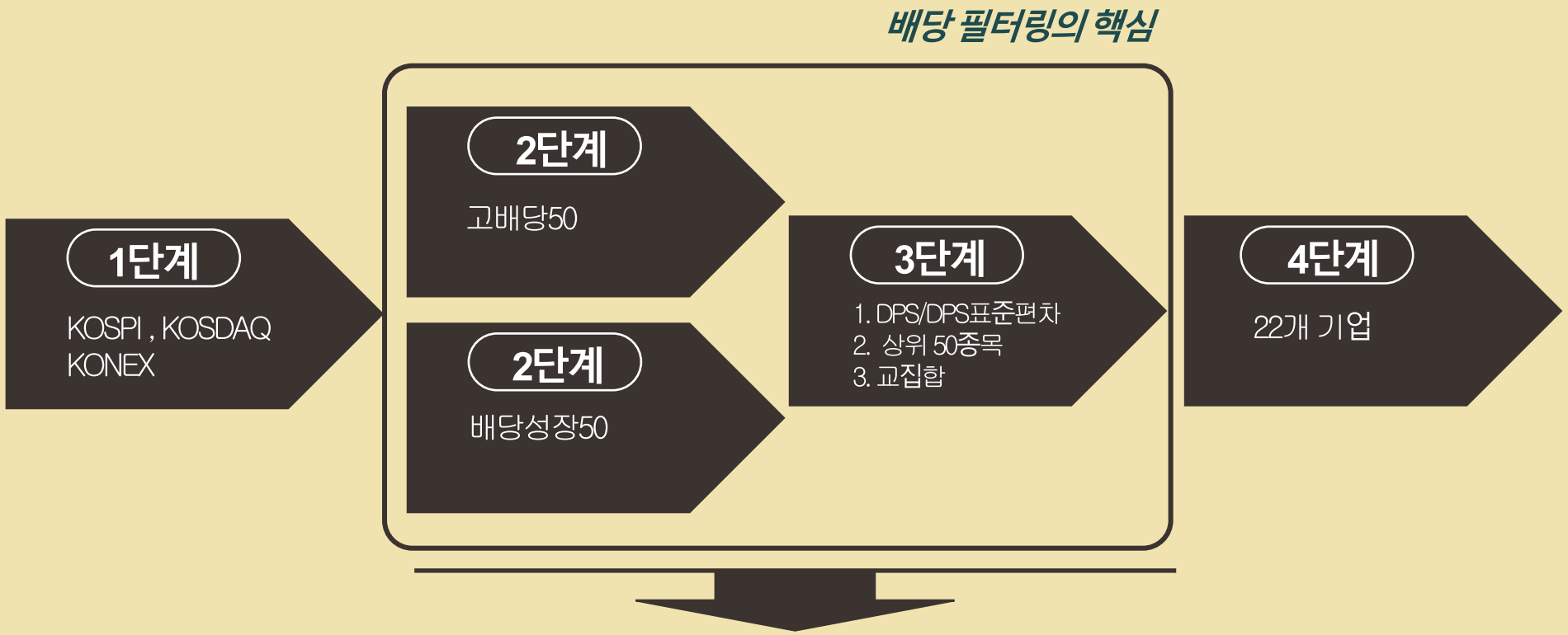
승률, 손익비를 고려한
트레이딩 우위 ODDS,
전략성과 INTERATIVE PLOT

총배당금 예측

배당관련 선행연구를 바탕으로
한 다중선형회귀분석을 통한
총 배당금 예측

필터 NO.1,2 : NEW 배당 필터링

01. 배당 필터링 개괄



KRX의 배당지수 선정 기준에 배당 안정성 필터를 넣다!

필터 NO.1 : KRX 배당지수 필터링

01-1. 배당지수 선정 기준

〈KOSPI〉 고배당50지수 심사기준

(1) 시장규모기준
: 일평균시가총액 순위 상위 80% 이내
(일평균시총 800억원 미만 종목 제외)
(2) 유동성기준
: 일평균거래대금 순위 상위 80% 이내
(3) 배당기준
: 최근 3사업연도 연속 배당 실시 및 평균 배당성향 90% 미만
(4) 재무기준
: 최근 3사업연도 연속 당기순이익 실현

출처 : KRX

〈KOSPI〉 배당성장50지수 심사기준

(1) 시장규모기준
: 일평균 시총 순위 상위 50%이내
(2) 유동성기준
: 일평균거래대금 순위 상위 70% 이내
(3) 배당기준
: 최근 7사업연도 연속 배당 실시 및 최근 5사업연도 평균 배당성향 60% 미만
(4) 배당성장기준
: 최근 사업연도 DPS 최근 7사업연도 평균 DPS 대비 증가
(5) 재무기준
: 최근 5사업연도 연속 당기순이익 실현

