석사학위논문

# 주식시세 예측을 위한 딥러닝 최적화 방법 연구

지도교수 이영구

경희대학교 대학원
소프트웨어융합과

정 욱

2018 년 2 월

# 주식시세 예측을 위한
# 딥러닝 최적화 방법 연구

지도교수 이영구

이 논문을 석사 학위논문으로 제출함


## 경희대학교 대학원
## 소프트웨어융합과

## 정 욱


2018 년 2 월

# 정 욱의 공학 석사학위 논문을 인준함

주심교수 채 옥 삼 ㊞

부심교수 홍 충 선 ㊞

부심교수 이 영 구 ㊞

## 경희대학교 대학원

2018 년 2 월

Table of Contents

**List of Table** ..... iii
**List of Figure** ..... iv

**국문 요약** ..... v

**I. 서 론** ..... 1

**II. 주가 예측과 딥러닝** ..... 5

**1. 딥 러닝** ..... 5
**2. 순환 신경망(Recurrent Neural Network, RNN)** ..... 5
**3. LSTM (Long Shorterm Memory)** ..... 6
**4. RNN-LSTM 하이퍼 파라미터** ..... 7
**5. Fin-Tech 와 인공지능** ..... 8
**6. 기존 시계열 예측 연구** ..... 11
**7. 주식시장의 생체 리듬** ..... 12

**III. 주가 예측을 위한 딥러닝 최적화** ..... 13

**1. 데이터 취득 및 전처리** ..... 13
**2. 하이퍼 파라미터 탐색 자동화** ..... 15
**3. 주가의 특징별 하이퍼 파라미터 적용 분석** ..... 16
**4. 거래시간 Feature 적용 학습시 예측 정확도 검토** ..... 16
**5. 실험 데이터** ..... 17
**6. 실험 환경** ..... 18
6.1. 데이터 수집기 ..... 19
6.2. 데이터 전처리기 및 모델러 ..... 19
6.3. 교차검증기 ..... 20
6.4. 예측 및 차트 분석 ..... 20
**7. 실험 방법** ..... 21
7.1. 거래시간 feature 학습 실험 ..... 21
7.2. 하이퍼 파라미터 탐색 실험 ..... 22

**IV. 주가 예측을 위한 딥러닝 최적화 평가** ..... 24

**1. 실험 결과 요약 ..... 24**
**2. 거래시간 Feature 학습 검토 ..... 25**
2.1. KODEX 기반 종목 거래시간 + 종가 feature 학습 ..... 25
2.2. 고 변동성 종목 거래시간 + 종가 feature 학습 ..... 27
**3. 하이퍼 파라미터 탐색 ..... 30**
3.1. Epoch 탐색 ..... 30
3.2. Batch 탐색 ..... 32
3.3. Hidden Node 탐색 ..... 34
3.4. Dropout 탐색 ..... 35
3.5. 최적 모델과 최악 모델 ..... 37
**IV. 결론 ..... 42**
**Grossary ..... 43**
**참고 문헌 ..... 44**

## List of Table

<표 1> 주요 로보어드바이저 현황 [15] ..... 10
<표 2> 하이퍼 파라미터와 범위 ..... 15
<표 3> 거래시간 Feature 학습 데이터셋 ..... 22
<표 4> 하이퍼 파라미터 리스트 ..... 22
<표 5> Kodex 기반 종목 거래 시간별 종가 특성 ..... 25
<표 6> kodex 기반 종목 모델 예측 정확도 ..... 27
<표 7> 고 변동성 종목 거래 시간 특성 ..... 28
<표 8> 고변동성 종목 모델 예측 정확도 ..... 29
<표 9> 하이퍼 파라미터 Epoch 검토 ..... 31
<표 10> 하이퍼 파라미터 Batch size 검토 ..... 32
<표 11> 하이퍼 파라미터 Hidden node 검토 ..... 34
<표 12> 하이퍼 파라미터 Dropout 검토 ..... 36
<표 13> 하이퍼 파라미터 s Bestfit vs 최악 모델 ..... 38

## List of Figure

<그림 1> Recurrent Neural Network [17] ..... 5
<그림 2> RNN-LSTM Shell [18] ..... 7
<그림 3> 단타매매 상위 10 개 종목 ..... 18
<그림 4> 실험 환경 구성 ..... 19
<그림 7> KODEX 지수 기반 종목 거래 시간 특성 ..... 26
<그림 8> KODEX Leverage 거래시간 + 종가 ..... 27
<그림 9> KODEX Leverage 종가 only ..... 27
<그림 10> 고변동성 종목 거래 시간 특성 ..... 29
<그림 11> 미래산업 거래시간 + 종가 ..... 29
<그림 12> 미래산업 종가 Only ..... 29
<그림 13> Epoch 탐색 ..... 31
<그림 14> Batch Size 탐색 ..... 33
<그림 15> Hidden nodes 탐색 ..... 35
<그림 16> Hidden nodes 탐색 ..... 37
<그림 17> Tiger 200 ETF 최적 모델 vs 최악 모델 ..... 39
<그림 18> 미래산업 최적 모델 vs 최악 모델 ..... 40

# 국문 요약

## 주식시세 예측을 위한
## 딥러닝 최적화 방법 연구

**경희대학교 대학원**

**소프트웨어융합과**

**정 욱**

본 연구에서는 딥러닝을 이용한 주가 예측 정확도를 높이기 위한 방법을 제시한다. 해당 주가 예측 플랫폼은 증권사 시스템 연계를 통한 주식시세 수집 및 예측 모델의 생성 및 하이퍼 파라미터  최적화를 통한 Model 튜닝 및 관련 거래 시간 피쳐의 제공을 추가하여 주가 예측치의 보다 높은 예측 정확도를 제공 한다. 주가 예측에 사용 되는 알고리즘은 순차적인 입력 데이터로 부터 이후의 데이터를 예측하는 순환 신경망(RNN : Recurrent Neural Network)을 사용한다.  순환 신경망은 기본적으로 일련의 순차적인 데이터를 학습하는데 정확도를 떨어뜨리고 과적합 문제를 발생시키는 장기의존성문제를 가지고 있는데, 이를 개선하기 위해 고안된 LSTM(Long Short Term Memory) 을 사용하여 주가 예측을 수행 하였다. 주가의 패턴에 따른 튜닝점을 탐색 하고자 지수 기반의 Kodex ETF 류의 종목과 단타 매매가 많이 이루어지는 변동성이 큰 종목의 데이터 를 수집하여 각 하이퍼 파라미터  의 자동 검사를 구현 하였고, 모델 튜닝 의 결과를 검토 하였다. 추가로 각 종목의 시간별 주식 거래 패턴을 피쳐로 선정하여, 시간 피쳐의 적용시 주가 예측의 정확도가 증가함을 확인 하였다.

키워드
주가 예측, RNN, LSTM, cross validation, 모델 튜닝, 하이퍼 파라미터 , trading 거래시간 feature

# I. 서 론

주식 시장에 많은 개인 투자자 들이 거래에 참여 하고 있다. 하지만 시장에서 정보력과 전문성을 가지고 있는 외국인 투자자들 과 기관 투자자들을 상대로 거래에서 수익을 내기는 쉽지 않다. 개인 투자자로서 거래에서 수익을 내는 방법으로, 순수하게 데이터만을 가지고 기술적인 분석을 통해 짧은 순간에 매매를 하여 리스크를 줄이는 방법을 스켈핑 이라고 하는데, 이를 자동화하여 매매하는 시스템을 구현함에 있어서, 기존에 연구된 주가 예측 알고리즘을 검증 하고, 이를 활용하여 주가 예측 정확도를 높히는 방법을 제안하고 실험 한다.

기존 연구에서는 시계열 데이터를 예측 하는 방법에 있어서 알고리즘이나 공식을 제안하여 과거 데이터로 부터 패턴이나 트렌드를 파악하여 특정 시점에서의 예측을 진행하는데 최근에는 딥러닝을 통하여 시계열 데이터로부터 학습을 통한 예측을 시도 하는 연구가 활발히 진행 되고 있다. 주식 시세의 예측에 있어서 타 시계열 데이터의 예측과 많이 다르지 않지만, 예측치의 실제 거래에서의 사용성을 고려 한다면 아래와 같은 요구 사항을 만족 해야 한다.

첫번째로 데이터의 종류 및 시간 도메인에 대한 고려가 필요하다. 포털 사이트 및 증권 정보 사이트에서 얻을 수 있는 주식시세 데이터는 보통 하루 단위의 데이터로 순수하게 데이터만을 가지고 짧은 시간내에 거래하는 시스템에는 맞지 않다.

실제 거래에 사용하는 데이터 라면, 학습 / 예측 및 거래를 위한 API Call 지연등을 충분히 고려되어야하고, 시간대별로 의미 있는 차익을 얻을 수 있는 시간이어야 하며, 하루내에 충분한 거래 단위 수를 만들 수 있어야 한다. 본연구에서는 증권사의 API 를 연계하여 60 분 단위의 데이터를 학습 및 예측에 사용하였다. 종목 별로 각 시간별로 차이가 있어, 여러 시간 도메인에서 실험하면 더 좋을것이나, 실험 환경 및 시간의 한계등 어려움으로 60 분 데이터만을 살펴 보았다.

두번째로 예측의 정확도가 보장되어야 한다. 기존의 연구에서 살펴 보면, ARIMA, RNN, CNN 등의 방법을 사용하여 어느정도 트렌트를 맞추거나, 예측이 60~70% 정도로 맞는 정도를 제시 한다. 하지만, 수차례의 거래에서 차익을 실현해야 하는 목표를 가지는 예측이 필요하다면, 학습 및 모델을 생성 함에 있어서 예측 정확도를 최대로 끌어 올려야 한다. 하지만 기존의 연구들 에서는 예측 정확도가 어느 정도에 한정 되며, 이후 이를 높이는것 까지는 언급이 없었다.

본 연구에서는 RNN-LSTM 을 이용하여 주가 예측을 함에 있어서 종목별로 RNN-LSTM 의 특징을 정의 하는 하이퍼  파라미터 의 Tuning 을 통한 예측 정확도의 최대화를 제시 하였다.

세번째로는 학습데이터의 다변화 가 필요하다. 시계열 데이터를 예측 하기위해 학습 함에 있어서 필요한 최소 데이터 종류는 한종류, Target 이 되는 시계열 수치이며, 해당 수치만 있으면 어느정도 예측이 가능 하다. 하지만 주가는 단순한 패턴의 반복 혹은 단순한 트렌드 라기 보다는 수많은 지표 / 상황 / 거래심리 등이 반영된 복잡한, 예측이 힘든 시계열이라는점을 고려 할때 학습이 가능한한 다양한 종류의 학습데이터를 제공한다면, 좀더 나은 예측이 가능 하다.

본 연구에서는 보조 지표중 가장 명확하지만, 종목별로 복잡한 거래 심리 등을 손쉽게 반영 해볼 수 있는 feature 로써 거래 시간을 추가 하여, 단순히 종가 만 학습 했을 때와 학습 및 예측 정확도의 비교 분석을 수행 한다.

네번째로는 시스템의 성능이 실제 거래에 사용 가능 할 만큼 좋아야 한다. 딥러닝의 학습은 인공 신경망의 소프트웨어 구현체으로 제공 되는데, 이는 수많은 데이터를 수많은 연산을 통해 검증 하고 재 연산 하는 반복 작업으로 많은 컴퓨팅 자원을 필요로 한다. 그리고 보통은 충분한 학습이

수반되었을때 비로소 쓸만한 정확도를 보장 할 수 있는 모델을 생성 할 수 있는데, 실험환경의 제한된 컴퓨팅 리소스 안에서 실제 거래에 사용 할 만큼 빠르게 정확한 예측을 제공하기는 어려웠다. 추후 예측 정확도를 보장하며 더 빠른 학습이 가능 하도록 하는 연구 개발이 더 필요 하다.

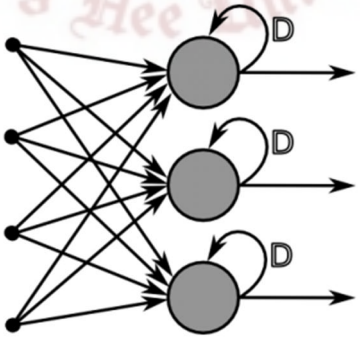
# Ⅱ. 주가 예측과 딥러닝

## 1. 딥 러닝

기존의 통계학적인 접근으로 예측치를 구하거나, 분류나 상황을 판별을 하는 방법으로 여러가지 알고리즘이 있지만 최근에는 기계학습을 통해 문제의 해를 구하는 딥러닝이 대두 되고 있다. 문제나 상황을 정의 하기위한 일련의 데이터를 컴퓨터에 제공하고, 컴퓨터가 이를 학습하기 위한 구조를 정의하는 연구가 활발히 진행 되고 있으며, 이런 연구의 결과로 Deep neural networks, recurrent nueral network, convolutional deep neural networks 등이 고안 되었고, 이는 자연어, 시계열데이터, 음성인식 등의 처리에 사용 되고 있다.

