기계학습 및 인공지능을 이용한 주식투자 기술 동향

2017년 4월 26일 수요일

유성준

sjyoo@sejong.ac.kr

세종대학교 컴퓨터공학과

빅데이터산학연구센터

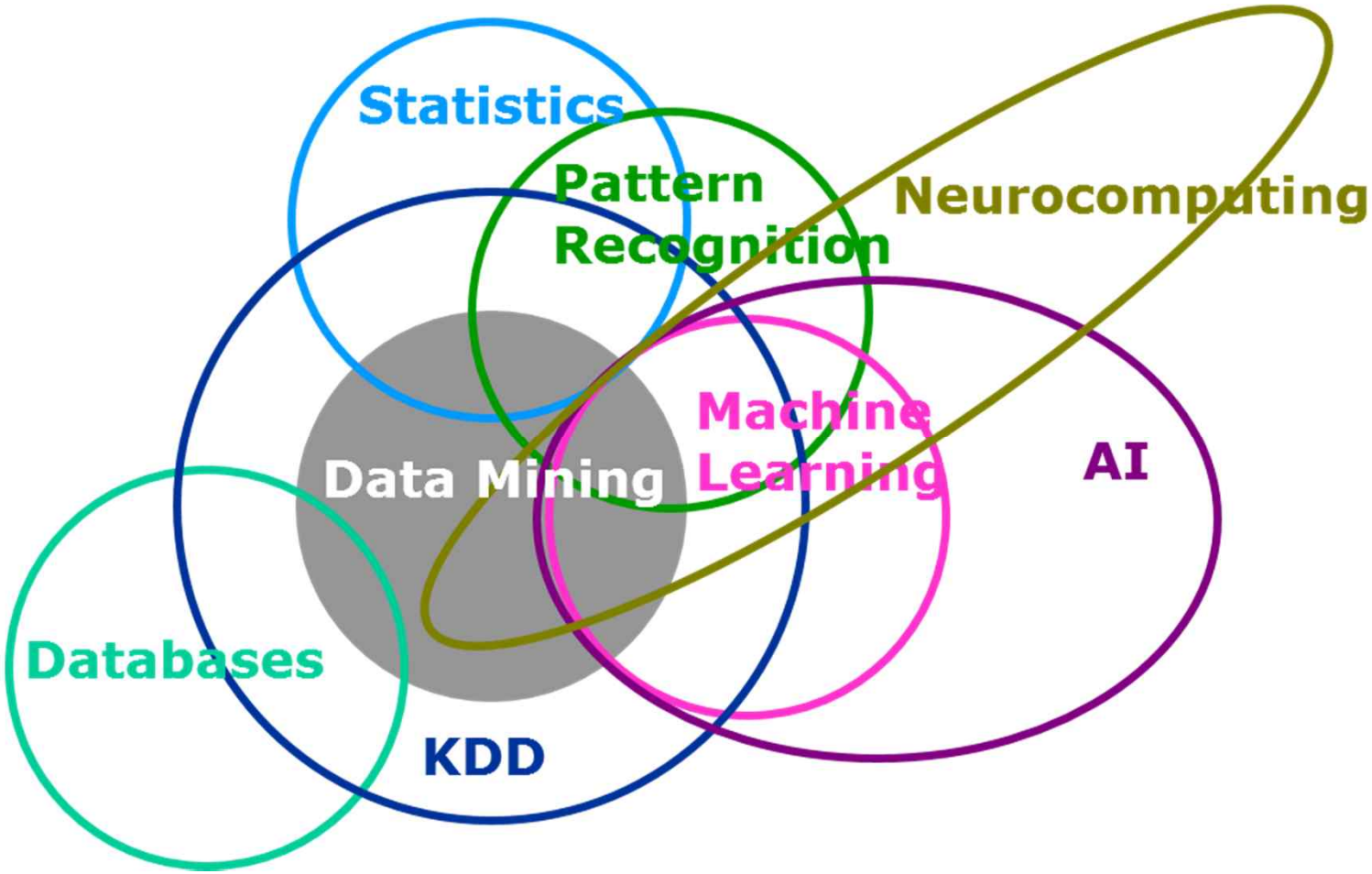
목차

I. 기계학습 및 AI 이론 복습

개요

- 기계학습 및 인공지능 기술 기반 투자이론
  - Portfolio 선정 및 최적화(이 강의에서는 이 내용만 다룹니다)
  - 예측 기반 Algorithmic Trading 및 High Frequency Trading
  - 텍스트(News 등) 내용 분석 등을 통한 예측 또는 종목 선정

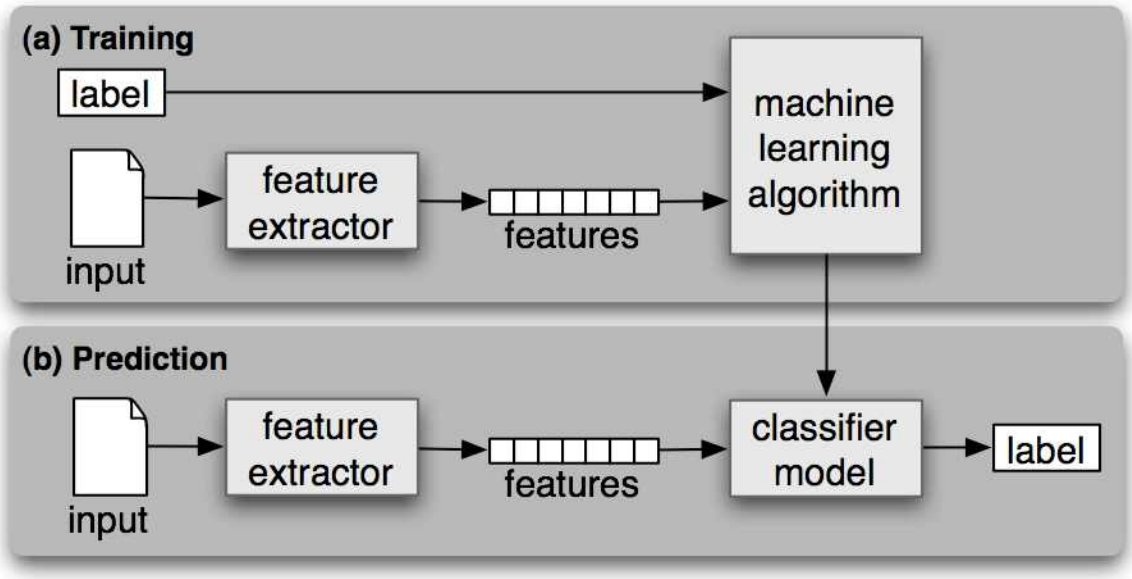
기계학습, 인공지능, 데이터마이닝, …



그림출처: https://www.analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/

기계학습 이론 다시 보기 - 지도학습

- 지도학습:
  - 훈련 데이터를 보여주면서 해당 데이터가 어디에 속하는지 지도하여 학습시킴
  - Support Vector Machine(SVM), Naïve Bayes Classification, Hidden Markov Model, Regression, Neural Network 외 다수 (https://en.wikipedia.org/wiki/Supervised_learning 참고)



그림출처: http://www.nltk.org/book/ch06.html

Feature의 정의 및 중요성

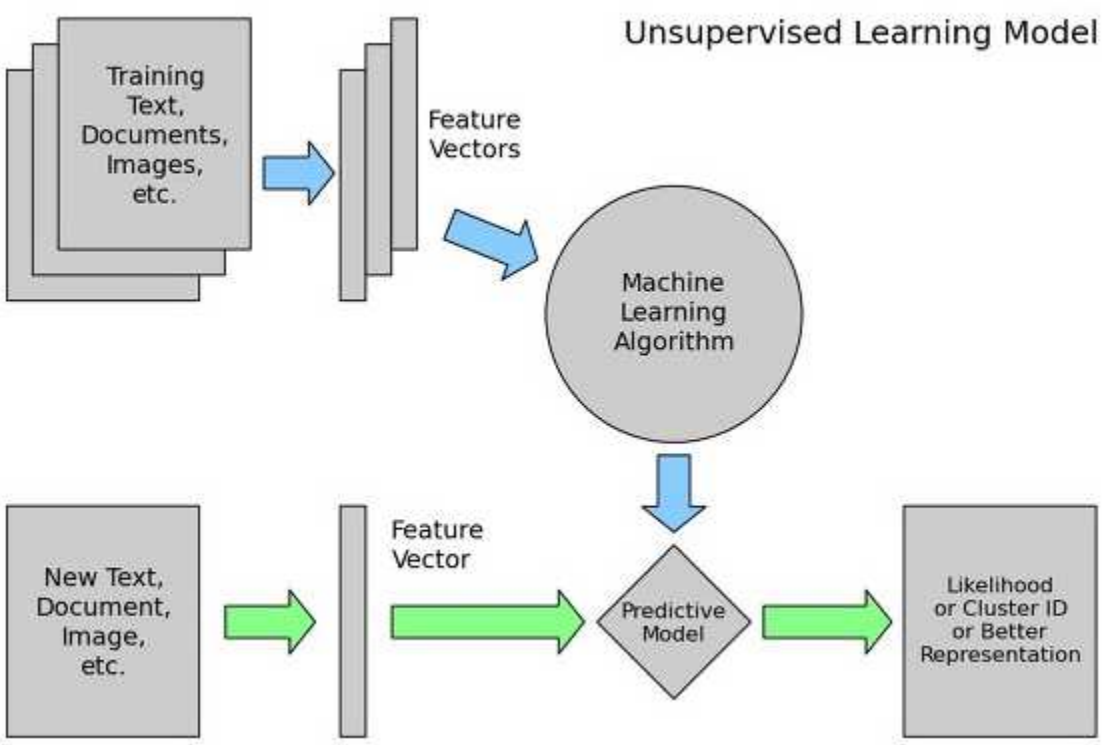
- 기계학습은 보통 중간 단계인 특징 추출(**feature extraction**)을 거쳐 data→feature→knowledge 의 단계로 학습하는 것이 보통입니다.

- 예를 들어 사진 속에서 사물을 인식하기 위해 픽셀 값에서 먼저 특징적인 선이나 특징적인 색 분포 등을 먼저 추출한 후 이를 기반으로 '이건 사과다' '이건 바나나다'라는 판단을 내리는 것이죠. 이러한 중간 표현단계를 특징 지도(feature map)이라고 하는데요, 기계학습의 성능은 **얼만큼 좋은 특징들을 뽑아내느냐**에 따라 그 성능이 매우 크게 좌지우지 됩니다. (이는 이미지 처리 뿐만 아니라 음성 인식, 자연어 분석 등 대부분의 기계학습에 적용되는 이야기입니다.) .

- 출처: http://t-robotics.blogspot.kr/2015/05/deep-learning.html#.WILUn_mLSUk

기계학습 이론 다시 보기 - 비지도학습

- 비지도학습:
  - 훈련 데이터는 레이블되어 있지 않음
  - 데이터 자체에서 패턴을 찾아냄
- 예
  - Autoencoder, K-means clustering 등

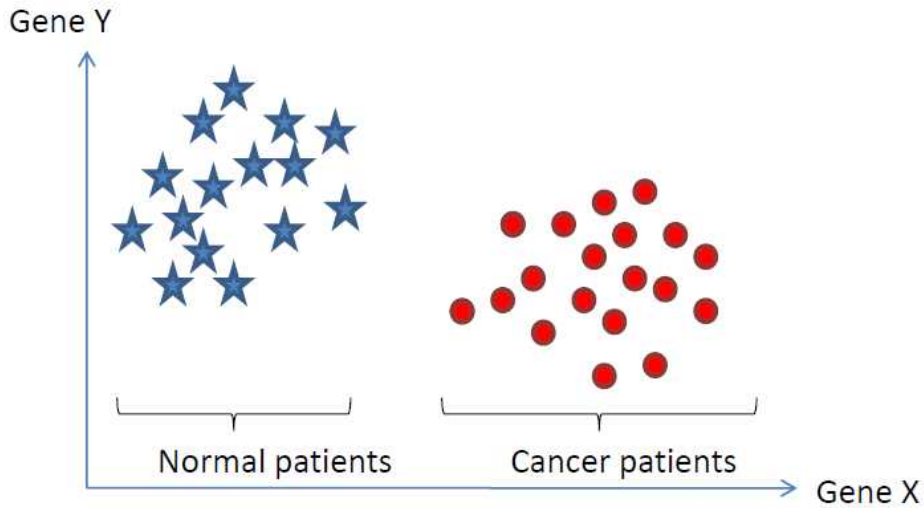


그림출처: http://overface.tistory.com/entry/Machine-Learning

기계학습 기술 적용 분야 다시 보기

- 예측
  - Regression, Support Vector Regression, Random Forest 등
- 분류
  - Naïve Bayes, Support Vector Machine 등
- 영상/음성 인식
- 자연어처리
- 기타

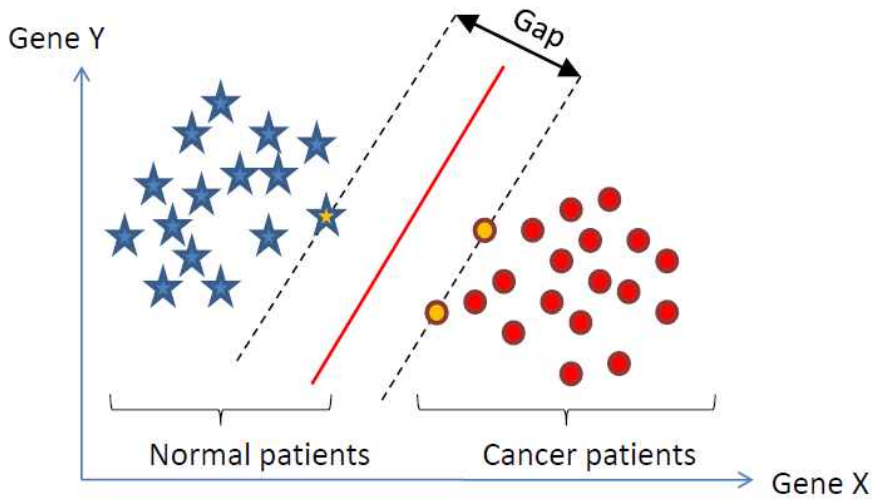
Support Vector Machine 개념 다시보기



- Consider example dataset described by 2 genes, gene X and gene Y
- Represent patients geometrically (by “vectors”)

출처: **Alexander Statnikov, Douglas Hardin, Isabelle Guyon, Constantin F. Aliferis,** A Gentle Introduction to Support Vector Machines in Biomedicine, AMIA 2009

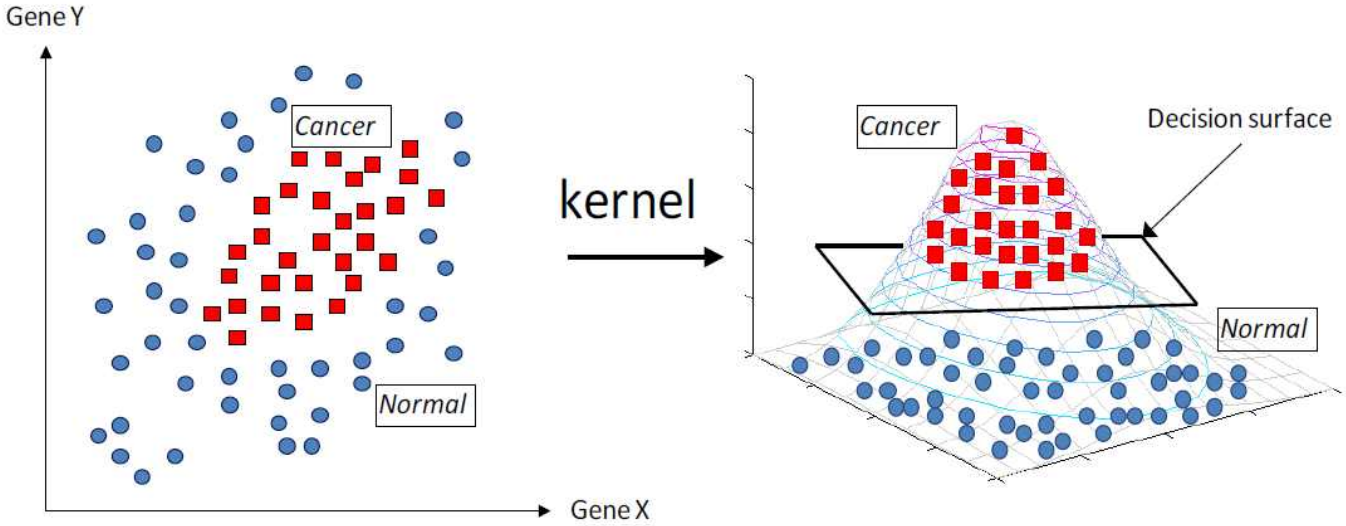
Support Vector Machine 개념 다시 보기



- Find a linear decision surface ("hyperplane") that can separate patient classes and has the largest distance (i.e., largest "gap" or "margin") between border-line patients (i.e., "support vectors");

출처: **Alexander Statnikov, Douglas Hardin, Isabelle Guyon, Constantin F. Aliferis,** A Gentle Introduction to Support Vector Machines in Biomedicine, AMIA 2009

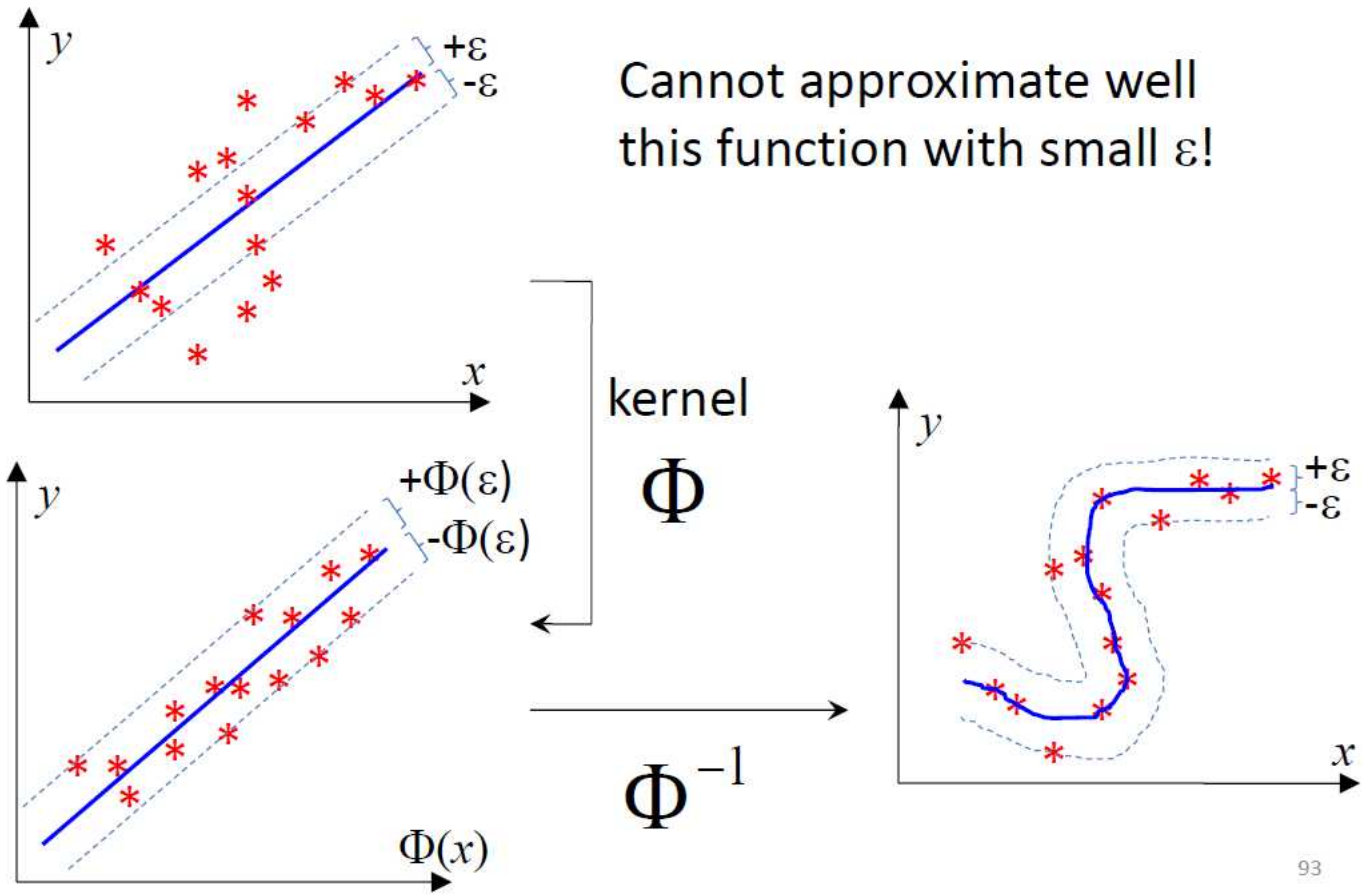
Support Vector Machine 개념 다시보기



- If such linear decision surface does not exist, the data is mapped into a much higher dimensional space ("feature space") where the separating decision surface is found;
- The feature space is constructed via very clever mathematical projection ("kernel trick").

출처: **Alexander Statnikov, Douglas Hardin, Isabelle Guyon, Constantin F. Aliferis,** A Gentle Introduction to Support Vector Machines in Biomedicine, AMIA 2009

비선형 Support Vector Regression 개념 다시보기



출처: **Alexander Statnikov, Douglas Hardin, Isabelle Guyon, Constantin F. Aliferis,** A Gentle Introduction to Support Vector Machines in Biomedicine, AMIA 2009

유전알고리즘(Genetic Algorithm) 개념 다시보기

- 유전법칙의 연산과정 일부(교차, 돌연변이, 대치 등)를 반복적으로 적용하여 최적에 가까운 또는 최적의 해를 구하는 알고리즘

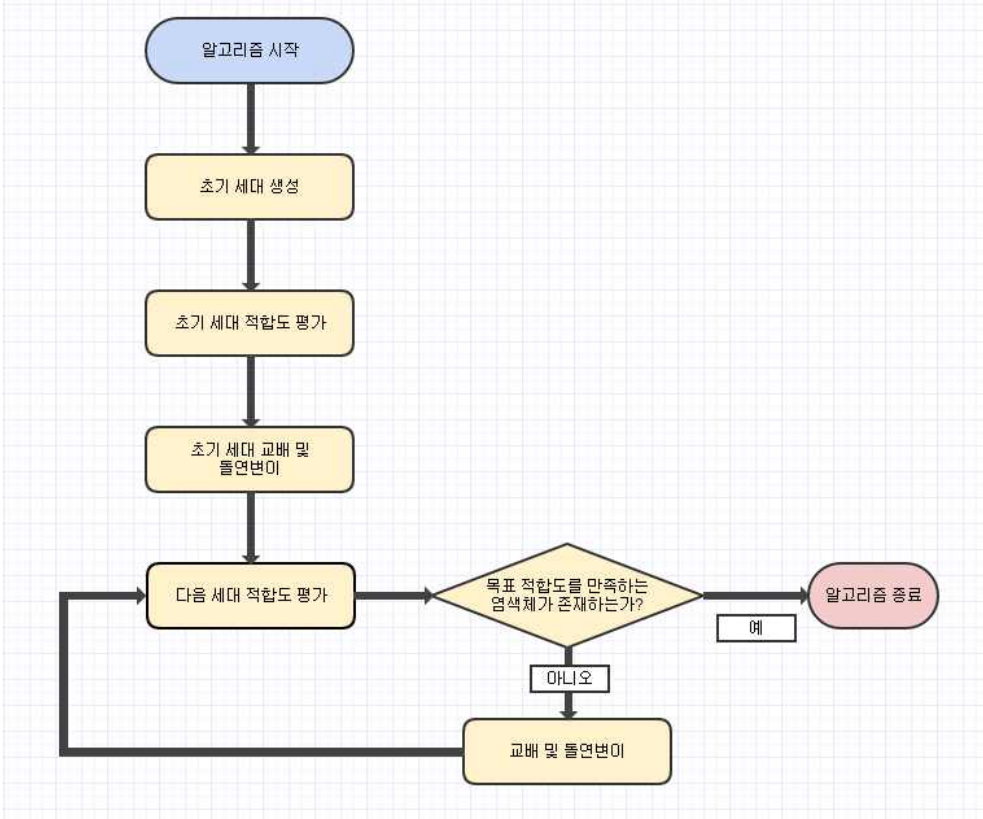
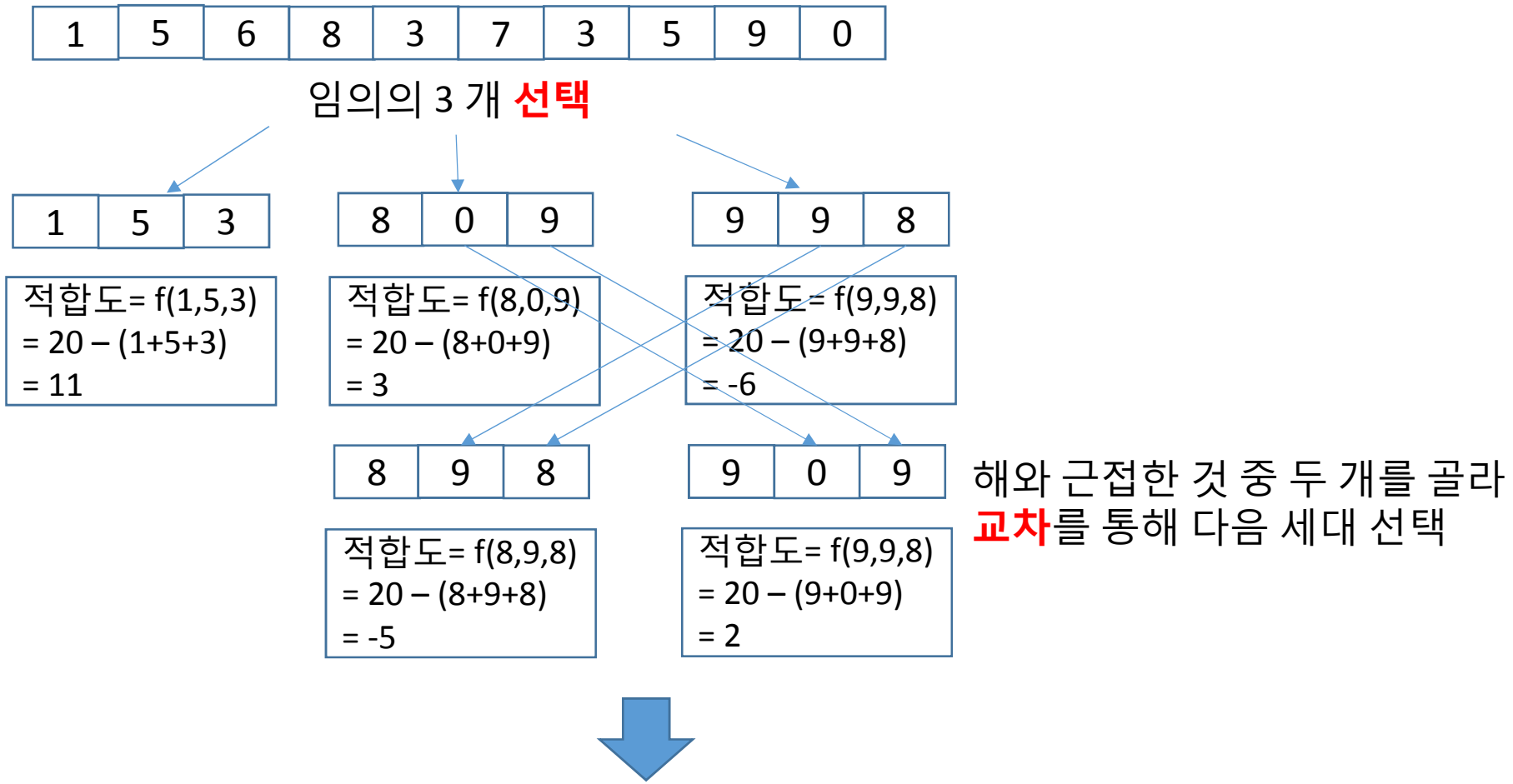


그림 출처: http://twinw.tistory.com/1

유전알고리즘(Genetic Algorithm) 개념 예

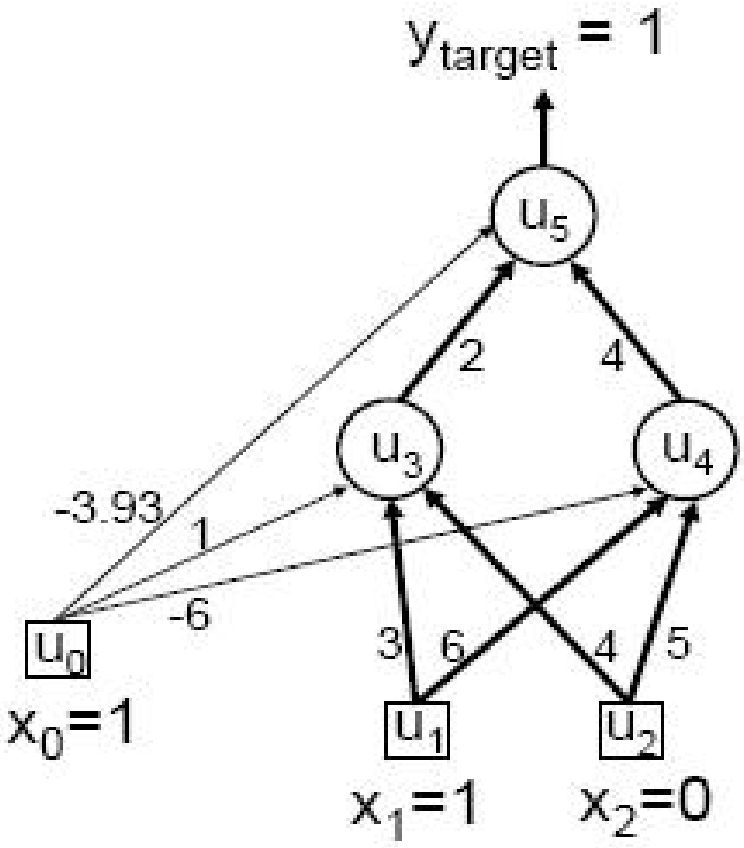
문제: 아래 숫자중 3개를 골라 20을 만들어라



해와 근접한 것 중 두 개를 골라 **교차**를 통해 다음 세대 선택

이후 종료 조건을 만족하거나 그 전에 정해진 횟수만큼 반복하면 종료

Artificial Neural Network 개념



Current state:

- Weights on arrows e.g. $w_{13} = 3$, $w_{35} = 2$, $w_{24} = 5$
- Bias weights, e.g.