출처 : KRX

필터 NO.2 : NEW 배당 안정성 필터링

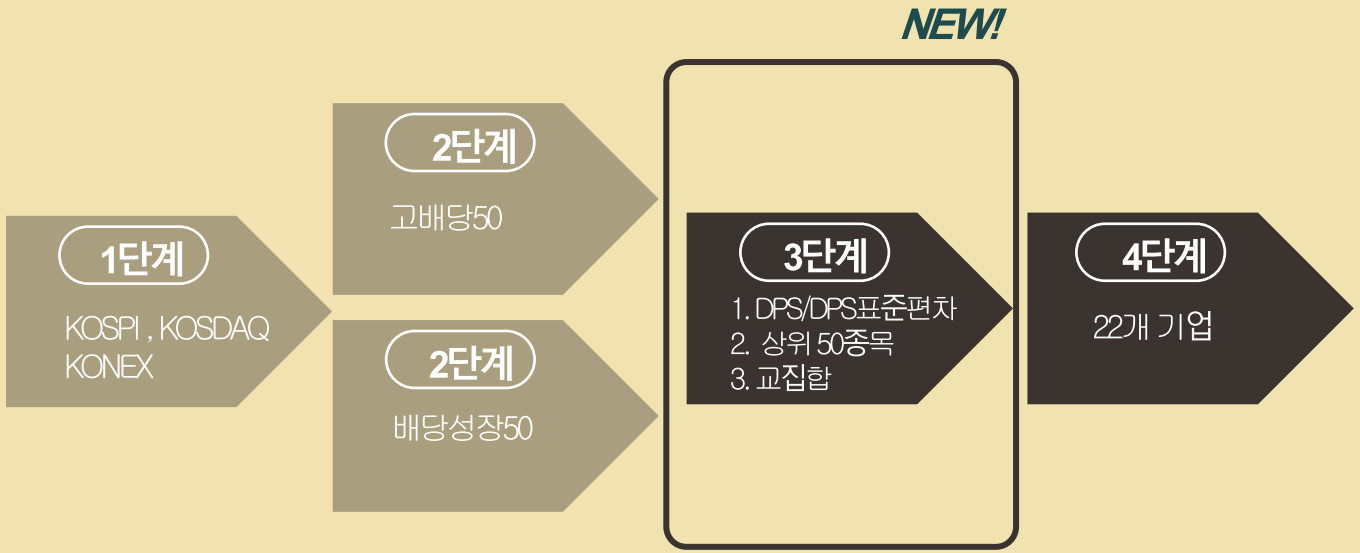
02. 배당안정성 기준

《배당안정성》

DPS/DPS표준편차

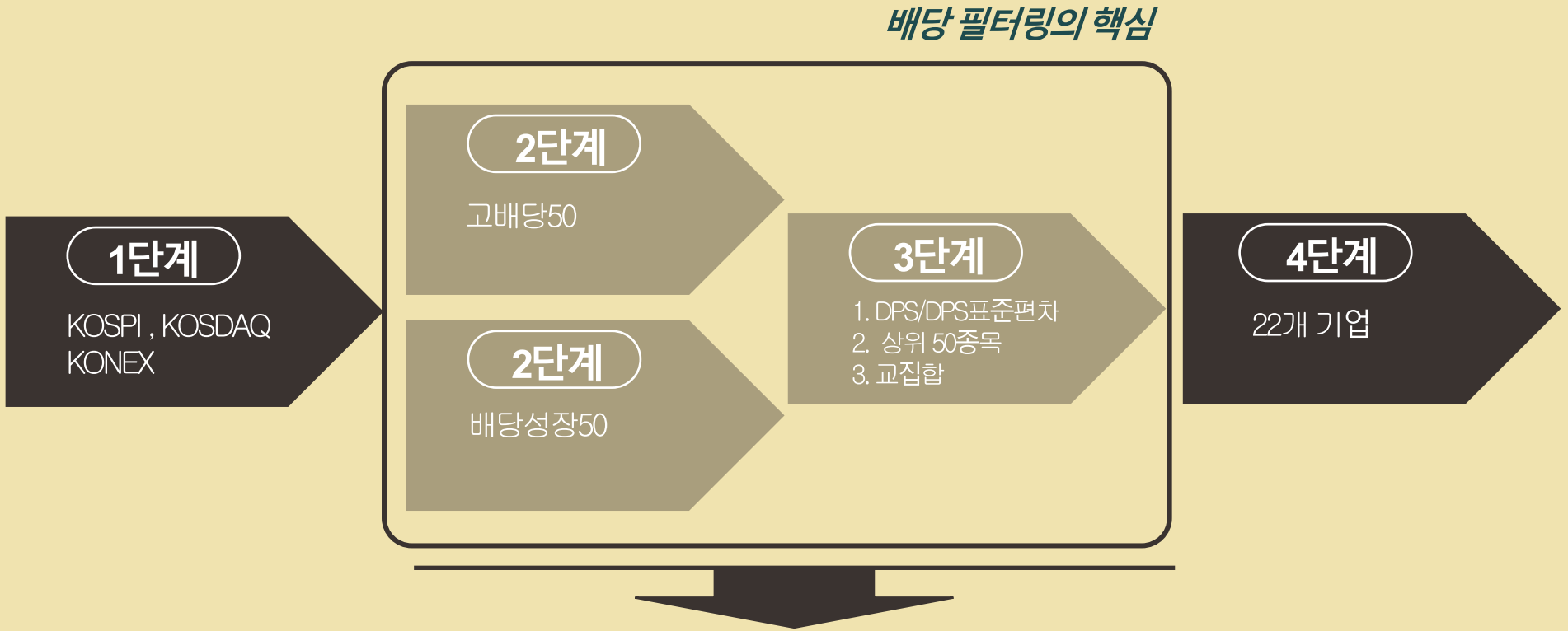
주당 배당금(DPS, Divedend per Shares)을 표준편차로 나누어 이를 필터링으로 사용하는 이유는,
배당금의 변동성이 큰 기업에 대한 패널티를 적용하여 배당을 안정적으로 지급하는 기업을 추출하기 위함이다.

출처 : KRX



필터 NO.2 : NEW 배당 필터링

02-1. 결과



전체 지수에서 22개의 기업을 찾다!!

필터 NO.3 : 기대수익률>요구수익률 필터

03. 요구수익률 필터링

22개의 기업에서
요구수익률을 만족시키는
21개의 기업으로 필터링

STEP 3 D+g > WACC 필터링

21개 기업 최종 모델 포트폴리오 선정

$$WACC = Re\left(\frac{E}{V}\right) + Rd(1-t)\left(\frac{D}{V}\right)$$

STEP 2 가중평균자본비용 WACC 구하기

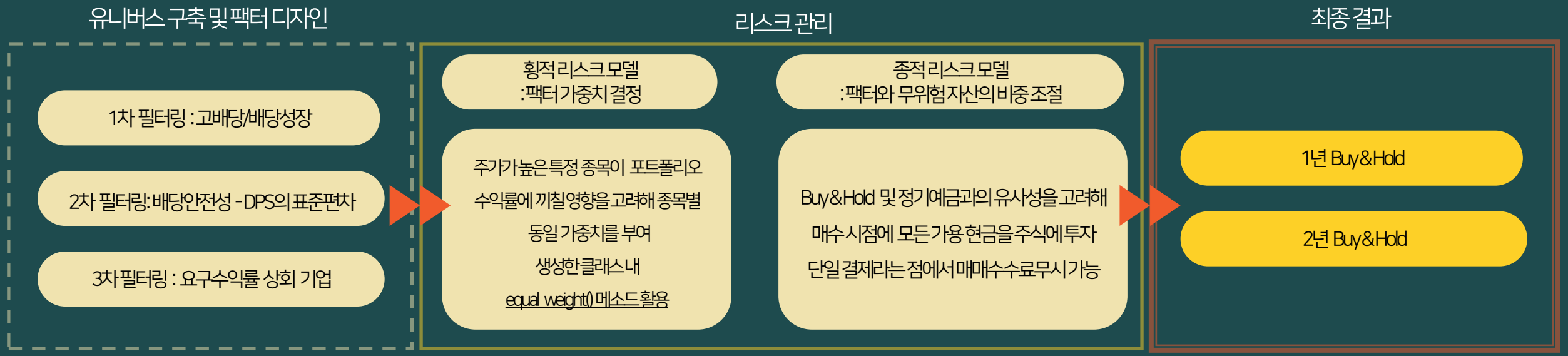
1) Re(주주자본비용) : CAPM 모델을 통해 계산 Re = Rf + Beta(Rm-rf)
2) Rf : 국고채 3년물의 3년 평균
3) Rm : KOSPI의 5개년 평균 수익률
4) Beta : 5개년 월별 주식의 시장대비 민감도
5) Rd(이자비용) : 기업이 자금 차입을 위해 지불하는 이자율