## 2. 순환 신경망(Recurrent Neural Network, RNN)

순환 신경망은 인공신경망을 구성함에 있어서 특정 부분이 반복 되는 구조를 가진다는 의미인데, 기존의 신경망이 단순히 각층의 뉴런이 연결되는 구조 였다면, RNN 은 아래 <그림 1>과 같이 은닉층에 자신을 가리키는 구조를 가진다.

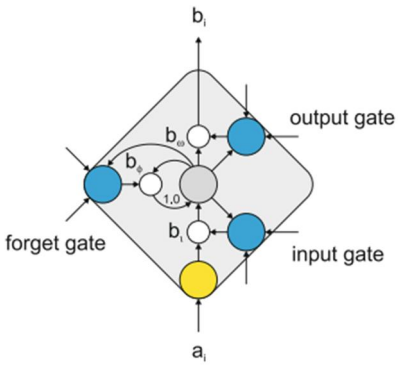



*<그림 1> Recurrent Neural Network [17]*

이렇게 자신을 가리키는 부분이 인공신경망에 추가된 구조로서 Recurrent Weight 라고 부른다. 이 구조는 과거 데이터에 대한 기억을 어렴풋이 기억하도록 하는데, 이구조가 연결되 데이터를 학습하고 이로부터 예측을 할 수 있도록 하는데 도움이 된다. 그러나 이구조는 Gradient Vanishing Problem 을 발생 시키는 문제가 있다. 이는 학습을 계속 함에 있어서 반복되어 곱해지는 Recurrent Weight 에 의해 기울기가 급하게 줄어들거나 혹은 반대로 기울기가 너무 급해저셔 Gradient Exploding Problem 이 발생 한다.

### 3. LSTM (Long Shorterm Memory)

RNN 의 Long-Term Dependency 를 해결하기 위해 LSTM 은 아래 <그림 2>와 같이 내부가 Simple RNN 보다 훨씬 복잡한데 하나의 셀에 history gate, output gate, input gate 뿐만 아니라, forget gate 가 추가로 존재 한다. 이는 history state 의 전달에 있어서 Simple RNN 과는 다르게 무조건 history state 를 전달 하기 보다, 설정된 forget 상수에 따라 가끔 history state 를 망각 하게하여, 데이터 입출력을 조절하여 필요할때만 history state 를 update 하고 데이터를 넣고, 빼는 기능을 가지고 있다. 이로써 Long term Dependency 문제인 Gradient Vanishing Problem, 또는 Gradient Exploding Problem 을 어느정도 해결하고, LSTM 을 이용한 반복 학습시 과적합 문제를 방지 할 수 있다.




*<그림 2> RNN-LSTM Shell [18]*

## 4. RNN-LSTM 하이퍼 파라미터

RNN-LSTM 의 학습 및 예측 정확도를 튜닝 하기 위한 하이퍼 파라미터는 Epoch count, Batch count, Hidden Node count, Drooput 이 있다. 이 하이퍼 파라미터 의 튜닝 을 통하여 종목 별 모델을 만듦에 있어서 최적의 예측 정확도를 제시 할 수 있다.

- Epoch : 주어진 학습 set 의 전체를 한번 학습하는 횟수를 정의 하며, 얼만큼 반복하여 학습 할것인지를 결정하는 파라미터이다. 매 epoch 시 마다 weight 값이 바뀌게 되는데, 이를 잘 모니터링 할 필요가 있음. 부족하면 학습 이 덜 되고, 과하면 과적합이 발생 할 수 있다.
- Batch : 주어진 학습 set 을 어떻게 나누어서 학습 할 것인지를 결정하는 파라미터 , 크기에 따라 패턴 학습 의 정도가 달라질 수 있다.

- Hidden Node : Hidden 수에 따른 신경망의 복잡도와 메모리 크기 결정하는 파라미터 - 복잡해질 수록 계산량이 많아지며, 복잡한 패턴의 계산이 가능하다.
- Dropout : Hidden Layer 사이에 과적합을 방지하기위해서 state Drop rate 를 설정함. 이는 forget gate 에서 history state 를 갱신함에 있어서 dropout 상수로 정해진 만큼의 forget 을 통해 새로운 입력을 학습하기 용이하게 하게 하여, long-term dependency 문제를 해결하는 장치로 사용된다.

## 5. Fin-Tech 와 인공지능

2015 년의 알파고 사태 이후 인공지능은 마치 요술 방망이와 같은 처지에 놓여있는것 같다. 다들 있으면 좋겠다고 생각 하고 있고, 누군가는 뛰어난 인공지능 시스템을 갖추어 시대를 앞서 나가고 있다고 선전하는데, 정작 개인으로써는 만능에 가깝고 대적할만한 상대가 없다고 하는 인공지능을 내손으로 쓰이기는 어려운것이 사실이다. 시중에 나와있는 인공 지능 제품들은 대게 말을 알아 듣거나, 영상을 인지 하여 기존에 사용자가 직접 수행하던 검색을 대신 해주는 등의, 컴퓨터 입력의 다변화 정도가 실제로 사용되어지고 있다.

하지만 분명한것은 이제 인터넷을 하는 사람들이라면, 바둑에 있어서는 사람이 알파고를 이기기 어렵고, 자동차 회사들은 AI 기술을 이용한 자율주행차들을 테스트 하고 있다는것을 잘 알고 있다. 금융거래에 있어서도 로보어드바이저가 초기 단계 이긴 하지만, 관심을 받으며 조심스러운 성장을 하고 있다. 시간도 돈도 부족한 현대인들의 현실에 로보어드바이저는 가장 좋은 대안을 제시 할 수 있다.

첫번째로는 금융거래를 자동화 함 으로써 개인의 판단을 최소화 하고, 정신 노동을 최소화 할 수 있는 점이 장점이다. 개인이 주식을 하면서 가장 어려운점이 원칙을 지키기 어렵다는 점인데, 시스템이 이를 대신하게 되면, 개인은 원칙을 100% 지키게 되면서도, 언제팔지, 언제 살지에 대해 고민을 하지 않을 수 있게 된다. 물론 이를 위해서는 시스템이 안전하게 수익을 내며 거래한다는 100% 믿음이 있어야 한다. 실제로 미국의 Knight Capical 사는 2016 년 4.4 억 달러의 오류 주문으로 인해 파산의 위기에 봉착 하기도 했다.[15]

둘째로는 로보어드바이저는 개인이 살펴보기 힘든 찰나의 시간에도, 수 많은 종목의 데이터를 꾸준히 검토 할 수 있다. 바쁜 현대인들이 짬을 내어서 혹은 상사 몰래 핸드폰으로 가끔 거래 하게 되는데, 기술적으로, 시간적으로 전문 딜러보다 부족할 수 밖에 없다. 이를 로보어드바이저가 대신 하게 되면 사람이 볼 수 있는 시간의 영역을 넘어선 분석이 가능하게 되고, 시간에 구애 받지 않고 거래를 계속 할 수 있게 해준다. 하지만 이것도 역시 로보어드바이저의 알고리즘과 거래 로직이 먼저 검증되어야 한다.

| 구분 | 금융회사 | 내용 |
|---|---|---|
| 은행 | 우리은행 | 파운트와 제휴한 '로보어드알파' 출시, ISA에 접목 가능 |
| | KB국민은행 | 은행권 최초 쿼터백투자자문과 MOU 체결해 '쿼터백 R-1' 출시 |
| | KEB하나은행 | 2016년 2월 금융상품 포트폴리오 서비스 'Cyber PB(Private Bank)' 출시, 국내 은행 최초 자체 개발, ISA에 접목 예정 |
| | 신한은행 | 2016년 4월 로보어드바이저 기반 펀드추천서비스 'S로보플러스' 출시 |
| | NH농협은행 | 2016년 8월 은퇴설계와 퇴직연금 자산운용기능 연계한 'NH로보-프로(NH Robo-Pro)' 출시 |
| 증권 | 삼성증권 | 국내 최초로 로보어드바이저 핵심기술인 '투자성과 정밀검증 알고리즘 시스템' 관련 특허 출원 완료 |
| | 신한금융투자 | 2016년 4월 밸류시스템 로보어드바이저인 '아이로보' 포트폴리오를 기반으로 하는 '신한명품 밸류시스템 자문형 로보랩' 출시 |
| | 한국투자증권 | 2016년 4월 어카운트상품 운용 과정에 로보어드바이저의 판단이 반영된 '한국투자로보랩' 출시 |
| | 현대증권 | 2016년 2월 로보어드바이저에 기반한 일임형랩인 '현대able로보랩' 출시 |
| | NH투자증권 | 2011년부터 운용했던 온 스마트인베스터 특허기술을 기반으로 ETF를 자동매매하는 'QV로보어카운트' 서비스 오픈 |
| 자산운용 | 미래에셋자산운용 | 2015년 5월부터 국내 최초의 로보어드바이저 시스템인 '글로벌 자산배분 솔루션' 제공 |
| | 삼성자산운용 | 연내 모멘텀 솔루션 등 퀀트분석에 기반한 ETF 자산배분 서비스 출시 예정 |
| | KB자산운용 | 연내 계열은행 증권사 통한 로보어드바이저 투자솔루션 제공 예정 |

*<표 1> 주요 로보어드바이저 현황 [15]*

국내에도 위의 <표 1> 와 같이 초기 단계의 수동적인 범위의 로보어드바이저가 많이 존재한다. 하지만 현재 로보어드바이저는 대부분 개인 성향별 포트폴리오를 관리 하는데 중점을 두고 있다. 아직까지 개별 종목의 주가 예측을 이용한 자동거래는 아직 위험 하기도 하고, 로직에 의한 이익률을 증명하기 어려워서 실제로 서비스로 운영되기는 어려운 측면이 존재 한다. 이를 발전시키기 위해서는 가장 첫 단계로 개별 종목 주가의 예측 정확도를 높힐 수 있어야 하겠고, 궁극적으로는 이를 이용하여 실제 거래에서 의미 있는 이익을 낼증명 할 수 있을때 로보어드바이저를 넘어선 주식 자동 매매 시스템으로 발전 할 수 있다.

## 6. 기존 시계열 예측 연구

최근 주가 변동을 딥러닝 또는 Linear regression 을 통해 예측 해보는 연구는 활발히 진행 되고 있다. 다중 퍼셉트론 모델을 이용한 연구에서는 특정 종목에 대해서 정해진 하이퍼 파라미터  으로 종가를 예측 하여 50%를 상회하는 정도의 예측률을 내는 결과를 보여준다.[12] 해당 연구에서는 딥러닝을 통하여 주가를 예측 한 것에 의미를 두고 있으며, 추후 여러가지 설정을 실험하여 예측도를 높히는 것을 추후 목표로 지정 하고 있다.

또한 다른 연구에서는 순환신경망(RNN) 하이퍼 파라미터 의 알고리즘 평가를 하는데 대상 데이터 가 특정 상황에 정해진 한가지 데이터 만을 대상으로 하여, 그 상황에서 가장 적절한 하이퍼 파라미터  이 무엇인지를 평가하는 작업을 보여준다. [13] RNN 은 해당 데이터가 어떤 상황인지, 어떤 시간을 표현하고 있는지 알지 못하며, 만약 데이터의 종류가 바뀌거나, 시간 도메인이 변경 될 시, 모델생성을 위한 학습이 새로 필요할 수 있다. 이때 적절한 하이퍼 파라미터 은 항상 같다고 단정 할 수 없으며, 만약 하이퍼 파라미터 이 고정된채로 학습이 이루어 진다면, 기존의 예측 정확도는 새로 학습된 모델에서는 같다고 할 수 없다. 다시 얘기 하자면, 데이터가 바뀌면 최적의 하이퍼 파라미터 도 역시 바뀔 수 있다.

국내외 연구에서의 순환 신경망이나 Regression 을 사용하여 시계열의 예측을 평가 하는 논문에서는 대체로 특정 시계열 데이터를 정하여 해당 데이터의 특정 시점을 예측 하여, 예측 정확도 를 평가 한다. 이는 해당 논문에서 평가 하듯이 어느정도 (60~70% 이상) 의 적중률을 내는것으로 결론이 나지만, 데이터가 바뀔시 다시 어느정도 예측이 되는지에 대해서는 얘기 하지 않는다. 그리고 제시된 60~70 의 예측 정확도를 좀더 높이기 위한

노력은 다양한 데이터 및 다양한 시간 도메인 상에서의 시도가 아쉬운 부분이다.