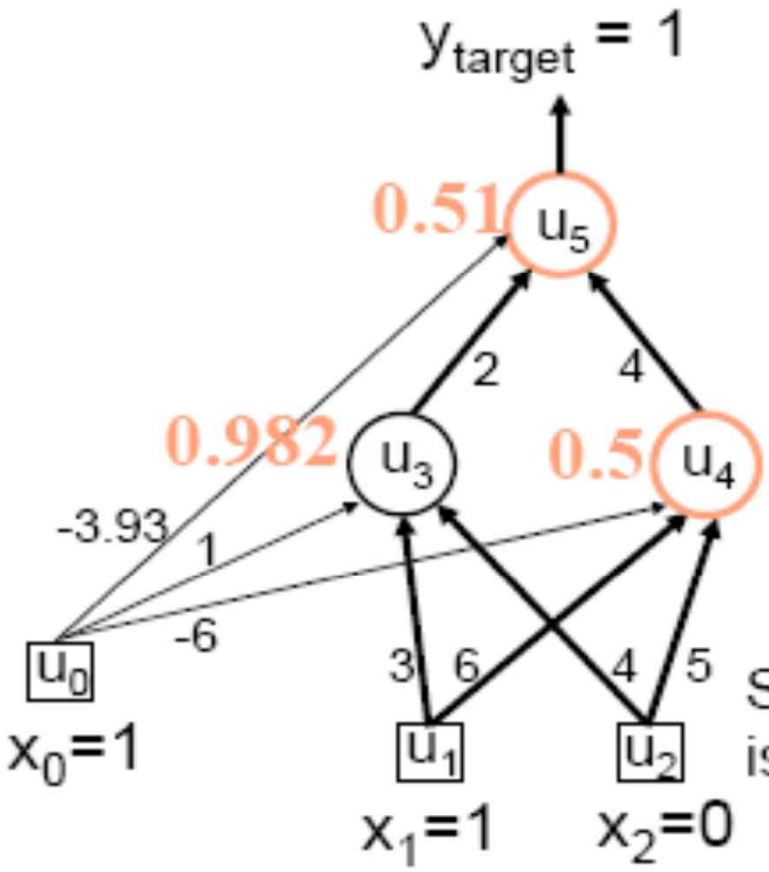
bias for unit 4 (u_4) is $w_{04} = -6$

Training example (e.g. for logical OR problem):

- Input pattern is $x_1=1$, $x_2=0$
- Target output is $y_{target}=1$

- 출처: 도용태 외, 인공지능: 개념 및 응용, 2013

Artificial Neural Network 개념



- Target 값 1과 계산된 값 0.51을 비교후 차이를 줄이기 위해 weight를 재계산
- 차이가 acceptable할 때가지 반복

Unit	activation a_j	output y_j
u_3	4.00	0.982
u_4	0.00	0.500
u_5	0.04	**0.510**

(network output)

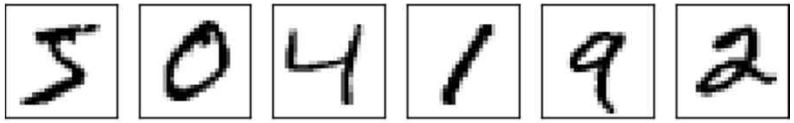
So the error for this training example is: $(y_{target} - y_5) = (1 - 0.510) = 0.490$

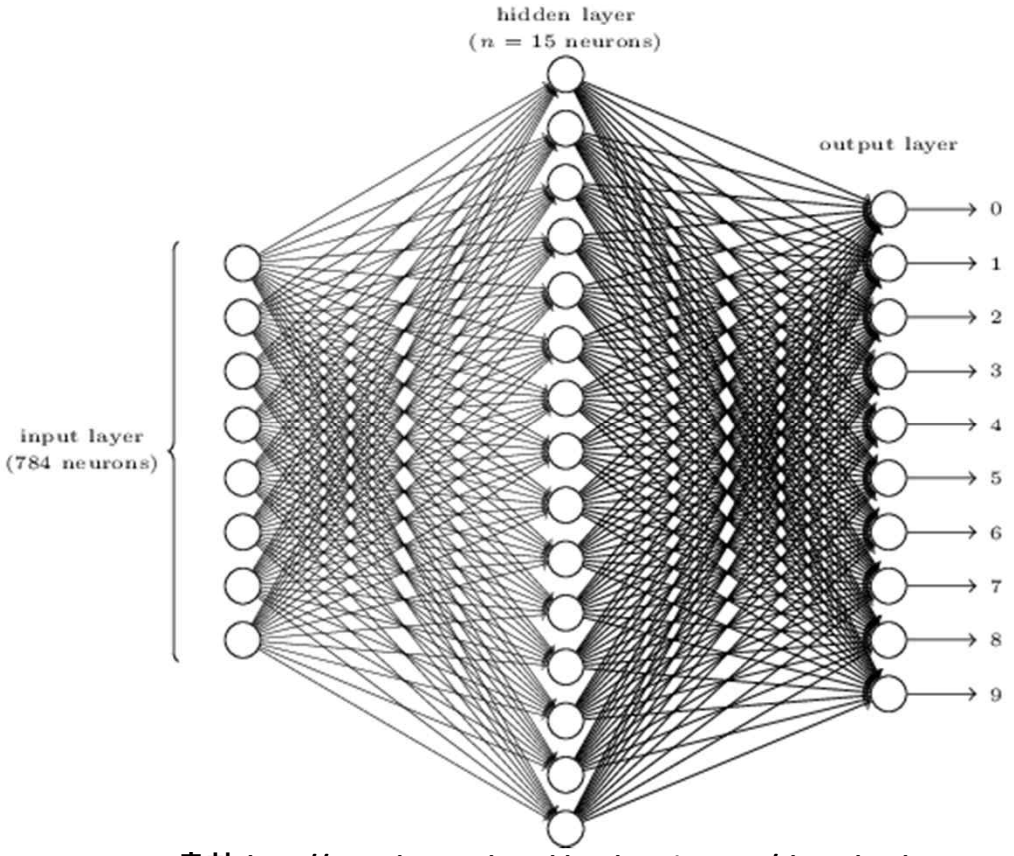
- 출처: 도용태 외, 인공지능: 개념 및 응용, 2013

신경망 작동 예제 - 분류

- 신경망을 이용한 필기체 숫자 인식 문제

입력데이터





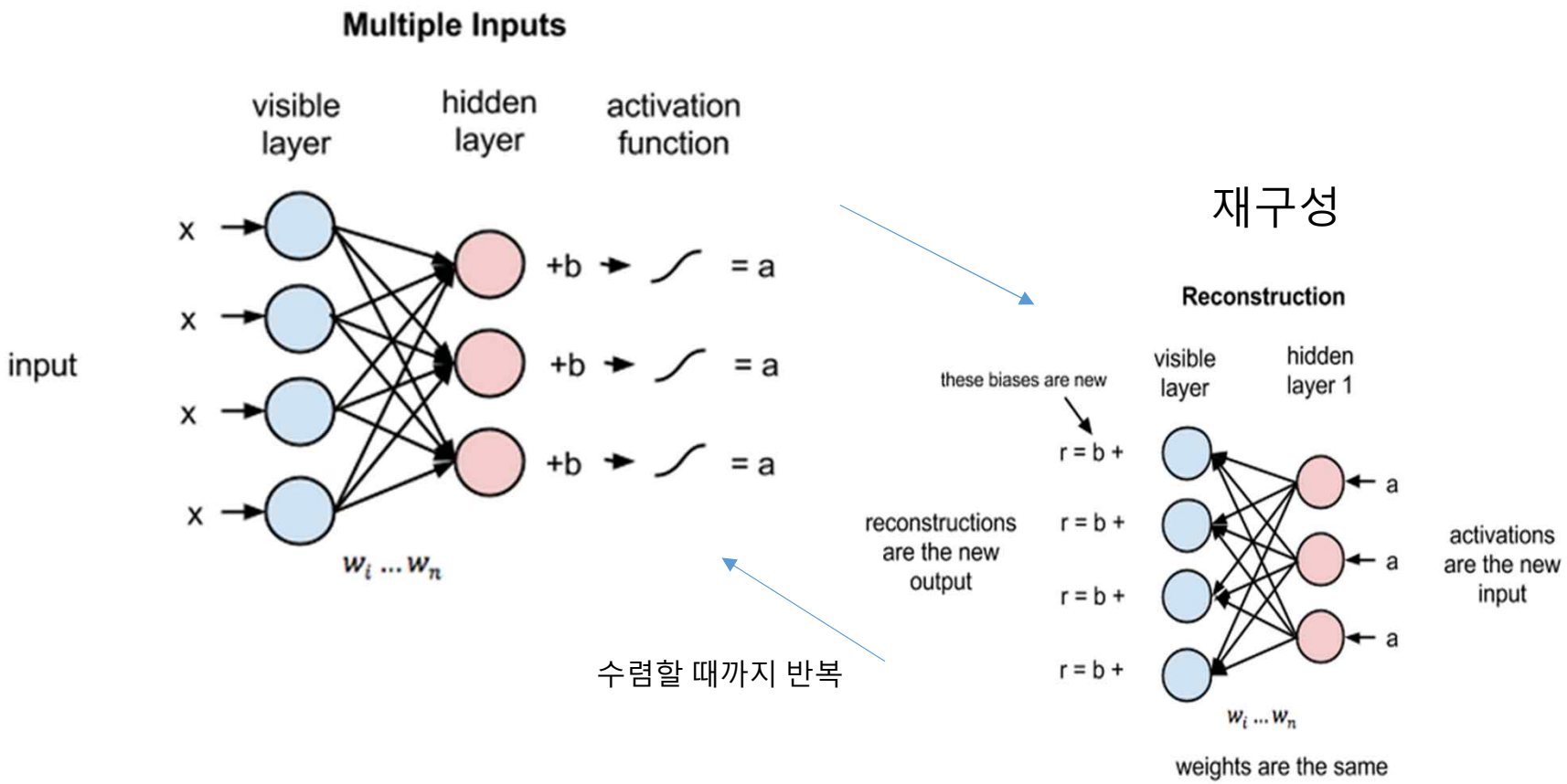
여기가 1이고 나머지가 0이면 해당 숫자는 0

여기가 1이고 나머지가 0이면 해당 숫자는 9

- 출처: http://neuralnetworksanddeeplearning.com/chap1.html

Autoencoder

순방향 학습- 2 Layers



- **출처: https://deeplearning4j.org/kr/kr-restrictedboltzmannmachine**

Ⅱ. 포트폴리오 이론 복습

Motivation

- 투자 모델의 주요 목표
  - 리스크 최소화 및 수익률 최대화

- 포트폴리오
  - 이익을 최대화하고 위험을 최소로 하는 목적으로 최적의 비율로 나눠진 투자자산의 집합

- Mean Variance Model
  - Markowitz, 1952.
  - This model can be used by investors to achieve desired returns from portfolio with minimum possible risk. In fact, this theory has wide spread acceptance and has been used as a practical tool for portfolio optimization.

기본지식정리 – 기대 수익률 및 분산

경제상황	확률	주식A	주식B
호황	0.3	8%	4%
보통	0.4	6%	5%
불황	0.3	5%	6%

기대수익률 구하기

$E(R_A) = 0.3 \times 8\% + 0.4 \times 6\% + 0.3 \times 0.5\% = 6.3\%$

표준편차 구하기

$\sigma_A^2 = 0.3 \times (8\% - 6.3\%)^2 + 0.4 \times (6\% - 6.3\%)^2 + 0.3 \times (5\% - 6.3\%)^2 = 0.11874$

$\sigma_A = 0.108968$

기본지식정리 – 포트폴리오의 기대수익률 및 volatility

- Expected return:

$$\mathrm{E}(R_p) = \sum_i w_i \, \mathrm{E}(R_i)$$

where R_p is the return on the portfolio, R_i is the return on asset i and w_i is the weighting of component asset i (that is, the proportion of asset "i" in the portfolio).

- Portfolio return variance:

$$\sigma_p^2 = \sum_i w_i^2 \sigma_i^2 + \sum_i \sum_{j \neq i} w_i w_j \sigma_i \sigma_j \rho_{ij},$$

where σ is the (sample) standard deviation of the periodic returns on an asset, and ρ_{ij} is the correlation coefficient between the returns on assets i and j. Alternatively the expression can be written as:

$$\sigma_p^2 = \sum_i \sum_j w_i w_j \sigma_i \sigma_j \rho_{ij},$$

where $\rho_{ij} = 1$ for $i = j$, or

$$\sigma_p^2 = \sum_i \sum_j w_i w_j \sigma_{ij},$$

where $\sigma_{ij} = \sigma_i \sigma_j \rho_{ij}$ is the (sample) covariance of the periodic returns on the two assets, or alternatively denoted as $\sigma(i,j)$, cov_{ij} or $cov(i,j)$.

- Portfolio return volatility (standard deviation):

$$\sigma_p = \sqrt{\sigma_p^2}$$

출처: https://en.wikipedia.org/wiki/Modern_portfolio_theory

기본지식정리 – 포트폴리오의 기대수익률 및 volatility

For a **two asset** portfolio:

- Portfolio return: $\mathrm{E}(R_p) = w_A\,\mathrm{E}(R_A) + w_B\,\mathrm{E}(R_B) = w_A\,\mathrm{E}(R_A) + (1 - w_A)\,\mathrm{E}(R_B).$
- Portfolio variance: $\sigma_p^2 = w_A^2\sigma_A^2 + w_B^2\sigma_B^2 + 2w_Aw_B\sigma_A\sigma_B\rho_{AB}$

<출처: https://en.wikipedia.org/wiki/Modern_portfolio_theory>

주식 A에 50%, 주식 B에 50% 투자한 포트폴리오 구성

기대수익률 구하기

$$E(R_P) = 50\% \times 6.3\% + 50\% \times 5\% = 5.65\%$$

표준편차(return volatility) 구하기

$$\sigma_P^2 = 0.5^2 \times \sigma_A^2 + 0.5^2 \times \sigma_B^2 + 2 \times 0.5^2 \times \sigma_A \times \sigma_B \times (-0.978) = 0.0000005273$$

$$\sigma_P = 0.002296$$

가정치

Motivation

15.401

What Is A Portfolio and Why Is It Useful?

- A **portfolio** is simply a specific combination of securities, usually defined by **portfolio weights** that sum to 1:

$$\omega = \{ \omega_1 , \omega_2 , \ldots , \omega_n \}$$

$$\omega_i = \frac{N_i P_i}{N_1 P_1 + \cdots + N_n P_n}$$

$$1 = \omega_1 + \omega_2 + \cdots + \omega_n$$

- Portfolio weights can sum to 0 (dollar-neutral portfolios), and weights can be positive (long positions) or negative (short positions).
- Assumption: Portfolio weights summarize all relevant information.

출처: Andrew W. Lo, Finance Theory, Lecture Note 13-14, MIT Sloan School, 2007-2008

Mean-Variance Analysis

15.401

Example: From 1946 – 2001, Motorola had an average monthly return of 1.75% and a std dev of 9.73%. GM had an average return of 1.08% and a std dev of 6.23%. Their correlation is 0.37. How would a portfolio of the two stocks perform?

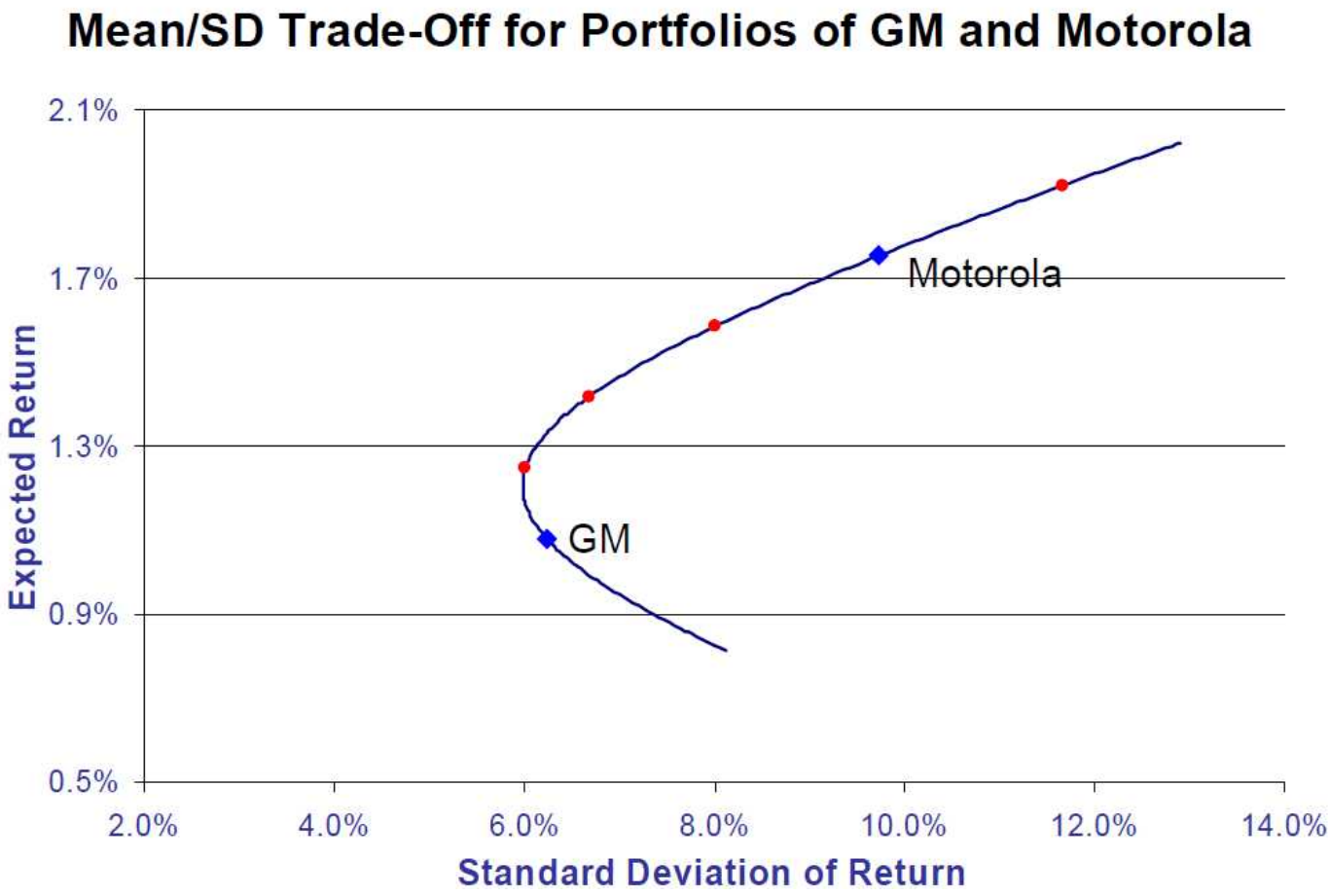
$$\begin{aligned} \mathrm{E}[R_p] &= \omega_{\mathrm{GM}}\, 1.08 + \omega_{\mathrm{MOT}}\, 1.75 \\ \mathrm{Var}[R_p] &= \omega_{\mathrm{GM}}^2\, 6.23^2 + \omega_{\mathrm{MOT}}^2\, 9.73^2 + \\ &\qquad 2\omega_{\mathrm{GM}}\omega_{\mathrm{MOT}}\,(0.37 \times 6.23 \times 9.73) \end{aligned}$$

w_{Mot}	w_{GM}	$E[R_P]$	$var(R_P)$	$stdev(R_P)$
0	1	1.08	38.8	6.23
0.25	0.75	1.25	36.2	6.01
0.50	0.50	1.42	44.6	6.68
0.75	0.25	1.58	64.1	8.00
1	0	1.75	94.6	9.73
1.25	-0.25	1.92	136.3	11.67

출처: Andrew W. Lo, Finance Theory, Lecture Note 13-14, MIT Sloan School, 2007-2008

Mean-Variance Analysis

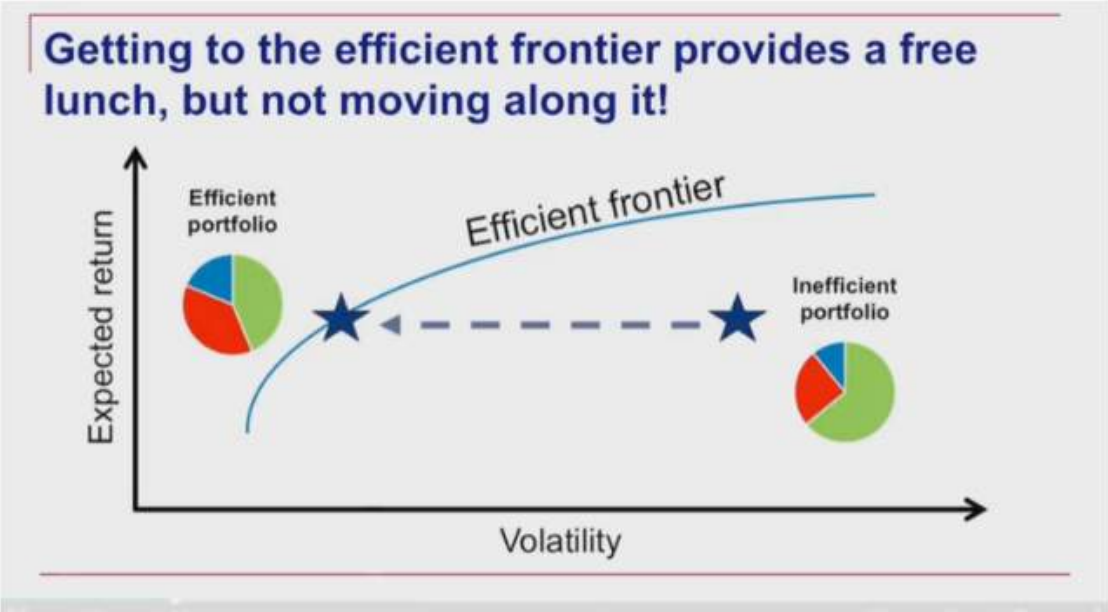
15.401



출처: Andrew W. Lo, Finance Theory, Lecture Note 13-14, MIT Sloan School, 2007-2008

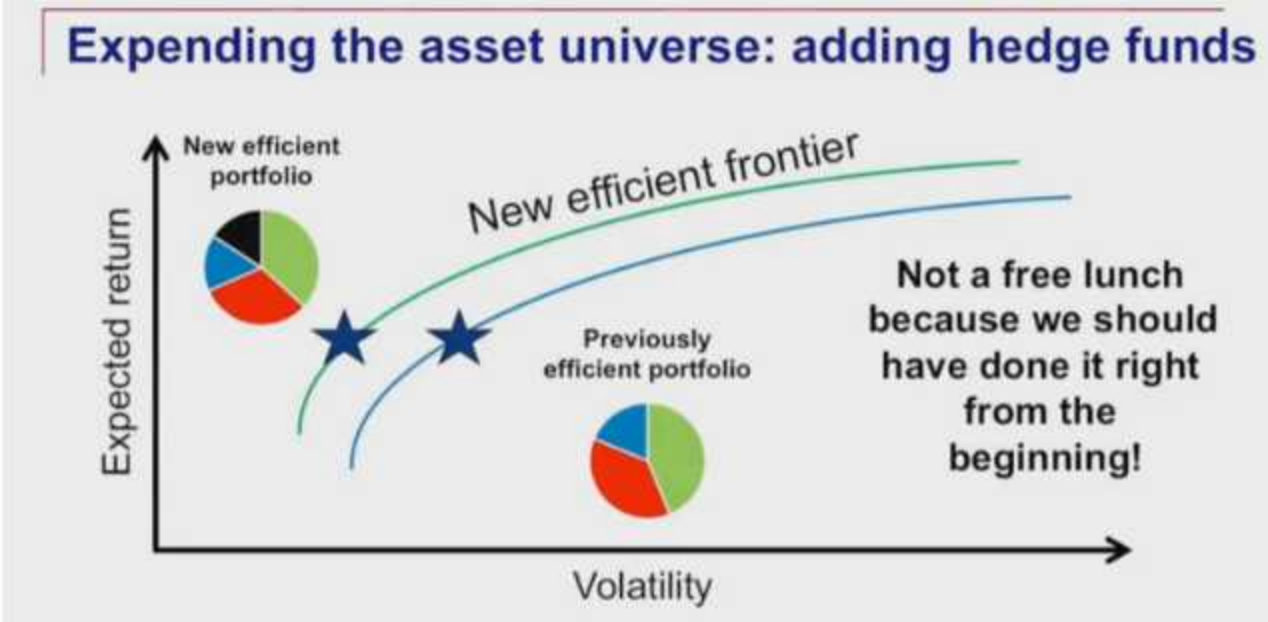
기본지식정리

- 다음은 효율적 프론티어 곡선임
- 수익이 커질 수록 위험성은 커지고 가장 최적의 효율은 효율적 프론티어 곡선 위에서 발생
- 오른쪽에 있는 inefficient portfolio는 왼쪽에 있는 efficient portfolio와 같은 수익을 가지면서 위험성은 높음
- 따라서 포트폴리오를 조정해줌으로써 위험성을 낮출 수 있음(주식의 비중을 낮추고 채권과 현금의 비중을 올리는 등)



기본지식정리

- 또 다른 헤지펀드를 추가 시 새로운 효율적 프론티어 곡선이 생김
- 같은 수익에 리스크는 낮아짐
- 이것은 그냥 투자할 수 있는 universe를 넓힌 것 뿐임

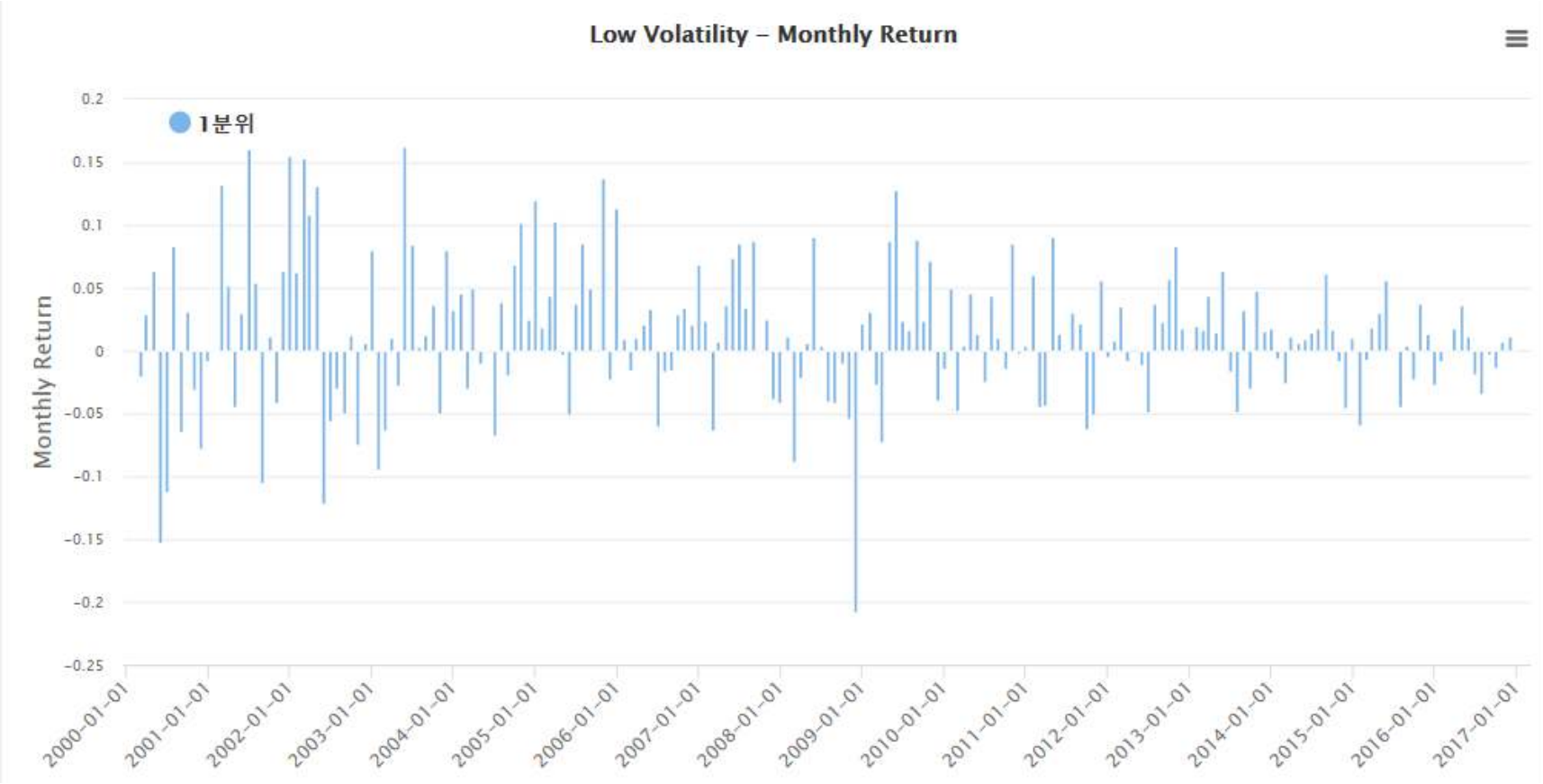


포트폴리오 종목 선정 예 – Style Factors

Performance of a Model

누적수익률(cumulative return)
= (Current Price of Security) - (Original Price of Security) / (Original Price of Security)

Performance of a Model



Performance of a Model

초기값이 $500, 일정기간동안(over a period of time)의 최고값이 $750까지 도달 후 $400로 하락했다가 다시 $600로 상승,
다시 $350까지 하락후 $800까지 재상승할 경우
Maximum Drawdown = ($350 - $750)/$750 = -53.3%

Performance of a Model

Sharpe ratio = (Mean portfolio return − Risk-free rate)/Standard deviation of portfolio return

$$= \frac{\bar{r}_p - r_f}{\sigma_p}$$

Where:

$\bar{r}_p$ = Expected porfolio return

r_f = Risk free rate

σ_p = Portfolio standard deviation

예를 들어 1년동안 코스피가 20% 오르고 내 포트폴리오가 30% 올랐는데, 내 포트폴리오의 월간 수익률에서 코스피의 월간 수익률을 뺀 값의 표준편차가 5%라면, 사후적 샤프 비율은 (0.3-0.2)/(0.05*12^(1/2)) = 약 0.57이 된다.[3]

높은 샤프지수를 기록하는 투자일수록 더 적은 위험으로 더 높은 수익률을 내는 것이라고 볼 수 있다. 그러므로 샤프지수가 높은 펀드 혹은 포트폴리오가 더 좋은 포트폴리오라고 볼 수 있다. 기본적으로 샤프지수가 0 이상이어야 투자의 고려대상이 될 수 있고[4] 1 이상이면 상당히 괜찮은 투자라고 볼 수 있다.[5]

출처: https://namu.wiki/w/%EC%83%A4%ED%94%84%20%EB%B9%84%EC%9C%A8

Ⅲ. ML 및 AI 기술을 포트폴리오 이론에 적용한 사례

A hybrid approach to portfolio composition based on fundamental and technical indicators

- Feature로 펀더멘탈(fundamental)과 기술적 지표(technical indicators)를 혼합하여 사용
- 다중 사물 진화 알고리즘(multi-objective evolutionary algorithms, MOEA)을 수익률과 위험의 최적화에 적용
- 펀더멘탈 분석으로는 가장 좋은 회사를 선택하게 하고 기술적 지표 분석으로는 주식의 매수, 매도 시점을 결정

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

A hybrid approach to portfolio composition based on fundamental and technical indicators

- 기술적 분석은 시장의 패턴(주식의 공급과 수요)을 분석함
- 가치 분석은 회사의 현재 상황을 분석(재무재표)함
  - 펀더멘탈 지표들로는 부채비율(DR), 자기자본 수익률(ROE), 당기순이익(PM) 등 10개를 사용함
- 포트폴리오의 핵심은 자본 배분인데 목적은 위험을 최소로, 수익은 최대로 하여 최적의 포트폴리오를 구성하는 것
- 포트폴리오를 최적화하기 위해 많은 알고리즘들이 나와 있음(local search, tabu search, simulated annealing, particle swarm optimization, evolutionary algorithm)
- 이와 같은 많은 최적화 알고리즘 중 진화 알고리즘(evolutionary algorithm)을 사용함

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

A hybrid approach to portfolio composition based on fundamental and technical indicators

- 제안한 시스템은 크게 Investors Simulator, Optimization, Data로 구성되었음
- Data 모듈은 Simulator에 의해 액세스되며 전략을 테스트함
- Optimization 모듈은 MOEA 알고리즘을 적용
- MOEA 알고리즘은 각 지표들에 가중치를 부여함으로써 여러 가지 전략을 세울 수 있음

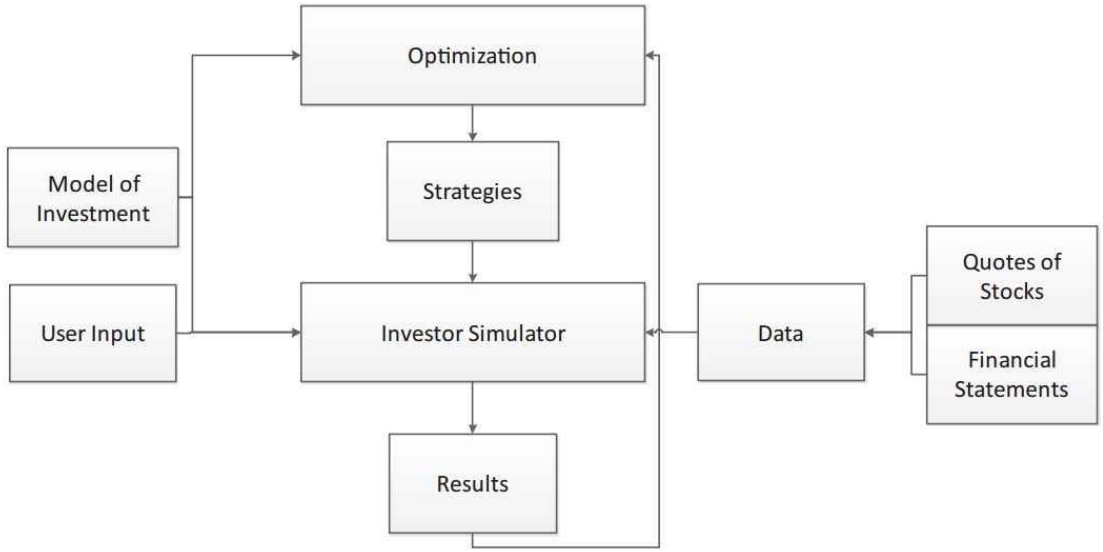


Fig. 1. System architecture.