STEP 1 고든의 배당성장모형(DDM)으로 배당수익률(d)+매출액성장률(g)을 구하다!

$$P_0 = \frac{D_0(1+g)}{(k_e - g)} = \frac{D_1}{(k_e - g)}$$

요구수익률(ke)
= 배당수익률(d=D1/P0) + 성장률(g)

Portfolio Modeling



필터링 포트폴리오 백테스팅

1) KRX API로 포트폴리오 기업 주식 정보 가져오기

```
# KRX API 인 pykrx로 필터링된 기업리스트 주식 정보 가져오기
stock_code = portfolio_code
res = pd.DataFrame()
for ticker in stock_code:
    df = stock.get_market_ohlcv_by_date(fromdate=start_sp, todate=end_sp, ticker=ticker)
    df = df.assign(종목코드=ticker, 종목명=stock.get_market_ticker_name(ticker))
    res = pd.concat([res, df], axis=0)
    time.sleep(1)
res = res.reset_index()
res
```

2-1) 1년 배당금 데이터가져오기

```
# portfolio_df['종목명']으로 된 csv파일을 각각 파일 이름으로 불러오기
import os
folders = os.listdir('./KRX')
folders

for i in folders:
    globals()[i.split('.')[0]] = pd.read_csv(f'./KRX/{i}',encoding='CP949')
    globals()[i.split('.')[0]].drop(columns=['종가','주당배당금','대비','등락률','EPS','PER','선행 EPS','선행 PER','BPS','PBR'],axis=1,inplace=True)
    globals()[i.split('.')[0]].set_index('일자',inplace=True)

div_df = pd.concat([LF,LS,NI스틸,SK텔레콤,고려산업,금강철강,노루페인트,대동,대상홀딩스,대정화금,대한제당,대한제분,삼천리,세방전지,유니드,유한양행,태경산업,태경케미컬,한섬,한화,현대백화점],axis=1)
div_df.columns = ['LF','LS','NI스틸','SK텔레콤','고려산업','금강철강','노루페인트','대동','대상홀딩스','대정화금','대한제당','대한제분','삼천리','세방전지','유니드','유한양행','태경산업','태경케미컬','한섬','한화','
div_df.dropna(inplace=True)
div_df = div_df.sort_index()
div_df.index = div_df.index.str.replace('/','-')
```

필터링 포트폴리오 백테스팅

2-2) 2년 배당금 데이터가져오기

```
# portfolio_df['종목명']으로 된 csv파일을 각각 파일 이름으로 불러오기
folders = os.listdir('./KRX/')
for i in folders:
    globals()[i.split('.')[0]] = pd.read_csv(f'./KRX/{i}',encoding='CP949')
    globals()[i.split('.')[0]].drop(columns=['배당수익률','대비','등락률','EPS','PER','선행 EPS','선행 PER','BPS','PBR'],axis=1,inplace=True)
    globals()[i.split('.')[0]].set_index('일자',inplace=True)

div_df2 = pd.concat([LF,LS,NI스틸,SK텔레콤,고려산업,금강철강,노루페인트,대동,대상홀딩스,대정화금,대한제당,대한제분,삼천리,세방전지,유니드,유한양행,태경산업,태경케미컬,한섬,한화,현대백화점],axis=1)
div_df2.columns = [['LF','LF','LS','LS','NI스틸','NI스틸','SK텔레콤','SK텔레콤','고려산업','고려산업','금강철강','금강철강','노루페인트','노루페인트','대동','대동','대상홀딩스','대상홀딩스','대정화금','대정화금',
div_df2.dropna(inplace=True)
div_df2 = div_df2.sort_index()
div_df2.index = div_df2.index.str.replace('/','-')
div_df2 = div_df2[::250]
list = ('LF','LS','NI스틸','SK텔레콤','고려산업','금강철강','노루페인트','대동','대상홀딩스','대정화금','대한제당','대한제분','삼천리','세방전지','유니드','유한양행','태경산업','태경케미컬','한섬','한화','현대백화점')

for i in list :
    div_df2[i,'2년배당수익률'] = (div_df2[i,'주당배당금']+div_df2[i,'주당배당금'].shift(-1))/div_df2[i,'종가']
div_df2=div_df2.swaplevel(0,1, axis=1)['2년배당수익률']
div_df2=(div_df2*100).dropna()
```

일자	LF	LS	NI스틸	SK텔레콤	고려산업	금강철강	노루페인트	대동	대상홀딩스	대정화금	...	대한제분	삼천리	세방전지	유니드	유한양행	태경산업	태경케미컬
2010-12-20	2.503912	1.951220	3.125000	10.681818	5.882353	8.602151	7.621951	2.469136	7.621951	4.347826	...	4.406780	3.773585	1.592357	3.874539	1.208459	6.329114	6.849315
2011-12-21	1.777778	2.848101	5.617978	12.287582	7.032349	6.956522	11.725293	2.604167	7.177033	6.545455	...	4.529617	5.434783	1.508121	3.451493	1.711027	9.202454	8.797654
2012-12-21	2.523659	2.714441	5.813953	11.677019	3.592814	8.064516	8.978676	1.848429	4.231975	7.486631	...	4.181818	4.800000	1.472135	4.021448	1.453488	8.862629	6.593407
2013-12-27	2.420575	3.132832	5.420054	8.263736	4.803493	7.874016	7.812500	1.700680	2.857143	5.365854	...	2.884615	4.669261	1.181818	2.936242	1.486486	6.185567	5.617978
2015-01-07	3.169014	4.807692	4.405286	6.975881	3.888889	7.434944	4.175365	1.128205	1.624650	5.313496	...	3.020134	3.745318	1.608579	3.756708	1.963746	5.110733	5.504587
2016-01-11	3.960396	6.459948	3.976143	9.556650	3.865979	4.494382	4.012036	1.230425	1.823056	3.875969	...	2.601156	4.782147	1.724138	4.899777	1.264755	6.407767	4.958678