## 7. 주식시장의 생체 리듬

시간대별 주식시장의 생체리듬이 있다 라는 주장이 있는데, 이는 시장 참여자들의 공유 심리변화가 시장에 영향이 있다는 설이다. [16] 각 단계별로 하루를 나누어 볼 수 있는데, 예를 들면, 오전 9 시 부터 2 시간 반 정도는 어제의 미국 시장의 분위기가 장초반에 반영 될 수 있어 미국 지수에 민감하게 움직이고, 외국인 대규모 투자자들의 매매가 이루어져서 급격한 지수 변화를 보이는데, 국내 개인투자자 들도 가세하여, 시장 분위기의 변화가 큰 시간이다.
이후 오후 1 시까지는 오전에 크던 변동성이 갑자기 잠잠해지거나, 잠잠했던 시장이 요동치는 경우가 있다는 설이다. 이는 점심 시간을 중심으로 잠잠하던 거래가 다시 활성화 되어, 하락 중이던 장세가 상승으로 반전 하거나, 상승 하고 있던 장세가 하락 하게 되는 계기를 맞는다고 한다.
이후 2 시 반 까지는 가장 진지한 심리가 작용하는 시간대로, 단기교란에 가장 영향을 덜 받고, 트렌드가 반영되는 시간 대라고 한다.
이후 장 마감까지는 종가 관리 차원의 안정된 트렌드 위에서 시세가 움직이는 경우가 있다고 한다.
위에서 본 것 처럼, 실제 주가의 움직임에는 각 시간대 별로 특징을 보이는데 본연구에서는 각 종목 별로 거래 시간대별 종가의 특징을 먼저 살펴 보고, 실제 딥러닝 학습시 피쳐로 추가하여, 시간대가 가지는 특징을 딥러닝의 뉴럴네트워크가 판단하여 예측의 정확도를 높 힐 수 있도록 검토 한다.

# III. 주가 예측을 위한 딥러닝 최적화

본연구에서는 시계열 데이터인 주가의 효과적인 예측을 위하여, 첫번째로 딥러닝 방법론중 RNN-LSTM 인공 신경망을 구성함에 있어서, 하이퍼 파라미터 (Epoch / Batch / Hidden Layer node / dropout )별 교차 설정을 통해 주가 예측 정확도 를 비교 검토하고, 종목별, 종목 특성별 최적의 하이퍼 파라미터 를 어떻게 찾을 것인가 하는 점에 집중하고, 이를 위한 실험을 진행한다.

인공 신경망의 구성과는 별개로 데이터를 제공하는 방법을 통해 주가 예측의 정확도를 높이는 방법의 일환으로, 주가의 움직임과 상관 관계가 있는 거래 시간 피쳐를 추가 하여, 주식시세 데이터의 특징과 딥러닝 학습 및 예측 정확도 의 상관 관계를 파악 한다.

## 1. 데이터 취득 및 전처리

실험을 위해 실제 거래에 사용 할 수 있는 1 시간 단위 주가 1 년 분을 증권사 서버로부터 취득 하였다. 증권 거래에 사용할 데이터의 시간 도메인은 첫번째로 하루중에 최대한 많은 거래를 할  있도록 최대한 짧은 것이 좋다. 하지만, 예를 들어 거래 틱 단위, 1 분 단위의 데이터를 사용하면 모델의 갱신이 필요한 시점에 학습을 할 시간이 턱없이 부족한점, 너무 짧은 시간텀은 거래에서 의미있는 이득을 바랄 만큼 종가의 차이가 적은점, 그리고 시스템상에서 거래에 사용할 수있는 시간이 HTS 시스템 데이터 취득 – 예측 – 거래 판단 – HTS 거래 수행 사이에 많이 부족하다. 그래서 거래에 사용할 만큼 종가의 차이가 있으면서, 재학습을 할만하고, 시스템

거래 싸이클에도 부담이 없을 만한 시간은 60 분으로 초기 설정 하였다. 재학습과 시스템 거래 사이클은 시간이 지남에 따라 숙련도가 높아지면서 또는 하드웨어 가 보충 되에 따라 점차 나아질 것으로 예상한다. 추후 기회가 된다면 60 분에서 10 분 사이까지도 시도 해 볼 수 있을 것으로 기대한다. 60 분 데이터의 단점은 1 년치 데이터 가 1660 건 으로 Sample 수가 많지 않다는 것이다. 이는 반복 학습을 많이 해야 모델이 정확도가 높아지는데, 반대로 과적합이 되기 쉽다는 점이다. 이를 방지 하기 위해서 학습중에 MSE 가 높아지는 과적합점을 잘 모니터링 하고, Dropout 등의 하이퍼  파라미터 을 신중히 조절해 볼 필요가 있다.

분석에 사용된 주가는 지수 기반의 Kodex ETF 류의 비교적 변동성이 적고 거래량이 많아 안정적인 종목과 단타 매매가 전체 매매의 50% 이상을 차지하는 변동성이 큰 종목으로 두가지 형태의 데이터 를 수집하였다. ETF 종목은 KODEX 대표 지수 기반으로 종가는 각각 다르나, 트렌드는 종목별로 큰 차이가 없이 대동 소이한 모습을 보인다. 지수 기반이 아닌 변동성 위주의 종목은 각각 다른 트렌드를 가지며, 이슈에 따라 시간에 따라 큰 변동폭을 가져서, 예측이 쉽지 않은 특징을 가진다.

데이터가 수집되면 수집서비스에서 모델링 및 데이터 분석 서비스 쪽으로 데이터를 전달 하게 되고, 해당 데이터는 학습을 위해 피처(종가  only , 종가 + 거래시간 (거래시간) )를 정리 하고,  한타임 앞의 종가  를 해로 제공하여, Supervised 학습을 위한 데이터 형태로 재 가공 하였다.

데이터는 RNN-LSTM 에서 학습 및 예측의 MSE (Mean Squared Error) 를 효과적으로 모니터링 하기 위하여, 0 to 1 Scaled Normalize 를 진행하였다. 이후 딥러닝 모듈에서는 해당 데이터에 대한 학습을 진행하여 모델을 생성 한다.

## 2. 하이퍼 파라미터 탐색 자동화

전처리를 마친 데이터는 RNN-LSTM model 생성을 위한 학습을 진행 하였다. 각 하이퍼 파라미터 (Epoch / Batch / Hidden Layer node / dropout) 별 후보 list 를 먼저 대략적인 범위를 지정 하였다. 이후 실험을 진행 하며, 적절히 Value range 를 재지정 하여 최종적으로 아래 <표 2>와 같이 설정 하였다.
Epoch 는 전체 학습데이터에 대한 학습의 반복 회수로서 최소 5 회 부터 최대 100 회를 지정했으며, Batch count 는 전체 학습 데이터를 어떻게 나누어 학습 할것인가에 대한 파라메터로 10 에서 부터 150 으로 지정하였다. RNN Hidden Layer Node count 는 RNN 내부의 숨겨진 노드의 개수를 1 개 에서 부터 160 개까지를 지정 하였고, Dropout 상수는 0.05 에서부터 0.6 까지로 지정하였다.

| Item | Values |
|---|---|
| Epoch count | 5 ~100 |
| Batch count | 10 ~ 150 |
| Hidden Layer Node count | 1 ~ 160 |
| Dropout rate | 0.05 ~ 0.6 |

*<표 2> 하이퍼 파라미터와 범위*

Scikit Learn 의 Cross Validation framework 을 사용하여 위에 정의된 Epoch, Batch, HiddenLayer Node count, Dropout rate 의 하이퍼 파라미터의 탐색을 진행 하였다. 각 하이퍼 파라미터의 최적 조합을 찾기 위해서는 교차검증 framework 안에서 각 대상 파라미터를 격리 시키고, 모든 조합의 매트릭스 안에서 최고 점수를 찾아내는 방식으로 진행 된다.

실험환경의 경우 GPU(Nvidia CUDA 프레임워크) 를 사용하여 기존의 CPU 만 사용 하는 경우보다 훨씬 빠른 연산이 가능함에도 불구하고, 1fit, 한

세트의 학습을 진행시 최소한 30 초가 소요 되어서 전체 Cross Validatin 을 자동으로 수행하기 위해서는 10 시간이 넘게 소요 되었다.

### 3. 주가의 특징별 하이퍼 파라미터  적용 분석

변동이 적은 지수기반의 ETF 종목, 그리고 단타 매매가 많아 변동성이 큰종목을 구분하여, 각 특징을 대표하는 종목에 어떤 하이퍼 파라미터 을 적용 했을때 예측 정확도가 높아지는지 검토 하였다. 이를 위해서 ETF 종목은 세가지 Kodex leverage, Kodex Kospi, Tiger 200 을 사용하며, 단타 매매가 많은 종목으로는 미래산업, 에이프로젠 제약, 이아이디를 사용 하여 검증 해보았다.

### 4. 거래시간  Feature 적용 학습시 예측 정확도 검토

주가예측의 결과 Value 로 종가를 사용하는데 기존의 방법과 는 다른게 예측의 1 차적인 대상인 종가 뿐만 아니라, 종목별로 시간별 거래 패턴이 반영된 예측을 진행 하고자, 거래 시간을 추가 Feature 로써 검토 하였다. 기존 time stamp 인 Datetime 형식에서 시간 특성만을 추출하여 10, 11, 12, 13, 14, 15 와 같이 정리 하였고, 종목별로 시간 별 특성을 먼저 확인 하고, 실제 RNN-LSTM 학습에서 사용 하여, 학습과 예측시의 정확도 를 평가 하였다.
각 종목의 거래시간 별로 주가 변화율은 차이가 많았다. 어떤 종목은 장시작 시에 가장 많이 상승하고 점점 상승폭을 줄이는가 하면, 특정 종목은 장 시작과 마감시에는 잠잠 하고 점심 이후에 강세를 보이는 종목 도 있었다.
추가 feature 로서 Year, Month, Season 등도 검토 했으나, 증권사에서 제공하는 데이터가 현재로 부터 과거 1 년 뿐이라서, year, month, season 을 검토 하기에는 sample 수가 부족하였다.

## 5. 실험 데이터

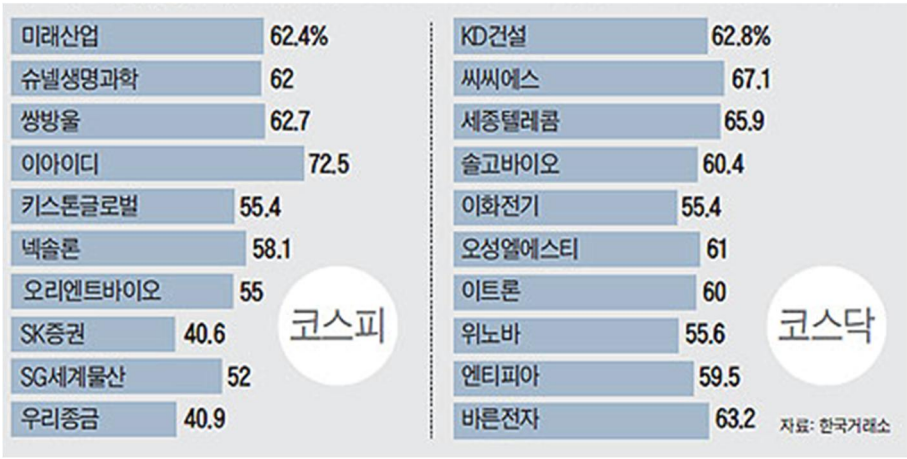
실험 데이터 로는 EBEST 증권사의 API 로부터 취득한 1 년치 주식시세를 사용 한다. 딥러닝시 데이터는 많으면 많을 수록 좋지만, 증권사에서 제공하는 데이터는 최근 1 년치만 을 제공한다. 그래서 거래에 사용할 만큼 종가의 차이가 있으면서, 재학습을 할만하고, 시스템 거래 싸이클에도 부담이 없을 만한 시간은 60 분 단위 데이터를 수집 하였다. 추후 기회가 된다면 60 분에서 10 분 사이까지도 시도 해 볼 수 있을 것으로 기대 한다.

주식 시세의 종류는 시간 도메인의 구분(10 분(추후)/30 분(추후)/60 분) 과 추세의 종류에 따라 상대적으로 변동성이 적은 지수 기반 종목, 볼륨이 큰 대형주, 상대적으로 변동석이 큰 소형주 구분으로 사용한다.