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

A hybrid approach to portfolio composition based on fundamental and technical indicators

- Data 모듈은 회사의 재무재표와 주식 시세(quotes)를 사용하여 각종 비율(ratio)을 계산함
- 이러한 각종 비율은 investor simulator 모듈에 사용되며 optimization 모듈에서 생성된 전략을 테스트 함
- Data 모듈은 매일 주식을 평가, 누적 수익률을 구하고 월 수익률의 분산을 구함

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

A hybrid approach to portfolio composition based on fundamental and technical indicators

- 회사의 재무제표 정보를 이용해 퀀트 측정하고 각종 비율을 구해 일정기간 회사의 수익성(profitability), 유동성(liquidity), 채무(debt), 성장률(growth)을 구할 수 있음
- 각종 비율을 사용하면 같은 업종에 속해 있는 여러 회사들을 비교할 수 있음
- 펀더멘탈 각종 지표를 사용하는 목적은 회사를 평가하고 그 가운데서 가장 좋은 회사를 고르는데 있음

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

글로벌 값(Global Value)

- 각 주식의 글로벌 가치는 재무비율과 일일 PER를 사용하여 매일 계산되는 평가임
- 글로벌 값은 각 염색체의 가중치에 각각의 재무 비율을 곱한 것 임
- 글로벌 값을 구하는 공식은 아래와 같음(Ratio: fundamental 값)

$$Global_{value} = \sum_{i=1}^{n} weight_i * Ratio_i$$

  - n은 펀더멘탈 지수의 개수를 표시함
- 포트폴리오를 구성 시 글로벌 값이 큰 것을 사용함

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

손절 및 익절

- 손절 공식

$$P_{Stop} = P_{Entry} \times (1 - StopLoss)$$

- 익절 공식

$$P_{profit} = P_{Entry} \times (1 + TakeProfit)$$

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

첫 번째 염색체 모델

- 염색체 구조는 펀더멘탈 비율과 트레이딩 파라미터 두 가지 파트로 나눌 수 있음
- 7개의 펀더멘탈 가중치는 DR(Debt Ratio), ROE(Return on Equity), PM(Profit Margin Ratio), PER(Price Earnings Ratio), GR(Grwoth Rate), CSO(Common Stock Outstanding), NI(Net Income)
- 5개의 트레이딩 파라미터는 최소 글로벌 값, 손절선, 익절선, 주식선을 사용하여 각 자산에 자본 배분을 진행함

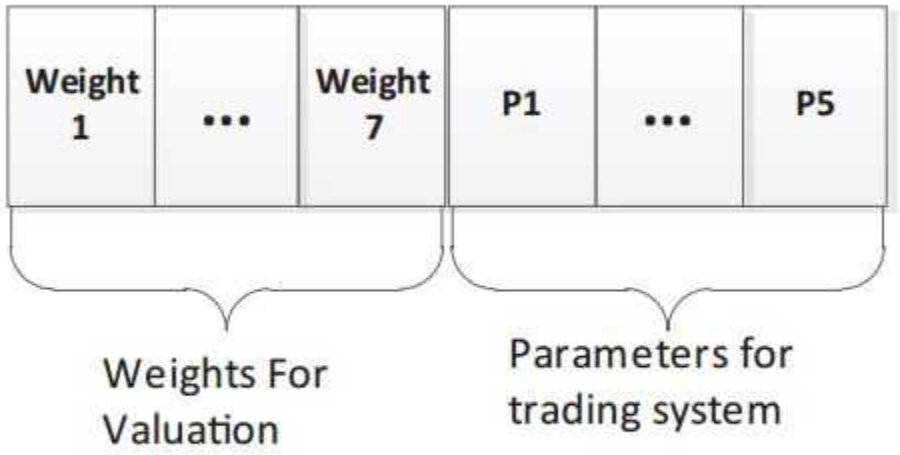


Fig. 13. Chromossome that represent the first model.

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

첫 번째 모델 결과

- 각 유전자의 값

Genes of the chromosomes.

Genes	C1	C27	C36	C51	C54
Debt ratio	0,53	0,53	0,28	0,28	0,28
ROE	0,64	0,12	2,24	0,12	2,24
profit margin	3,01	0,61	0,49	0,96	1,01
PER	0,54	0,40	0,23	0,23	0,27
Δ of Rev.	0,34	1,13	0,62	0,65	0,57
Δ Common stock out	1,33	1,08	1,04	0,97	1,25
Δ net income	1,79	2,07	1,24	1,70	1,95
Min Global Value	0,00	0,00	0,00	0,00	0,00
Stop Loss	1,11	0,88	1,98	0,88	0,82
Take profit	1,89	2,46	1,18	0,12	1,36
Days of MMA	1,00	5,00	4,00	1,00	1,00
Position size	0,20	0,20	0,20	0,05	0,05

각 주식을 5%씩
균등 보유

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

첫 번째 모델 결과

- 각 염색체와 인덱스의 수익률과 분산 결과표

Table 2
Results from the test.

	Return (%)	Variance (%)
SP&500	28.69	14,20
Chromosome ID		
1	50.24	19.71
27	45.83	17.70
36	38.81	14.28
51	13.66	4.94
54	22.73	6.25

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

첫 번째 모델 결과

- 훈련 데이터 기간: 2010-06-17 ~ 2012-01-03
- 알고리즘: MOEA와 두 개의 객체(수익률, 분산)
- 테스트 데이터 기간: 2012-01-04 ~ 2013-06-07

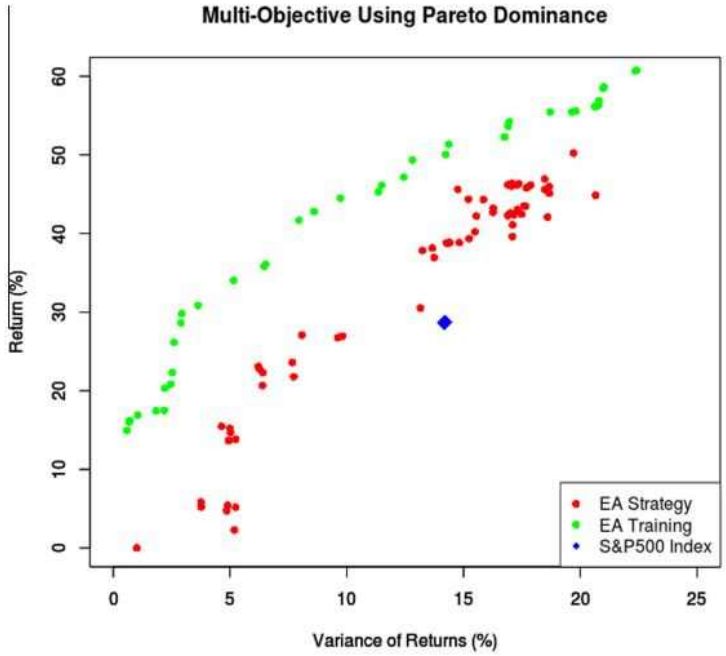
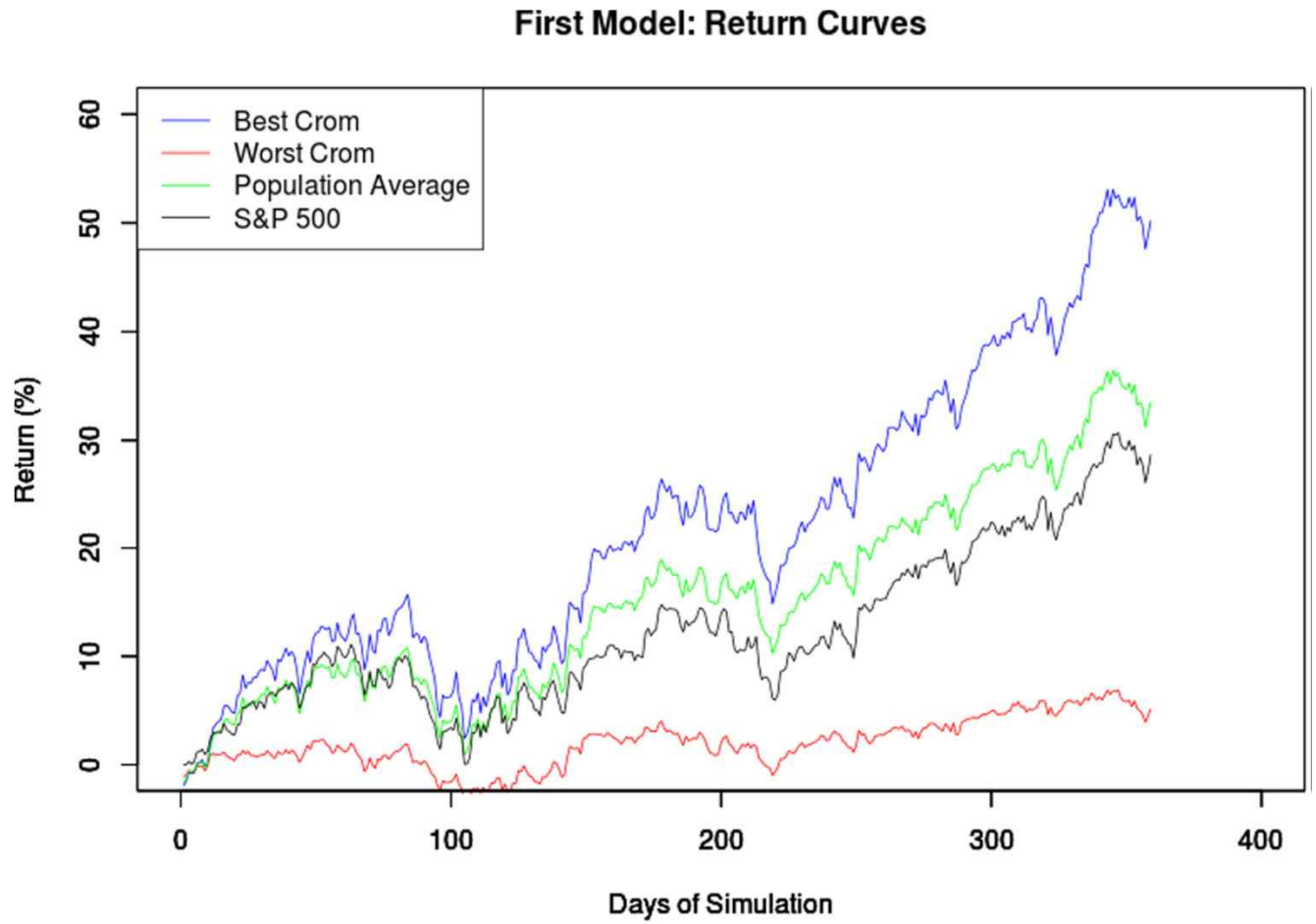


Fig. 17. Results from real test using the objectives return and variance of return.

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

첫번째 모델 수익률 곡선



출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

두 번째 염색체 구조

- 이 모델은 항상 결정된 수의 주식을 보유하는 포트폴리오를 시뮬레이션을 하기 위해 개발 되었음
- 이 모델의 목적은 가장 높은 값을 가진 포트폴리오의 주식을 보유하는 것

Fig. 15. Chromossome representing the second model.

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

두 번째 모델 설명

- 두 번째 모델 내용

Table 5
Genes of the Chromosomes.

Genes	C2	C10	C23	C29	C0
Debt ratio	0,000006	0,000002	0,017	0,0003	0,000002
ROE	0,00015	0,00015	0,31	0,0029	0,31
Profit margin	0,3578	0,0059	0,02617	0,2156	0,00286
PER	1,11	1,11	2,099	0,8637	2.099
Δ of Rev.	0,0739	0,0739	0.8549	1.0786	0,01089
Δ Common stock out	1,3989	1.3983	1.9047	1.0967	1.5
Δ Net income	2.9453	1,8516	0.331	0,043	0,638
Stocks number	8	7	9	6	4
Position size	0,18	0,18	0,063855	0,0638	0,0638

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

두 번째 모델 결과

- 두 번째 모델 테스트 결과표

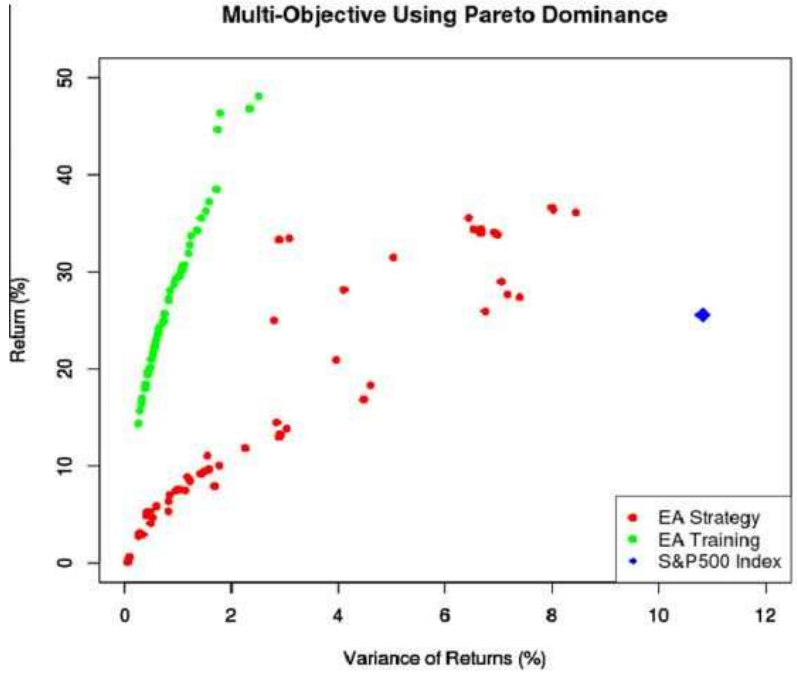
Table 4
Results of the real test.

	Return (%)	Variance (%)
SP&500	25.55	10.82
Chromosome ID		
2	36.4	8.02
10	27.59	7.18
23	18.31	4.58
29	11.84	2.25
40	8.89	1.16

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

두 번째 모델 결과

- 훈련 데이터 기간: 2010-06-17 ~ 2012-01-03
- 알고리즘: MOEA와 두 개의 객체(수익률, 분산)
- 테스트 데이터 기간: 2012-01-04 ~ 2013-06-07
- 두 번째 모델은 낮은 위험을 갖는 목적으로 만들었음



출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

두 번째 모델 결과

- 두 번째 모델 수익률 곡선

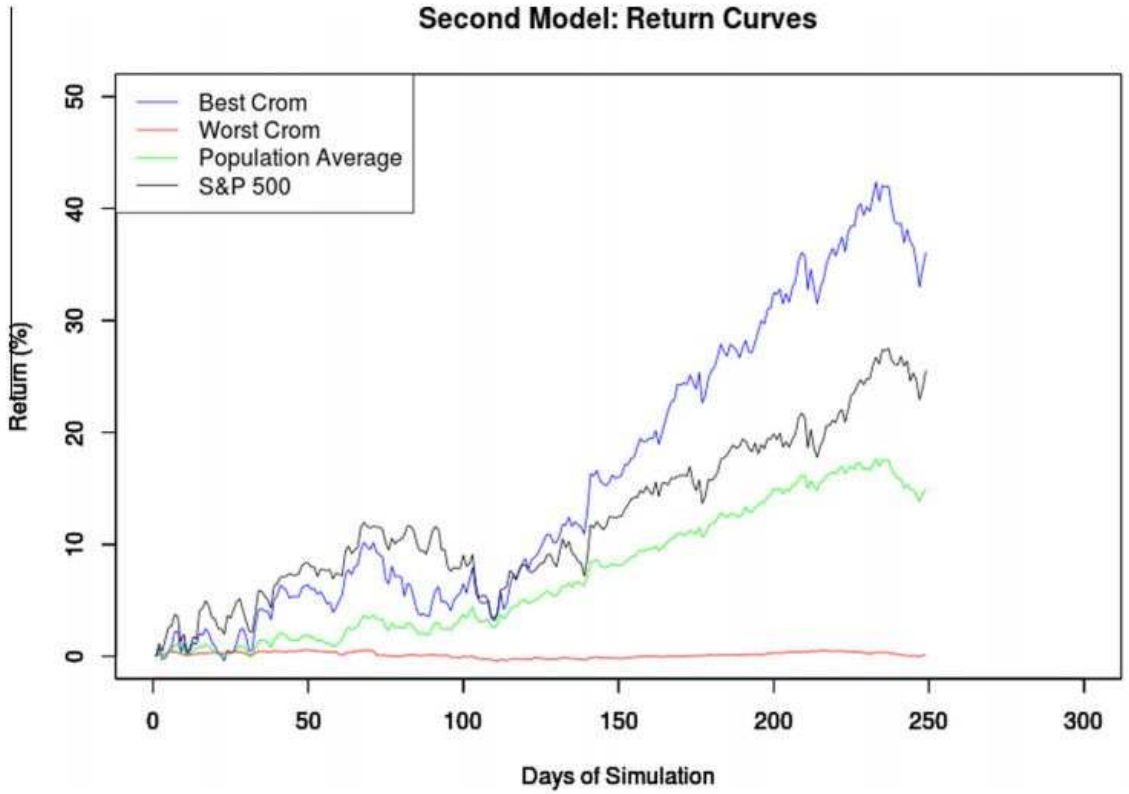


Fig. 20. Result of the real test using the second model.

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

세 번째 염색체 구조

- Payout ratio, CFOA, CE를 추가하여 10개의 펀더멘탈 지표를 사용

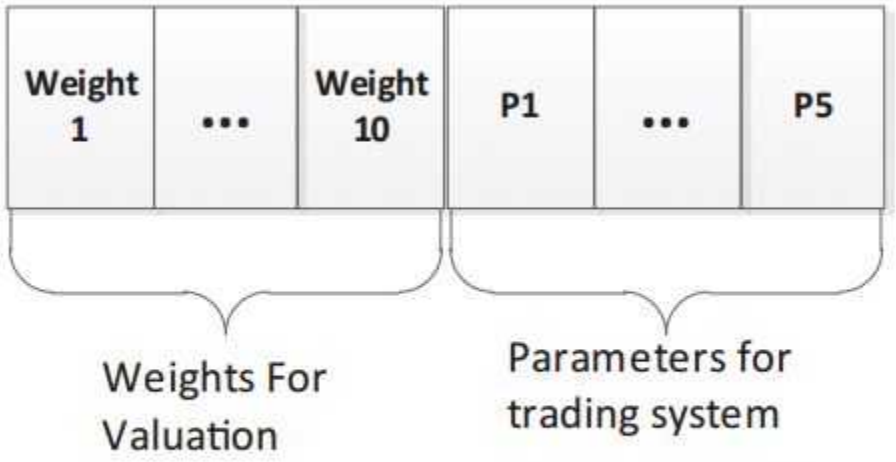


Fig. 16. Chromossome representing the third model.

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

세 번째 모델

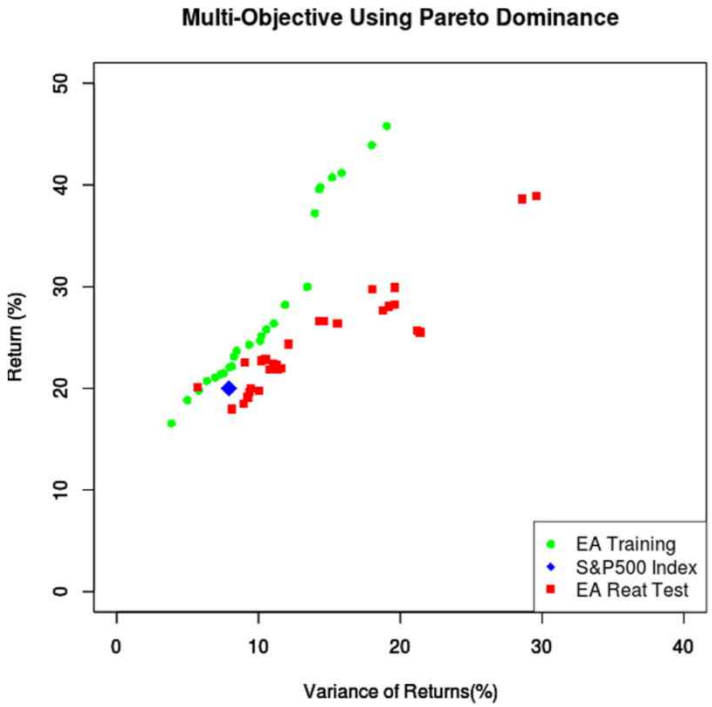
- 세 번째 모델

Genes	C4	C6	C7	C15	C29
PER	0,206	0,206	0,15	0,206	0,206
Debt ratio	0,754	0,751	0,868	0,869	1,053
ROE	2,360	0,985	1.2	1,390	2,335
Profit margin	2,309	1,239	2.31	2,011	0,724
Payout ratio	0,397	0,429	0,17	0,623	0,571
Δ of Rev.	0,170	0,319	0,375	0,236	0,480
Δ Common stock out	0,301	0,727	0,375	0,189	0,216
Δ Net income	0,341	0,730	0,341	0,730	0,621
ΔCE	1,182	0,227	1.41	1,778	0,227
ΔCFOA	0,995	0,769	0,756	1,308	1,434
MIN GLOBAL VALUE	0,355	0,430	0,493	0,554	0,462
Stop loss	0,635	0,359	0,635	0,233	0,310
Take profit	0,930	0,727	0,923	0,727	0,316
Days of MMA	0,029	6,264	0.03	0,043	0,156
Position size	0.2	0,2	0,2	0,034	0,045

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

세 번째 모델 결과

- 훈련 데이터 기간: 2010-06-17 ~ 2012-01-03
- 알고리즘: MOEA와 두 개의 객체(수익률, 분산)
- 테스트 데이터 기간: 2012-01-04 ~ 2013-06-07



출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

세 번째 모델 결과

- 세 번째 모델 테스트 결과표

Table 6
Results of the real test.

	Return (%)	Variance (%)
SP&500	20	7.89
Chromosome ID		
4	28.3	19.6
6	38.7	28.6
7	25.7	21.7
15	18.5	9.0
29	20.15	5.7

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

세 번째 모델 결과

- 세 번째 모델 수익률 곡선

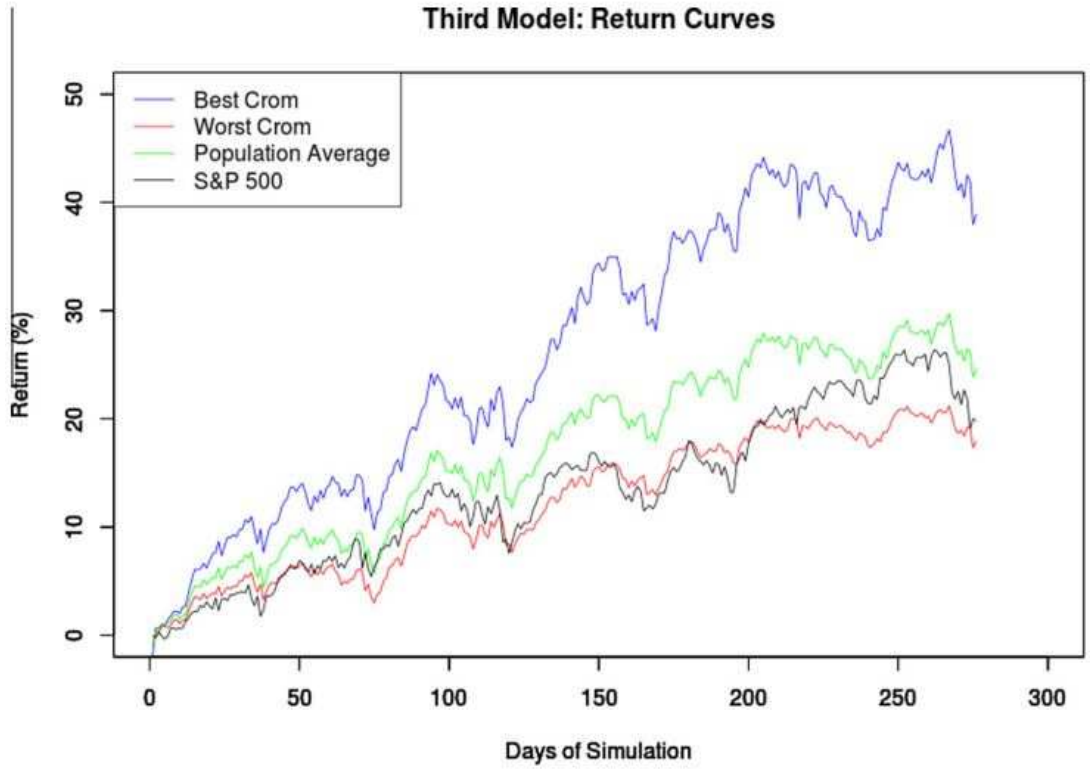


Fig. 22. Result of the real test using the first model.