필터링 포트폴리오 백테스팅

3) 모델포트폴리오 클래스화

i) 바이앤 홀딩 수익률을 구하는 알고리즘
ii) 해당 종목에 동일 비중을 주는 알고리즘
iii) 매매차익과 배당금을 고려한 백테스팅 알고리즘을 클래스화

```
class ModelPortfolio:
    def __init__(self, prices, holding_period, df_div):
        self.holding_returns = self.get_holding_returns(prices,holding_period)
        self.cs_risk_weight = self.equal_weight(prices)
        self.port_rets = self.backtest(df_div,self.holding_returns,self.cs_risk_weight)

    def get_holding_returns(self,prices, holding_period):
        holding_returns = (prices.pct_change(periods=holding_period).shift(-holding_period).dropna(0)) *100
        return holding_returns

    def equal_weight(self,prices):
        #prices = prices.iloc[-holding_period:,:]
        weight = pd.DataFrame(index=prices.index,columns=prices.columns).fillna(1/len(prices.columns))
        return weight

    def backtest(self,df_div,holding_returns,weight):
        hold_rets = ((holding_returns*weight).dropna()).sum(axis=1)
        div = ((df_div*weight).dropna()).sum(axis=1)
        port_rets = (hold_rets + div).dropna()
        return port_rets
# price 가격 가져오는 함수
def get_price_df(path):
    df = pd.read_csv(path).dropna()
    df = df.drop(columns=['Unnamed: 0'])
    return df
```

필터링 포트폴리오 백테스팅

4) 주가 데이터와 배당금 데이터를 가지고 class 함수를 실행하는 알고리즘

```
def portfolio_returns(prices,holding_period,df_div):
        c1 = ModelPortfolio(prices, holding_period,df_div)
        holding_returns = c1.get_holding_returns(prices,holding_period)
        weight = c1.equal_weight(prices)
        port_rets = c1.backtest(df_div,holding_returns,weight)
        return port_rets
```

4-1) 1년 홀딩 수익률 계산

```
port_rets1 = portfolio_returns(prices=prices1,holding_period=250,df_div=div_df)
port_rets1.index = pd.to_datetime(port_rets1.index)
port_rets1
```

4-2) 2년 홀딩 수익률 계산

```
port_rets2 = portfolio_returns(prices=prices1,holding_period=500,df_div=div_df)
port_rets2.index = pd.to_datetime(port_rets2.index)
port_rets2
```

필터링 포트폴리오 백테스팅

5) 요약통계함수

```
# 손익비를 보여주는 요약통계 함수
def describe_winloss(df):
    _desc_df = df.describe()

    total = df.notna().sum()
    _win_df, _loss_df = df.applymap(lambda e: e if e>=0 else np.nan), df.applymap(lambda e: e if e<0 else np.nan)

    _desc_df = _desc_df.append(pd.Series(((df >= 0).sum() / total), name='win_rate'))
    _desc_df = _desc_df.append(pd.Series(_win_df.count(), name='win_count'))
    _desc_df = _desc_df.append(pd.Series(_win_df.mean(), name='win_return_mean'))
    _desc_df = _desc_df.append(pd.Series(_win_df.median(), name='win_return_median'))

    _desc_df = _desc_df.append(pd.Series(((df < 0).sum() / total), name='loss_rate'))
    _desc_df = _desc_df.append(pd.Series(_loss_df.count(), name='loss_count'))
    _desc_df = _desc_df.append(pd.Series(_loss_df.mean(), name='loss_return_mean'))
    _desc_df = _desc_df.append(pd.Series(_loss_df.median(), name='loss_return_median'))

    _desc_df = _desc_df.append(pd.Series(_desc_df.loc['win_return_mean']/(-1*_desc_df.loc['loss_return_mean']), name='return_dist'))
    _desc_df = _desc_df.append(pd.Series((_desc_df.loc['win_rate']*_desc_df.loc['return_dist'])-(-1*_desc_df.loc['loss_rate']*1))*100,name='trading_odds'))

    return _desc_df
```

5-1) 1년 홀딩 수익률 승률

```
port_rets1_df = pd.DataFrame(port_rets1,columns=['1y'])
describe_winloss(port_rets1_df)
```

5-2) 2년 홀딩 수익률 승률

```
port_rets2_df = pd.DataFrame(port_rets2,columns=['2y'])
describe_winloss(port_rets2_df)
```

필터링 포트폴리오 백테스팅

6) cufflinks와 plotly로 interactive plot을 생성

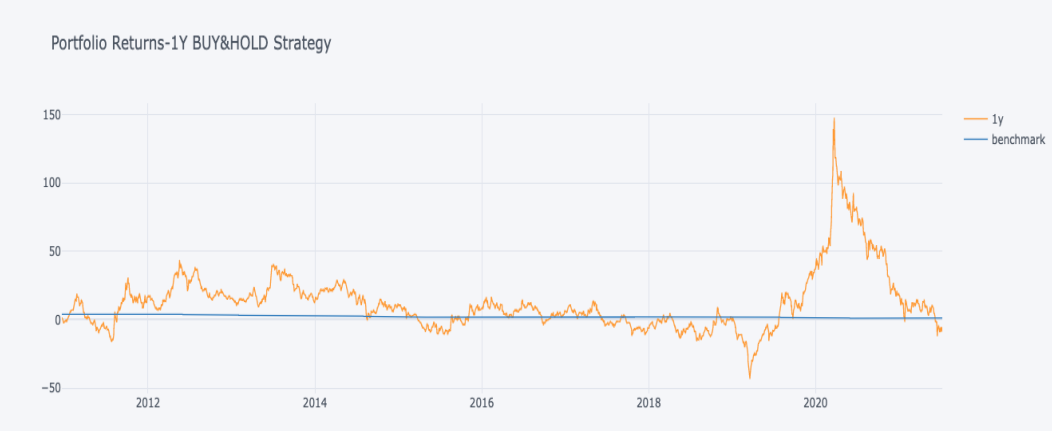
```
import cufflinks as cf
from plotly.offline import iplot, init_notebook_mode

# set up settings (run it once) :  스크립트를 다시 실행하지 않고 도면을 볼 수 있는 옵션으로 노트북 공유
cf.set_config_file(world_readable=True, theme='pearl',
                   offline=True)

# initialize notebook display
init_notebook_mode()
```

6-1) 1년 BUY&HOLD 전략 수익률 plot

```
# Portfolio Returns-1Y BUY&HOLD Strategy
final1.iplot(title='Portfolio Returns-1Y BUY&HOLD Strategy')
```



6-2) 2년 BUY&HOLD 전략 수익률 plot

```
# Portfolio Returns-2Y BUY&HOLD Strategy
final2.iplot(title='Portfolio Returns-2Y BUY&HOLD Strategy')
```

전략의 확률적 우위

승률
Hit Ratio

1회 트레이딩 시 승리인지 패배인지 알려주는 확률
요약 통계 'win_rate'