지수 기반 종목 - kodex 기반 ETF 종목
- kodex leverage : 122630
- KODEX 코스피 : 226490
- tiger 200 : 102110

변동성이 큰 종목의 탐색은 실제 시스템상에서도 가능 하겠지만, 단타 매매가 많다고 매스컴 상에 알려진 종목을 선택 하였다. <그림 3> 의 2016 년 조선일보의 기사중에서 거래량이 많고, 변동성이 큰 종목으로 아래 종목으로 선정 하였다.

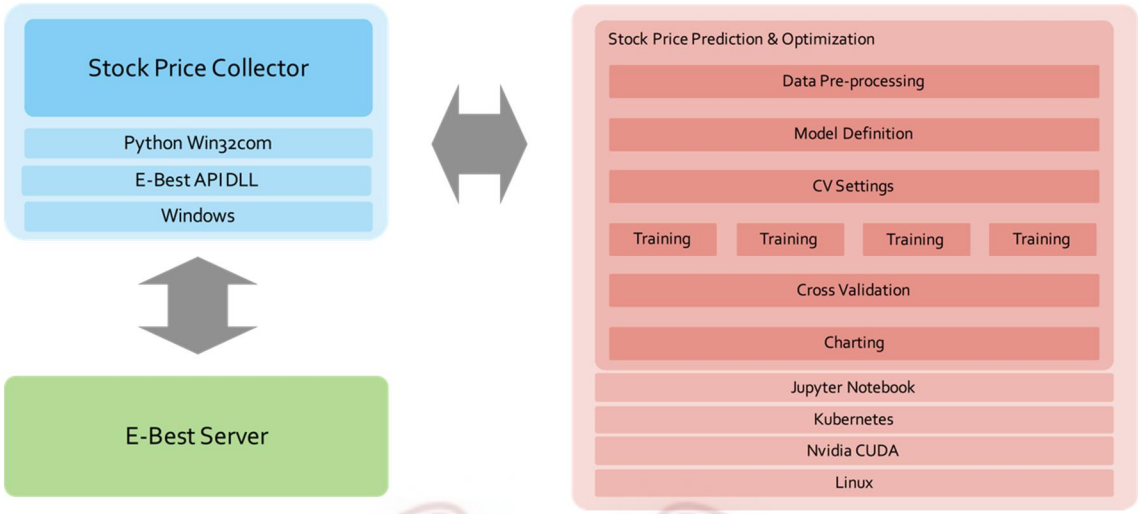
변동성이 큰 종목 : 코스피/코스닥에서 단타 매매가 많은 종목
- 미래산업 : 025560
- 에이프로젠제약 : 003060
- 이아이디 : 093230




*<그림 3> 단타매매 상위 10 개 종목 , 2016, 조선일보*

## 6. 실험 환경

실험 환경은 크게는 <그림 4>와 같이, 요구되는 종류별 데이터를 취득하여 딥 러닝모듈에 전달 하는 “데이터 취득 모듈”과 데이터를 수신 하여 학습과 평가를 진행하는 “딥러닝 모듈”, 하이퍼 파라미터 별 성능 평가를 담당하는 “하이퍼 파라미터 탐색 ing 모듈”, 평가할 하이퍼 파라미터 의 설정을 담당하는 “설정 모듈”로 나뉜다.




*<그림 4> 실험 환경 구성*

## 6.1. 데이터 수집기

1 시간단위 주가 1 년 분을 E 증권사 서버로부 터 취득 하였다. E 증권사에서 제공하는 API 를 사용하기 위해 Windows Machine 에 데이터 취득부를 구성 하였고, 추후 주문에도 사용 할 수 있도록 API layer 를 구성 하였다.

## 6.2. 데이터 전처리기 및 모델러

전처리를 마친 데이터는 LSTM model 생성을 위해 Deep learning 부에 전달 된다. 데이터를 학습에 적절하도록 시간순에 따라 학습 용과 테스트 용으로 나누며, 이때 Network 입력 길이등을 반영한 Look-back count 에 의한 3d 처리등을 진행한다. 거래시간 등 추가 feature 를 학습, 검증 데이터 에 넣고, 학습 효율 향상을 위해 Normalization 을 수행한다. Model 은 기존 참고

모델에서 차용 하되, LSTM 의 하이퍼 파라미터  을 변경 하여 시험 할 수 있게 구성 하였다.

### 6.3. 교차검증기

LSTM Model 의 하이퍼 파라미터  을 변경 하기 위해 설정 및 검증 기준을 설정 하였다. 각 하이퍼 파라미터  별 후보 list 및 범위를 지정 하여, 교차검증 프레임워크를 사용하여 Epoch, Batch, HiddenLayer Node count, Dropout rate 의 하이퍼 파라미터 의 탐색을 진행 하였다. RNN-LSTM 의 학습의 Cross validation 은 학습시간이 오래 걸리기 때문에 이를 최적화 하기 위해 Parallel fit 을 진행 하였다. 하나의 큐를 이용하여 전체 학습및 교차 검증을 진행하는 것보다 수배 빠르게 처리가 가능했다. GPU 를 사용하여 RNN-LSTM 학습을 하기 위하여, Nvidia CUDA framework 을 사용하였다. 그럼에도 불구하고, 전체 교차검증 프로세스는 종목당 10 시간 정도를 소모 하였다. 모든 종목의 매일 학습을 위해서는 첫번째로는 교차 검증 에 쓰이는 파라미터 리스트의 정제가 필요해 보인다. 추후에는 경험적으로 얻어진 각 종목별로 가능성이 높은 파라미터들만을 선별하여 진행 할 수 있을 것으로 보인다. 두번째로는 시스템 고도화 에 따른 학습 시간 단축을 진행 할 수 있다. 지금은 pararell Processing 에 한정된 컴퓨터 자원만이 사용이 되는데 전체 컴퓨터 자원을 할당 하여 교차검증 에 사용한다면 더 빠른 연산을 가능 하게 할 수 있다.

### 6.4. 예측 및 차트 분석

각 종목 별로 전체 교차검증 프로세스를 통과 하여 가장 Loss(MSE) 가 적은 하이퍼 파라미터 의 조합을 찾게 되면, 해당 하이퍼 파라미터 을 이용하여 학습 데이터 에 대한 재 학습을 진행 하여 최적 모델을 생성한다. 학습이후 해당 모델을 이용하여, 최근 데이터를 입력으로 사용하여, 알아보고자 하는 예측을 진행 하고 결과를 차트로 그려 검토 하였다.

## 7. 실험 방법

### 7.1. 거래시간 feature 학습 실험

주가는 거래 시간대별로 특징을 가진다라는 전제를 검토 하기 위해 실험데이터의 주가를 거래 시간대별로 분리 하여 각 시간대별 종가의 특성을 분석 하였다.
비교 지표는 1 년데이터 의 각 시간대별 종가의 평균치이며, 분석 방법은 10, 11, 12, 13, 14, 15 시 종가의 노말라이즈를 수행 하였다.

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

위 공식을 이용하여 종목별로 하루중에 가장 낮은 시간은 0 으로 표시되고 가장 높은시간은 1 로 표시되는 거래시간별 종가 특징을 알아볼 수 있다.

거래 시간대별로 종가의 특성이 다름을 확인하고, 딥러닝시 학습 feature 로서 거래시간 feature 를 추가하여 학습 효과의 결과를 검토 한다.
기존에 주가예측의 결과 수치로 종가 를 사용하는데 기존의 방법과 는 다른게 예측의 1 차적인 대상인 종가 뿐만 아니라, 종목별로 시간별 거래 패턴이 반영된 예측을 진행 하고자, 거래 시간을 추가 Feature 로써 검토

하였다. 기존 time stamp 인 Datetime 형식에서 시간 특성만을 추출하여 10, 11, 12, 13, 14, 15 와 같이 정리 하였고, 종목별로 시간 별 특성을 먼저 확인 하고, 실제 RNN-LSTM 학습에서 사용 하여, 학습과 예측시의 정확도 를 평가 하였다. 학습에 사용되는 종가 데이터는 해당 시간으로부터 4 시간 전 까지의 데이터로 총 5 개를 사용 하여 학습 하였다.

| | 학습용 데이터 | | 검증용 데이터 |
|---|---|---|---|
| 비교 데이터 셋 | 종가 | | 종가 |
| | $Close_{t-4}, Close_{t-3}, Close_{t-2}, Close_{t-1}, Close_t$ | | $Close_{t+1}$ |
| 실험 데이터 셋 | 종가 | 거래시간 | 종가 |
| | $Close_{t-4}, Close_{t-3}, Close_{t-2}, Close_{t-1}, Close_t$ | $Hour_t$ | $Close_{t+1}$ |

*<표 3> 거래시간 Feature 학습 데이터셋*

### 7.2. 하이퍼 파라미터 탐색 실험

각 종목별 최고의 예측 적중률을 찾기 위한 하이퍼 파라미터 의 탐색은 각 하이퍼 파라미터 즉 Epoch count, Batch count, Hidden node count, Dropout rate 의 파라미터 array list 를 아래 <표 4>와 같이 정의하였다.

| 파라미터 | values |
|---|---|
| epochs | 5, 10, 50, 100, 200, 400, 600, 1000, 1500 |
| batches | 10, 20, 30, 40, 50, 75, 100, 150 |
| Hidden nodes | 1, 2, 5, 10, 20, 40, 80, 160 |
| dropouts | 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 |

*<표 4> 하이퍼 파라미터 리스트*

하이퍼 파라미터 탐색을 위해 학습별 파라미터 list 는 epoch 9 개, batches 8 개, hidden nodes 8 개, dropout 7 개 의 조합으로 파라미터 그리드를 생성하여 각 조합에 대한 3 회 검증을 포함하여 search(9+8+8+7)*3 = 96 회 학습을 검토하였다.

이때 모델에 의한 예측 종가와 실제 종가와의 비교 검토는 Mean Squared Error 를 사용하였다.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_{test} - Y_{pred}\right)^2$$

# IV. 주가 예측을 위한 딥러닝 최적화 평가

## 1. 실험 결과 요약

Epoch 의 경우 현재 데이터가 60 분 단위의 1500 여개 Sample 로 데이터수가 많지 않은 점 때문인지 1000 번 넘게 했을때 정확도 가 높아졌다. 과적합(MSE Loss 가 더 높아 지는 점)을 모니터링 하며 까지 추가로 더 실험을 해볼 필요가 있다. Batch / hidden node 는 데이터 Sample 의 패턴과 길이에 영향을 받을텐데 좀더 다양한 종목의 데이터 sample 로 추가 실험을 한다면, 데이터 타입과 상관관계를 규명해볼 수 있을 것 같다. Dropout 은 모든 경우에서 0.2 였을 때 가장 좋은 성능을 보여줬다. RNN – LSTM 으로 주가 예측을 할때 하이퍼 파라미터 의 튜닝 여부에 따라 학습 정확도는 9.68%, 예측 정확도는 6.43% 차이를 보였다. RNN-LSTM 을 시계열 예측에 사용할 경우 종목별로 최적 모델의 하이퍼 파라미터의 구성은 모두 다를 수 있고, 특히 패턴이 불규칙적이고 변동이 큰 경우는 실제 학습을 통한 하이퍼 파라미터의 계속적인 튜닝이 필요하다. 거래시간 feature + 종가 feature 를 함께 학습한 경우에는 종가 feature 만 학습한 경우보다 0.5%~4.6% 예측 정확도가 증가 하였다.

각 종목의 거래시간 별로 주가 변화율은 차이가 많았다. 어떤 종목은 장시작 시에 가장 많이 상승하고 점점 상승폭을 줄이는가 하면, 특정 종목은 장 시작과 마감시에는 잠잠 하고 점심 이후에 강세를 보이는 종목 도 있었다.

추가 feature 로서 연도, 월, 계절등도 검토 했으나, 증권사에서 제공하는 데이터가 현재로 부터 과거 1 년 뿐이라서 연도, 월, 계절을 검토 하기에는 sample 수가 부족하였다.