출처: Antonio Silva, Rui Neves, Nuno Horta, "A hybrid approach to portfolio composition based on fundamental and technical indicators" Expert Systems with Applications 42 (2015) 2036–2048

Style Investing with Machine Learning

- 종목 선정 방법
  - 섹터별로 종목선정
  - Style Factor별로 다음달 수익률 예측
  - 예측 수익률이 높은 것부터 낮은 순으로 정렬
  - 5개 그룹으로 나누어 efficiency rate 측정 비교
- 예측 방법
  - Multiple Linear Regression
  - SVR

Table 2. Average efficiency rates of the linear versus the SVR model for different quintiles and time periods

		Quintile				
		1	2	3	4	5
Linear model	07/31/1996 to 07/31/2006	0.07	0.18	0.35	0.57	0.79
	07/31/2006 to 06/30/2016	-0.10	0.11	0.26	0.44	0.53
SVR model	07/31/1996 to 07/31/2006	-0.46	0.09	0.33	0.65	1.28
	07/31/2006 to 06/30/2016	-0.57	-0.12	0.25	0.56	1.14

출처: Philipp Kallerhoff, "Style Investing with Machine Learning," International Business Research Vol 9, No 12 (2016)

Style Investing with Machine Learning

Table 4. Statistics of the rolling linear and SVR models for each sector, the combined strategy (equal weighted) and the MSCI World Index (equal weighted) based on daily data between 01/01/1996 to 06/30/16

	Materials	Communication	Consumer, Cyclical	Consumer, Non-cyclical	Energy	Financial	Industrials	Technology	Utilities	Combined	MSCI World TR Index (equal weighted)
Linear model											
Annual Mean	0.11	0.15	0.09	0.11	0.12	0.11	0.12	0.13	0.10	0.12	0.09
Annual Vol	0.22	0.20	0.14	0.14	0.28	0.21	0.15	0.26	0.16	0.14	0.15
Eff. Ratio	0.49	0.78	0.64	0.81	0.43	0.52	0.81	0.49	0.66	0.82	0.62
Min Day	-0.09	-0.09	-0.06	-0.06	-0.15	-0.12	-0.06	-0.10	-0.11	-0.06	-0.06
Max Day	0.14	0.13	0.08	0.09	0.14	0.17	0.07	0.30	0.13	0.07	0.07
Skew	0.39	0.36	0.01	-0.07	-0.22	0.81	-0.15	1.43	0.31	-0.23	-0.24
Kurt	11.79	9.34	8.32	10.41	11.50	28.64	7.77	30.97	22.70	9.79	10.03
Max. Drawdown	0.61	0.61	0.43	0.40	0.70	0.70	0.44	0.86	0.29	0.42	0.56
SVR model											
Annual Mean	0.11	0.09	0.10	0.14	0.14	0.10	0.12	0.09	0.10	0.11	0.09
Annual Vol	0.22	0.23	0.19	0.15	0.26	0.20	0.20	0.32	0.16	0.15	0.15
Eff. Ratio	0.49	0.41	0.54	0.93	0.55	0.50	0.63	0.28	0.62	0.72	0.62
Min Day	-0.12	-0.09	-0.08	-0.08	-0.13	-0.09	-0.09	-0.23	-0.07	-0.06	-0.06
Max Day	0.10	0.08	0.12	0.08	0.15	0.10	0.10	0.14	0.08	0.07	0.07
Skew	-0.00	-0.14	0.14	-0.05	-0.18	0.15	-0.01	0.01	-0.23	-0.21	-0.24
Kurt	9.29	7.16	9.88	11.00	8.29	11.51	7.92	12.10	8.07	9.14	10.03
Max. Drawdown	0.62	0.83	0.65	0.54	0.58	0.69	0.57	0.90	0.39	0.52	0.56

출처: Philipp Kallerhoff, "Style Investing with Machine Learning," International Business Research Vol 9, No 12 (2016)

Style Investing with Machine Learning

- Combined Strategy
  - 각 섹터별 상위 수익률 종목 동일 가중 평균
  - 매월 리밸런싱
- Skew, Kurtosis와 Maximum Drawdown 모두 개선
- 선형회귀 성능 > SVR의 성능

출처: Philipp Kallerhoff, "Style Investing with Machine Learning," International Business Research Vol 9, No 12 (2016)

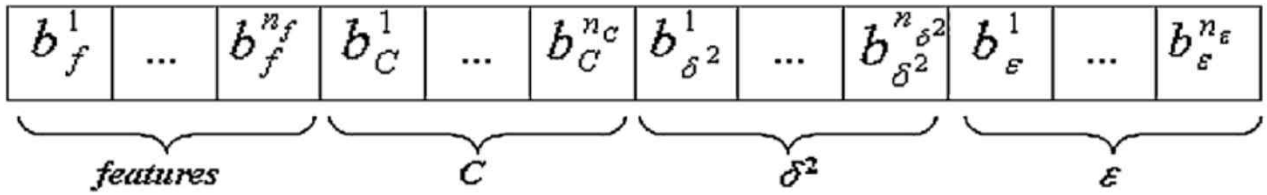
A hybrid stock selection model using genetic algorithms and support vector regression

- 예측 수익률 Top k개를 선택하여 포트폴리오 구성
- the SVR predicts the return of stock i at time *t* as $y_{i,t}$ (***F***, θ) $\in R$, for $i = 1, \ldots, n$,
- where
  - ***F*** **:** the set of input features
  - θ *:* kernel parameters

출처: Chien-Feng Huang, "A hybrid stock selection model using genetic algorithms and support vector regression," Applied Soft Computing 12 (2012) 807–818

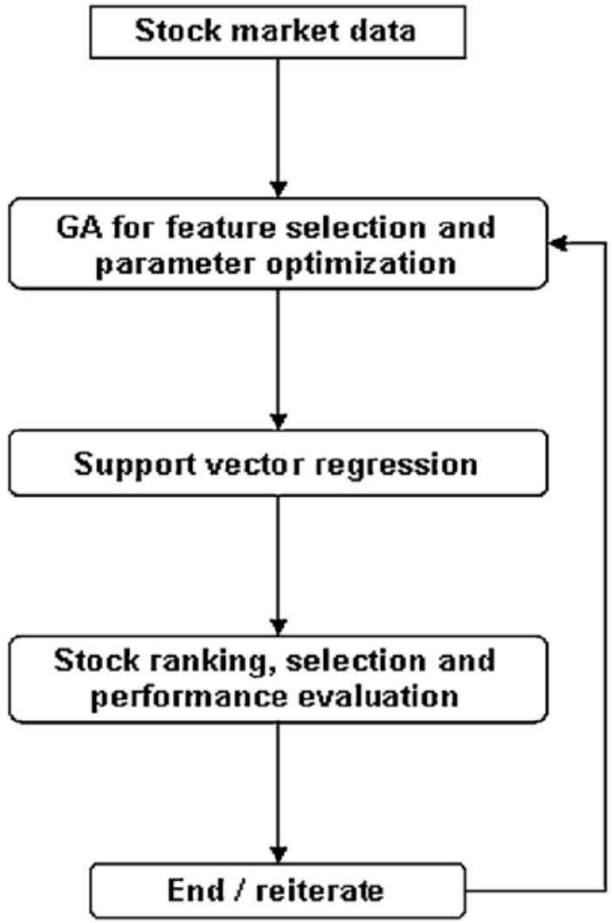
A hybrid stock selection model using genetic algorithms and support vector regression

- GA를 이용한 최적화
- 아래의 그림과 같이 feature 및 SVR의 패러미터인 C, δ^2, ε를 염록체의 구성요소로 하여 이 값들을 최적화



출처: Chien-Feng Huang, "A hybrid stock selection model using genetic algorithms and support vector regression," Applied Soft Computing 12 (2012) 807–818

A hybrid stock selection model using genetic algorithms and support vector regression



출처: Chien-Feng Huang, "A hybrid stock selection model using genetic algorithms and support vector regression," Applied Soft Computing 12 (2012) 807–818

A hybrid stock selection model using genetic algorithms and support vector regression

Table 1
Attributes used in the stock selection model.

Attribute	Ratios	Description	Refs.
Share price rationality	(1) PE ratio	Price-to-earnings ratio = share price/earnings per share	[38,40,41,43]
	(2) PB ratio	Price-to-book ratio = share price/book value per share	[38–42]
	(3) PS Ratio	Price-to-sales ratio = share price/sales per share	[38]
Profitability	(4) ROE	Return on equity (after tax) = net income after tax/shareholders' equity	[44,45]
	(5) ROA	Return on asset (after tax) = net income after tax/total assets	[44]
	(6) OPM	Operating profit margin = operating income/net sales	[46]
	(7) NPM	Net profit margin = net income after tax/net sales	[45]
Leverage	(8) DE ratio	Debt-to-equity ratio = total liabilities/shareholders' equity	[44]
Liquidity	(9) CR	Current ratio = current assets/current liabilities	[44]
	(10) QR	Quick ratio = quick assets/current liabilities	[44]
Efficiency	(11) ITR	Inventory turnover rate = cost of goods sold/average inventory	[44]
	(12) RTR	Receivables turnover rate = net credit sales/average accounts receivable	[47]
Growth	(13) OIG	Operating income growth rate = (operating income at the current year – operating income at the previous year)/operating income at the previous year	[48]
	(14) NIG	Net income growth rate = (net income after tax at the current year – net income after tax at the previous year)/net income after tax at the previous year	[49]

출처: Chien-Feng Huang, "A hybrid stock selection model using genetic algorithms and support vector regression," Applied Soft Computing 12 (2012) 807–818

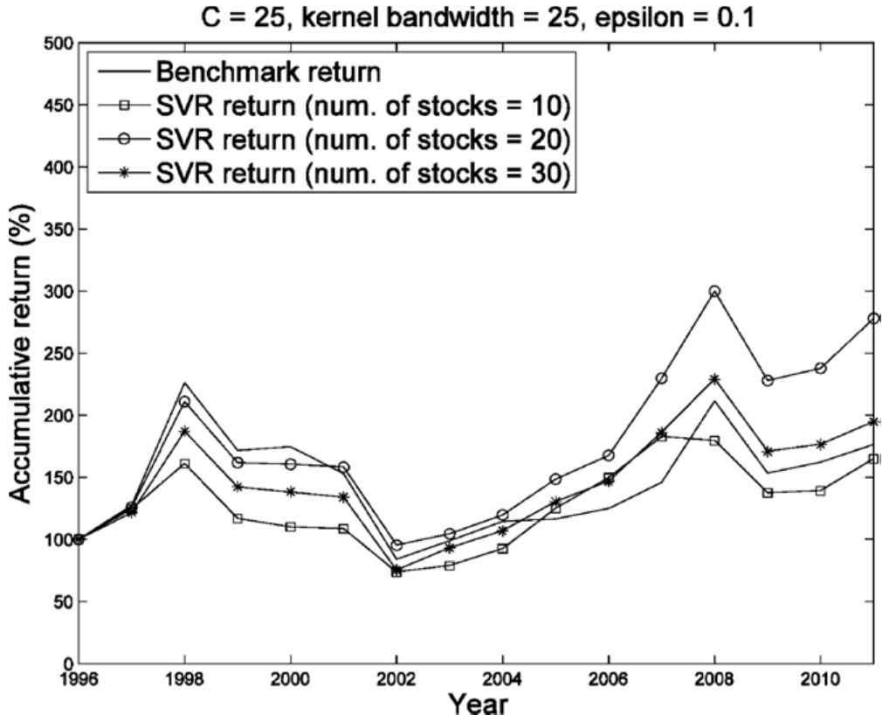
A hybrid stock selection model using genetic algorithms and support vector regression

- Comparative Study
  - (1) SVR using several illustrative values from the parameter ranges suggested by and all the features;
  - (2) SVR using parameters optimized by the GA and all the features;
  - (3) SVR using several illustrative values from the parameter ranges, but using an optimal subset of features selected by the GA;
  - and (4) SVR using both the optimal parameters and subsets of features optimized by the GA

출처: Chien-Feng Huang, "A hybrid stock selection model using genetic algorithms and support vector regression," Applied Soft Computing 12 (2012) 807–818

A hybrid stock selection model using genetic algorithms and support vector regression

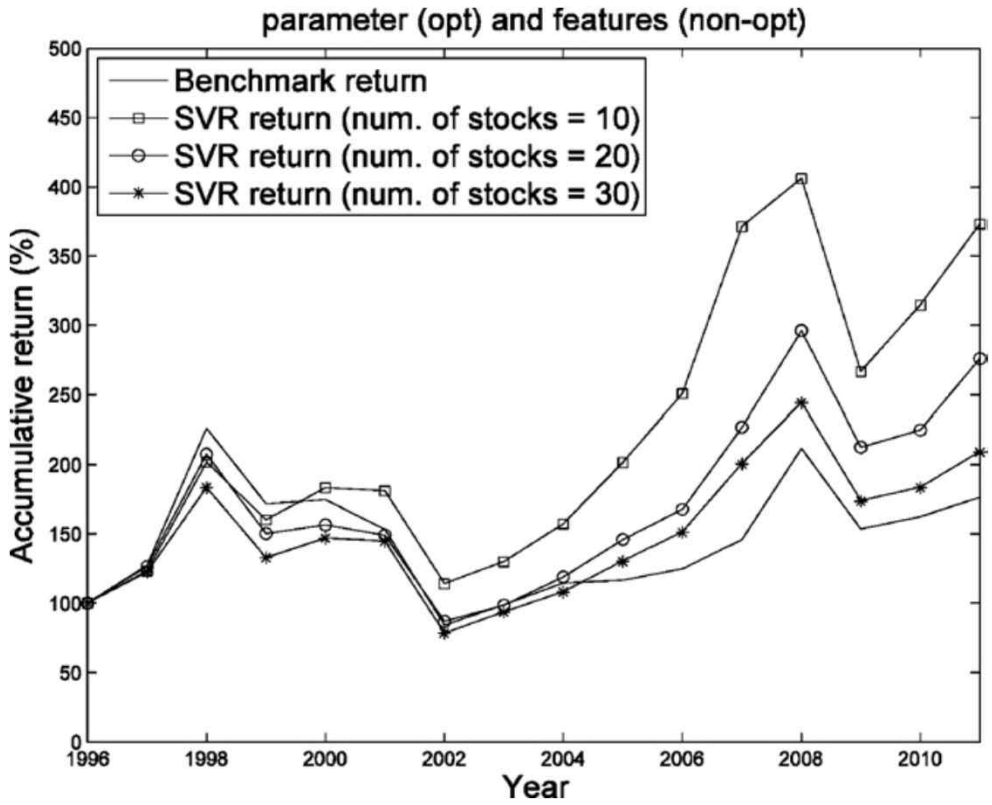
- SVR using non-optimized parameters and all the features



출처: Chien-Feng Huang, "A hybrid stock selection model using genetic algorithms and support vector regression," Applied Soft Computing 12 (2012) 807–818

A hybrid stock selection model using genetic algorithms and support vector regression

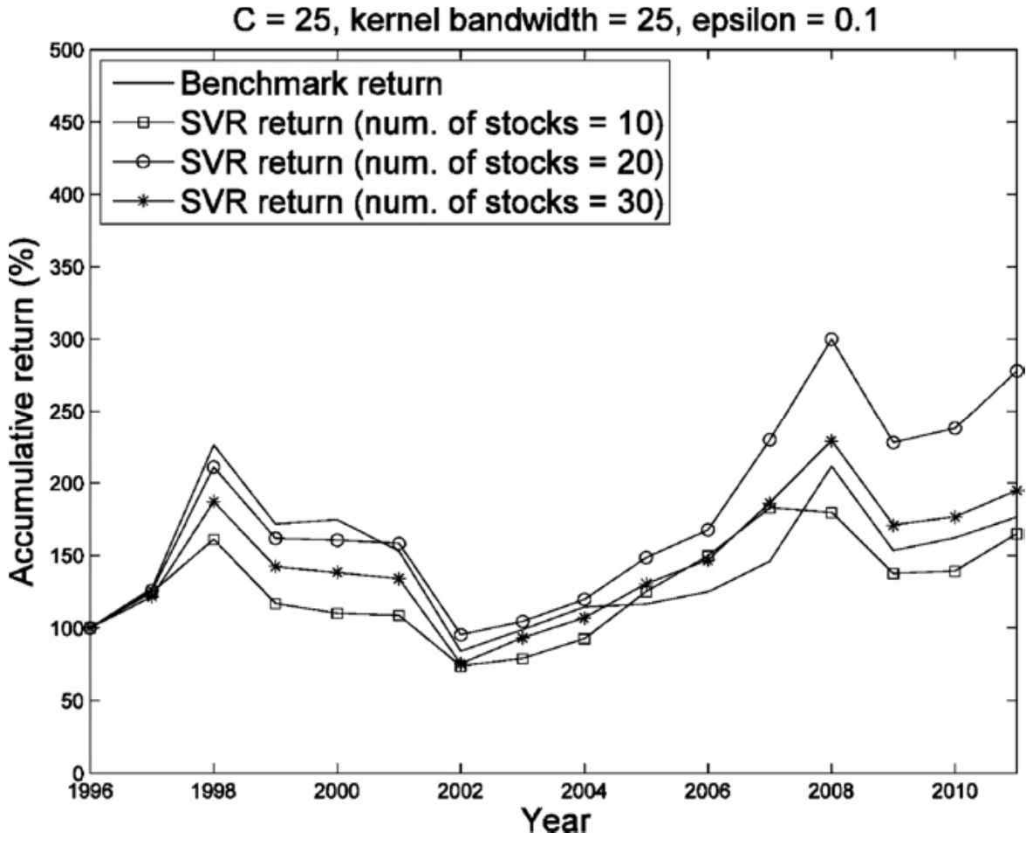
- SVR using non-optimized parameters and optimal subsets of features



출처: Chien-Feng Huang, "A hybrid stock selection model using genetic algorithms and support vector regression," Applied Soft Computing 12 (2012) 807–818

A hybrid stock selection model using genetic algorithms and support vector regression

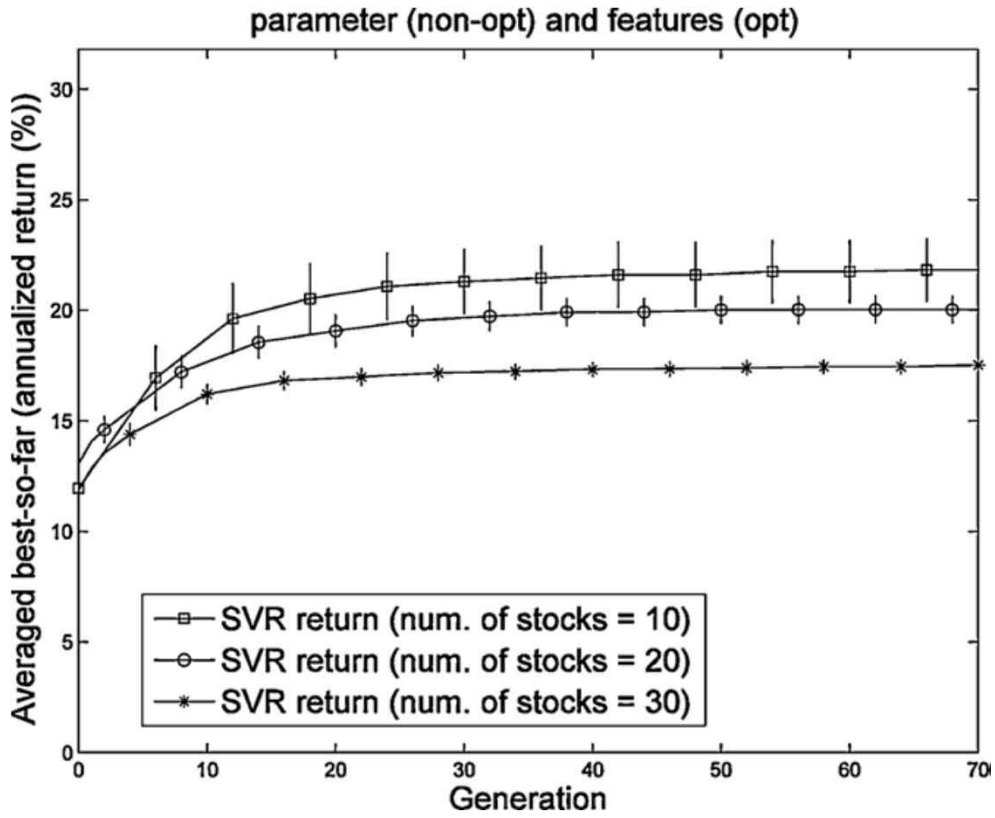
- SVR using optimized parameters and all the features



출처: Chien-Feng Huang, "A hybrid stock selection model using genetic algorithms and support vector regression," Applied Soft Computing 12 (2012) 807–818

A hybrid stock selection model using genetic algorithms and support vector regression

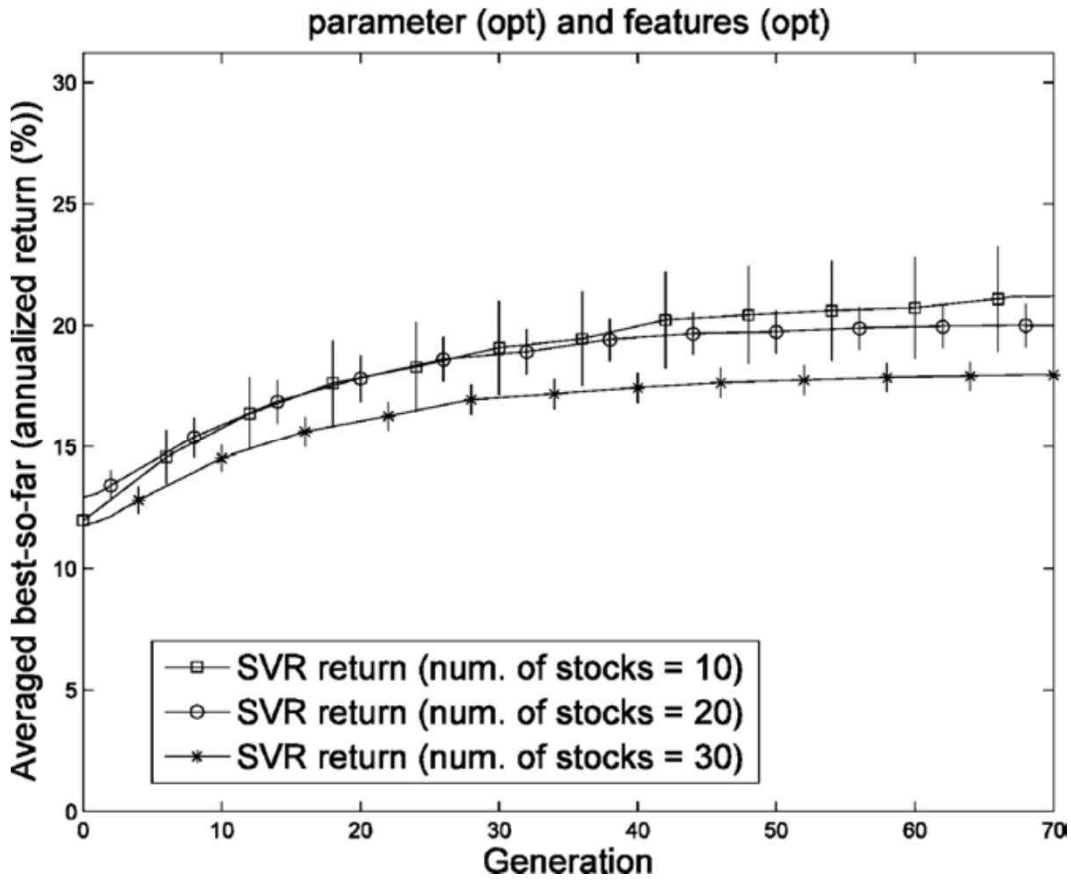
- GA 최적화 회수에 따른 종목 갯수별 최고 수익률 평균값의 변화



출처: Chien-Feng Huang, "A hybrid stock selection model using genetic algorithms and support vector regression," Applied Soft Computing 12 (2012) 807–818

A hybrid stock selection model using genetic algorithms and support vector regression

- GA 최적화 회수에 따른 종목 갯수별 최고 수익률 평균값의 변화



출처: Chien-Feng Huang, "A hybrid stock selection model using genetic algorithms and support vector regression," Applied Soft Computing 12 (2012) 807–818

A hybrid stock selection model using genetic algorithms and support vector regression

- 10개 주식에 대한 제안 모델의 성능

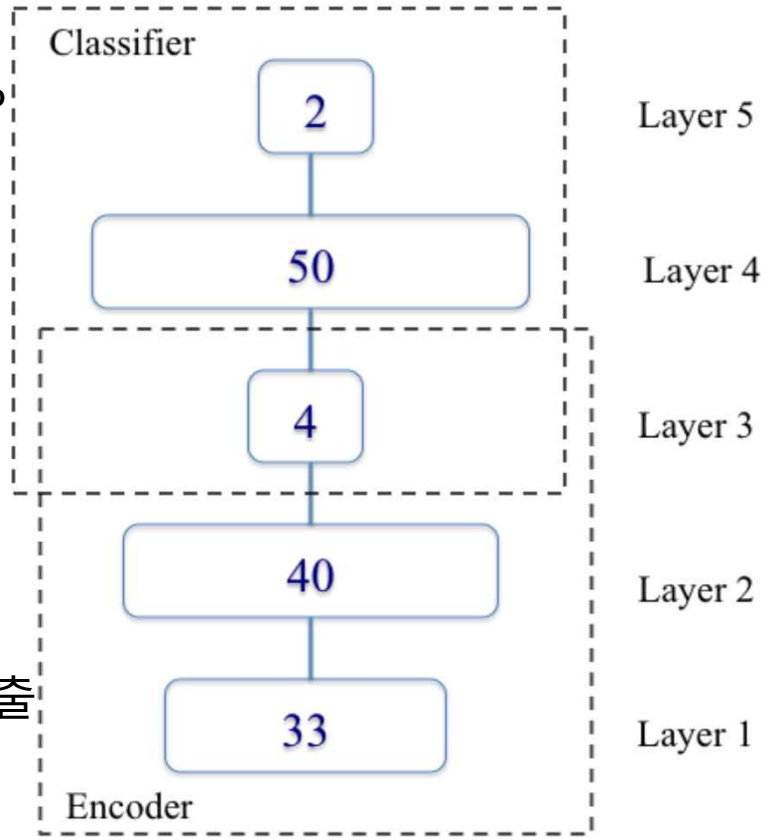
Table 3
Statistics of the benchmark and SVR–GA stock selection models for 10 stocks.

Case index	Training period	Annualized benchmark return (%)	Mean of annualized model return (%)	Variance of annualized model return (%)	Testing period	Annualized benchmark return (%)	Mean of annualized model return (%)	Variance of annualized model return (%)
1	1	27.2615	61.7676	5.5770	2–15	2.3489	5.2850	7.7316
2	1–2	50.4192	107.7513	1768.1	3–15	−1.9077	2.9041	22.9912
3	1–3	19.7369	41.7102	261.3085	4–15	0.2147	4.5146	37.3409
4	1–4	14.9839	39.6706	229.5432	5–15	0.0694	6.0917	25.3487
5	1–5	8.9005	24.9487	67.3695	6–15	1.4078	7.3426	19.7575
6	1–6	−2.8317	14.0150	49.0752	7–15	8.5513	13.6560	28.4809
7	1–7	−0.1497	19.5937	102.3874	8–15	7.4737	16.0384	26.6916
8	1–8	1.7285	24.4311	67.8455	9–15	6.3206	14.4583	18.5463
9	1–9	1.7348	29.2812	53.8386	10–15	7.0959	10.4337	12.8360
10	1–10	2.2505	29.7401	56.8345	11–15	7.1129	5.8753	19.9693
11	1–11	3.4922	33.5983	52.1087	12–15	4.8263	−1.6058	2.7269
12	1–12	6.4443	30.1030	52.3017	13–15	−5.9271	−5.4475	5.5000
13	1–13	3.3426	23.7139	29.7553	14–15	7.1805	14.0516	21.5848
14	1–14	3.5111	21.5515	35.0980	15	8.6541	22.1951	51.9317

출처: Chien-Feng Huang, “A hybrid stock selection model using genetic algorithms and support vector regression,” Applied Soft Computing 12 (2012) 807–818

Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks

예측치가
평균 이상 상승?
또는 이하 분류



Feature 자동 추출

출처: Lawrence Takeuchi and Yu-Ying (Albert) Lee, "Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks" Stanford Univ.

Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks

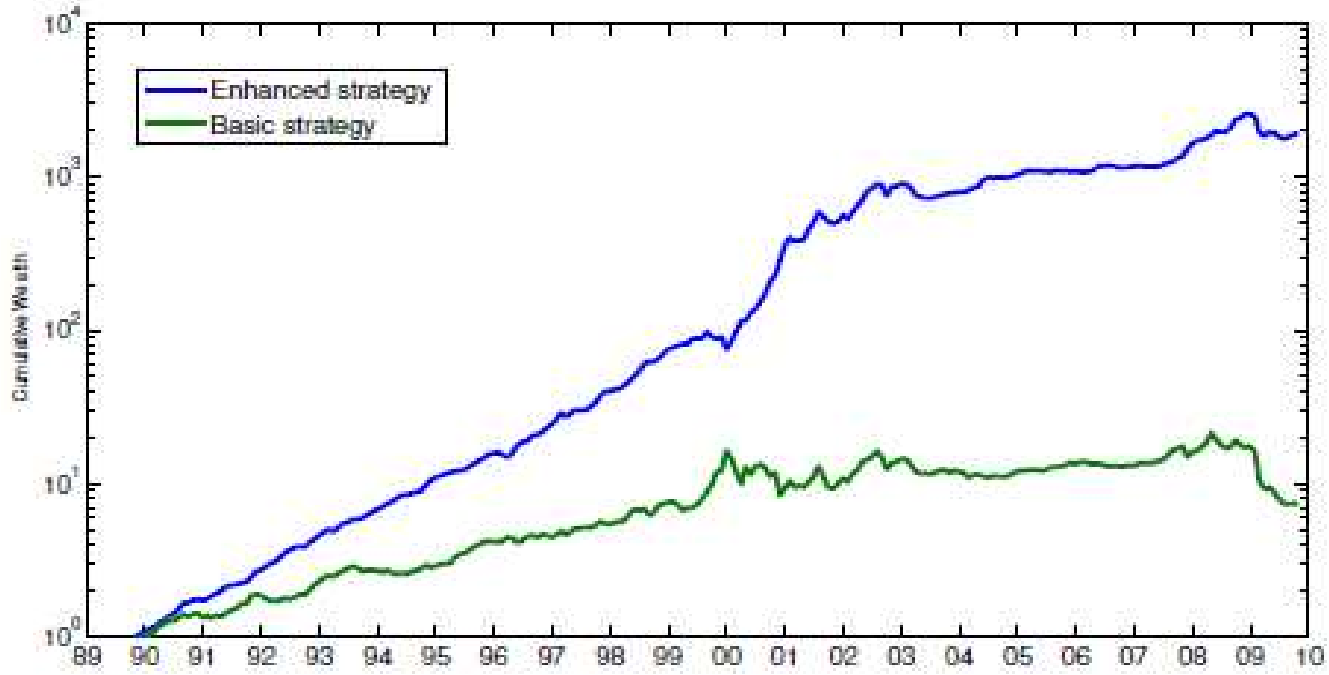


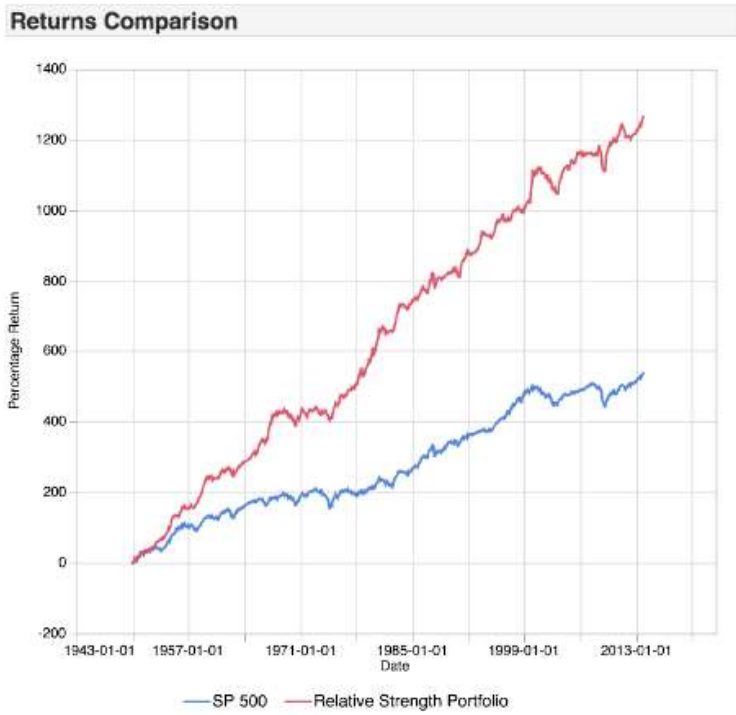
Figure 4. Log Growth in Cumulative Wealth

출처: Lawrence Takeuchi and Yu-Ying (Albert) Lee, ‘‘Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks” Stanford Univ.