Hit Ratio = 0 이상의 수익률을 기록한 횟수/0 미만 수익률 횟수

확률적 우위
Trading Edge
Trading Odds

승률과 손익비를 동시에 고려한 값으로, 0보다 클 때 효과적인 트레이딩 전략

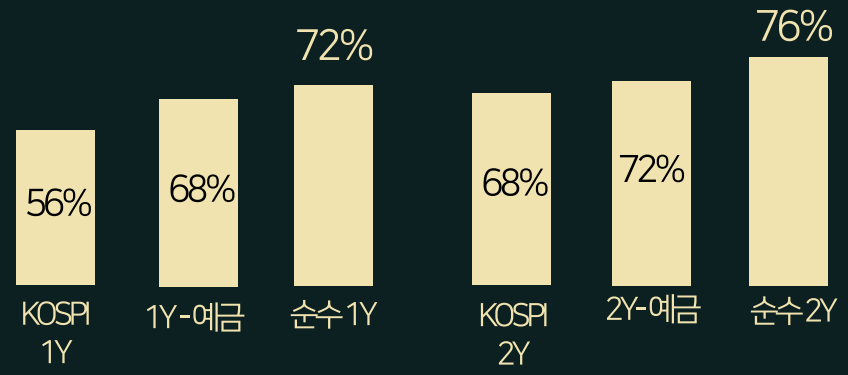
= 승률 * 수익의 평균 - (1 - 승률) * 손실의 평균

손익비
Return Distribution
P&L Ratio

평균적 하락 대비 평균적 수익의 비율
요약 통계 'Return_dist'

P&L Ratio= 수익의 평균/손실의 평균

<포트폴리오 승률>



1Y 포트폴리오 확률적 우위(예금 대비) :
124.3624%

2Y 포트폴리오 확률적 우위 (예금 대비) :
188.8333 %

<포트폴리오 손익비>

Multiple Regression 방법

EDA를 통한 이상치 및 결측치 확인
변수 추출 방법

- 결측치
- Box-plot
- Heatmap, VIF
- Stepwise 방법

다중회귀모델 생성

- Box-plot 을 통해 이상치 값 확인
- 다중공선성이 있다고 판단되어 지는 변수들을 VIF를 통해 하나씩 제거
- 해당 모델의 적합한 모델 선정
- OLS함수를 이용한 다중회귀모델

결과(모델의 적합성 확인

- Train data : 80%
  Test data : 20%
- R-squared 및
  Adj. R-squared 확인
- AIC, BIC
  Durbin-Watson
  Cond. No
- P-value
  Skew(왜도)
  Kurtosis(첨도)

OLS Regressions Results

1) 다중 회귀 분석 summary

	OLS Regression	〈 유의성 변수 〉		〈 Stepwise 〉		〈 Stepwise + 유의성〉	
Dep. Variable:	총배당금	R-squared:	0.674	R-squared:	0.667	R-squared:	0.667
Model:	OLS	Adj. R-squared:	0.665	Adj. R-squared:	0.654	Adj. R-squared:	0.654
Method:	Least Squares	F-statistic:	77.09	F-statistic:	50.40	F-statistic:	50.40
Date:	Mon, 01 Aug 2022	Prob (F-statistic):	9.81e-52	Prob (F-statistic):	7.66e-39	rob (F-statistic):	7.66e-39
Time:	09:38:37	Log-Likelihood:	-1882.2	Log-Likelihood:	-1503.8	Log-Likelihood:	-1503.8
No. Observations:	231	AIC:	3778.	AIC:	3024.	AIC:	3024.
Df Residuals:	224	BIC:	3803.	BIC:	3049.	BIC:	3049.
Df Model:	6						
Covariance Type:	nonrobust						

Adj.R-squared

0.65대로 유지되는 모습을 보임

AIC / BIC

stepwise 실행 이후 줄어들었으나 여전히 높은 편

OLS Regressions Results

2) 잔차의 독립성, 왜도, 첨도, 다중공선성 확인

〈 유의성 변수 〉

Omnibus:	21.273	Durbin-Watson:	2.007
Prob(Omnibus):	0.000	Jarque-Bera (JB):	98.305
Skew:	-0.024	Prob(JB):	4.50e-22
Kurtosis:	6.581	Cond. No.	4.76

〈 Stepwise 〉

Omnibus:	20.786	Durbin-Watson:	1.961
Prob(Omnibus):	0.000	Jarque-Bera (JB):	93.110
Skew:	0.035	Prob(JB):	6.04e-21
Kurtosis:	6.484	Cond. No.	4.84