## 2. 거래시간 Feature 학습 검토

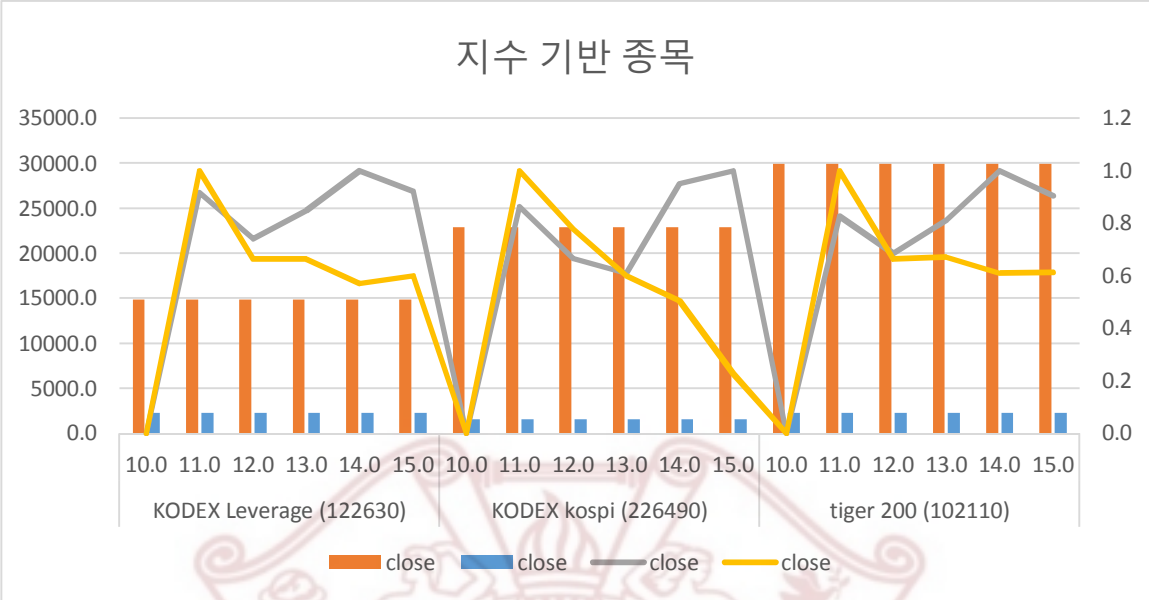
거래시간 Feature 는 주가의 움직임에 시장참여자의 단체 심리를 반영한다는 점에서 영향이 크다고 할 수 있다. 각 종목에서 어떤 특징을 보이는지 살펴 보도록 한다.

### 2.1. KODEX 기반 종목 거래시간 + 종가 feature 학습

| | | KODEX Leverage (122630) | | | | | | KODEX kospi (226490) | | | | | | tiger 200 (102110) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 10.0 | 11.0 | 12.0 | 13.0 | 14.0 | 15.0 | 10.0 | 11.0 | 12.0 | 13.0 | 14.0 | 15.0 | 10.0 | 11.0 | 12.0 | 13.0 | 14.0 | 15.0 |
| | mean | 14882.7 | 14898.6 | 14895.6 | 14897.5 | 14900.1 | 14898.7 | 22940.5 | 22948.0 | 22946.3 | 22945.8 | 22948.8 | 22949.2 | 29944.4 | 29960.6 | 29957.8 | 29960.3 | 29963.9 | 29962.1 |
| | std | 2281.5 | 2303.2 | 2295.9 | 2295.9 | 2293.9 | 2294.5 | 1572.1 | 1585.1 | 1582.2 | 1579.9 | 1578.7 | 1575.1 | 2259.6 | 2281.4 | 2274.1 | 2274.2 | 2272.9 | 2272.9 |
| | norm mea | 0.0 | 0.9 | 0.7 | 0.8 | 1.0 | 0.9 | 0.0 | 0.9 | 0.7 | 0.6 | 1.0 | 1.0 | 0.0 | 0.8 | 0.7 | 0.8 | 1.0 | 0.9 |
| | norm std | 0.0 | 1.0 | 0.7 | 0.7 | 0.6 | 0.6 | 0.0 | 1.0 | 0.8 | 0.6 | 0.5 | 0.2 | 0.0 | 1.0 | 0.7 | 0.7 | 0.6 | 0.6 |
| | min | 11105.0 | 11135.0 | 11085.0 | 11095.0 | 11120.0 | 11115.0 | 20190.0 | 20200.0 | 20140.0 | 20155.0 | 20175.0 | 20175.0 | 26005.0 | 26050.0 | 25990.0 | 26010.0 | 26040.0 | 26025.0 |
| | 25% | 12520.0 | 12505.0 | 12517.5 | 12525.0 | 12518.8 | 12510.0 | 21352.5 | 21350.0 | 21330.0 | 21358.8 | 21343.8 | 21338.8 | 27665.0 | 27640.0 | 27646.3 | 27656.3 | 27660.0 | 27651.3 |
| | 50% | 15645.0 | 15597.5 | 15580.0 | 15587.5 | 15610.0 | 15640.0 | 23535.0 | 23507.5 | 23505.0 | 23500.0 | 23505.0 | 23525.0 | 30625.0 | 30592.5 | 30627.5 | 30590.0 | 30597.5 | 30625.0 |
| | 75% | 16467.5 | 16507.5 | 16505.0 | 16502.5 | 16511.3 | 16523.8 | 24015.0 | 24046.3 | 24016.3 | 24030.0 | 24015.0 | 24023.8 | 31480.0 | 31610.0 | 31573.8 | 31557.5 | 31548.8 | 31498.8 |
| close | max | 18960.0 | 18940.0 | 18925.0 | 18885.0 | 18895.0 | 18915.0 | 25690.0 | 25640.0 | 25610.0 | 25625.0 | 25625.0 | 25640.0 | 33935.0 | 33920.0 | 33900.0 | 33870.0 | 33885.0 | 33895.0 |
| | mean | 22.1 | 1.1 | -3.0 | 1.8 | 2.6 | -1.4 | -4.0 | -2.8 | -1.7 | -0.5 | 3.0 | 0.4 | 20.0 | 2.0 | -2.8 | 2.5 | 3.6 | -1.9 |
| | std | 131.7 | 65.4 | 49.7 | 37.8 | 32.2 | 45.6 | 97.4 | 49.9 | 45.9 | 35.1 | 35.0 | 37.0 | 137.1 | 67.4 | 49.6 | 39.0 | 33.3 | 45.4 |
| | min | -500.0 | -255.0 | -285.0 | -155.0 | -170.0 | -230.0 | -420.0 | -175.0 | -220.0 | -140.0 | -100.0 | -155.0 | -490.0 | -245.0 | -270.0 | -165.0 | -155.0 | -205.0 |
| | 25% | -45.0 | -30.0 | -20.0 | -15.0 | -10.0 | -15.0 | -55.0 | -35.0 | -25.0 | -20.0 | -15.0 | -18.8 | -45.0 | -30.0 | -25.0 | -15.0 | -15.0 | -20.0 |
| | 50% | 12.5 | 2.5 | 0.0 | 0.0 | 5.0 | 0.0 | 0.0 | 0.0 | -5.0 | 0.0 | 5.0 | 0.0 | 10.0 | 5.0 | 0.0 | 2.5 | 5.0 | 0.0 |
| | 75% | 97.5 | 35.0 | 20.0 | 20.0 | 20.0 | 20.0 | 50.0 | 25.0 | 25.0 | 20.0 | 20.0 | 20.0 | 90.0 | 40.0 | 20.0 | 20.0 | 25.0 | 20.0 |
| diff | max | 600.0 | 215.0 | 170.0 | 170.0 | 105.0 | 275.0 | 335.0 | 140.0 | 145.0 | 105.0 | 130.0 | 135.0 | 575.0 | 210.0 | 170.0 | 165.0 | 110.0 | 275.0 |

*<표 5> Kodex 기반 종목 거래 시간별 종가 특성*
*(종가 및 normalized(0~1) 종가)*

<표 5>과 <그림 7> 에서 보이는 바와 같이, Kodex 지수 기반 종목의 각 시간별 주가를 10 시 ~ 15 시로 구분하여 평균 주가를 최소 주가를 0, 최대 주가를 1 로 normalize 해 보면, 아침 첫 시간인 10 시 시간대가 항상 낮게 시작하여, 바로 다음 시간대에 강세를 보이고, 점심시간대에 진정세를 보인후, 조금 오른 다음 다시 한두시간대 낮아 지는 경향을 보인다.




*<그림 5> kodex 지수 기반 종목 거래 시간 특성*
*( X: 종목별 거래 시간, 좌 Y: 종가, 우 Y : normalized 종가 )*

각 종목의 가격은 차이가 좀 있지만, 첫시간대가 낮고 바로 다음 시간대에 올랐다가 그다음 시간대에 조금 떨어지는 경향이 보인다.

| | KODEX Leverage (122630) | | KODEX kospi (226490) | | tiger 200 (102110) | | Average | | |
|---|---|---|---|---|---|---|---|---|---|
| | 거래 시간 + 종가 | 종가 only | 거래 시간 + 종가 | 종가 only | 거래시간 + 종가 | 종가 only | 거래 시간 + 종가 | 종가 only | diff |
| 학습 정확도 | 99.14 | 97.27 | 99.04 | 97.58 | 99.16 | 97.43 | 99.1 | 97.4 | 1.7 |
| 테스트 정확도 | 98.18 | 94.73 | 97.8 | 95.15 | 98.11 | 94.8 | 98.0 | 94.9 | 3.1 |

*<표 6> kodex 기반 종목 모델 예측 정확도 (%)*

Kodex 기반 각 종목을 종가 feature 만 학습했을때와 거래시간 feature + 종가 feature 를 함께 학습 했을때는 <표 6>과 같이 학습 정확도가 1.7% 개선 되었고, 테스트 정확도는 3.1% 가 개선 되었다. 다만, 학습 feature 가 늘어남에 따라 학습 시간이 1.5 배 정도 더 소요 되었다.

*<그림 6> KODEX Leverage 거래시간 + 종가 (X: 시간, Y: 종가)*

*<그림 7> KODEX Leverage 종가 Only (X: 시간, Y: 종가)*

가장 큰 차이를 보이는 Kodex Leveage 종목의 경우 <그림 8> 과 <그림 9>에서 와 같이 예측시 퍼센티지 로 보면 3.5% 이지만, 차트상으로는 학습이 종가의 트렌드를 그려내지 못하는것 처럼 차이를 보였다.