Quantitative Tactical Asset Allocation Using Ensemble Machine Learning Methods

- 각 섹터별로 지난달의 월간 수익률을 구하고 그 중 최고의 수익률을 거둔 섹터를 buy
- 지금이 1월이라고 하면 지난 12월의 수익률을 구함

$$monthlyReturn = 100 * \left[\left(\frac{price(t-1)}{price(t-2)}\right) - 1\right]$$



출처: Kemal Oflus, "Quantitative Tactical Asset Allocation Using Ensemble Machine Learning Methods" http://ssrn.com/abstract=2438522

Quantitative Tactical Asset Allocation Using Ensemble Machine Learning Methods

- ETF로 투자할 경우의 시뮬레이션 결과

Table 4: Industry Descriptions

ETF	Industry/Index
XLY(4)	Consumer Discretionary
XLP(4)	Consumer Staples
XLE	Energy
XLF	Financials
XLV	Health Care
XLI	Industrials
XLB	Materials
XLK	Technology
XLU	Utilities
TLT(5)	20+ Year Treasury Bond
IYR	US Real Estate
EEM	MSCI Emerging Markets
EFA	MSCI EAFE

출처: Kemal Oflus, "Quantitative Tactical Asset Allocation Using Ensemble Machine Learning Methods" http://ssrn.com/abstract=2438522

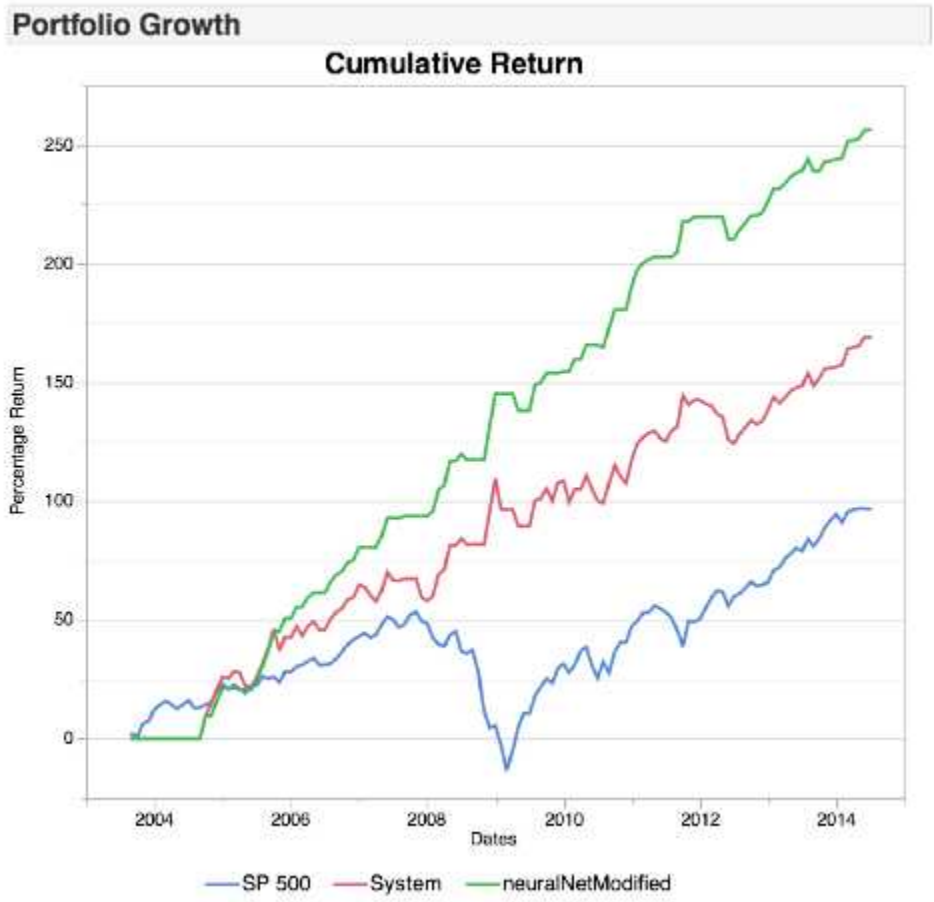
Quantitative Tactical Asset Allocation Using Ensemble Machine Learning Methods

- 모멘텀 Filter를 사용하는 경우
- Cumulative Return이 Moving Average 위에 있을 때에만 해당 섹터 buy
- 그렇지 않은 경우는 no position
- Improved CAGR(compound Annual Growth Rate) by 150 basis points by increasing from 9.04% to 10.58%.

출처: Kemal Oflus, "Quantitative Tactical Asset Allocation Using Ensemble Machine Learning Methods" http://ssrn.com/abstract=2438522

Quantitative Tactical Asset Allocation Using Ensemble Machine Learning Methods

- 앙상블 모델을 이용하여 주가 예측 후 섹터 선정(논문에서는 더 이상 상세 내용 없음)
- The ensemble model consists of Gradient boosted decision trees and neural network models



출처: Kemal Oflus, "Quantitative Tactical Asset Allocation Using Ensemble Machine Learning Methods" http://ssrn.com/abstract=2438522

포트 최적화에 AI Algorithm 적용 사례 다수

<약어표>

AES = Adaptive Exponential Smoothing,
AHP = Analytical Hierarchy Process,
ARIMA = Autoregressive Integrated Moving Average,
ARM = Association Rule Mining,
ARMS = Autoregressive Markov-Switching Model,
ARX = Autoregressive Exogenous,
EA = Evolutionary Algorithm,
ES = Exponential Smoothing,
FT = Fuzzy Theory,
GA = Genetic Algorithm,
GBM = Geometric Brownian Motion,
GMM = Generalized Method of Moments,
MOEA = Multi-Objective Evolutionary Algorithm,
MPM = Minmax Probability Machine,
NCP = Nadir Compromising Programming,
NSGA-II = Nondominated Sorting Genetic Algorithm II,
PESA-II = Pareto Envelope Based Selection,
PQP = Parametric Quadratic Programming,
PSO = Particle Swarm Optimization,
RBF = Radial Basis Function,
RS = Rough Set,
SA = Simulated Annealing,

TABLE I. TECHNIQUES USED FOR PORTFOLIO OPTIMIZATION BY DIFFERENT RESEARCHERS

Paper	GA	FT	PSO	Others
[4]	Y	-	-	TS, SA
[5]	Y	-	-	ANN, TS, SA
[6]	-	-	-	Lagrange multiplier theory in optimization, RBF ANN, ARIMA, AES
[8]	-	-	-	ARX
[9]	-	-	-	ARX, RS, GS
[10]	Y	-	-	-
[11]	Y	-	-	-
[12]	Y	-	-	-
[13]	Y	-	-	-
[14]	-	-	Y	-
[15]	Y	-	-	SMO
[16]	-	Y	-	RS, GS
[17]	-	Y	-	-
[18]	-	-	-	EA
[19]	-	-	-	ARX
[20]	-	Y	-	AHP
[21]	-	-	Y	-
[22]	-	-	-	NCP
[23]	-	-	Y	-
[24]	-	-	Y	-
[25]	-	-	-	Kernel Method
[26]	-	Y	-	-
[27]	-	Y	Y	Monte Carlo
[28]	-	Y	-	-
[29]	Y	Y	-	-
[30]	Y	Y	-	-
[31]	Y	-	-	-
[32]	-	Y	-	-
[33]	-	Y	-	Clustering, SOM
[34]	Y	-	-	-
[35]	-	Y	-	ARM
[36]	Y	-	-	SVR
[37]	Y	-	-	-
[38]	-	Y	-	-
[39]	Y	-	-	Hybrid of GA, SA
[40]	-	-	-	DEA
[41]	-	Y	-	-
[42]	Y	Y	-	-
[43]	-	-	Y	-
[44]	Y	-	-	-
[45]	Y	-	-	-
[46]	-	Y	-	MOEA
[47]	Y	-	-	-
[48]	Y	-	-	SA

출처: Mukesh Kumar Pareek and Priyank Thakkar, "Surveying Stock Market Portfolio Optimization Techniques," 2015 5th Nirma University International Conference on Engineering (NUiCONE)

Ⅳ. ML 기반
알고리즘 트레이딩 사례

System overview

- **An adaptive stock direction prediction system has been developed.**
- The system utilizes an artificial neural network to forecast the movement of future stock prices.
- Direction information is then used as trading signal in order to assist users in making trading decisions.
- The output from the neural network will be standardized between zero and one.

Forecasting methodology(adaptive methodology)

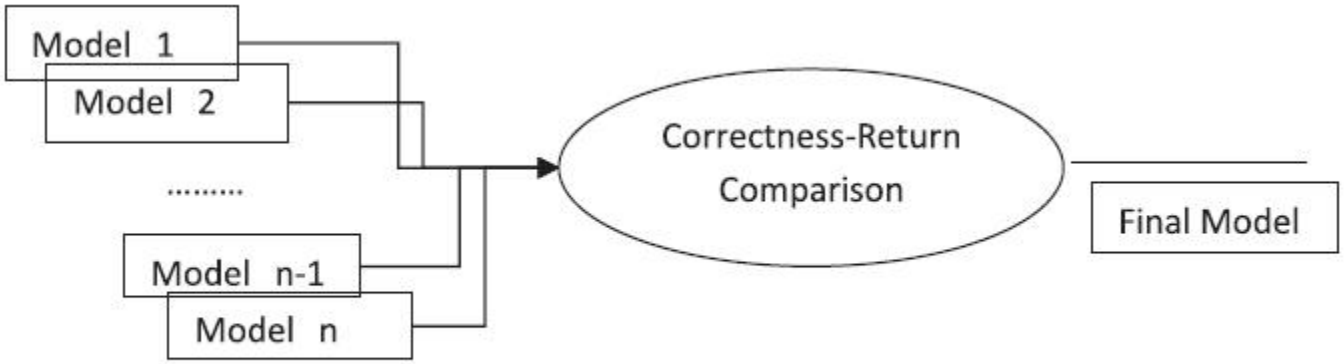


Fig. 3. Adaptive model selection.

Forecasting methodology(adaptive methodology)

- The accuracy of the prediction and network performance are greatly impacted by the selection of the inputs for the neural network.
- Initially, a general model was developed to describe the relationship among stock index variables.
- Principal component analysis, genetic algorithms, and decision trees are widely used for filtering representative variables.

Forecasting methodology(adaptive methodology)

- With N potential variables, the different possible combination of inputs for the neural network will reach 2^N -1, which can be a huge number with increasing N.

- Based on a number of experiments we conducted in our research, we find that most of the **best performing models have five variables or less.**

- As a result, we set the maximum number of the inputs for the neural network at five, such that the number of combination for inputs can be reduced to the following, where N is the number of potential variables.

$$C = C_N^1 + C_N^2 + C_N^3 + C_N^4 + C_N^5 \qquad (1)$$

System architecture

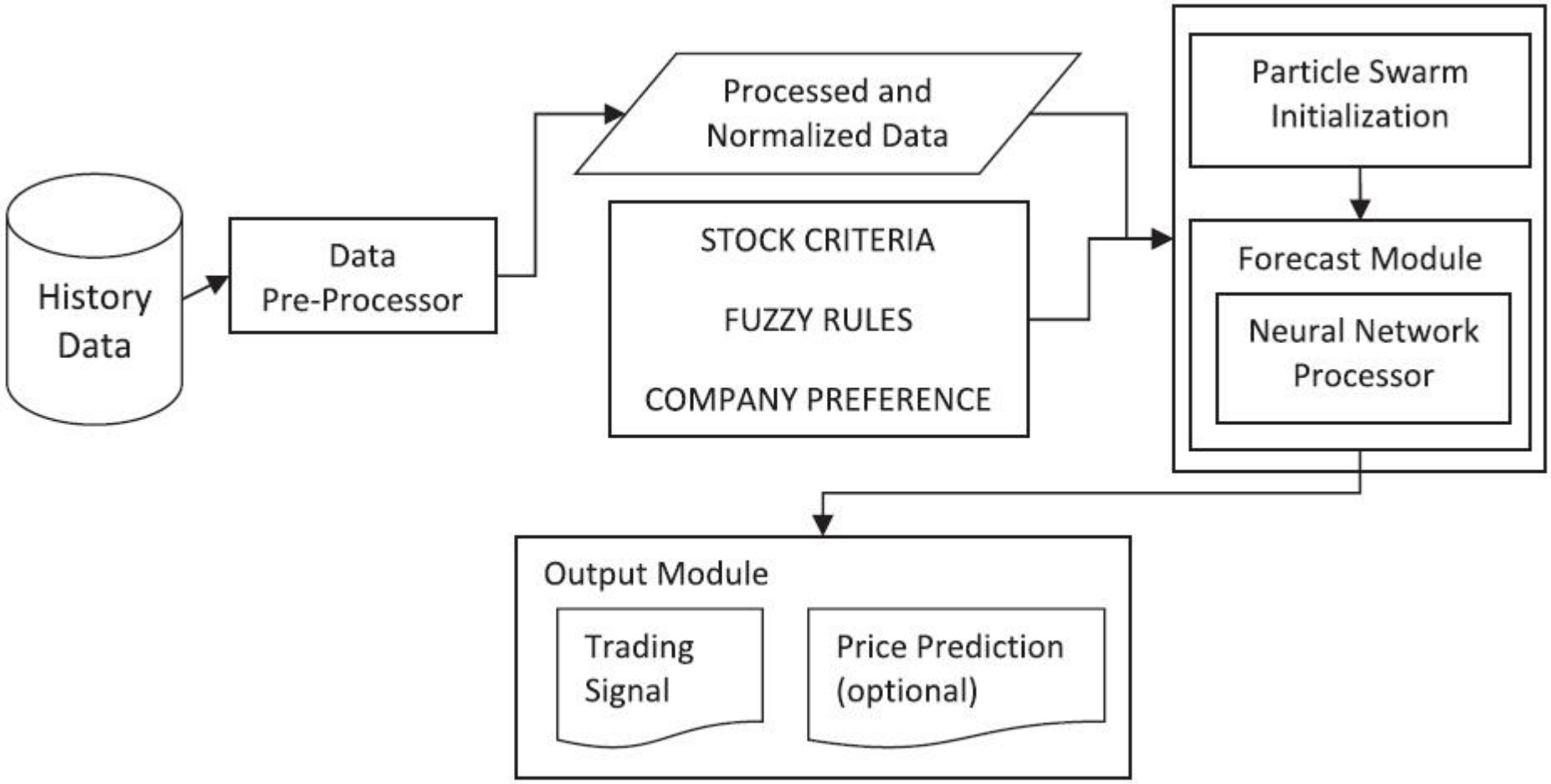


Fig. 4. System architecture.

System architecture(Particle swarm in neural network initialization module)

- In order to improve the efficiency, the **particle swarm algorithm** has been adopted in this module.
- The particle swarm algorithm has a larger searching space, which will help the neural network **avoid the local optimal**.
- **The output of the particle swarm optimization will be used as the initial weights for the neural network**.

System architecture(Output module)

- The trading signal will be presented in the output module.
- There are three different scenarios: buy, hold, and sell.
- **If traders are not holding any stock indexes when the direction prediction is greater than 0.5 (buy signal), they will buy the stock index, otherwise, the trader will do nothing.**
- **If the trader is holding the stock index when the direction prediction is greater than 0.5, they will hold the stock index, otherwise, when the direction prediction is less than 0.5 (sell signal), the stock index will be sold.**

Independent variables

- Economic variables are usually released monthly and quarterly and typically stay the same for a specific period.
- As a result, economic variables are excluded from our daily forecast.
- **For daily prediction, the most important parameters are the closing price and volume of the stock index for the last trading day.**

Independent variables

Table 1
Potential variables.

Variable name	Description
DP_{t-1}	The change of price from time t–1 and t–2
DP_{t-2}	The change of price from time t–2 and t–3
DV_{t-1}	The change of volume from time t–1 and t–2
DV_{t-2}	The change of volume from time t–2 and t–3
MA_{t-1} **(5 days)**	The moving average of the last 5 days at time t–1
MA_{t-1} **(10 days)**	The moving average of the last 10 days at time t–1
MA_{t-1} **(30 days)**	The moving average of the last 30 days at time t–1
MPP_{t-1} **(30 days)**	The moving price level in the last 30 days at time t–1
MPP_{t-1} **(120 days)**	The moving price level in the last 120 days at time t–1
PPO_{t-1}	The percentage price oscillator at time t–1

Independent variables(MA)

- Moving averages are a frequently used measurement in technical analysis, indicating short and long-term direction trends based on the period in which the average is calculated.
- The moving average of the previous five trading days' closing price is considered in the proposed system, providing a short-term trend analysis.
- This shorter period is better suited for daily trading. For the analysis, a simple Moving Average (MA), which is the non-weighted mean of the previous n data points, is calculated as
- where $P\ t{-}n$ is the stock price at time $t - n$.

$$MA_{t-1} = \frac{\sum_1^n P_{t-n}}{n} \qquad (3)$$

Independent variables(MPP)

- For this research, a Moving Price level Percentage (MPP) measure is used as an indicator for support and resistance.
- To obtain the MPP for the last K days, we initially calculate the increment between the current price and the minimum price for the last K days.
- Dividing the increment by the difference of the maximum and minimum stock index price over the K trading days gives the MPP.
- **A larger MPP suggests a higher level of price, and potential bigger level of resistance (smaller value otherwise).**

$$MPP = \frac{P - P_{min}}{P_{max} - P_{min}} \qquad (4)$$

Independent variables(PPO)

- The Percentage Price Oscillator (PPO) is a technical momentum indicator showing the relationship between the short-term and long-term averages.

- In a traditional view, a buy signal is issued when PPO measures above 0, while a sell signal is issued when PPO is below 0.

$$PPO = \frac{9\ Day\ EMA - 26\ Day\ EMA}{26\ Day\ EMA} \quad (5)$$

Independent variables(PPO)

- where EMA stands for the Exponential Moving Average.
- Closing prices and volumes were collected from finance. yahoo.com.
- The time series used was from January 29, 2004 to June 24, 2011. The various price and volume changes, as well as the chosen technical indicators, were calculated in Excel based on the collected data.
- In total, 12 variables are selected to form a potential parameter pool as shown in Table 1 .

Experimental results

- The proposed system trained the neural network with two years of daily data from January 2008 through December 2009.
- A twelve-month sample from January 2010 through December 2010 was retained to access the prediction performance of the neural network.
- In order to evaluate the performance of the neural network for generating trading signals, a Direction Correctness Percentage (DCP) was calculated.
- The DCP measures what percentage of the direction predictions are inline with the real direction changes in the historical data.
- If the DCP is greater than 50%, it may be possible to profit from the system.

Experimental results

- To illustrate how the proposed system adaptively develops the final model, the simulation results from testing the SPY (S&P 500 Index ETF) and IXIC (NASDAQ Composite Index) are used as examples.
- For each scenario, direction predictions were conducted over the 252 trading days in year 2010 (typically 365 days minus weekends and holidays).
- In those simulations, trading was conducted in accordance with the signals suggested by the proposed system.
- An accumulative return over the one-year period was then calculated.

Experimental results

Table 2
Selected simulation return for SPY.

SPY	Training data Jan 02, 2008 to Dec 31, 2009		Training data Mar 02, 2008 to Dec 31, 2009	
Neural network inputs	DCP	Return	DCP	Return
DPt–1, DPt–2, MPPt–1(30days), MPPt–2(120days), PPOt–1	56.75%	–0.36%	55.56%	–4.83%
Vt–1, Vt–2, Pt–1, Pt–2, MPPt–1(30days)	54.37%	9.81%	55.56%	2.39%
DVt–1, DVt–2, DPt–1, DPt–2, MPPt–1(30days)	53.97%	–0.62%	56.75%	2.25%
DVt–1, DPt–1, MPPt–1(30days)	55.56%	7.20%	53.97%	–8.76%
Vt–1, Pt–1, MPPt–1(30days)	53.17%	8.55%	54.37%	6.74%
Pt–1, Pt–2, MPPt–1(30days)	57.14%	13.67%	57.94%	11.00%
Pt–1, Pt–2, MPPt–1(30days), PPOt–1	54.76%	1.26%	56.75%	6.52%
MAt–1(5days), Pt–1, Pt–2, MPPt–1(30days)	57.94%	11.76%	55.56%	3.80%
MAt–1(5days), DPt–1, DPt–2, MPPt–1(30days)	57.54%	**14.70%**	57.14%	7.22%
MAt–1(5days), DPt–1, DPt–2	56.75%	12.04%	58.73%	11.05%
MAt–1(5days), DPt–1, DPt–2, PPOt–1	57.54%	7.41%	57.54%	5.49%
MAt–1(5days), DPt–1, DPt–2, MPPt–1(30days), MPPt–2(120days)	57.14%	–4.35%	58.33%	0.58%

Table 3
Selected simulation return for IXIC.

IXIC	Training data Jan 02, 2008 to Dec 31, 2009		Training data Mar 02, 2008 to Dec 31, 2009	
Neural network inputs	DCP	Return	DCP	Return
DPt–1, DPt–2, MPPt–1(30days), MPPt–2(120days), PPOt–1	55.16%	–6.21%	53.57%	–2.48%
Vt–1, Vt–2, Pt–1, Pt–2, MPPt–1(30days)	53.87%	7.25%	50.00%	–0.36%
DVt–1, DVt–2, DPt–1, DPt–2, MPPt–1(30days)	55.95%	8.72%	53.97%	1.74%
DVt–1, DPt–1, MPPt–1(30days)	57.54%	18.55%	58.33%	13.08%
Vt–1, Pt–1, MPPt–1(30days)	54.37%	8.64%	54.37%	8.83%
Pt–1, Pt–2, MPPt–1(30days)	55.56%	11.62%	52.78%	5.97%
Pt–1, Pt–2, MPPt–1(30days), PPOt–1	53.57%	1.31%	54.37%	11.77%
MAt–1(5days), Pt–1, Pt–2, MPPt–1(30days)	58.33%	18.21%	53.17%	12.31%
MAt–1(5days), DPt–1, DPt–2, MPPt–1(30days)	59.13%	16.67%	55.56%	7.54%
MAt–1(5days), DPt–1, DPt–2	59.52%	**20.24%**	56.35%	5.72%
MAt–1(5days), DPt–1, DPt–2, PPOt–1	60.32%	17.04%	57.14%	3.45%
MAt–1(5days), DPt–1, DPt–2, MPPt–1(30days), MPPt–2(120days)	56.35%	4.84%	55.16%	1.39%

Experimental results

- It can be clearly observed that the combination of parameters significantly affects the performance of prediction.
- Such results illustrate that a single model may not fit every stock index.
- Another interesting observation is that higher prediction accuracy does not always guarantee higher return, further highlighting why return is used as the criteria for determining the final predictive model.

Experimental results

Table 4
List of indices.

Index	Description	Index	Description
IXIC	NASDAQ Composite	**XLF**	Select Sector Financial
SPY	SP500 Index (large cap index)	**XLY**	Select Sector Consumer Discretionary
QQQ	Nasdaq 100 (large / medium tech stocks)	**XLE**	Select Sector Energy
MDY	S&P Midcap 400 (mid cap index)	**XLV**	Select Sector Health Care
IWM	iShares Russell 2000 index (small cap)	**XLK**	Select Sector Technology
FXI	iShares China	**SMH**	Semiconductor Holders
EFA	iShares EAFE Index, Europe	**OIH**	Oil Service Holders
EEM	iShares Emerging Markets Index	**FTSE**	FTSE 100 Index (UK)
HSI	Heng Seng Index (Hong Kong)	**ATX**	Austrian Traded Index in EUR
TWII	TSEC weighted index (Taiwan)	**FCHI**	CAC 40 (French)
BSESN	Bombay Stock Exchange (India)	**AEX**	Amsterdam Exchange index

Experimental results

Table 5
Simulation result for U.S. indices.

U.S. indices	DCP	Return (Signal)	Return (buy-hold)	ΔReturn	Improvement in %
IXIC	62.20%	**24.75%**	14.92%	9.83%	65.88%
SPY	61.11%	**20.34%**	13.69%	6.65%	48.57%
QQQ	58.33%	**29.09%**	19.29%	9.80%	50.80%
MDY	56.35%	**35.14%**	24.52%	10.62%	43.31%
IWM	53.97%	**31.48%**	25.60%	5.88%	22.96%

Table 6
Simulation result for selected sectors.

Selected sectors	DCP	Return (Signal)	Return (buy-hold)	ΔReturn	Improvement in %
XLE	53.17%	**34.01%**	20.75%	13.26%	63.90%
XLF	50.00%	**22.41%**	13.45%	8.96%	66.61%
XLK	59.13%	**20.26%**	11.15%	9.11%	81.70%
XLV	54.76%	**16.48%**	2.49%	13.99%	562%
XLY	54.37%	**27.50%**	24.98%	2.52%	10.08%

Experimental results

Table 7
Simulation result for iShares indices.

iShares indices	DCP	Return (Signal)	Return (buy-hold)	ΔReturn	Improvement in %
EEM	55.56%	**36.48%**	17.07%	19.41%	114%
EFA	54.76%	**30.09%**	8.13%	21.96%	270%
FXI	52.78%	**28.07%**	4.54%	23.53%	518%

Table 8
Simulation result for commodity indices.

Commodity	DCP	Return (Signal)	Return (buy-hold)	ΔReturn	Improvement in %
OIH	53.57%	**24.17%**	21.71%	2.46%	11.33%
SMH	52.78%	**29.66%**	18.74%	10.92%	58.27%

Experimental results

Table 9
Simulation result for international indices.

International indices	DCP	Return (Signal)	Return (buy-hold)	ΔReturn	Improvement in %
FTSE	51.98%	**20.15%**	12.57%	7.58%	60.30%
ATX	52.62%	**25.98%**	21.75%	4.23%	19.44%
FCHI	54.37%	**23.11%**	4.35%	18.76%	431.2%
AEX	54.37%	**28.64%**	10.53%	18.12%	172.1%
HSI	55.95%	**20.69%**	6.85%	13.84%	202.0%
TWII	57.94%	**29.13%**	15.31%	13.82%	90.26%
BSESN	55.95%	**27.58%**	13.24%	14.34%	108.3%

Experimental results

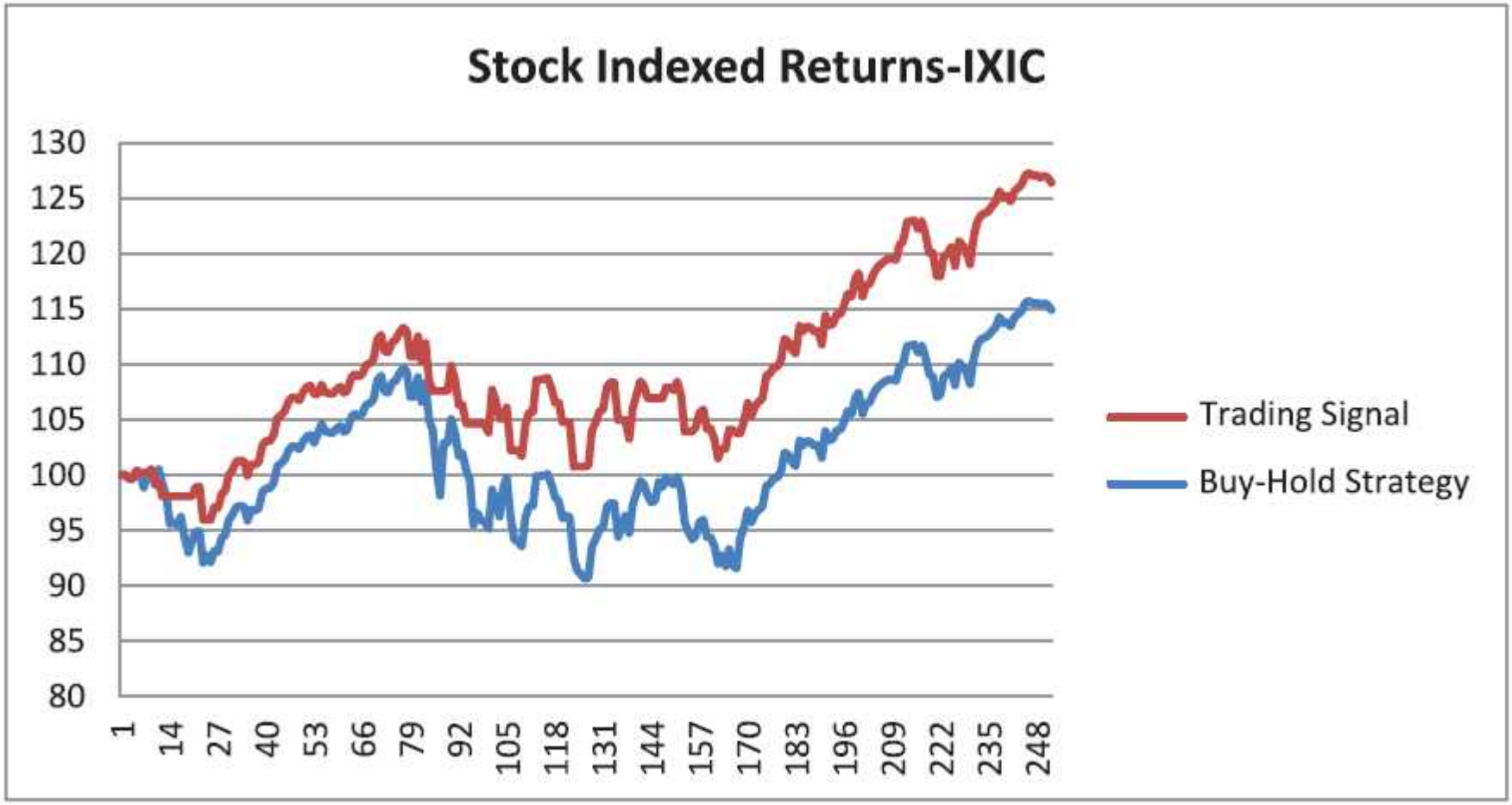


Fig. 5. Trading simulation result over time for IXIC.

Experimental results

- Investigation of the simulation results reveals that the system sometimes fails to recognize small price changes, but predicts the major changes in price movement.
- It can be observed that the system helped avoid several sharp downturns during the simulation, which occurred at the 12th, 78th, 89th, 122nd, 150th, and 160th days.
- Nonetheless, for this simulation there were still 50 transactions that occurred during the 252 trading days, including 25 buy and 25 sell signals.
- This suggest that even without small movement trades, excessive trading transition cost may still eliminate the benefits of the trading system in those cases where the additional return generated by the model was not significant.