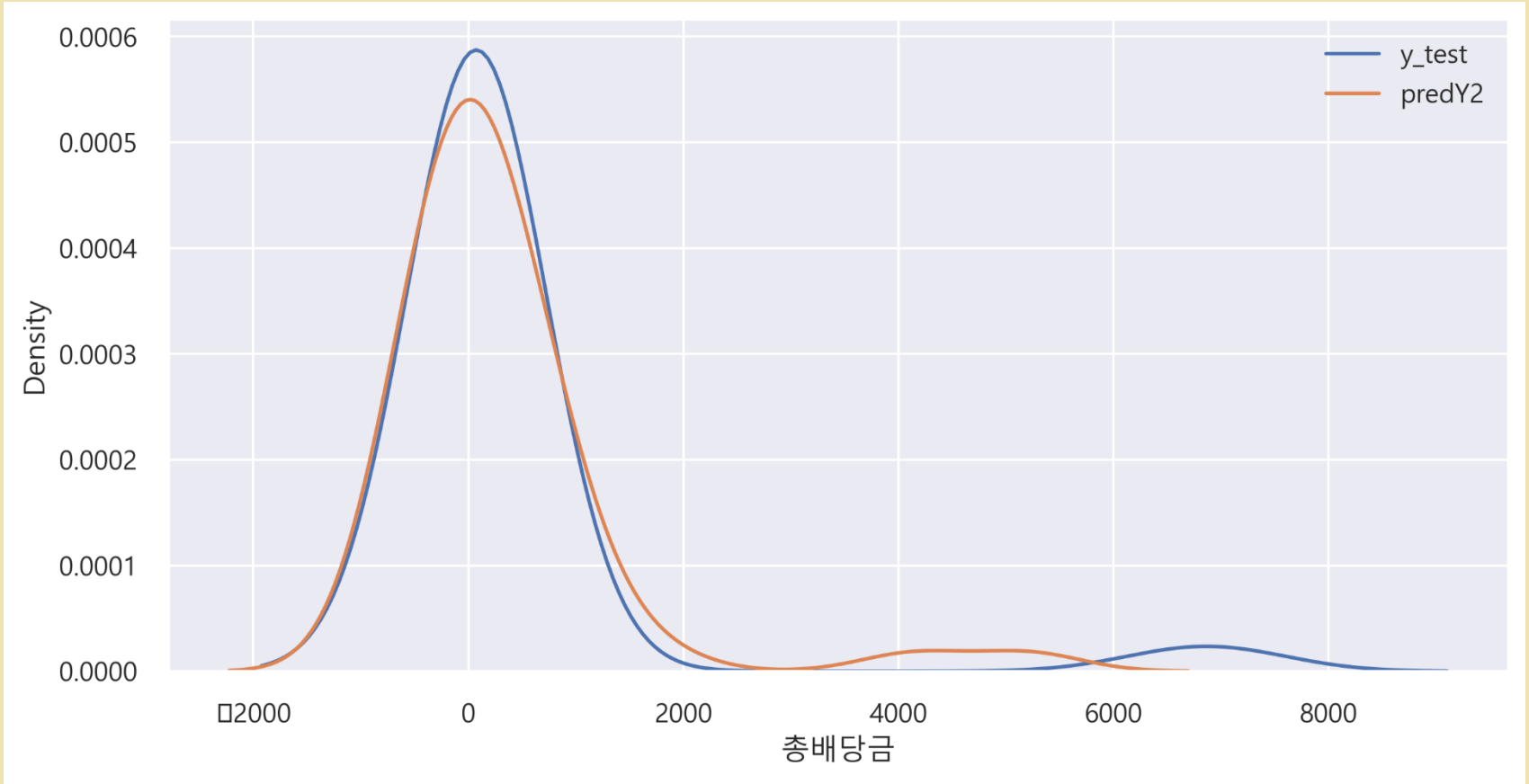
〈 Stepwise + 유의성〉

Omnibus:	20.786	Durbin-Watson:	1.961
Prob(Omnibus):	0.000	Jarque-Bera (JB):	93.110
Skew:	0.035	Prob(JB):	6.04e-21
Kurtosis:	6.484	Cond. No.	4.84

Pred와 ytest 비교 – 1) Stepwise로 변수 선택

MSE & RMSE :

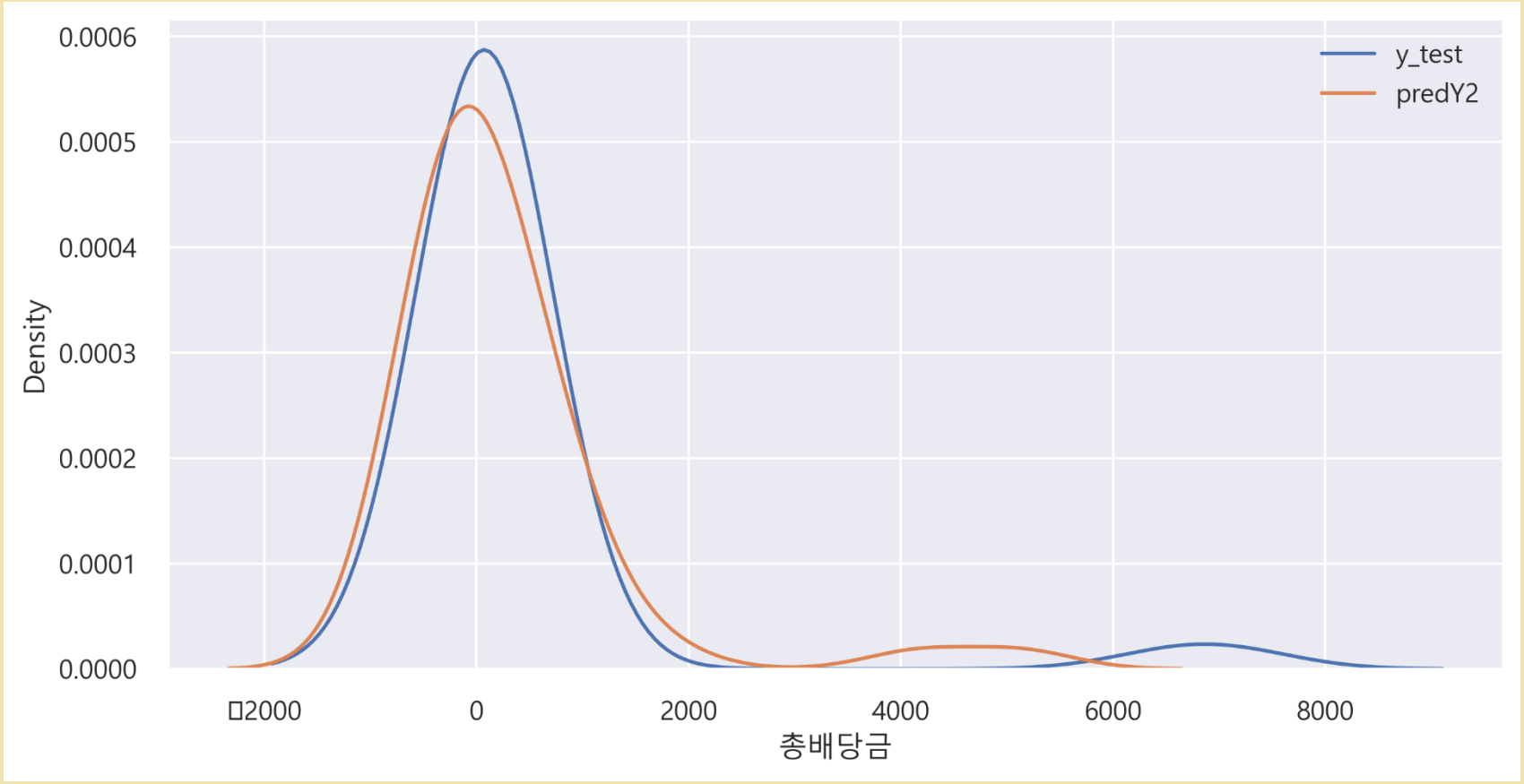
```
459956.63993009843
678.2010320915904
```



Pred와 ytest 비교 – 2) 유의성 검정을 통한 변수 선택

MSE & RMSE :