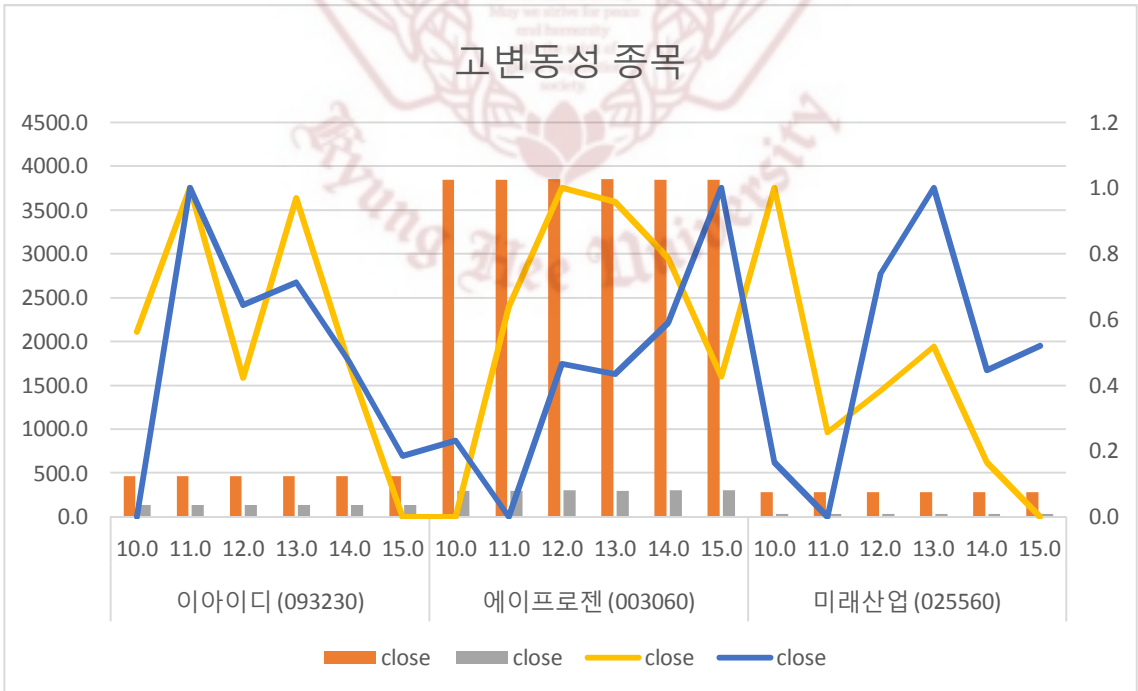
## 2.2. 고 변동성 종목 거래시간 + 종가 feature 학습

| | | 이아이디 (093230) | | | | | | 에이프로젠 (003060) | | | | | | 미래산업 (025560) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 10.0 | 11.0 | 12.0 | 13.0 | 14.0 | 15.0 | 10.0 | 11.0 | 12.0 | 13.0 | 14.0 | 15.0 | 10.0 | 11.0 | 12.0 | 13.0 | 14.0 | 15.0 |
| | mean | 465.2 | 465.7 | 465.1 | 465.6 | 465.1 | 464.7 | 3841.1 | 3846.2 | 3849.0 | 3848.7 | 3847.3 | 3844.5 | 278.1 | 277.6 | 277.7 | 277.8 | 277.5 | 277.4 |
| | std | 131.2 | 132.6 | 132.1 | 132.2 | 131.9 | 131.5 | 296.5 | 294.1 | 299.1 | 298.7 | 300.4 | 304.8 | 34.1 | 34.0 | 34.3 | 34.4 | 34.2 | 34.2 |
| | norm mea | 0.6 | 1.0 | 0.4 | 1.0 | 0.5 | 0.0 | 0.0 | 0.6 | 1.0 | 1.0 | 0.8 | 0.4 | 1.0 | 0.3 | 0.4 | 0.5 | 0.2 | 0.0 |
| | norm std | 0.0 | 1.0 | 0.6 | 0.7 | 0.5 | 0.2 | 0.2 | 0.0 | 0.5 | 0.4 | 0.6 | 1.0 | 0.2 | 0.0 | 0.7 | 1.0 | 0.4 | 0.5 |
| | min | 320.0 | 317.0 | 318.0 | 317.0 | 317.0 | 318.0 | 3235.0 | 3250.0 | 3255.0 | 3245.0 | 3250.0 | 3255.0 | 222.0 | 222.0 | 224.0 | 223.0 | 224.0 | 223.0 |
| | 25% | 364.0 | 365.0 | 364.3 | 365.3 | 365.0 | 365.0 | 3650.0 | 3656.3 | 3657.5 | 3646.3 | 3656.3 | 3656.3 | 250.0 | 249.0 | 249.0 | 249.3 | 249.0 | 249.0 |
| | 50% | 420.0 | 419.5 | 420.0 | 420.5 | 420.5 | 420.0 | 3815.0 | 3820.0 | 3825.0 | 3827.5 | 3822.5 | 3817.5 | 279.0 | 279.0 | 279.0 | 279.0 | 279.0 | 279.0 |
| | 75% | 465.5 | 462.8 | 462.0 | 462.8 | 465.0 | 463.8 | 4002.5 | 4023.8 | 4028.8 | 4023.8 | 4033.8 | 4018.8 | 295.5 | 294.8 | 294.0 | 294.8 | 294.8 | 295.0 |
| close | max | 762.0 | 752.0 | 753.0 | 758.0 | 758.0 | 752.0 | 5200.0 | 5300.0 | 5400.0 | 5360.0 | 5260.0 | 5370.0 | 419.0 | 423.0 | 431.0 | 424.0 | 425.0 | 419.0 |
| | mean | 0.0 | 0.2 | -0.5 | 0.5 | -0.5 | -0.4 | 5.3 | -0.3 | 2.8 | -0.3 | -1.3 | -2.8 | 0.5 | -0.3 | 0.1 | 0.1 | -0.3 | -0.1 |
| | std | 9.7 | 5.7 | 4.1 | 4.4 | 5.7 | 4.6 | 79.8 | 53.9 | 34.8 | 26.5 | 41.2 | 43.6 | 6.6 | 3.6 | 2.6 | 2.0 | 1.9 | 2.7 |
| | min | -91.0 | -22.0 | -25.0 | -14.0 | -72.0 | -23.0 | -270.0 | -350.0 | -130.0 | -115.0 | -105.0 | -190.0 | -25.0 | -18.0 | -12.0 | -9.0 | -15.0 | -29.0 |
| | 25% | -3.0 | -2.0 | -2.0 | -2.0 | -2.0 | -2.0 | -40.0 | -25.0 | -15.0 | -15.0 | -15.0 | -15.0 | -2.0 | -1.0 | -1.0 | -1.0 | -1.0 | -1.0 |
| | 50% | 0.0 | 0.0 | -1.0 | 0.0 | 0.0 | 0.0 | 0.0 | -5.0 | 0.0 | 0.0 | -5.0 | -5.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | 75% | 3.0 | 2.0 | 1.0 | 1.8 | 1.0 | 1.0 | 43.8 | 15.0 | 15.0 | 10.0 | 10.0 | 10.0 | 2.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| diff | max | 35.0 | 47.0 | 17.0 | 36.0 | 17.0 | 47.0 | 665.0 | 420.0 | 195.0 | 155.0 | 475.0 | 450.0 | 45.0 | 39.0 | 24.0 | 13.0 | 8.0 | 16.0 |

*<표 7> 고 변동성 종목 거래 시간 특성*

*(종가 및 normalized(0~1) 종가)*

변동성이 많은 종목은 각 종목별로 각기 다른 시간대 특성을 가지고 있었다. <표 7>와 <그림 10> 에서 보이는것 처럼 에이아이디는 점심시간 양쪽으로 장 중반에 고가이고 장마감시 저가를 보인다. 에이프로젠은 장초반에 저가로 시작해서 중반에 고가를 보인다. 미래산업은 고가로 시작하여 저가로 마감하는 특성을 보였다.



*<그림 8> 고변동성 종목 거래 시간 특성*
*( X: 종목별 거래 시간, 좌 Y: 종가, 우 Y : normalized 종가 )*

| | 이아이디 | | 에이프로 | | 미래 | | Average | | |
|---|---|---|---|---|---|---|---|---|---|
| | 거래시간 + 종가 | 종가 only | 거래시간 + 종가 | 종가 only | 거래시간 + 종가 | 종가 only | 거래시간 + 종가 | 종가 only | diff |
| 학습 정확도 | 99.1 | 98.66 | 99.33 | 99.26 | 98.57 | 96.74 | 99.0 | 98.2 | 0.8 |
| 테스트 정확도 | 98.75 | 98.6 | 98.55 | 98.45 | 98.94 | 94.33 | 98.7 | 97.1 | 1.6 |

*<표 8> 고변동성 종목 모델 예측 정확도 (%)*

고 변동성 각 종목을 종가 feature 만 학습했을때와 거래시간 feature + 종가 feature 를 함께 학습 했을때는 <표 8>과 같이 학습 정확도는 0.8% 개선 되었고, 테스트 정확도 는 1.6% 가 개선 되었다. 역시 해당 실험도 거래시간 + 종가 feature 학습 시간이 비교군에 비해 1.5 배 정도 더 소요 되었다.

*<그림 9> 미래산업 거래시간 + 종가*
*(X: 시간, Y: 종가)*

*<그림 10> 미래산업 종가 Only*
*(X: 시간, Y: 종가)*

가장 큰 차이를 보이는 미래산업 종목의 경우 <그림 11> 과 <그림 12>와 같이 예측시 평균으로는 퍼센티지 로 보면 4.6% 이지만, 실체 차트 상으로는 피크와 최근 트렌드를 잘 그려주지 못하는 정도로 차이가 많았다.

## 3. 하이퍼 파라미터 탐색

하이퍼 파라미터 탐색 는 탐색 대상 외에 다른 파라미터를 고정하고, 해당 파라미터만 변화하여 Mean squared error 값이 최소한이 되는 지점에서 최적 모델을 선정 한다. 교차 검증 이후, 최적 모델을 통해 테스트 예측을 수행하여 차트를 그려보았다.

### 3.1. Epoch 탐색

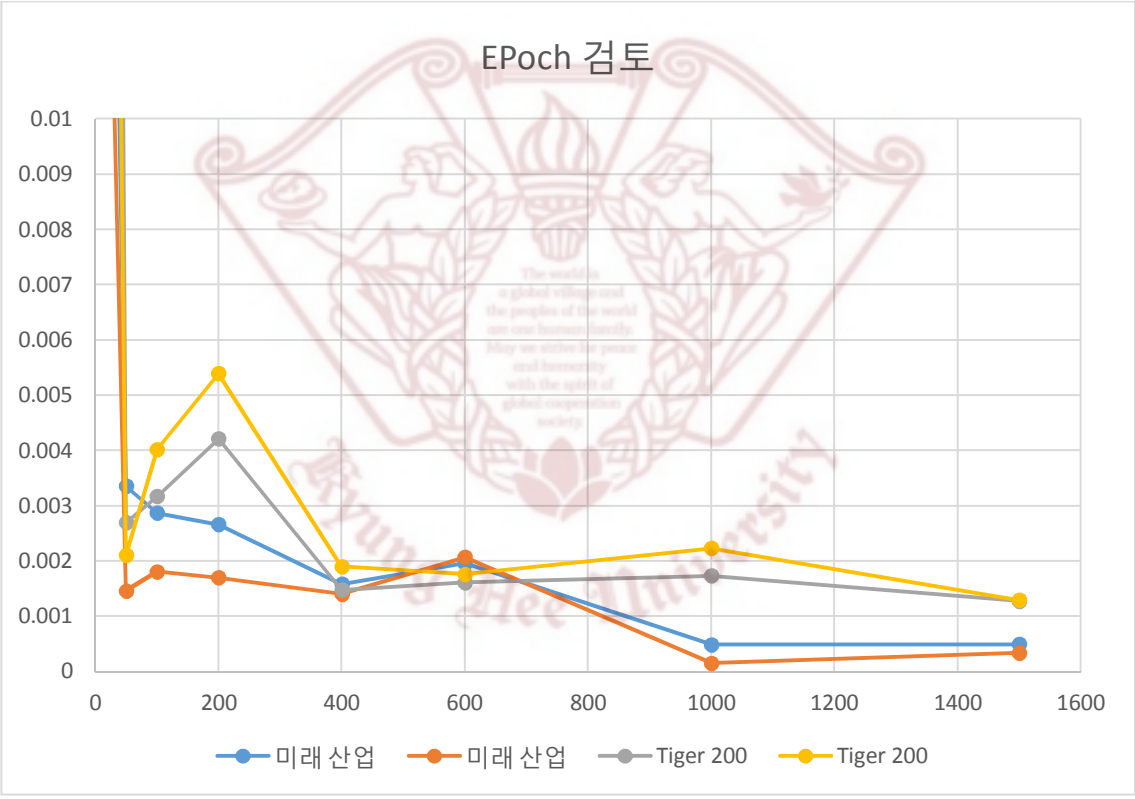
Epoch 은 전체 데이터를 한번 처리하는 단위로 너무 많으면 과적합이 발생할 수 있고 너무 적으면 학습이 완성되지 않아 예측 정확도가 떨어지는 문제가 발생 할 수 있다.

Epoch 이외에 다른 하이퍼 파라미터 는 아래와 같이 고정 하였다.

- Hidden nodes: 80,
- Batch size: 40,
- dropout: 0.2

| Epochs | MSE | SDEV | MSE | SDEV |
|---|---|---|---|---|
| 5 | 0.031417 | 0.017821 | 0.103677 | 0.060706 |
| 10 | 0.027981 | 0.019873 | 0.07162 | 0.042513 |
| 50 | 0.003356 | 0.001464 | 0.002699 | 0.002109 |
| 100 | 0.00287 | 0.001811 | 0.003174 | 0.004023 |
| 200 | 0.00266 | 0.0017 | 0.004216 | 0.005393 |
| 400 | 0.001587 | 0.001403 | 0.001483 | 0.001906 |
| 600 | 0.001959 | 0.002069 | 0.001613 | 0.001764 |
| 1000 | 0.000488 | 0.000157 | 0.001733 | 0.002235 |
| 1500 | 0.000499 | 0.00034 | 0.001277 | 0.001297 |

*<표 9> 하이퍼 파라미터 Epoch 검토*




*<그림 11> Epoch 탐색*

<표 9> 과 <그림 13> 에서 나타나는 것 같이, 지수기반의 대표 종목 Tiger 200 의 경우, Epoch 이 1500 일때 MSE 가 가장 낮았고, 고변동성 대표

종목인 미래 산업의 경우, Epoch 이 1000 일때 MSE 가 가장 낮았다. 데이터가 60 분 간격이라서 증권사에서 제공 할 수 있는 데이터 sample 수가 1660 으로 많지 않아서 1500 까지는 MSE 가 증가 하지 않고 낮아 지는 형태를 보였다.