Experimental results

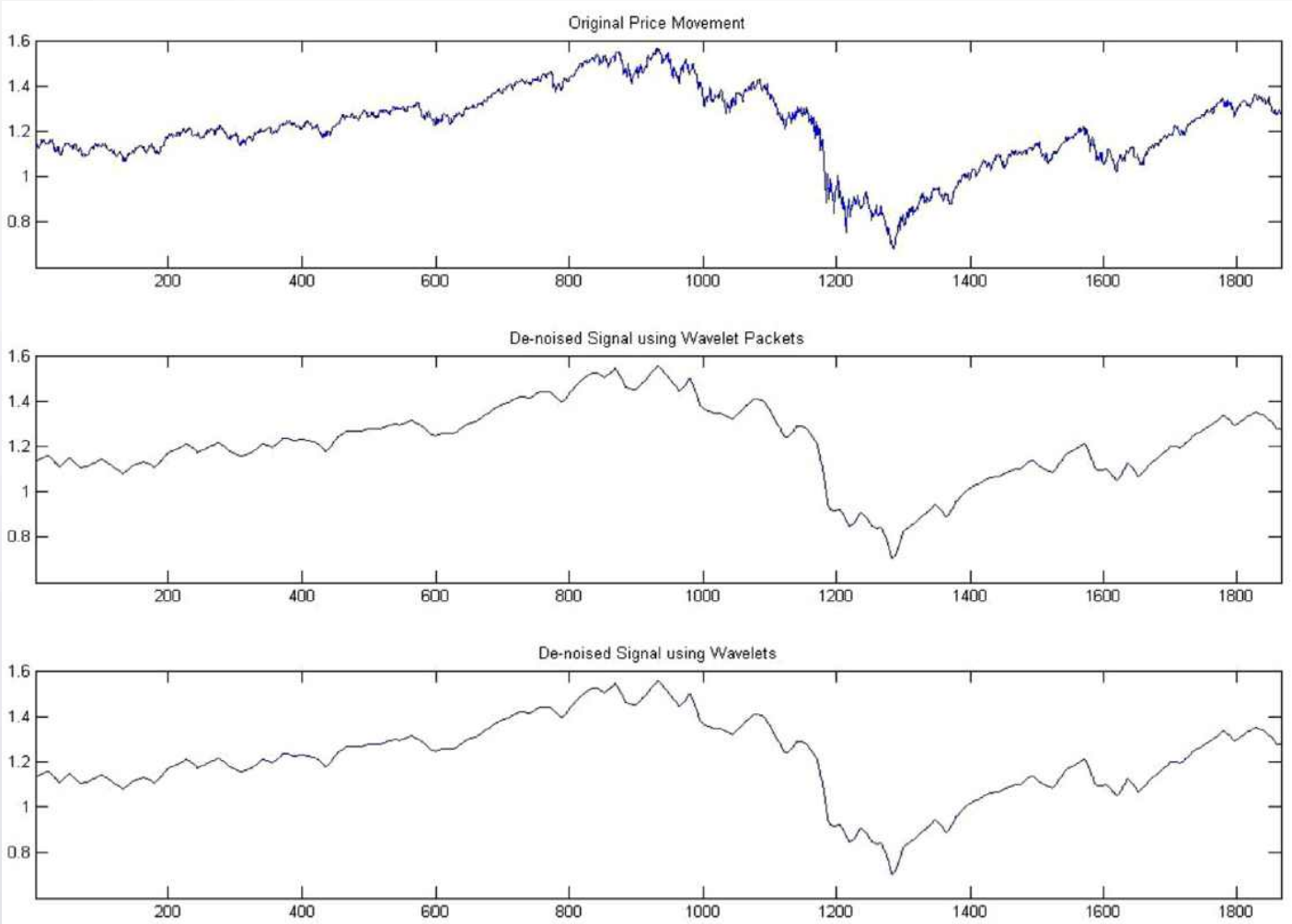
- This is due in part to various noises in the stock market, such as those caused by speculation and program trading, among others.
- In addition, selling stocks at every downturn will result in numerous transactions, generating higher transaction fees.
- As a result of the noise in the stock market, there exist meaningless movements during a short-term period.
- Therefore, as an additional test, **the data was smooth using a wavelet transformation before providing it to the neural network.**
- Wavelets are considered a very useful tool for extracting information from various types of data.

Experimental results

- where s(n) = noisy signal, f(n) = principal content, σ = volatility, and e(n) = the noise.
- The denoising process is designed to filter out the noise and recover signal f .
- Signal f is considered as useful data with less noise. Fig. 6 illustrates the transformation of the original data into de-noised data for the SPY ETF.
- From Fig. 6 , one can observe that the de-noised price and volume movement are smoothed.
- The trends of these data are thus clearer.
- **The system can be improved by ignoring small fluctuations but remain effective in detecting larger trends.**

$$s(n) = f(n) + \sigma e(n) \qquad (6)$$

Experimental results



Experimental results

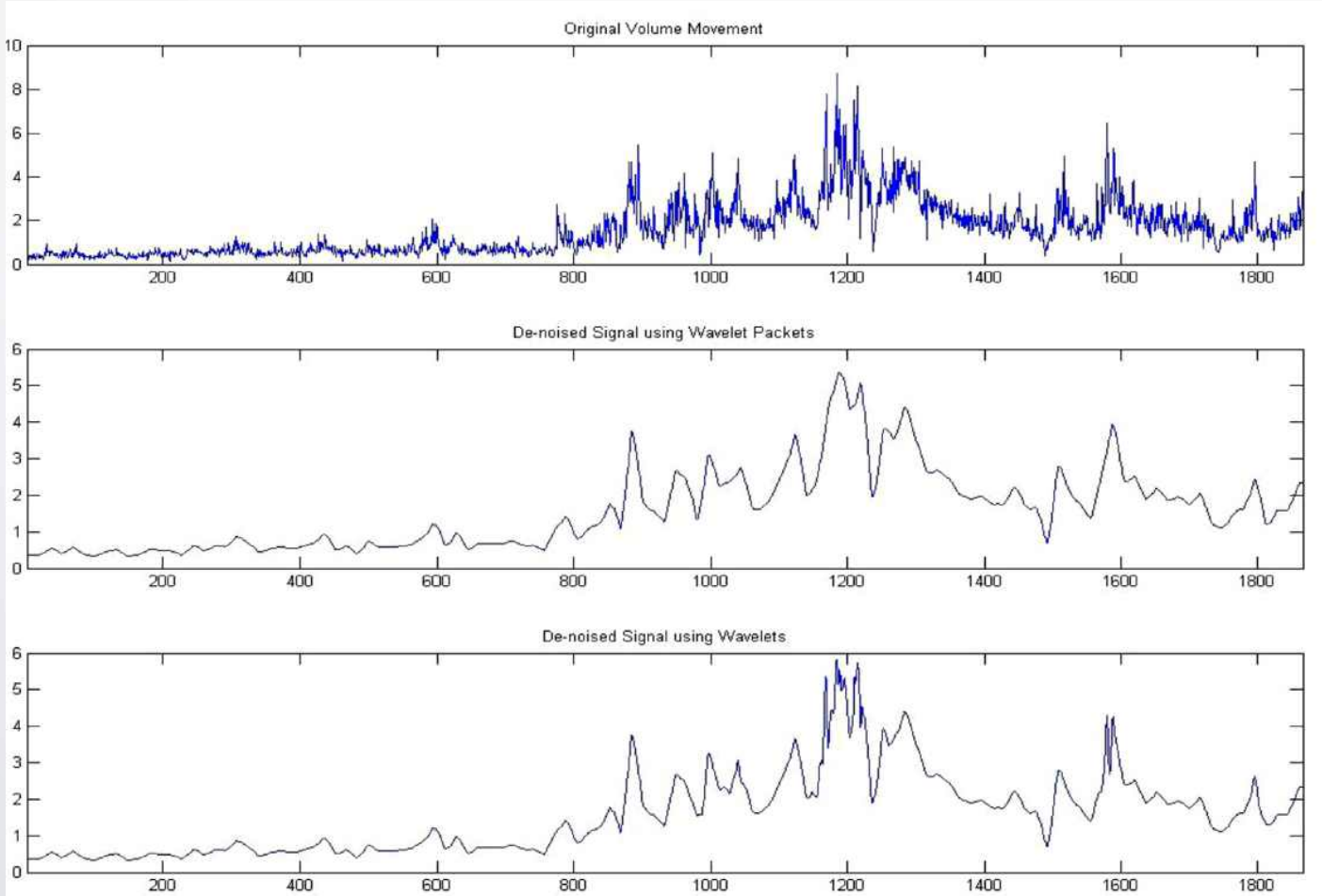


Fig. 6. Transformation of the original data into denoised data for the SPY ETF.

Experimental results

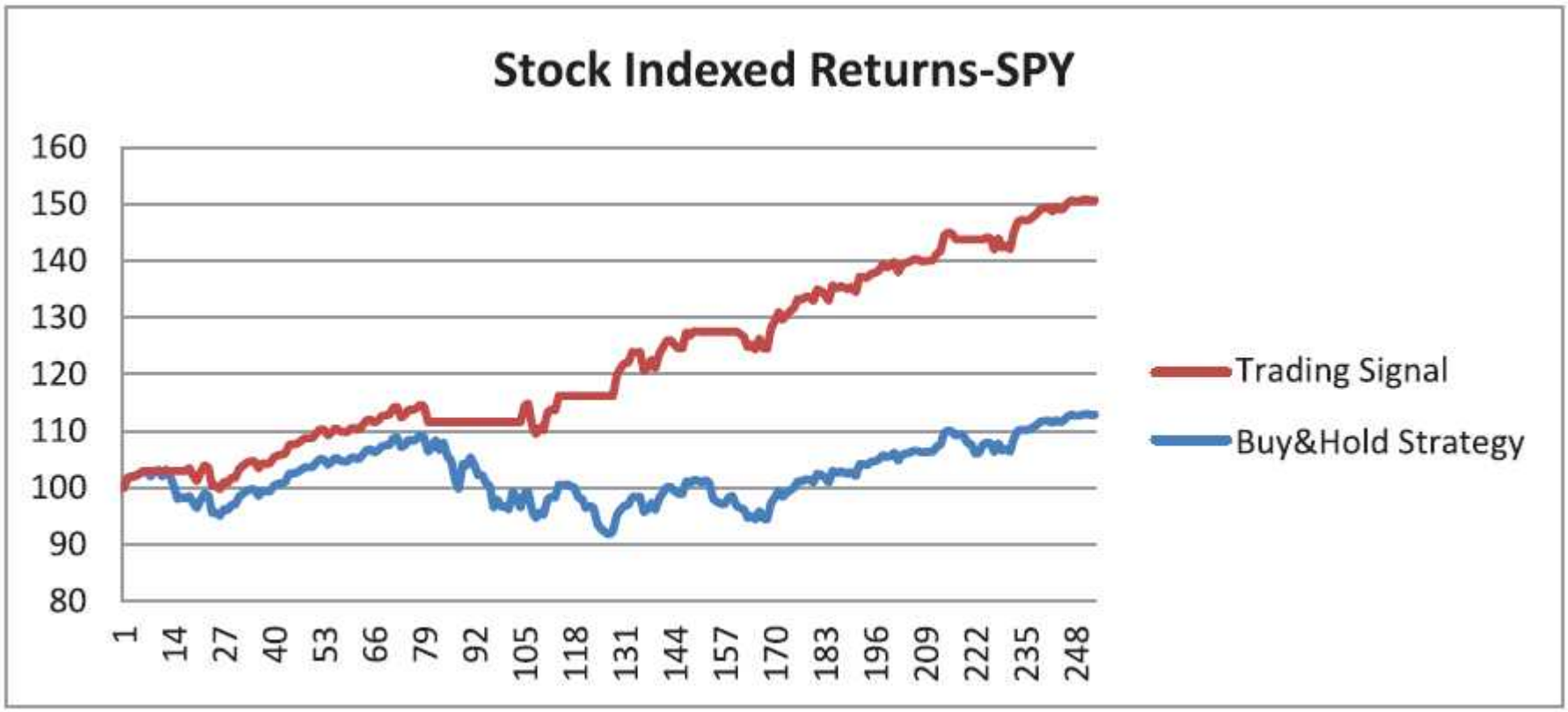


Fig. 7. Simulation trading result for SPY using denoised data.

Experimental results

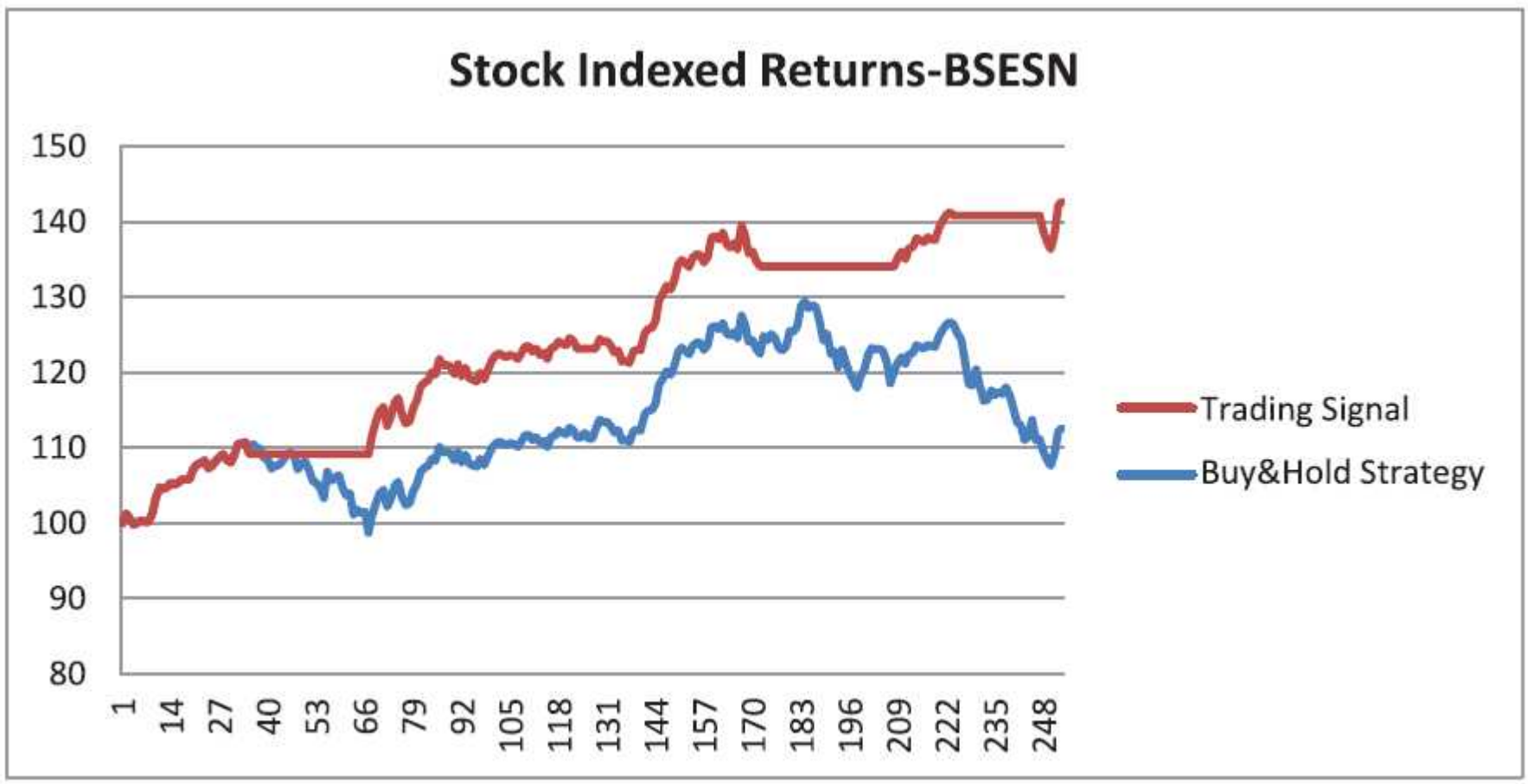


Fig. 8. Simulation trading result for BSESN using denoised data.

Experimental results

Table 10
Simulation result with the denoising approach.

Denoised

Indices	Return (buy-hold)	Return (Signal)	Return (Signal)	Number of Transactions (denoised based)
BSESN	13.24%	27.58%	**41.54%**	**10**
SPY	13.69%	20.34%	**41.89%**	**14**
EEM	17.06%	36.48%	**54.20%**	**18**
EFA	8.13%	30.09%	**50.42%**	**16**
IWM	25.60%	31.48%	**61.84%**	**12**
MDY	24.52%	35.14%	**54.29%**	**16**
QQQ	19.29%	29.09%	**50.78%**	**18**
XLF	13.45%	22.41%	**48.28%**	**20**
XLY	24.98%	27.50%	**59.24%**	**20**
AEX	10.53%	28.64%	**30.81%**	**12**
ATX	21.75%	25.98%	**55.91%**	**10**
FCHI	4.35%	23.11%	**42.88%**	**16**
FTSE	12.57%	20.15%	**34.61%**	**12**
HSI	6.85%	20.69%	**45.30%**	**12**
OIH	21.71%	24.17%	**71.08%**	**14**
SMH	18.74%	24.17%	**58.80%**	**16**
TWII	15.31%	29.13%	**39.05%**	**8**
XLE	20.75%	34.01%	**59.31%**	**10**
XLK	11.15%	20.26%	**46.67%**	**18**
XLV	2.49%	16.48%	**24.05%**	**12**
FXI	4.54%	28.07%	**54.40%**	**10**

Experimental results

- The number of transactions was on average14 (including both buying and selling), down from an average of 50 without denoising.
- As can be seen from Table 10 , the denoising approach helped the trading system in two aspects.
- First, the returns are greatly improved.
- Second, the number of transactions is dramatically lower, and hence the transaction costs are reduced.
- Often, when prices are noisy, there is an increase in the number of false signals.
- The use of denoising has the benefit of not only reducing the amount of unprofitable trades, but it does so by helping to better isolate the underlying trend, even in periods with noisy price action.
- This can have a positive impact in the areas of financial risk management and general trend forecasting.

Experimental results

Table 11
Simulation result with transaction costs.

	Buy-and-hold	Trading following signals	Denoising approach
Investment	$100,000	$100,000	$100,000
Ending Balance	$113,000	$128,000	$142,000
Transaction Cost	$20	$500	$140
Profit	$12,980	$27,500	$41,860
Adjusted Return	12.98%	27.50%	41.86%

Experimental results

Table 12
Simulation result with short selling.

Indices	Return (Signal)	Return (with Short)
BSESN	41.54%	61.14%
SPY	41.89%	81.77%
EEM	54.20%	82.81%
EFA	50.42%	78.62%
IWM	61.84%	85.32%
MDY	54.29%	83.64%
QQQ	50.78%	73.58%
XLF	48.28%	65.88%
XLY	59.24%	86.28%
AEX	30.81%	45.95%
ATX	55.91%	73.98%
FCHI	42.88%	73.00%
FTSE	34.61%	55.77%
HSI	45.30%	76.40%
OIH	71.08%	87.59%
SMH	58.80%	69.32%
TWII	39.05%	49.57%
XLE	59.31%	90.33%
XLK	46.67%	65.73%
XLV	24.05%	45.08%
FXI	54.40%	103.79%

Denoised

Conclusions

- The proposed adaptive model is unique in that it optimizes network performance for each individual stock index, and only uses those technical indictors that facilitate the best network performance.
- Different from other research approaches regarding stock and stock index selection, the adaptability and flexibility of the proposed system makes it applicable to real life situations, without the disadvantage of trying to develop a new system each time the input data and/or desired output change.
- Furthermore, rather than predicting the stock index price, a strength of the proposed system is that it generates trading signals by predicting the direction of the stock index price.
- The uses of denoising and short selling were also shown to increase trading returns by reducing transaction cost and increasing trading opportunities, respectively.

이동평균선 기준 매수, 매도 전략

- 이 절에서는 15일 이동평균선을 사용하여 매수, 매도를 하는 방법에 대한 투자 성과를 측정함

이동평균선 기준 매수, 매도 전략

- 추세(Trend) 분석하는 소스코드

In [12]:
```
import pandas as pd
import numpy as np
import pandas.io.data as data
from datetime import datetime
import matplotlib.pyplot as plt
%matplotlib inline

start = datetime(2015, 11 ,9)
end = datetime(2016, 11, 18)
symbol = '000660.KS'
f = data.get_data_yahoo(symbol, start=start, end=end)

f = f.drop(f.columns[[0,1,2,4,5]], axis =1)
f['MA15'] = pd.rolling_mean(f['Close'], 15)
f = f.dropna()
df = f.copy()
f['Status'] = np.where(f['Close'] > f['MA15'], 1, 0)
f['Action'] = np.where(f['Status'].rolling(window = 5).sum() == 5, 'Up'
                       , np.where(f['Status'].rolling(window = 5).sum() == 0, 'Down', 'No'))

date_list = []
for i in f.index:
    date_list.append(i.strftime('%Y-%m-%d'))
f['Date'] = date_list
f = f.reset_index(drop = True)
```

이동평균선 기준 매수, 매도 전략

- 트레이딩 시그널(Trading Signal) 생성하는 소스코드

```
date_list = []
for i in f.index:
    date_list.append(i.strftime('%Y-%m-%d'))
f['Date'] = date_list
f = f.reset_index(drop = True)

trading_signal = []
for i in f.index:
    if f['Action'][i] == 'Up':
        if i+2 >= len(f) or i+1 >= len(f):
            trading_signal.append(0.0)
        else:
            max = f['Close'][i:i+3].rolling(window = 3).max()
            min = f['Close'][i:i+3].rolling(window = 3).min()
            trading = round((f['Close'][i] - min[i+2]) / (max[i+2] - min[i+2]) * 0.5 + 0.5, 4)
            trading_signal.append(trading)
    elif f['Action'][i] == 'Down':
        if i+2 >= len(f) or i+1 >= len(f):
            trading_signal.append(0.0)
        else:
            max = f['Close'][i:i+3].rolling(window = 3).max()
            min = f['Close'][i:i+3].rolling(window = 3).min()
            trading = round((f['Close'][i] - min[i+2]) / (max[i+2] - min[i+2]) * 0.5, 4)
            trading_signal.append(trading)
    else:
        trading_signal.append(0.0)

f['Trading signal'] = trading_signal
```

이동평균선 기준 매수, 매도 전략

- 트레이딩 시스널을 이용해 사고 팔기(Buy and Sell)하는 소스코드

```
return_list = []
buy_date_list = []
sell_date_list = []
index = 0
count = 0
for i in f.index:
    if index == 0:
        if(f['Trading signal'][i] >= 0.5):
            print('Buy')
            buy_price = f['Close'][i]
            print('Buy Price: '+str(buy_price))
            buy_date_list.append(i)
            index = 1
            print(i)
            print(buy_price)

    elif index == 1:
        #print('Index number:' +str(i)+'/'+str(len(f)))
        if(i == len(f)-1):
            sell_price = f['Close'][i]
            sell_date_list.append(i)
            return_list.append((sell_price - buy_price) / buy_price)
            print('It is final sell')
        else:
            if(f['Trading signal'][i] == 0):
                count += 1
                print('Hold')
                print('The count :' +str(count))

            elif(f['Trading signal'][i] < 0.5):
                print('Sell')
                sell_price = f['Close'][i]
                print('Sell Price: '+str(sell_price))
                index = 0
                sell_date_list.append(i)
                return_list.append((sell_price - buy_price) / buy_price)


print(str(sum(return_list) * 100) +' %')
```

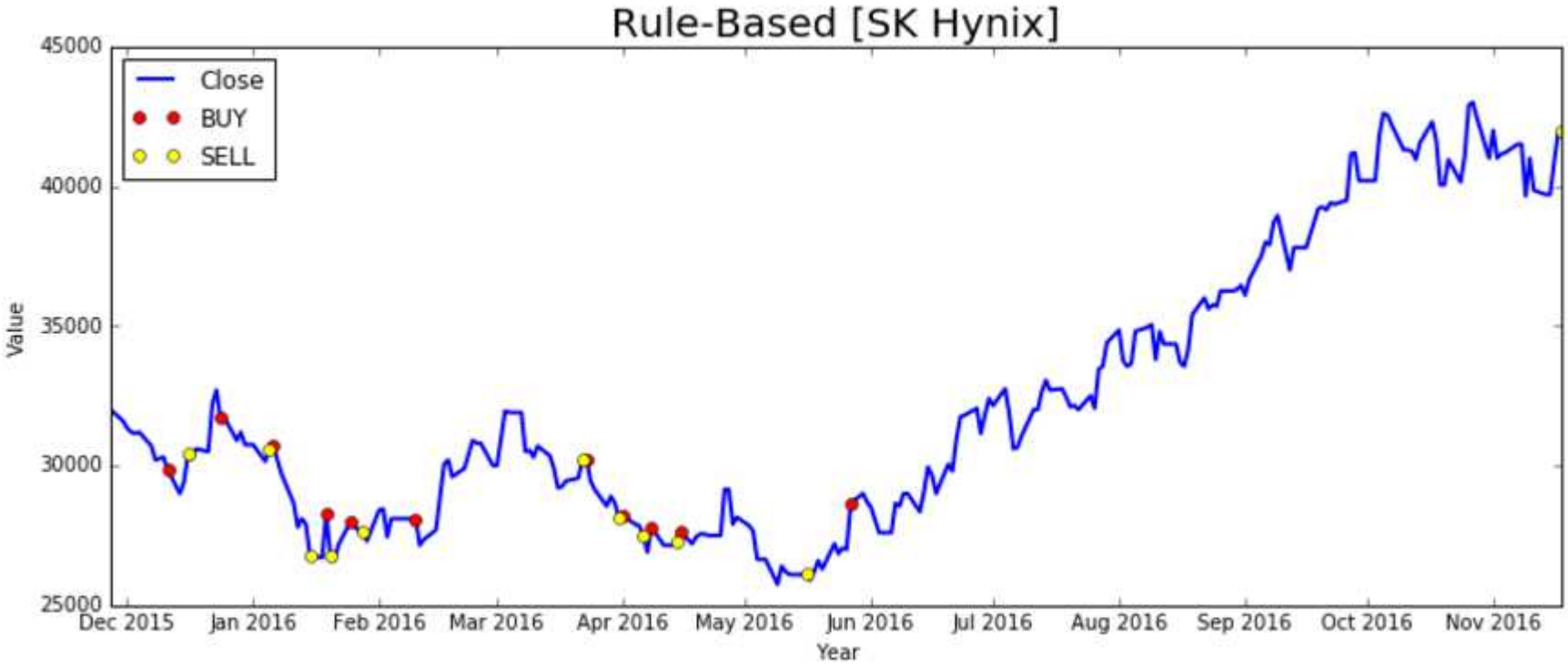
이동평균선 기준 매수, 매도 전략

- 시각화 소스코드

```
plt.figure(figsize=(13,5))
plt.xlabel('Year')
plt.ylabel('Value')
plt.plot(df['Close'], linewidth=2.0, color='blue')
plt.plot(df['Close'][buy_date_list], 'o', color='red', label = 'BUY')
plt.plot(df['Close'][sell_date_list], 'o', color='yellow', label = 'SELL')
plt.legend(loc='upper left')
plt.title('Rule-Based [BSE SENSEX]',size=20)
plt.show()
```

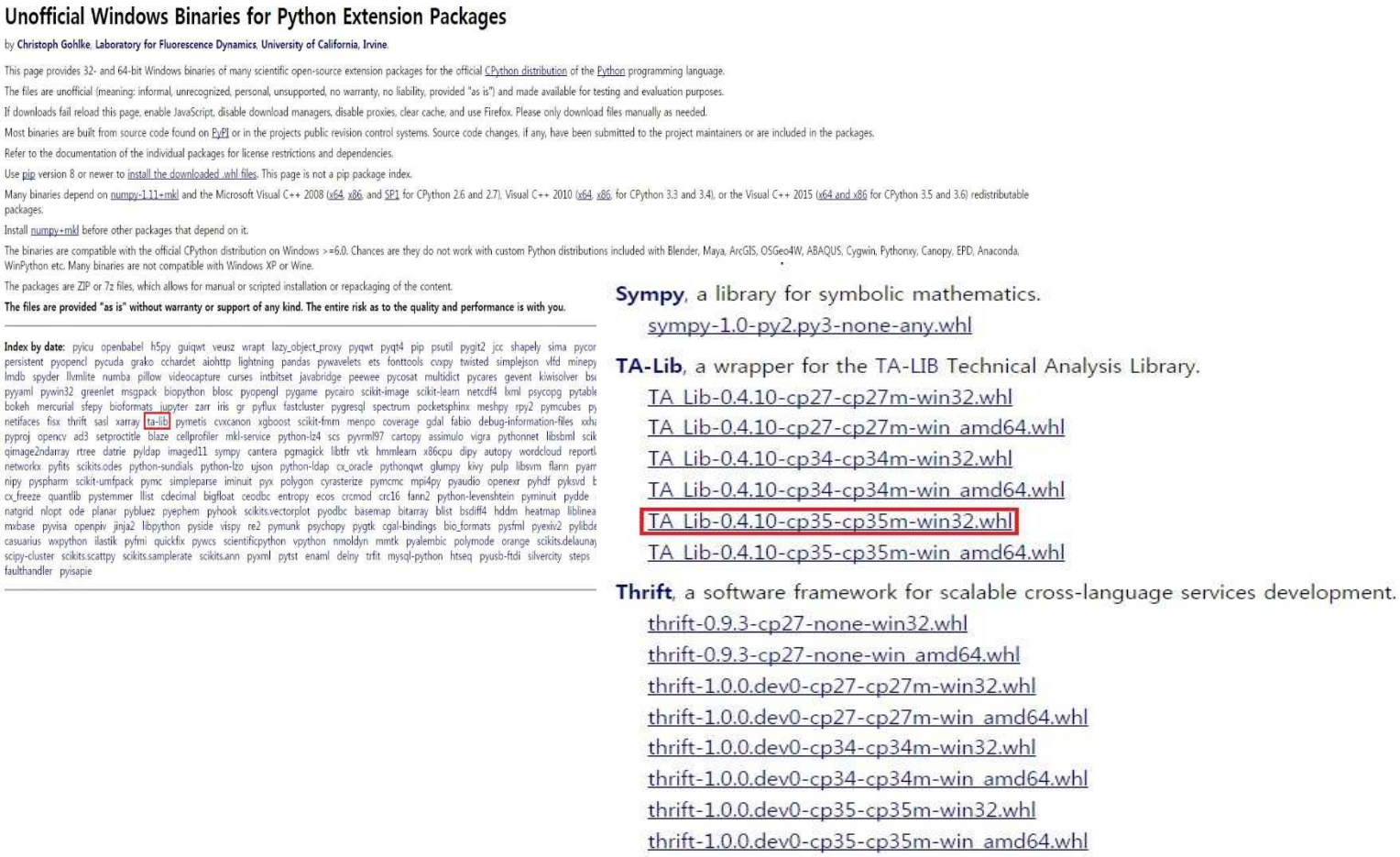
이동평균선 기준 매수, 매도 전략

- 사고 팔기 시점을 나타내는 주가 그래프



기술적 지표(Technical Indicator)

- 기술적 지표를 나타내는 Technical Analysis Library 패키지 다운로드
  - 자신의 Python 버전에 맞는 라이브러리 버전 설치
  - 다운로드 주소 - http://www.lfd.uci.edu/~gohlke/pythonlibs/#ta-lib



Unofficial Windows Binaries for Python Extension Packages

by **Christoph Gohlke, Laboratory for Fluorescence Dynamics, University of California, Irvine.**

This page provides 32- and 64-bit Windows binaries of many scientific open-source extension packages for the official CPython distribution of the Python programming language.

The files are unofficial (meaning: informal, unrecognized, personal, unsupported, no warranty, no liability, provided "as is") and made available for testing and evaluation purposes.

If downloads fail reload this page, enable JavaScript, disable download managers, disable proxies, clear cache, and use Firefox. Please only download files manually as needed.

Most binaries are built from source code found on PyPI or in the projects public revision control systems. Source code changes, if any, have been submitted to the project maintainers or are included in the packages.

Refer to the documentation of the individual packages for license restrictions and dependencies.

Use pip version 8 or newer to install the downloaded .whl files. This page is not a pip package index.

Many binaries depend on numpy-1.11+mkl and the Microsoft Visual C++ 2008 (x64, x86, and SP1 for CPython 2.6 and 2.7), Visual C++ 2010 (x64, x86, for CPython 3.3 and 3.4), or the Visual C++ 2015 (x64 and x86 for CPython 3.5 and 3.6) redistributable packages.

Install numpy+mkl before other packages that depend on it.

The binaries are compatible with the official CPython distribution on Windows >=6.0. Chances are they do not work with custom Python distributions included with Blender, Maya, ArcGIS, OSGeo4W, ABAQUS, Cygwin, Pythonxy, Canopy, EPD, Anaconda, WinPython etc. Many binaries are not compatible with Windows XP or Wine.