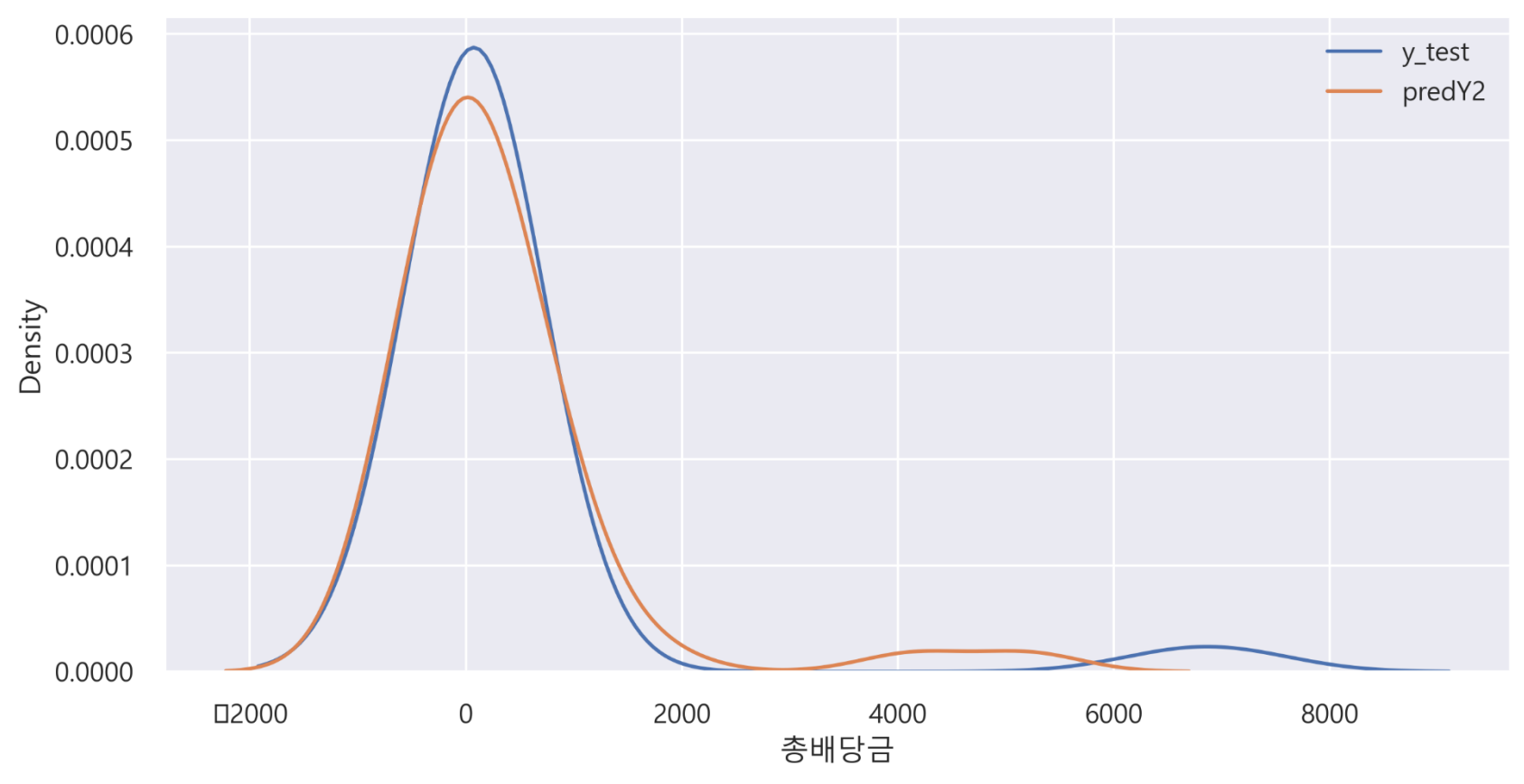
```
470402.70355191285
685.859098905827
```



Pred와 ytest 비교 – 3) Stepwise + 유의성검정을 통한 변수 선택

MSE & RMSE :

```
459956.63993009843
678.2010320915904
```



결론

기대효과

하나 안전성을 추가로 고려한 새로운 배당 안전과 관련된 정보상품 제공 가능

두울. 백테스트 시 기업들의 **배당수익률과 매매차익을 동시에 고려**

- 예금상품의 이자를 대체하고 매매차익을 통한 알파 창출 기회 제공

세엣. 총배당금 예측을 통한 개선된 포트폴리오 수익률

- 연도별 총배당금 예측함으로써 전 회계연도의 주당배당금으로 배당수익률을 추정했던 기존의 포트폴리오 수익률 개선

한계

하나. 주식 포트폴리오에 따른 리스크 무시 못함

- 승률이 100%인 예금에 비해서는 승률이 낮음
- 횡적,종적 리스크를 고려하지 않은 단순한 전략(필터링, Buy&Hold)만으로도 충분한 전략적 우위가 있다는 점에서 의의가 있음

두울. 예측을 위한 변수 데이터가 패널데이터

- 기업과 연도에 상관없다는 가정하에 다중회귀분석을 진행했지만 특정 기업의 고유한 특성이 있어 Adj.R-squared 값이 0.654로 나옴
- 포트폴리오 기업과 연도에 영향을 받는 데이터 특성상 패널 회귀모델이 생략된 변수의 위험을 줄이고 다중 공선성도 줄일 수 있을 거라 기대됨

$$Y_{it} = \beta_0 + \beta_1 X_{it} + \beta_2 Z_i + \mu_{it}$$

LIMITED

끝내기 전에…

기대효과

하나 안전성을 추가로 고려한 새로운 배당 안전과 관련된 정보상품 제공 가능

두울. 백테스트 시 기업들의 배당수[illegible] 차익을 동시에 고려

- 예금상품의 이자를 대체하고 매매차익을 통한 알파 창출 기회 제공

세엣. 총배당금 예측을 통한 개선된 포트폴리오 수익률

- 연도별 총배당금 예측함으로써 전 회계연도의 주당배당금으로 배당수익률을 추정했던 기존의 포트폴리오 수익률 개선

한계

하나. 주식 포트폴리오에 따른 리스크 무시 못함

- 승률이 100%인 예금에 비해서는 승률이 낮음
- 횡적,종적 리스크를 고려하지 않은 단순한 전략(필터링, Buy&Hold)만으로도 충분한 전략적 우[illegible] 있다는 점에서 의의가 있음

두울. 예측을 위한 변수 데이터가 패널데이터

- 기업과 연도에 상관없다는 가정하에 다중회귀분석을 진행했지만 특정 기업의 고유한 특성이 있어 Adj.R-squared 값이 0.654로 나옴
- 포트폴리오 기업과 연도에 영향을 받는 데이터 특성상 패널 회귀모델이 생략된 변수의 위험을 줄이고 다중 공선성도 줄일 수 있을 거라 기대됨