### 3.2. Batch 탐색

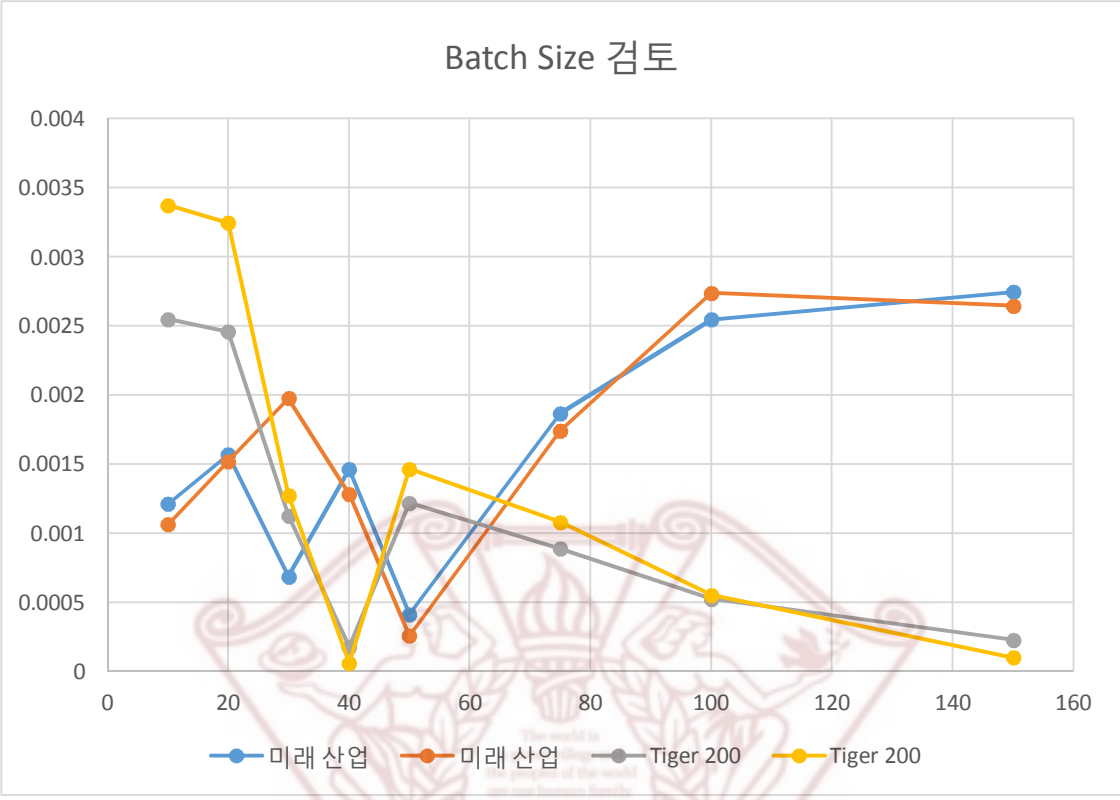
Batch size 는 한번에 처리 하는 노드 수 를 말하며, 주가의 패턴에 따라 큰 배치 사이즈와 작은 배치 사이즈의 계산 범위가 달라서 서로 다른 결과를 보여 줄 수 있다.

Batch size 이외에 다른 하이퍼 파라미터 는 아래와 같이 고정 하였다.

- Hidden nodes: 80,
- Epoch: 400,
- dropout: 0.2

| | 미래 산업 | | Tiger 200 | |
|---|---|---|---|---|
| Batch size | MSE | SDEV | MSE | SDEV |
| 10 | 0.001214 | 0.001064 | 0.00255 | 0.003375 |
| 20 | 0.001571 | 0.001522 | 0.00246 | 0.003247 |
| 30 | 0.000688 | 0.001978 | 0.001128 | 0.001275 |
| 40 | 0.001464 | 0.001286 | 0.000177 | 0.000062 |
| 50 | 0.000414 | 0.000258 | 0.00122 | 0.001465 |
| 75 | 0.001869 | 0.001743 | 0.00089 | 0.001082 |
| 100 | 0.002547 | 0.002739 | 0.000525 | 0.000554 |
| 150 | 0.002746 | 0.002648 | 0.000228 | 0.000102 |

*<표 10> 하이퍼 파라미터 Batch size 검토*




*<그림 12> Batch Size 탐색*

<표 10> 과 <그림 14> 에서 보여지는 것과 같이 지수기반의 대표 종목 Tiger 200 의 경우, BatchSize 가 40 일때 MSE 가 가장 낮았고, 고변동성 대표 종목인 미래 산업의 경우, BatchSize 가 50 일때 MSE 가 가장 낮았다. 가장 낮은 지점은 40, 50 으로 비슷했지만, 이후 미래 산업의 경우 Batchsize 가 커질 수록 MSE 가 커지는 형태를 보였고, Tiger 200 의 경우 50 이후 Batchsize 가 커질 수록 MSE 가 작아지는 형태를 보였다.

### 3.3. Hidden Node 탐색

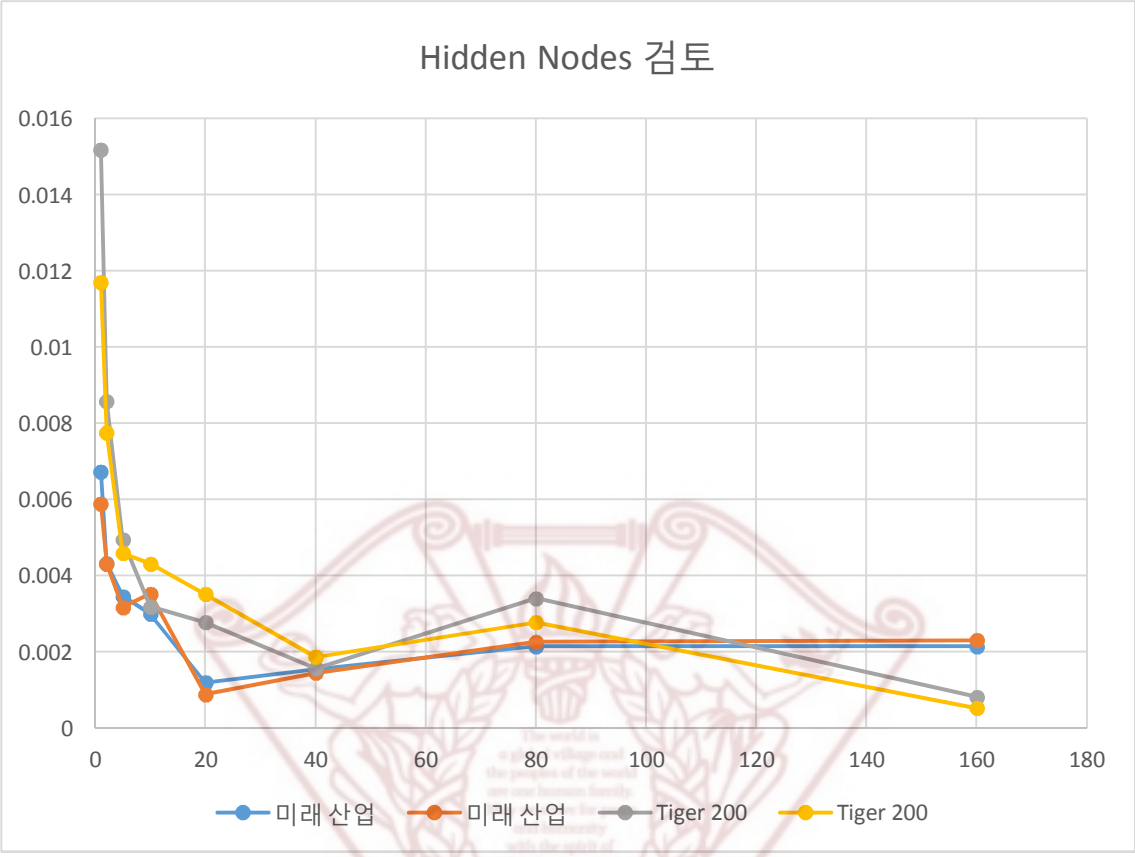
Hidden node 는 수에 따른 신경망의 복잡도와 메모리 크기 결정하는 파라미터 이며 복잡해질 수록 계산량이 많아지지만, 복잡한 패턴의 계산이 가능하다고함

Hidden node 이외에 다른 하이퍼 파라미터는 아래와 같이 고정 하였다.

- Batch Size: 40,
- Epoch: 400,
- dropout: 0.2

| | 미래 산업 | | Tiger 200 | |
|---|---|---|---|---|
| Hidden Nodes | MSE | SDEV | MSE | SDEV |
| 1 | 0.006738 | 0.005895 | 0.01518 | 0.011705 |
| 2 | 0.004328 | 0.004316 | 0.008588 | 0.007758 |
| 5 | 0.003461 | 0.003178 | 0.004948 | 0.004592 |
| 10 | 0.002999 | 0.003525 | 0.003196 | 0.004314 |
| 20 | 0.001204 | 0.000889 | 0.002783 | 0.003519 |
| 40 | 0.001557 | 0.001452 | 0.00157 | 0.001869 |
| 80 | 0.002149 | 0.002271 | 0.003408 | 0.002784 |
| 160 | 0.002152 | 0.002306 | 0.000823 | 0.000526 |

*<표 11> 하이퍼 파라미터 Hidden node 검토*




*<그림 13> Hidden nodes 탐색*

<표 11> 와 <그림 15>에서와 같이, 지수기반의 대표 종목 Tiger 200 의 경우, hidden node 가 160 일때 MSE 가 가장 낮았고, 고변동성 대표 종목인 미래 산업의 경우, hidden node 가 20 일때 MSE 가 가장 낮았다.

### 3.4. Dropout 탐색

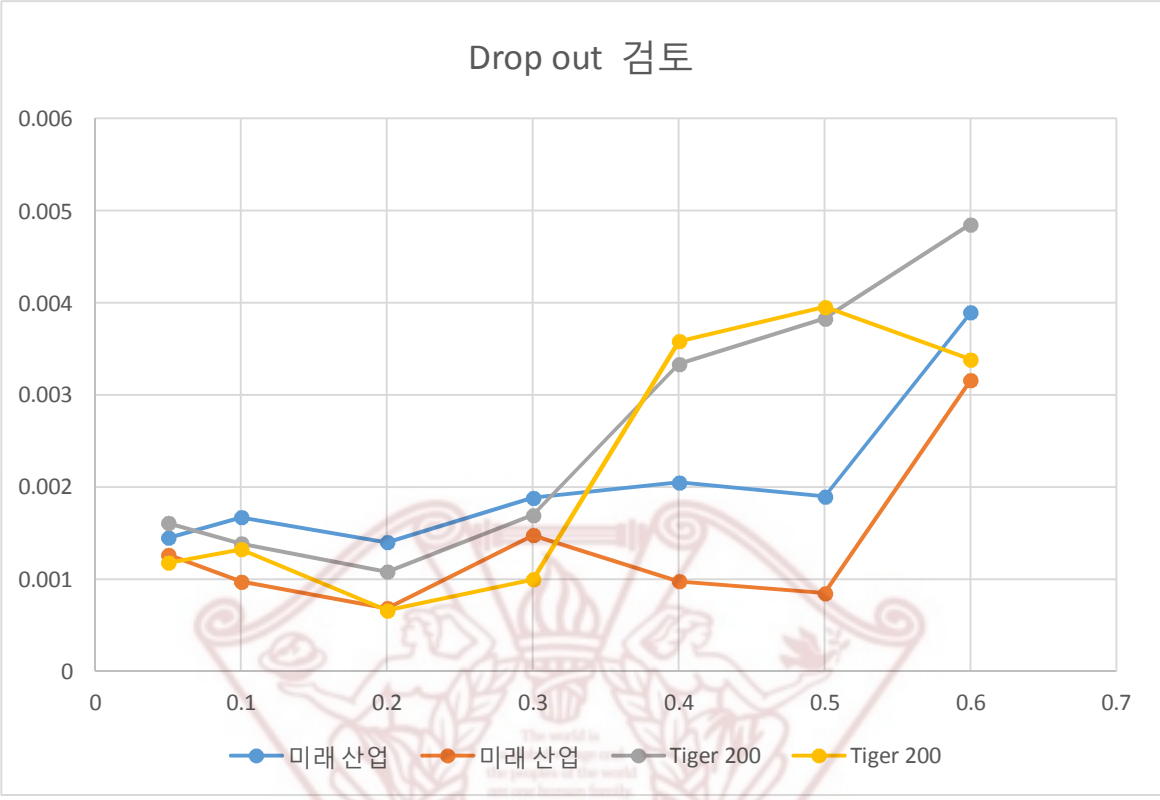
Hidden Layer 사이에 과적합을 방지하기위해서 state Drop rate 를 사용하는데 이는 layer 사이에 임의의 node 의 학습을 건너뛰는 방법으로 과적합을 피하도록 한다.

Drop out rate 이외에 다른 하이퍼 파라미터 는 아래와 같이 고정 하였다.