The packages are ZIP or 7z files, which allows for manual or scripted installation or repackaging of the content.

The files are provided "as is" without warranty or support of any kind. The entire risk as to the quality and performance is with you.

Index by date: pyicu openbabel h5py guiqwt veusz wrapt lazy_object_proxy pyqwt pyqt4 pip psutil pygit2 jcc shapely sima pycor persistent pyopencl pycuda grako cchardet aiohttp lightning pandas pywavelets ets fonttools cvxpy twisted simplejson vlfd minepy lmdb spyder llvmlite numba pillow videocapture curses intbitset javabridge peewee pycosat multidict pycares gevent kiwisolver bsd pyyaml pywin32 greenlet msgpack biopython blosc pyopengl pygame pycairo scikit-image scikit-learn netcdf4 lxml psycopg pytable bokeh mercurial sfepy bioformats jupyter zarr iris gr pyflux fastcluster pygresql spectrum pocketsphinx meshpy rpy2 pymcubes py netifaces fisx thrift sasl xarray ta-lib pymetis cvxcanon xgboost scikit-fmm menpo coverage gdal fabio debug-information-files xxha pyproj opencv ad3 setproctitle blaze cellprofiler mkl-service python-lz4 scs pyvrml97 cartopy assimulo vigra pythonnet libsbml scik qimage2ndarray rtree datrie pyldap imaged11 sympy cantera pgmagick libtfr vtk hmmlearn x86cpu dipy autopy wordcloud reportl networkx pyfits scikits.odes python-sundials python-lzo ujson python-ldap cx_oracle pythonqwt glumpy kivy pulp libsvm flann pyarr ripy pyspharm scikit-umfpack pymc simpleparse iminuit pyx polygon cyrasterize pymcmc mpi4py pyaudio openexr pyhdf pyksvd b cx_freeze quantlib pystemmer llist cdecimal bigfloat ceodbc entropy ecos crcmod crc16 fann2 python-levenshtein pyminuit pydde natgrid nlopt ode planar pybluez pyephem pyhook scikits.vectorplot pyodbc basemap bitarray blist bsdiff4 hddm heatmap liblinea mxbase pyvisa openpiv jinja2 libpython pyside vispy re2 pymunk psychopy pygtk cgal-bindings bio_formats pysfml pyexiv2 pylibde casuarius wxpython ilastik pyfmi quickfix pywcs scientificpython vpython nmoldyn mmtk pyalembic polymode orange scikits.delaunay scipy-cluster scikits.scattpy scikits.samplerate scikits.ann pyxml pytst enaml delny trfit mysql-python htseq pyusb-ftdi silvercity steps faulthandler pyisapie

Sympy, a library for symbolic mathematics.

sympy-1.0-py2.py3-none-any.whl

TA-Lib, a wrapper for the TA-LIB Technical Analysis Library.

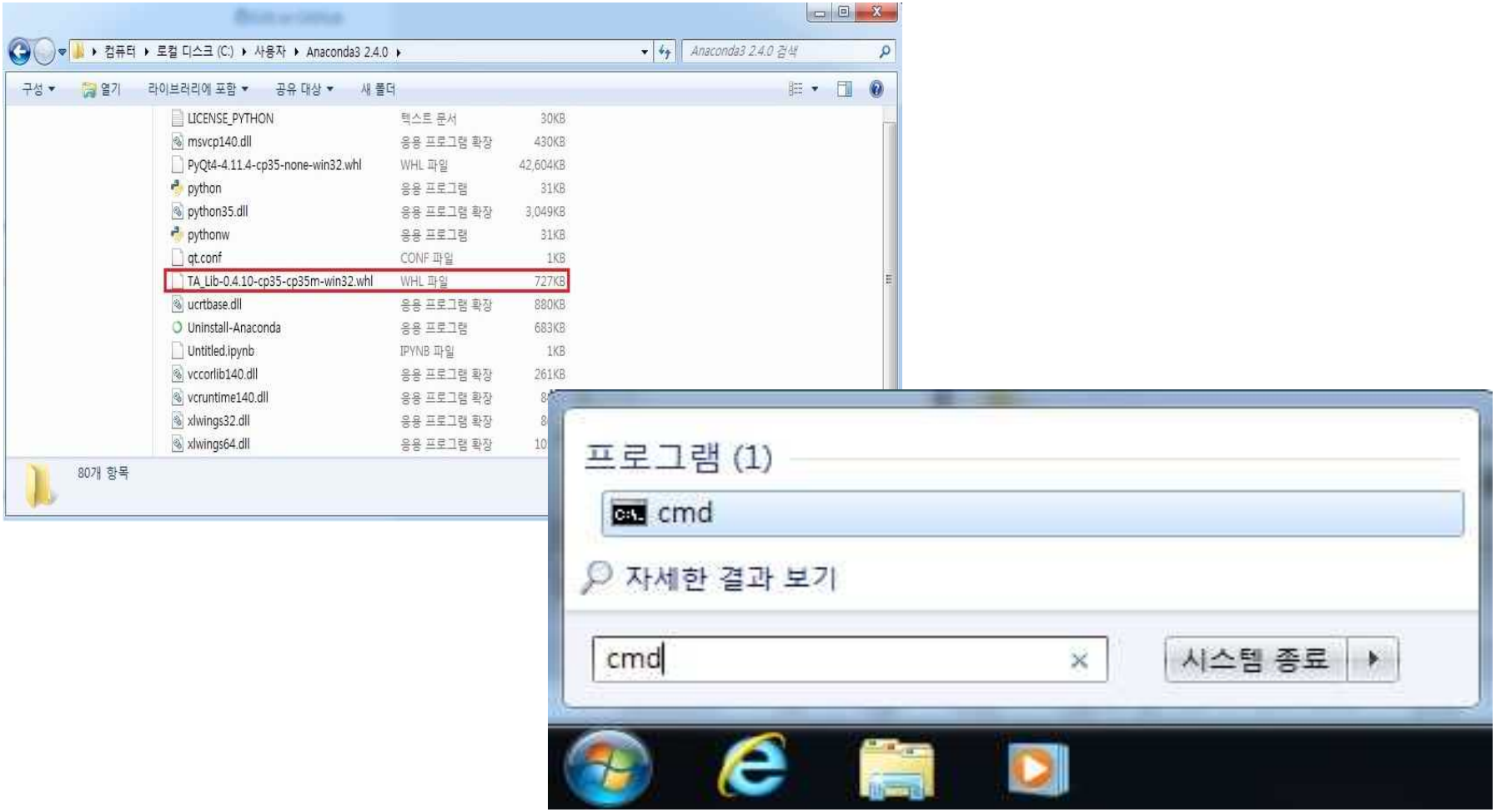
TA_Lib-0.4.10-cp27-cp27m-win32.whl
TA_Lib-0.4.10-cp27-cp27m-win_amd64.whl
TA_Lib-0.4.10-cp34-cp34m-win32.whl
TA_Lib-0.4.10-cp34-cp34m-win_amd64.whl
TA_Lib-0.4.10-cp35-cp35m-win32.whl
TA_Lib-0.4.10-cp35-cp35m-win_amd64.whl

Thrift, a software framework for scalable cross-language services development.

thrift-0.9.3-cp27-none-win32.whl
thrift-0.9.3-cp27-none-win_amd64.whl
thrift-1.0.0.dev0-cp27-cp27m-win32.whl
thrift-1.0.0.dev0-cp27-cp27m-win_amd64.whl
thrift-1.0.0.dev0-cp34-cp34m-win32.whl
thrift-1.0.0.dev0-cp34-cp34m-win_amd64.whl
thrift-1.0.0.dev0-cp35-cp35m-win32.whl
thrift-1.0.0.dev0-cp35-cp35m-win_amd64.whl

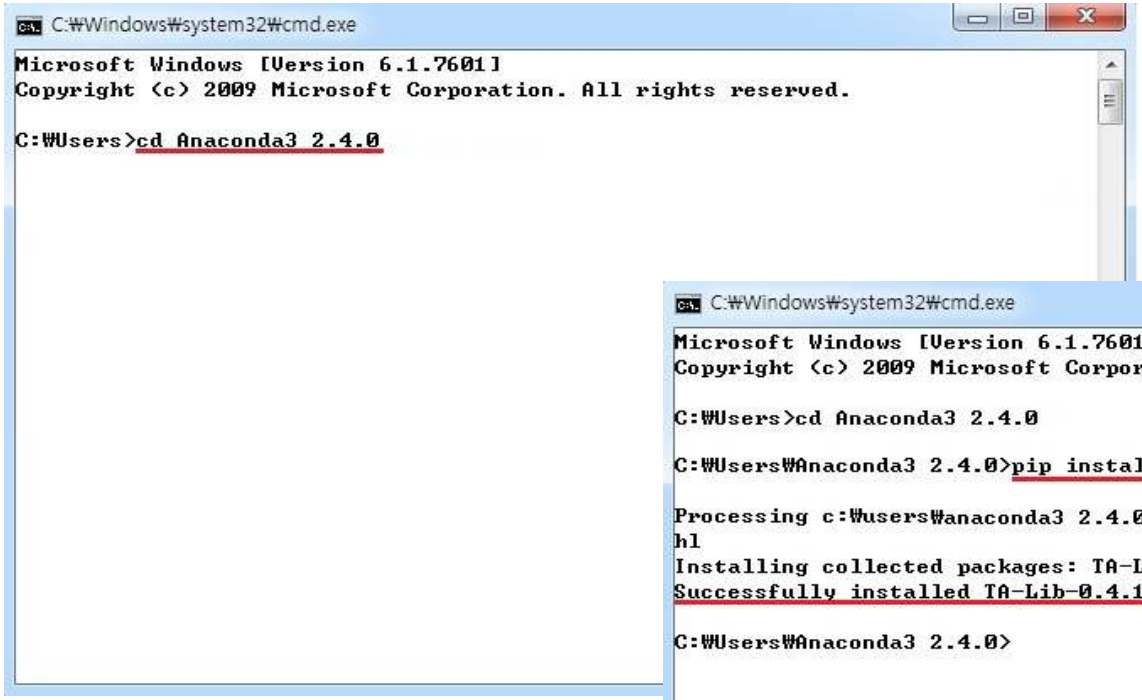
기술적 지표(Technical Indicator)

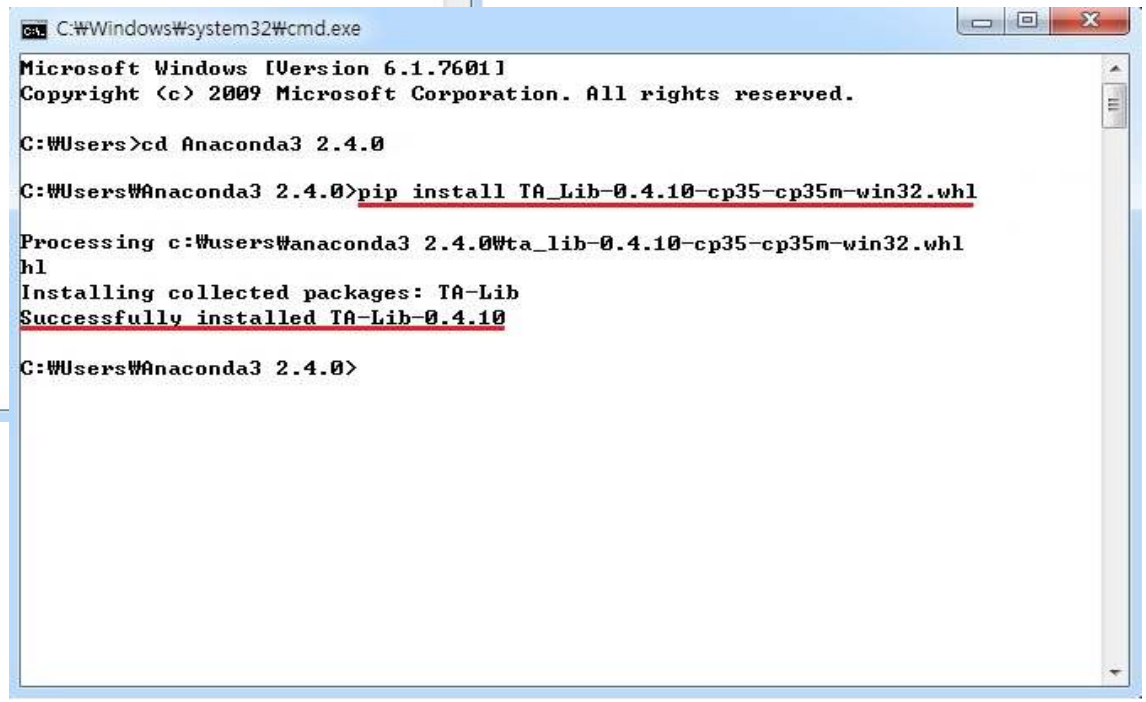
- 기술적 지표를 나타내는 Technical Analysis Library 패키지 Install
  - 다운로드 한 TA_Lib 파일을 Anaconda 폴더에 복사
  - Command 창 실행



기술적 지표(Technical Indicator)

- 기술적 지표를 나타내는 Technical Analysis Library 패키지 Install
  - Command 창에서 TA_Lib 파일을 복사해 넣은 Anaconda 폴더로 이동
  - Python 패키지 및 라이브러리를 install 하는 pip 명령어를 사용해 TA_Lib 라이브러리 Install





기술적 지표(Technical Indicator)

- 기술적 지표를 생성하는 패키지를 비롯한 전체 패키지 Import

```
import matplotlib.pyplot as plt
from datetime import datetime
from pandas import DataFrame
import pandas.io.data as data
import pandas as pd
import numpy as np
import talib as ta

%matplotlib inline
```

기술적 지표(Technical Indicator)

- 기술적 지표를 생성 전체코드 예)

```
def GetMarketData(Ticker):
    start = datetime(2005, 1 ,1)
    end = datetime(2016, 11, 18)
    symbol = Ticker
    f = data.get_data_yahoo(symbol, start = start,
                            end = end, interval = 'd')
    #SMA (Simple Moving Average)
    f['SMA'] = ta.SMA(np.array(f['Close']), timeperiod = 15)
    #MACD (Moving Average Convergence/Divergence)
    f['MACD'], f['MACDSignal'], f['MACDHist'] = ta.MACD(
        np.array(f['Close']), fastperiod = 12,
        slowperiod = 26, signalperiod = 9)
    #Stochastic K%, D%
    f['K%'], f['D%'] = ta.STOCH(np.array(f['High']),
                                np.array(f['Low']),
                                np.array(f['Close']),
                                slowk_period = 14,
                                slowd_period = 3)
    #RSI
    f['RSI'] = ta.RSI(np.array(f['Close']), timeperiod = 14)
    #Williams %R
    f['LW %R'] = ta.WILLR(np.array(f['High']),
                          np.array(f['Low']),
                          np.array(f['Close']),
                          timeperiod = 14)
    f = f.dropna()
    del f['Open'], f['High'], f['Low'], f['Adj Close']
    del f['Volume'], f['MACDSignal'], f['MACDHist']
    f = f.reset_index(drop = True)
    return f
#주식종목 입력
f = GetMarketData('^KS11')
f
```

기술적 지표(Technical Indicator)

- 주가 데이터 수집 코드

Start : 수집 시작 날짜
End : 수집 마지막 날짜
Symbol : 수집 대상 종목 명
f : KS11(KOSPI)에 대한 종목 수집 데이터를 DataFrame 형태로 반환

```
1  def GetMarketData(Ticker):
2      start = datetime(2005, 1 ,1)
3      end = datetime(2016, 11,18)
4      symbol = Ticker
5      f = data.get_data_yahoo(symbol, start = start,
6                              end = end, interval = 'd')
```

```
31 #주식종목 입력
32 f = GetMarketData('^KS11')
```

```
33 f
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2005-01-03	896.000000	897.590027	890.929993	893.710022	252600	893.710022
2005-01-04	890.559998	895.400024	884.940002	886.900024	331600	886.900024
2005-01-05	874.919983	885.200012	873.179993	885.190002	343200	885.190002
2005-01-06	878.229980	886.309998	871.280029	871.280029	394600	871.280029
2005-01-07	876.359985	878.890015	866.719971	870.840027	294800	870.840027
2005-01-10	872.820007	875.000000	866.169983	874.179993	276000	874.179993

기술적 지표(Technical Indicator)

- Simple Moving Average(SMA, 단순이동평균) 지표 생성 코드

TA_Lib 패키지의 SMA 함수를 이용해 단순 이동 평균 지표 생성

SMA 함수 parameter에 주식의 종가 f['Close']를 배열형태로 입력,

기간(timeperiod)을 15로 입력

```
7     #SMA (Simple Moving Average)
8     f['SMA'] = ta.SMA(np.array(f['Close']), timeperiod = 15)
```

```
31 #주식종목 입력
32 f = GetMarketData('^KS11')
```

```
33 f
```

	Open	High	Low	Close	Volume	Adj Close	SMA
Date							
2005-01-03	896.000000	897.590027	890.929993	893.710022	252600	893.710022	NaN
2005-01-04	890.559998	895.400024	884.940002	886.900024	331600	886.900024	NaN
2005-01-05	874.919983	885.200012	873.179993	885.190002	343200	885.190002	NaN
2005-01-06	878.229980	886.309998	871.280029	871.280029	394600	871.280029	NaN
2005-01-07	876.359985	878.890015	866.719971	870.840027	294800	870.840027	NaN
2005-01-10	872.820007	875.000000	866.169983	874.179993	276000	874.179993	NaN
2005-01-11	874.789978	884.289978	869.909973	884.289978	364400	884.289978	NaN
2005-01-12	884.469971	886.789978	878.030029	880.030029	321600	880.030029	NaN
2005-01-13	881.390015	886.729980	876.719971	885.539978	290600	885.539978	NaN
2005-01-14	879.830017	905.520020	877.440002	905.099976	356800	905.099976	NaN
2005-01-17	915.219971	925.010010	912.679993	923.080017	416600	923.080017	NaN
2005-01-18	923.520020	926.849976	920.080017	920.570007	432800	920.570007	NaN
2005-01-19	923.359985	926.710022	914.299988	916.270020	415600	916.270020	NaN
2005-01-20	910.400024	914.929993	907.510010	909.369995	467000	909.369995	NaN
2005-01-21	910.039978	920.349976	906.229980	919.609985	463400	919.609985	895.064005
2005-01-24	918.210022	928.520020	911.750000	923.109985	527600	923.109985	897.024003

기술적 지표(Technical Indicator)

- Moving Average Convergence and Divergence(MACD, 이동평균 수렴/확산지수) 지표 생성

TA_Lib 패키지의 MACD 함수를 이용해 MACD 지표 생성.

MACD 함수 parameter에 주식의 종가 f['Close']를 배열형태로 입력,

단기지수 이동평균의 기간(fastperiod)을 12로 입력,

장기지수 이동평균의 기간(slowperiod)을 26으로 입력,

신호선의 기간(signalperiod)을 9로 입력

```
9     #MACD (Moving Average Convergence/Divergence)
10    f['MACD'], f['MACDSignal'], f['MACDHist'] = ta.MACD(
11        np.array(f['Close']), fastperiod = 12,
12        slowperiod = 26, signalperiod = 9)
```

```
31 #주식종목 입력
32 f = GetMarketData('^KS11')
```

```
33 f
```

	...	SMA	MACD	MACDSignal	MACDHist
Date					
2005-02-15	...	933.435994	NaN	NaN	NaN
2005-02-16	...	936.899329	NaN	NaN	NaN
...	...	...	...	...	...
2016-10-10	...	2049.358024	6.134510	5.326544	0.807966
2016-10-11	...	2050.434692	4.028588	5.066953	-1.038365

기술적 지표(Technical Indicator)

▪ 스토캐스틱(Stochastic K%, D%) 지표 생성 코드

TA_Lib 패키지의 STOCH 함수를 이용해 스토캐스틱 지표 생성

STOCH 함수 parameter에 주식의 고가 f['High']를 배열형태로 입력,

주식의 저가 f['Low']를 배열형태로 입력,

주식의 종가 f['Close']를 배열형태로 입력,

K%의 이동평균기간(slowk_period)을 14로 입력,

D%의 이동평균기간(slowd_period)을 3으로 입력

```
13      #Stochastic K%, D%
14      f['K%'], f['D%'] = ta.STOCH(np.array(f['High']),
15                                  np.array(f['Low']),
16                                  np.array(f['Close']),
17                                  slowk_period = 14,
18                                  slowd_period = 3)
```

```
31  #주식종목 입력
32  f = GetMarketData('^KS11')
```

```
33  f
```

	...	SMA	MACD	MACDSignal	MACDHist	K%	D%
Date							
2005-02-15	...	933.435994	NaN	NaN	NaN	74.349290	71.988259
2005-02-16	...	936.899329	NaN	NaN	NaN	74.842154	73.677590
...	...	...	...	...	...	...	...
2016-10-10	...	2049.358024	6.134510	5.326544	0.807966	70.180727	67.838593
2016-10-11	...	2050.434692	4.028588	5.066953	-1.038365	66.205678	68.171242

기술적 지표(Technical Indicator)

- Relative Strength Index(RSI, 상대강도지수) 지표 생성 코드

TA_Lib 패키지의 RSI 함수를 이용해 상대강도지수 지표 생성

RSI 함수 parameter에 주식의 종가 f['Close']를 배열형태로 입력,

상대강도지수를 나타내는 기간(timeperiod)을 14로 입력

```
19      #RSI
20      f['RSI'] = ta.RSI(np.array(f['Close']), timeperiod = 14)
```

```
31 #주식종목 입력
32 f = GetMarketData('^KS11')
```

```
33 f
```

	...	SMA	MACD	MACDSignal	MACDHist	K%	D%	RSI
Date								
2005-02-15	...	933.435994	NaN	NaN	NaN	74.349290	71.988259	72.906802
2005-02-16	...	936.899329	NaN	NaN	NaN	74.842154	73.677590	73.619184
...	...	...	...	...	...	...	...	...
2016-10-10	...	2049.358024	6.134510	5.326544	0.807966	70.180727	67.838593	53.806369
2016-10-11	...	2050.434692	4.028588	5.066953	-1.038365	66.205678	68.171242	45.729525

기술적 지표(Technical Indicator)

- Williams R% 지표 생성 코드

TA_Lib 패키지의 WILLR 함수를 이용해 Williams R% 지표 생성

WILLR 함수 parameter에
주식의 고가 f['High']를 배열형태로 입력,

주식의 저가 f['Low']를 배열형태로 입력,

주식의 종가 f['Close']를 배열형태로 입력,

주가의 위치를 나타내는 기간(timeperiod)을 14로 입력

```
21      #Williams' R%
22      f['LW R%'] = ta.WILLR(np.array(f['High']),
23                            np.array(f['Low']),
24                            np.array(f['Close']),
25                            timeperiod = 14)
```

```
31  #주식종목 입력
32  f = GetMarketData('^KS11')
```

```
33  f
```

	...	SMA	MACD	MACDSignal	MACDHist	K%	D%	RSI	LW R%
Date									
2005-02-15	...	933.435994	NaN	NaN	NaN	74.349290	71.988259	72.906802	-0.000000
2005-02-16	...	936.899329	NaN	NaN	NaN	74.842154	73.677590	73.619184	-10.983176
...	...	...	...	...	...	...	...	...	...
2016-10-10	...	2049.358024	6.134510	5.326544	0.807966	70.180727	67.838593	53.806369	-24.692753
2016-10-11	...	2050.434692	4.028588	5.066953	-1.038365	66.205678	68.171242	45.729525	-79.825058

기술적 지표(Technical Indicator)

- DataFrame 재설정 코드

DataFrame f 내에 값이 없는 행을 제거하기 위해 dropna 사용

DataFrame f 내에 있는 시가, 고가, 저가, 수정종가 삭제

DataFrame f 내에 있는 거래량, f['MACDSignal'], f['MACDHist'] 삭제

DataFrame의 index를 0부터 시작하는 숫자형 index로 재설정

```
26      f = f.dropna()
27      del f['Open'], f['High'], f['Low'], f['Adj Close']
28      del f['Volume'], f['MACDSignal'], f['MACDHist']
29      f = f.reset_index(drop = True)
30      return f
31  #주식종목 입력
32  f = GetMarketData('^KS11')
33  f
```

	Close	SMA	MACD	K%	D%	RSI	LW R%
0	977.799988	952.438664	24.114274	79.276150	79.381471	69.621535	-19.805499
1	968.429993	955.561328	22.783502	73.704057	77.608764	63.338269	-32.464229
2	987.099976	959.187992	22.970577	76.413455	76.464554	69.286186	-7.241346
3	996.950012	964.071993	23.641130	80.688509	76.935340	71.878398	-4.178234
4	1011.359985	970.066659	25.046591	82.710949	79.937638	75.178819	-0.339028
5	1007.479980	975.312659	25.552789	82.749751	82.049736	72.704562	-10.121033
6	1010.919983	980.470658	25.932599	82.064000	82.508234	73.536160	-5.785260
7	1012.960022	984.721993	26.097380	81.690074	82.167942	74.041253	-5.208623
8	1007.500000	988.739994	25.493519	77.355239	80.369771	70.180471	-27.464412
9	1000.280029	991.105998	24.153933	71.280393	76.775235	65.329417	-40.536049
10	1008.789978	993.766663	23.507997	69.162139	72.599257	68.126029	-26.626342

데이터 Normalization

- 머신러닝 적용을 위한 데이터 Normalization
- Normalization 이유
  - 기술적 지표 값들에 다양성
  - 0~1사이 값으로 표준화하여 정확도 상관성을 높임
- 수식
  - Y = 데이터 Normalization 결과 값
  - X = 기술적 지표 값
  - X_{min} = 기술적 지표 전체기간 중 최소값
  - X_{max} = 기술적 지표 전체 기간 중 최대값

$$Y = \frac{X - X_{min}}{X_{max} - X_{min}}$$

데이터 Normalization 수식

데이터 Normalization

- Normalization Set: SMA, MACD, K%, D%, RSI, LW R%
  - MACD(Moving Average Convergence/Divergence)
  - SMA(Simple Moving Average)
  - Stochastic(K%, D%)
  - RSI(Relative Strength Index)
  - R%(Larry William's R%)

- 결과 값 수치 0~1사이
- Python Code 예

```
f['MA15'] = pd.rolling_mean(f['Close'], 15)
f = f.dropna()
f['Status'] = np.where(f['Close'] > f['MA15'], 1, 0)
f['Action'] = np.where(f['Status'].rolling(window = 5).sum() == 5,
                       'Up',np.where(
        f['Status'].rolling(window = 5).sum() == 0, 'Down', 'No'))
SMA_max=max(f['SMA'])
SMA_min=min(f['SMA'])
MACD_max=max(f['MACD'])
MACD_min=min(f['MACD'])
K_max=max(f['K%'])
K_min=min(f['K%'])
D_max=max(f['D%'])
D_min=min(f['D%'])
RSI_max=max(f['RSI'])
RSI_min=min(f['RSI'])
LWR_max=max(f['LW R%'])
LWR_min=min(f['LW R%'])
```

데이터 Normalization

```
SMA=[]
MACD=[]
K=[]
D=[]
RSI=[]
LWR=[]
f=f.reset_index(drop=True)
for i in range(0, len(f)):
    SMA.append((f['SMA'][i]-SMA_min)/(SMA_max-SMA_min))
    MACD.append((f['MACD'][i]-MACD_min)/(MACD_max-MACD_min))
    K.append((f['K%'][i]-K_min)/(K_max-K_min))
    D.append((f['D%'][i]-D_min)/(D_max-D_min))
    RSI.append((f['RSI'][i]-RSI_min)/(RSI_max-RSI_min))
    LWR.append((f['LW R%'][i]-LWR_min)/(LWR_max-LWR_min))
f['SMA']=SMA
f['MACD']=MACD
f['K%']=K
f['D%']=D
f['RSI']=RSI
f['LW R%']=LWR
```

데이터 Normalization

- 데이터 Normalization 전 기술적 지표 값

	Close	Date	SMA	MACD	K%	D%	RSI	LW R%	MA15	Status	Action
0	993.130005	2005-03-15	1001.589327	19.702749	61.804489	63.535163	54.273496	-52.222927	1001.589327	0	No
1	993.130005	2005-03-16	1002.611328	17.521246	62.521706	62.937039	54.273496	-59.674942	1002.611328	0	No
2	980.049988	2005-03-17	1003.385994	14.568999	57.712228	60.679474	48.855575	-86.297483	1003.385994	0	No
3	979.719971	2005-03-18	1002.893994	12.063630	52.166619	57.466851	48.723416	-86.929943	1002.893994	0	No
4	979.270020	2005-03-21	1001.715328	9.927363	46.397974	52.092274	48.530659	-81.614078	1001.715328	0	Down
5	980.409973	2005-03-22	999.651994	8.231454	43.226607	47.263733	49.080283	-79.583160	999.651994	0	Down

- 데이터 Normalization 후 결과 값

	Close	Date	SMA	MACD	K%	D%	RSI	LW R%	MA15	Status	Action
0	993.130005	2005-03-15	0.060243	0.748854	0.598953	0.623870	0.527774	0.477771	1001.589327	0	No
1	993.130005	2005-03-16	0.061054	0.737070	0.607816	0.616312	0.527774	0.403251	1002.611328	0	No
2	980.049988	2005-03-17	0.061669	0.721122	0.548383	0.587788	0.455658	0.137025	1003.385994	0	No
3	979.719971	2005-03-18	0.061278	0.707588	0.479853	0.547197	0.453899	0.130701	1002.893994	0	No
4	979.270020	2005-03-21	0.060343	0.696048	0.408566	0.479290	0.451333	0.183859	1001.715328	0	Down
5	980.409973	2005-03-22	0.058705	0.686886	0.369376	0.418282	0.458649	0.204168	999.651994	0	Down

Decision Tree 기반 예측 모델을 이용한 트레이딩

- Feature Set: SMA, MACD, K%, D%, RSI, LW R%
  - MACD(Moving Average Convergence/Divergence)
  - SMA(Simple Moving Average)
  - Stochastic(K%, D%)
  - RSI(Relative Strength Index)
  - R%(Larry William's R%)
- 종속변수의 값(feature_y): up or down
- Python Code 예