$$Y_{it} = \beta_0 + \beta_1 X_{it} + \beta_2 Z_i + \mu_{it}$$

SeeStock Member's Role

Kang Donghwa

#DB 관리자
#해커톤 최종발표자

Kim Bobae

#경제,통계,파이썬 총괄매니저
#노션관리자 #PPT 일러스트레이터

Im Hyeongjin

#통계, 파이썬 디벨로퍼
#해커톤 중간발표자

Jung Heedo

#경제 프로페서
#해커톤 아이디어발표자

SeeStock Member's Role

Q/A

Kang Donghwa

#DB 관리자
#해커톤 최종발표자

Kim Bobae

#경제,통계,파이썬 총괄매니저
#노션관리자 #PPT 일러스트레이터

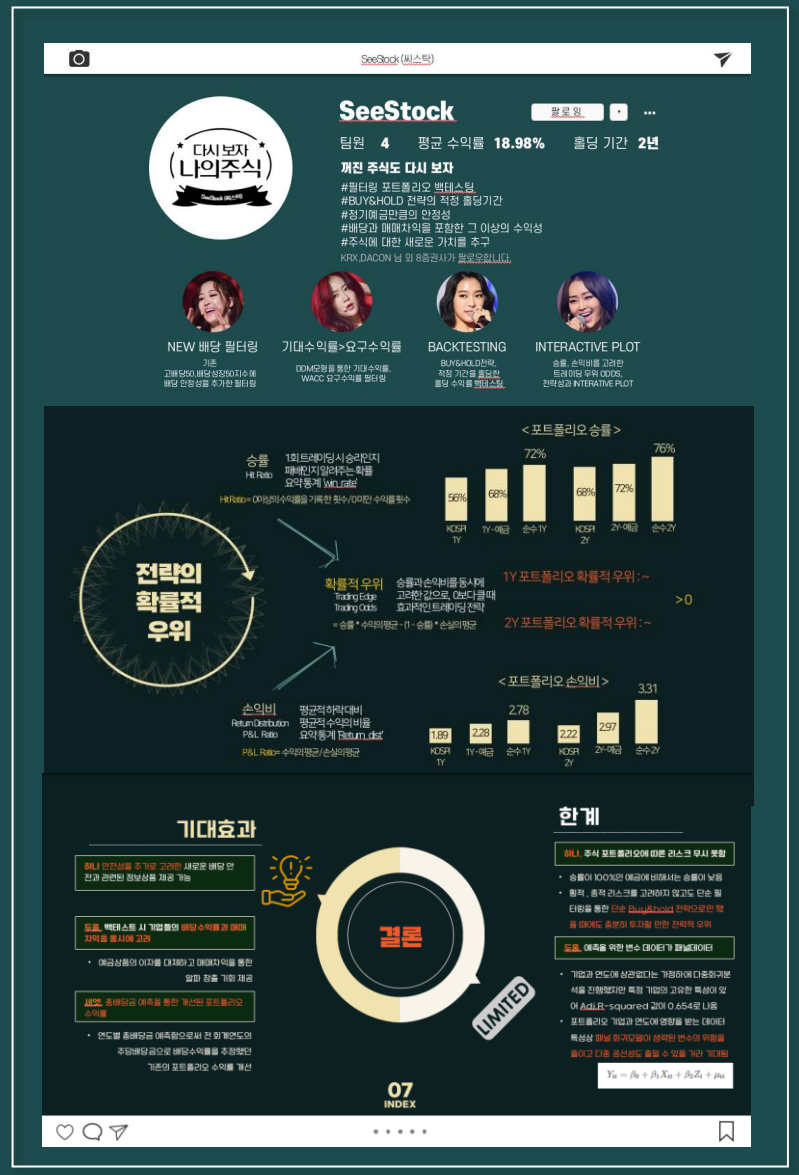
Im Hyeongjin

#통계, 파이썬 디벨로퍼
#해커톤 중간발표자

Jung Heedo

#경제 프로페서
#해커톤 아이디어발표자

THANK YOU


SeeStock (씨스탁)
SeeStock
팔로잉
팀원 4 평균 수익률 18.98% 홀딩 기간 2년
다시보자 나의주식
꺼진 주식도 다시 보자
#필터링 포트폴리오 백테스팅
#BUY&HOLD 전략의 적정 홀딩기간
#적기예금만큼의 안정성
#배당과 매매차익을 포함한 그 이상의 수익성
#주식에 대한 새로운 가치를 추구
KRX DACON 님 외 8종권사가 팔로우합니다.
NEW 배당 필터링
기존 고배당50,배당성장50지수에 배당 안정성을 추가한 필터링
기대수익률>요구수익률
DDM모형을 통한 기대수익률, WACC 요구수익률 필터링
BACKTESTING
BUY&HOLD전략, 적정 기간을 홀딩한 홀딩 수익률 백테스팅
INTERACTIVE PLOT
승률, 손익비를 고려한 트레이딩 우위 ODDS, 전략성과 INTERATIVE PLOT
<포트폴리오 승률>
승률
Hit Ratio
1회트레이딩시 승리인지 패배인지알려주는 확률
요약통계 'win_rate'
HitRatio= 이상의수익률을 기록한 횟수/이만한 수익률횟수
56%
68%
72%
68%
72%
76%
KOSPI 1Y
1Y-예금
순수 1Y
KOSPI 2Y
2Y-예금
순수 2Y
전략의 확률적 우위
확률적 우위
Trading Edge
Trading Odds
= 승률 * 수익의평균 - (1 - 승률) * 손실의평균
승률과 손익비를 동시에 고려한 값으로, 0보다클때 효과적인 트레이딩 전략
1Y 포트폴리오 확률적 우위 : ~
>0
2Y 포트폴리오 확률적 우위 : ~
<포트폴리오 손익비>
손익비
Return Distribution P&L Ratio
평균적 하락대비 평균적 수익의 비율
요약통계 'Return_dist'
P&L Ratio= 수익의평균/손실의평균
1.89
2.28
2.78
2.22
2.97
3.31
KOSPI 1Y
1Y-예금
순수 1Y
KOSPI 2Y
2Y-예금
순수 2Y
기대효과
하나 안전성을 추가로 고려한 새로운 배당 안전과 관련된 정보상품 제공 가능
도로, 백테스트 시 기업들의 배당수익률과 매매차익을 동시에 고려
예금상품의 이자를 대체하고 매매차익을 통한 알파 창출 기회 제공
세오, 종배당금 예측을 통한 개선된 포트폴리오 수익률
연도별 종배당금 예측함으로써 전 회계연도의 주당배당금으로 배당수익률을 추정했던 기존의 포트폴리오 수익률 개선
결론
LIMITED
한계
하나. 주식 포트폴리오에 따른 리스크 무시 못함
승률이 100%인 예금에 비해서는 승률이 낮음
횡적, 종적 리스크를 고려하지 않고도 단순 필터링을 통한 단순 Buy&hold 전략으로만 했을 때에도 충분히 투자할 만한 전략적 우위
도로, 예측을 위한 변수 데이터가 패널데이터
기업과 연도에 상관없다는 가정하에 다중회귀분석을 진행했지만 특정 기업의 고유한 특성이 있어 Adj.R-squared 값이 0.654로 나옴
포트폴리오 기업과 연도에 영향을 받는 데이터 특성상 패널 회귀모델이 생략한 변수의 위험을 줄이고 다중 공선성도 줄일 수 있을 거라 기대됨
07 INDEX