- Batch Size: 40,
- Epoch: 400,
- Hidden nodes: 80

| | 미래 산업 | | Tiger 200 | |
|---|---|---|---|---|
| Drop Out | MSE | SDEV | MSE | SDEV |
| 0.05 | 0.001452 | 0.001268 | 0.001612 | 0.00118 |
| 0.1 | 0.001675 | 0.000976 | 0.00139 | 0.001328 |
| 0.2 | 0.001402 | 0.000686 | 0.001083 | 0.000661 |
| 0.3 | 0.001885 | 0.001478 | 0.001698 | 0.001001 |
| 0.4 | 0.002052 | 0.000979 | 0.003339 | 0.003585 |
| 0.5 | 0.001899 | 0.000851 | 0.003832 | 0.003957 |
| 0.6 | 0.003899 | 0.00316 | 0.004852 | 0.003388 |

*<표 12> 하이퍼 파라미터 Dropout 검토*




*<그림 14> Hidden nodes 탐색*

<표 12> 과 <그림 16>에서 보여지는 바와 같이 Tiger 200 과 미래 산업의 경우 모두 dropout rate 가 0.2 일때 MSE 가 가장 낮았다. 과적합이 발생하지 않는 상황에서의 0.2 이상의 dropout 의 경우 학습 자체가 덜 되는 것 같이 MSE 가 높아져 갔다.

### 3.5. 최적 모델과 최악 모델

위와 같은 하이퍼 파라미터 탐색의 결과로 최적의 모델과 최악의 모델을 선정하였다. 최적 모델은 예측치와 시험치의 MSE 가 가장적은 모델이고, 최악 모델은 반대로 MSE 가 가장 큰 모델이다.

최악 모델의 경우, 특정 종목 학습시의 하이퍼 파라미터의 적정성에 대한 고려 없이 아무 하이퍼 파라미터를 사용 한다면, 최악의 상황에 나올 수 있는 모델이라는 점에서 최적 모델과 비교 할 만 하다.

하이퍼 파라미터 탐색을 진행한 결과 Tiger 200 ETF 종목과 미래산업 종목의 최적 모델을 아래와 같이 결정 되었다.

| | | Best | Worst |
|---|---|---|---|
| 미래 산업 | epochs | 1000 | 5 |
| | batches | 50 | 10 |
| | hidden layers | 20 | 1 |
| | dropout | 0.2 | 0.6 |
| Tiger 200 ETF | epochs | 1500 | 5 |
| | batches | 40 | 150 |
| | hidden layers | 160 | 1 |
| | dropout | 0.2 | 0.6 |
| Mean 학습 정확도 (%) | | 99.52 | 89.84 |
| Mean 테스트 정확도 (%) | | 99.39 | 92.96 |

*<표 13> 하이퍼 파라미터 s Bestfit vs 최악 모델*

최적 모델 일경우 최악 모델 일경우 보다 학습 정확도 는 평균 9.68 %, 테스트 정확도 는 평균 6.43% 개선 되었다.
이는 수치에서도 의미 있지만, 다음 차트에서 보여지는 것과 같이 시계열 예측을 위한 모델의 학습면에서는 추세를 만들어 갈수 있는가 없는가 의 정도로 차이를 보였다.

*<그림 15> Tiger 200 ETF 최적 모델 vs 최악 모델*

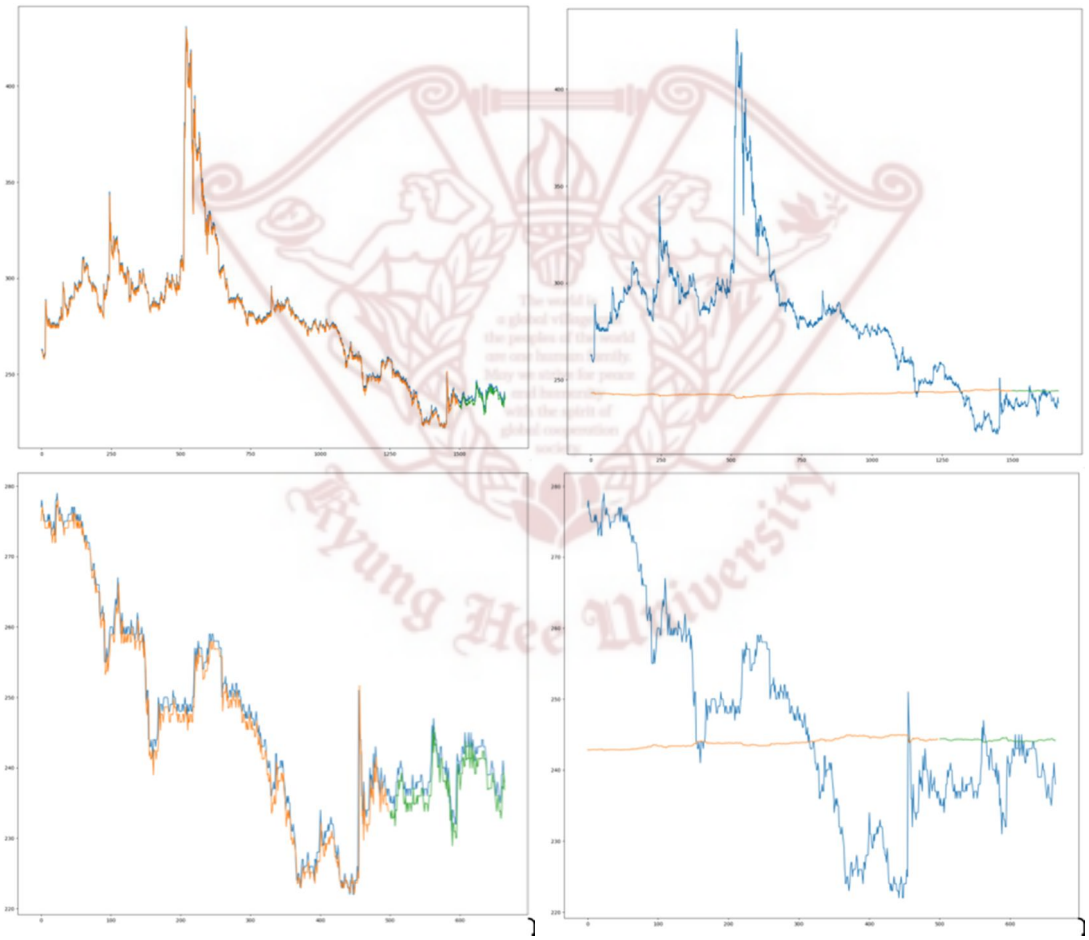
*[상:전체 시간, 하: 최근 600 샘플, 좌: 최적모델, 우: 최악모델]*

*(X: 시간, Y: 종가)*

<그림 17>은 Tiger 200 ETF 의 최적 모델 와 최악 모델 의 차이를 보여준다. 상측은 1 년간의 전체 학습 + 테스트 기간 모두를 보이며, 하측은 최근 600 sample(60 분 단위 600 개) 즉 100 일 정도를 보여준다. 파란선이 종가(종가 )

의 Original trend 이며, 주황색이 학습 종가, 녹색이 테스트 Prediction 종가 이다.

좌측 차트는 최적 모델 이고, 우측 차트는 최악 모델 인데 눈으로 보기에도 확연하게 좌측은 학습이 정상적으로 끝나서 모델이 잘 만들어 졌고, 우측은 학습이 완전치 않아서 직선형태를 보이고 있다.



*<그림 16> 미래산업 최적 모델 vs 최악 모델*

*[상:전체 시간, 하: 최근 600 샘플, 좌: 최적모델, 우: 최악모델]*

*(X: 시간, Y: 종가)*

미래산업의 경우에도 최적 모델은 학습이 정상적으로 끝나서 모델이 잘 만들어 졌고, 최악 모델은 학습이 완전치 않아서 Trend 를 만들지 못하고 직선형태를 보이고 있다.

# IV. 결론

주가를 예측함에 있어서 종가 이외에 거래시간 feature 를 추가하고, 하이퍼 파라미터를 튜닝 하는것이 그렇지 않은 상황에 비해 크게는 9.68% 까지 예측 정확도가 증가 하였다. 이는 단순히 예측 대상 데이터 만을 가지고 RNN-LSTM 으로 학습 및 예측 하는것 보다, 종목별로 거래 시간 특성을 고려한 feature 를 추가 했을때 학습 및 예측이 보다 정확하게 이루어지는 것을 알 수 있었다. 이를 통해 좀더 다양한 feature 예를 들면, Season(계절에 민감한 종목의 경우), 이슈 지수(테마주의 경우) 등을 발굴 하여 주가 예측을 진행 할 수 있다면, 좀더 다양한 데이터를 기반으로 하는 예측의 정확도를 높힐 수 있을것으로 기대된다.

실험의 결과와 같이 각 종목 별로 RNN – LSTM 으로 주가 예측을 할때 하이퍼 파라미터의 튜닝 여부에 따라 학습 정확도는 9.68%, 예측 정확도는 6.43% 차이를 보였다. 이로써 RNN-LSTM 을 시계열 예측에 사용할 경우 높은 정확도를 보장하기위해 하이퍼 파라미터 튜닝이 꼭 필요함을 알 수 있었다. 특히 종목별로 최적의 모델을 학습하는 하이퍼 파라미터의 구성은 모두 다를 수 있고, 특히 패턴이 불규칙 적이고 변동이 큰 경우는 실제 학습을 통한 하이퍼 파라미터의 튜닝이 필요한데, 이는 실제로는 자동화를 한다해도 시간이 많이 소요 되기 때문에 좀더 최적화된 S/W 와 H/W 의 뒷바침이 필요하다. 좀더 많은 샘플의 하이퍼 파라미터 탐색 을 통해 후보 파라미터 범위를 줄이고 시작한다면 탐색 시간을 몇 배로 단축 시킬 수 있다. 다만 매번 예측시 마다 학습과 하이퍼 파라미터의 탐색을 진행하기에는 학습 시간 및 컴퓨팅 리소스 비용이 들 수 밖에 없어서, 한번 생성된 모델에 대한 모델 품질 관리 및 모니터링을 통해 재학습을 결정하는 로직 또한 필요해 보인다.

# Grossary

MSE : Mean Squared Error

STDEV : Standard Deviation

ETF : Exchange Traded Fund

RNN : Recurrent Neural Network

LSTM : Long Shorterm Memory

KERAS : Neural network API

ARIMA : Autoregressive integrated moving average

GRU : Gated Recurrent Unit

교차검증 : Cross Validation

# 참고 문헌


[1] https://github.com/jaungiers/LSTM-Neural-Network-for-Time-Series-Prediction - LSTM Neural Network for Time Series Prediction
[2] http://scikit-learn.org Scikit-Learn
[3] https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/ RNN 과 LSTM 을 이해해보자

[4] 알디슨, http://blog.naver.com/PostView.nhn?blogId=silvury&logNo=220939233742 RNN 을 이용한 stock 종가 예측, 2017

[5] 김태영, https://tykimos.github.io/2017/07/09/Early_Stopping/, 케라스블로그, 학습 조기 종료 시키기, 2017
[6] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J Zhu. Persional Recomendation Using Deep Recurrent Neural Network in NetEase. 2016, International Conference on Data Engineering.
[7] SungManh Ahn, Deep Learning Architectures and Application. JIIS 2016,Vol. 22, No. 2, 127-142.
[8] S. Hochreiter and J. Schmidhuber (1997). Long short-term memory. Neural Computation, Vol. 9, No. 8, 1735-1780.

[9] 송유정, 이종우, 텐서플로우를이용한주가변동예측딥러닝모델설계및개발 , 한국정보과학회, 2017

[10] 서지혜, 텐서플로우를이용한 순환신경망들의알고리즘 성능평가, 이화여자대학교, 2016

[11] 김은정, http://biz.chosun.com/site/data/html_dir/2016/01/20/2016012004099, 조선일보, 2016

[12] https://ko.wikipedia.org/wiki/딥_러닝, wikipedia

[13] 편저:편집부 딥러닝과 빅데이터 산업동향. 하연출판 서울:2016.
[14] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with gradient descent is difficult. 1994, IEEE Transactions on Neural Networks, Vol. 5, No. 2, 157-166.

[15] 이근영, 국내외 로보어드바이저 동향 및 현황 분석, 금융보안원, 2016

[16] 김기성, 시간대별 주식시장의 생체리듬, edaily, 2012
[17] Gary Oldwood, Google's Deep-mind creates revolutionary "Neural Turing Machine" https://www.download3k.com/articles/Google-s-Deep-Mind-creates-revolutionary-Neural-Turing-Machine-00878, 2014
[18] kakack, http://kakack.github.io, 2016