```
from sklearn import tree
trainset_size = len(f)-(252*3)
feature_x=f[['SMA','MACD','K%','D%','RSI','LW R%']].values
feature_y=f[['Action']].values
x_train = feature_x[0:trainset_size]
y_train = feature_y[0:trainset_size]
x_test = feature_x[trainset_size+1:len(f)]
clf = tree.DecisionTreeClassifier()
DecisionTree=clf.fit(x_train, y_train)
DecisionTree_pred = DecisionTree.predict(x_test)
f_test=f[trainset_size+1:len(f)]
f_test=f_test.reset_index(drop=True)
del f_test['SMA'],f_test['MACD'],f_test['K%'],f_test['D%']
del f_test['RSI'],f_test['LW R%'],f_test['MA15']
del f_test['Status']
f_test['Predict']=DecisionTree_pred
index=0
trading_signal=[]
price=0
return_list=[]
buy_date_list=[]
sell_date_list=[]
```

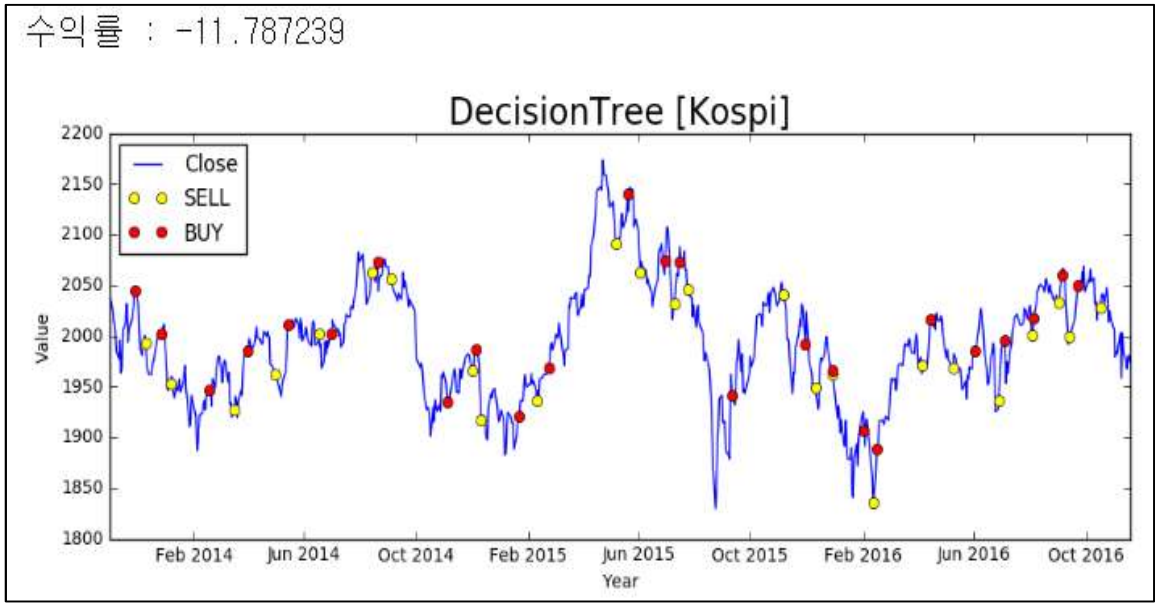
Decision Tree 기반 예측 모델을 이용한 트레이딩

```
for i in range(0, len(f_test)):
    if index==0:
        if f_test['Predict'][i]=='Up':
            trading_signal.append('BUY')
            price=f_test['Close'][i]
            buy_date_list.append(f_test['Date'][i])
            index=1
        else:
            trading_signal.append('')
    elif index==1:
        if f_test['Predict'][i]=='Down':
            trading_signal.append('SELL')
            return_list.append((f_test['Close'][i]-price)/
                               price*100)
            sell_date_list.append(f_test
            index=0
        else:
            trading_signal.append('')
f_test['Trading_Signal']=trading_signal
```

```
profit=sum(return_list)
print('수익률 : %f' %(profit))
f_test=f_test.set_index(['Date'])
plt.figure(figsize=(11,4))
plt.xlabel('Year')
plt.ylabel('Value')
plt.plot(f_test['Close'])
plt.plot(f_test['Close'][sell_date_list], 'o', color='yellow',
         label='SELL')
plt.plot(f_test['Close'][buy_date_list], 'o', color='red',
         label='BUY')
plt.legend(loc='upper left')
plt.title('DecisionTree [Kospi]',size=20)
plt.show()
```

Decision Tree 기반 예측 모델을 이용한 트레이딩

- 훈련데이터 기간: 2005년 1월 1일부터 2013년 12월 29일 까지
- 테스트 방법: 훈련된 모델에 2013년 12월 30일부터 2016년 11월 18일 까지 테스트데이터를 사용하여 예측한 매수 매도 신호에 수익률을 구함



Random Forest 기반 예측 모델을 이용한 트레이딩

- Feature Set: SMA, MACD, K%, D%, RSI, LW R%
  - MACD(Moving Average Convergence/Divergence)
  - SMA(Simple Moving Average)
  - Stochastic(K%, D%)
  - RSI(Relative Strength Index)
  - R%(Larry William's R%)
- 종속변수의 값(feature_y): up or down
- Python Code 예

```
from sklearn.ensemble import RandomForestClassifier
trainset_size = len(f)-(252*3)
feature_x=f[['SMA','MACD','K%','D%','RSI','LW R%']].values
feature_y=f[['Action']].values
x_train = feature_x[0:trainset_size]
y_train = feature_y[0:trainset_size]
x_test = feature_x[trainset_size+1:len(f)]
clf = RandomForestClassifier()
RandomForest=clf.fit(x_train, y_train)
RandomForest_pred = RandomForest.predict(x_test)
f_test=f[trainset_size+1:len(f)]
f_test=f_test.reset_index(drop=True)
del f_test['SMA'],f_test['MACD'],f_test['K%'],f_test['D%']
del f_test['RSI'],f_test['LW R%'],f_test['MA15']
del f_test['Status']
f_test['Predict']=RandomForest_pred
index=0
trading_signal=[]
price=0
return_list=[]
buy_date_list=[]
sell_date_list=[]
```

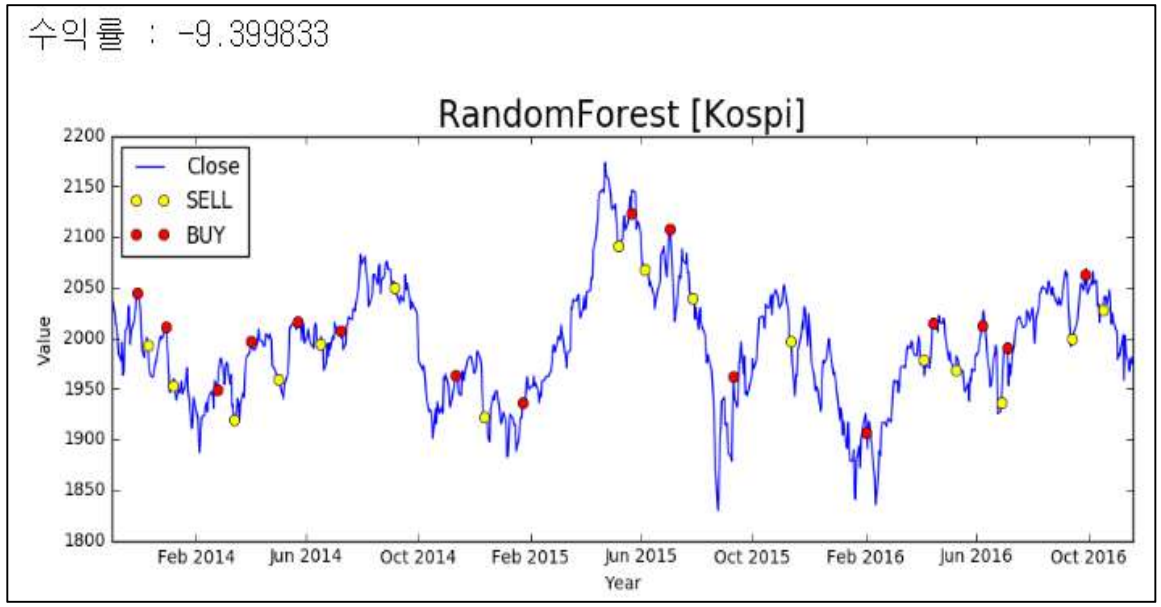
Random Forest 기반 예측 모델을 이용한 트레이딩

```
for i in range(0, len(f_test)):
    if index==0:
        if f_test['Predict'][i]=='Up':
            trading_signal.append('BUY')
            price=f_test['Close'][i]
            buy_date_list.append(f_test['Date'][i])
            index=1
        else:
            trading_signal.append('')
    elif index==1:
        if f_test['Predict'][i]=='Down':
            trading_signal.append('SELL')
            return_list.append((f_test['Close'][i]-price)/
                               price*100)
            sell_date_list.append(f_test
            index=0
        else:
            trading_signal.append('')
f_test['Trading_Signal']=trading_signal
```

```
profit=sum(return_list)
print('수익률 : %f' %(profit))
f_test=f_test.set_index(['Date'])
plt.figure(figsize=(11,4))
plt.xlabel('Year')
plt.ylabel('Value')
plt.plot(f_test['Close'])
plt.plot(f_test['Close'][sell_date_list], 'o', color='yellow',
         label='SELL')
plt.plot(f_test['Close'][buy_date_list], 'o', color='red',
         label='BUY')
plt.legend(loc='upper left')
plt.title('RandomForest [Kospi]',size=20)
plt.show()
```

Random Forest 기반 예측 모델을 이용한 트레이딩

- 훈련데이터 기간: 2005년 1월 1일부터 2013년 12월 29일 까지
- 테스트 방법: 훈련된 모델에 2013년 12월 30일부터 2016년 11월 18일 까지 테스트데이터를 사용하여 예측한 매수 매도 신호에 수익률을 구함



Linear SVM 기반 예측 모델을 이용한 트레이딩

- Feature Set: SMA, MACD, K%, D%, RSI, LW R%
  - MACD(Moving Average Convergence/Divergence)
  - SMA(Simple Moving Average)
  - Stochastic(K%, D%)
  - RSI(Relative Strength Index)
  - R%(Larry William's R%)
- 종속변수의 값(feature_y): up or down
- Python Code 예

```
from sklearn.svm import LinearSVC
trainset_size = len(f)-(252*3)
feature_x=f[['SMA','MACD','K%','D%','RSI','LW R%']].values
feature_y=f[['Action']].values
x_train = feature_x[0:trainset_size]
y_train = feature_y[0:trainset_size]
x_test = feature_x[trainset_size+1:len(f)]
clf = LinearSVC()
SVM=clf.fit(x_train, y_train)
SVM_pred = SVM.predict(x_test)
f_test=f[trainset_size+1:len(f)]
f_test=f_test.reset_index(drop=True)
del f_test['SMA'],f_test['MACD'],f_test['K%'],f_test['D%']
del f_test['RSI'],f_test['LW R%'],f_test['MA15']
del f_test['Status']
f_test['Predict']=SVM_pred
index=0
trading_signal=[]
price=0
return_list=[]
buy_date_list=[]
sell_date_list=[]
```

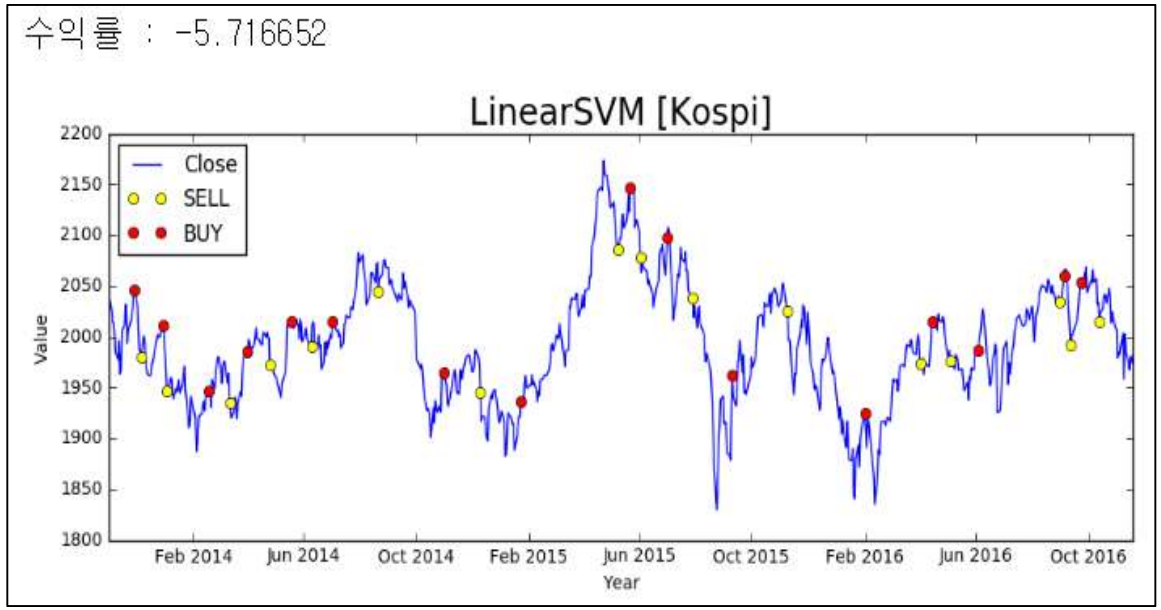
Linear SVM 기반 예측 모델을 이용한 트레이딩

```
for i in range(0, len(f_test)):
    if index==0:
        if f_test['Predict'][i]=='Up':
            trading_signal.append('BUY')
            price=f_test['Close'][i]
            buy_date_list.append(f_test['Date'][i])
            index=1
        else:
            trading_signal.append('')
    elif index==1:
        if f_test['Predict'][i]=='Down':
            trading_signal.append('SELL')
            return_list.append((f_test['Close'][i]-price)/
                               price*100)
            sell_date_list.append(f_tes
            index=0
        else:
            trading_signal.append('')
f_test['Trading_Signal']=trading_signal
```

```
profit=sum(return_list)
print('수익률 : %f' %(profit))
f_test=f_test.set_index(['Date'])
plt.figure(figsize=(11,4))
plt.xlabel('Year')
plt.ylabel('Value')
plt.plot(f_test['Close'])
plt.plot(f_test['Close'][sell_date_list], 'o', color='yellow',
         label='SELL')
plt.plot(f_test['Close'][buy_date_list], 'o', color='red',
         label='BUY')
plt.legend(loc='upper left')
plt.title('LinearSVM [Kospi]',size=20)
plt.show()
```

Linear SVM 기반 예측 모델을 이용한 트레이딩

- 훈련데이터 기간: 2005년 1월 1일부터 2013년 12월 29일 까지
- 테스트 방법: 훈련된 모델에 2013년 12월 30일부터 2016년 11월 18일 까지 테스트데이터를 사용하여 예측한 매수 매도 신호에 수익률을 구함



Naivebayes 기반 예측 모델을 이용한 트레이딩

- Feature Set: SMA, MACD, K%, D%, RSI, LW R%
  - MACD(Moving Average Convergence/Divergence)
  - SMA(Simple Moving Average)
  - Stochastic(K%, D%)
  - RSI(Relative Strength Index)
  - R%(Larry William's R%)
- 종속변수의 값(feature_y): up or down
- Python Code 예

```
from sklearn.naive_bayes import GaussianNB
trainset_size = len(f)-(252*3)
feature_x=f[['SMA','MACD','K%','D%','RSI','LW R%']].values
feature_y=f[['Action']].values
x_train = feature_x[0:trainset_size]
y_train = feature_y[0:trainset_size]
x_test = feature_x[trainset_size+1:len(f)]
clf = GaussianNB()
Naive_bayes = clf.fit(x_train, y_train)
Naive_bayes_pred = Naive_bayes.predict(x_test)
f_test=f[trainset_size+1:len(f)]
f_test=f_test.reset_index(drop=True)
del f_test['SMA'],f_test['MACD'],f_test['K%'],f_test['D%']
del f_test['RSI'],f_test['LW R%'],f_test['MA15']
del f_test['Status']
f_test['Predict']=Naive_bayes_pred
index=0
trading_signal=[]
price=0
return_list=[]
buy_date_list=[]
sell_date_list=[]
```

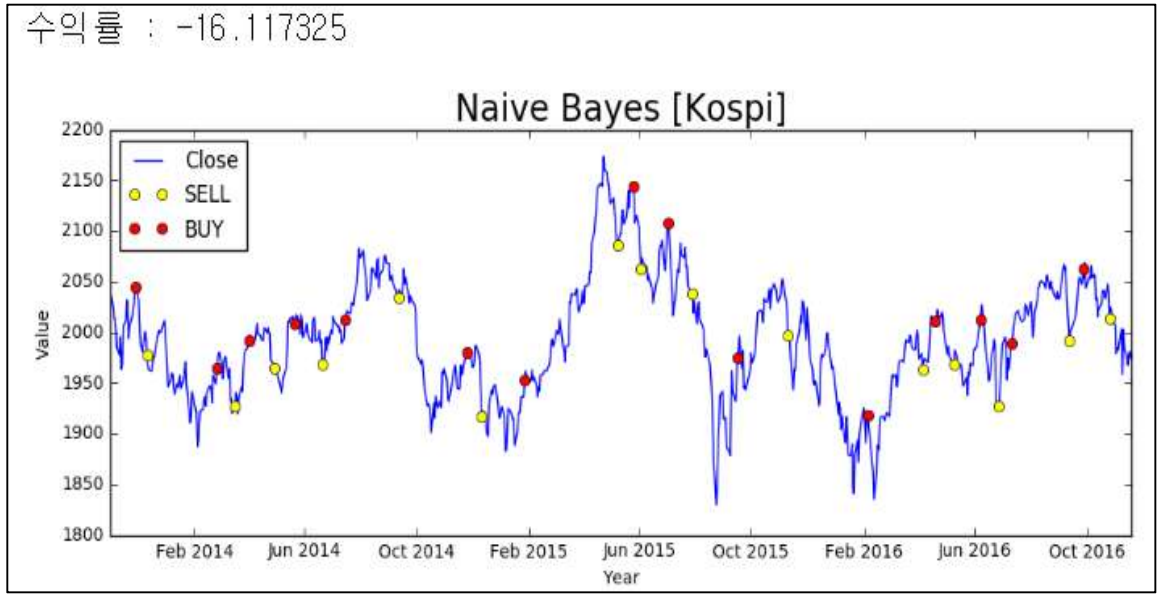
Naivebayes 기반 예측 모델을 이용한 트레이딩

```
for i in range(0, len(f_test)):
    if index==0:
        if f_test['Predict'][i]=='Up':
            trading_signal.append('BUY')
            price=f_test['Close'][i]
            buy_date_list.append(f_test['Date'][i])
            index=1
        else:
            trading_signal.append('')
    elif index==1:
        if f_test['Predict'][i]=='Down':
            trading_signal.append('SELL')
            return_list.append((f_test['Close'][i]-price)/
                               price*100)
            sell_date_list.append(f_tes
            index=0
        else:
            trading_signal.append('')
f_test['Trading_Signal']=trading_signal
```

```
profit=sum(return_list)
print('수익률 : %f' %(profit))
f_test=f_test.set_index(['Date'])
plt.figure(figsize=(11,4))
plt.xlabel('Year')
plt.ylabel('Value')
plt.plot(f_test['Close'])
plt.plot(f_test['Close'][sell_date_list], 'o', color='yellow',
         label='SELL')
plt.plot(f_test['Close'][buy_date_list], 'o', color='red',
         label='BUY')
plt.legend(loc='upper left')
plt.title('Naive Bayes [Kospi]',size=20)
plt.show()
```

Naivebayes 기반 예측 모델을 이용한 트레이딩

- 훈련데이터 기간: 2005년 1월 1일부터 2013년 12월 29일 까지
- 테스트 방법: 훈련된 모델에 2013년 12월 30일부터 2016년 11월 18일 까지 테스트데이터를 사용하여 예측한 매수 매도 신호에 수익률을 구함



Neural Network 기반 예측 모델을 이용한 트레이딩

- Feature Set: SMA, MACD, K%, D%, RSI, LW R%
  - MACD(Moving Average Convergence/Divergence)
  - SMA(Simple Moving Average)
  - Stochastic(K%, D%)
  - RSI(Relative Strength Index)
  - R%(Larry William's R%)
- 종속변수의 값(feature_y): up or down
- Python Code 예

```
from sklearn.neural_network import MLPClassifier
trainset_size = len(f)-(252*3)
feature_x=f[['SMA','MACD','K%','D%','RSI','LW R%']].values
feature_y=f[['Action']].values
x_train = feature_x[0:trainset_size]
y_train = feature_y[0:trainset_size]
x_test = feature_x[trainset_size+1:len(f)]
clf = MLPClassifier()
NN = clf.fit(x_train, y_train)
NN_pred = NN.predict(x_test)
f_test=f[trainset_size+1:len(f)]
f_test=f_test.reset_index(drop=True)
del f_test['SMA'],f_test['MACD'],f_test['K%'],f_test['D%']
del f_test['RSI'],f_test['LW R%'],f_test['MA15']
del f_test['Status']
f_test['Predict']=NN_pred
index=0
trading_signal=[]
price=0
return_list=[]
buy_date_list=[]
sell_date_list=[]
```

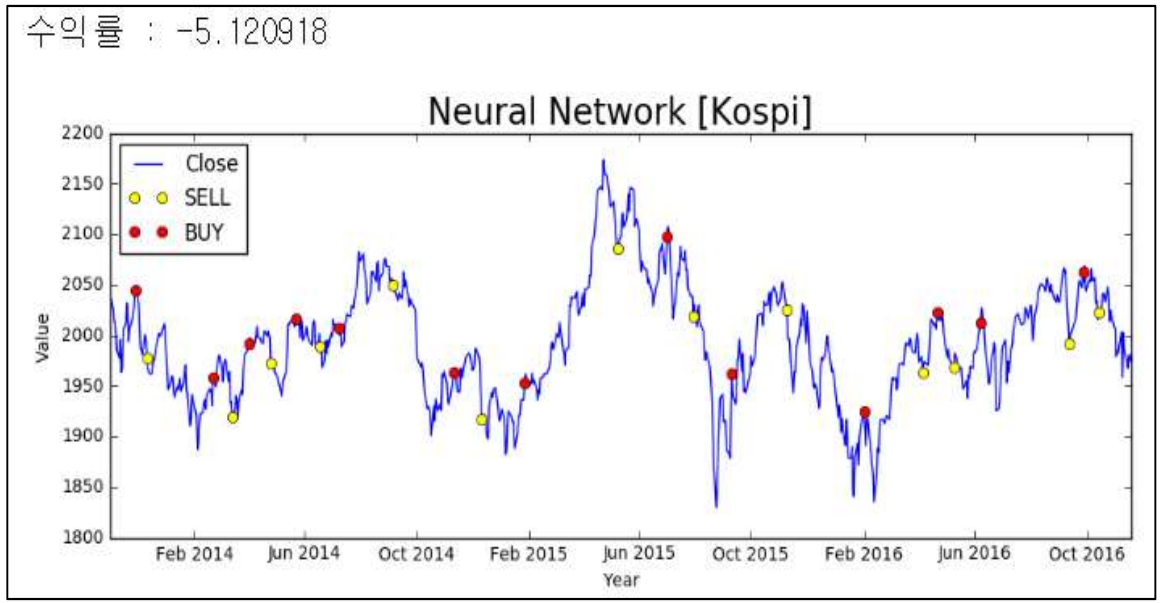
Neural Network 기반 예측 모델을 이용한 트레이딩

```
for i in range(0, len(f_test)):
    if index==0:
        if f_test['Predict'][i]=='Up':
            trading_signal.append('BUY')
            price=f_test['Close'][i]
            buy_date_list.append(f_test['Date'][i])
            index=1
        else:
            trading_signal.append('')
    elif index==1:
        if f_test['Predict'][i]=='Down':
            trading_signal.append('SELL')
            return_list.append((f_test['Close'][i]-price)/
                               price*100)
            sell_date_list.append(f_test
            index=0
        else:
            trading_signal.append('')
f_test['Trading_Signal']=trading_signal
```

```
profit=sum(return_list)
print('수익률 : %f' %(profit))
f_test=f_test.set_index(['Date'])
plt.figure(figsize=(11,4))
plt.xlabel('Year')
plt.ylabel('Value')
plt.plot(f_test['Close'])
plt.plot(f_test['Close'][sell_date_list], 'o', color='yellow',
         label='SELL')
plt.plot(f_test['Close'][buy_date_list], 'o', color='red',
         label='BUY')
plt.legend(loc='upper left')
plt.title('Neural Network [Kospi]',size=20)
plt.show()
```

Neural Network 기반 예측 모델을 이용한 트레이딩

- 훈련데이터 기간: 2005년 1월 1일부터 2013년 12월 29일 까지
- 테스트 방법: 훈련된 모델에 2013년 12월 30일부터 2016년 11월 18일 까지 테스트데이터를 사용하여 예측한 매수 매도 신호에 수익률을 구함



KNN 기반 예측 모델을 이용한 트레이딩

- Feature Set: SMA, MACD, K%, D%, RSI, LW R%
  - MACD(Moving Average Convergence/Divergence)
  - SMA(Simple Moving Average)
  - Stochastic(K%, D%)
  - RSI(Relative Strength Index)
  - R%(Larry William's R%)
- 종속변수의 값(feature_y): up or down
- Python Code 예

```
from sklearn.neighbors import KNeighborsClassifier
trainset_size = len(f)-(252*3)
feature_x=f[['SMA','MACD','K%','D%','RSI','LW R%']].values
feature_y=f[['Action']].values
x_train = feature_x[0:trainset_size]
y_train = feature_y[0:trainset_size]
x_test = feature_x[trainset_size+1:len(f)]
clf = KNeighborsClassifier()
KNN = clf.fit(x_train, y_train)
KNN_pred = KNN.predict(x_test)
f_test=f[trainset_size+1:len(f)]
f_test=f_test.reset_index(drop=True)
del f_test['SMA'],f_test['MACD'],f_test['K%'],f_test['D%']
del f_test['RSI'],f_test['LW R%'],f_test['MA15']
del f_test['Status']
f_test['Predict']=KNN_pred
index=0
trading_signal=[]
price=0
return_list=[]
buy_date_list=[]
sell_date_list=[]
```

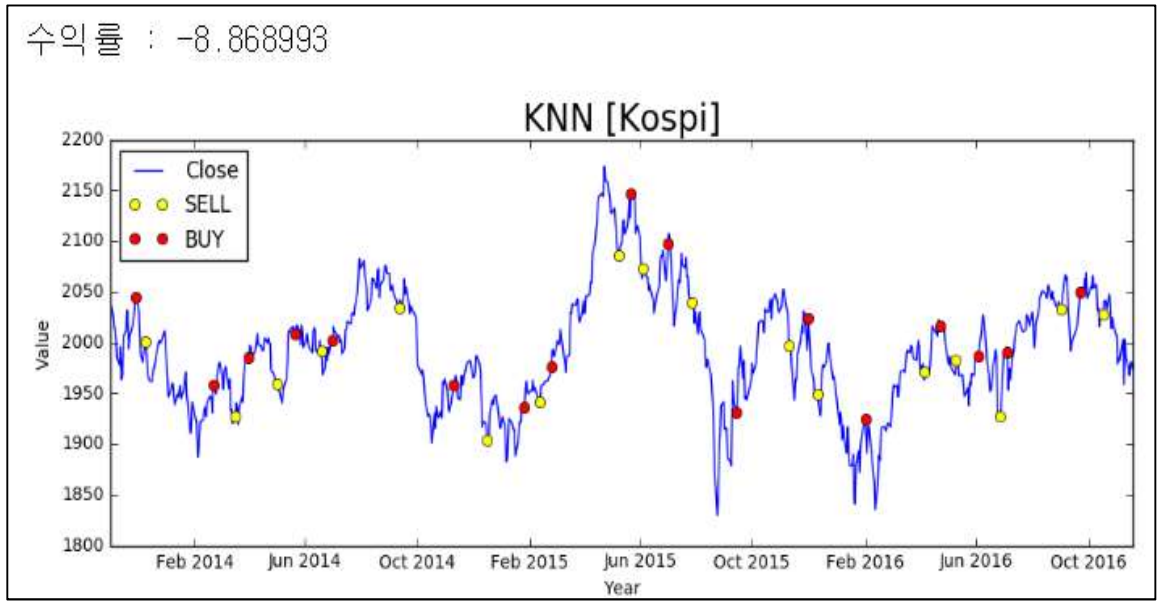
KNN 기반 예측 모델을 이용한 트레이딩

```
for i in range(0, len(f_test)):
    if index==0:
        if f_test['Predict'][i]=='Up':
            trading_signal.append('BUY')
            price=f_test['Close'][i]
            buy_date_list.append(f_test['Date'][i])
            index=1
        else:
            trading_signal.append('')
    elif index==1:
        if f_test['Predict'][i]=='Down':
            trading_signal.append('SELL')
            return_list.append((f_test['Close'][i]-price)/
                               price*100)
            sell_date_list.append(f_test[
            index=0
        else:
            trading_signal.append('')
f_test['Trading_Signal']=trading_signal
```

```
profit=sum(return_list)
print('수익률 : %f' %(profit))
f_test=f_test.set_index(['Date'])
plt.figure(figsize=(11,4))
plt.xlabel('Year')
plt.ylabel('Value')
plt.plot(f_test['Close'])
plt.plot(f_test['Close'][sell_date_list], 'o', color='yellow',
         label='SELL')
plt.plot(f_test['Close'][buy_date_list], 'o', color='red',
         label='BUY')
plt.legend(loc='upper left')
plt.title('KNN [Kospi]',size=20)
plt.show()
```

KNN 기반 예측 모델을 이용한 트레이딩

- 훈련데이터 기간: 2005년 1월 1일부터 2013년 12월 29일 까지
- 테스트 방법: 훈련된 모델에 2013년 12월 30일부터 2016년 11월 18일 까지 테스트데이터를 사용하여 예측한 매수 매도 신호에 수익률을 구함



예측 모델 성능 평가 및 분석

- 평가 대상 : Kospi, 삼성전자, SK하이닉스
- 룰 베이스 예측 모델과 머신러닝 알고리즘 6가지
  - 룰베이스 예측 모델
  - Decision Tree
  - Random Forest
  - Linear SVM
  - Naivebayes
  - Neural Network
- 훈련데이터 기간 : 2005년 1월 1일부터 2013년 12월 29일 까지
- 테스트 방법: 훈련된 모델에 2013년 12월 30일부터 2016년 11월 18일 까지 테스트 데이터를 사용하여 예측한 매수 매도 신호에 수익률을 구함

예측 모델 성능 평가 및 분석

- Kospi 예측 모델 성능 비교

구분	룰 베이스	Decision Tree	Random Forest	Linear SVM	Naive Bayes	Neural Network	KNN
매수 신호	15	26	15	16	15	13	17
보유 신호	207	374	354	374	357	374	384
매도 신호	15	26	15	16	15	13	17
수익률%	-3.54	-13.95	-7.87	-5.72	-16.12	**-3.51**	-8.87

예측 모델 성능 평가 및 분석

- 삼성전자 예측 모델 성능 비교

구분	룰 베이스	Decision Tree	Random Forest	Linear SVM	Naive Bayes	Neural Network	KNN
매수 신호	10	16	12	23	9	9	16
보유 신호	207	401	383	368	421	401	347
매도 신호	9	15	11	23	8	8	15
수익률%	**39.17**	4.9	6.32	14.21	17.63	23.41	-10.51

예측 모델 성능 평가 및 분석

- SK하이닉스 예측 모델 성능 비교

구분	룰 베이스	Decision Tree	Random Forest	Linear SVM	Naive Bayes	Neural Network	KNN
매수 신호	14	19	13	13	8	8	8
보유 신호	199	319	331	327	407	337	365
매도 신호	13	19	13	13	7	8	7
수익률%	-25.43	20.85	14.03	28.35	-9.74	**33.97**	19.66

Q & A