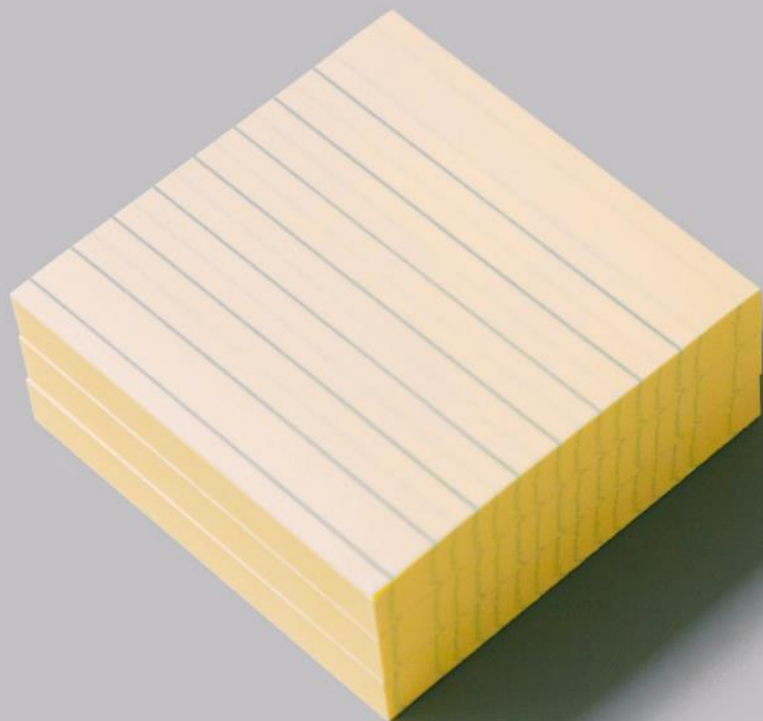


오픈소스 SW 프로젝트

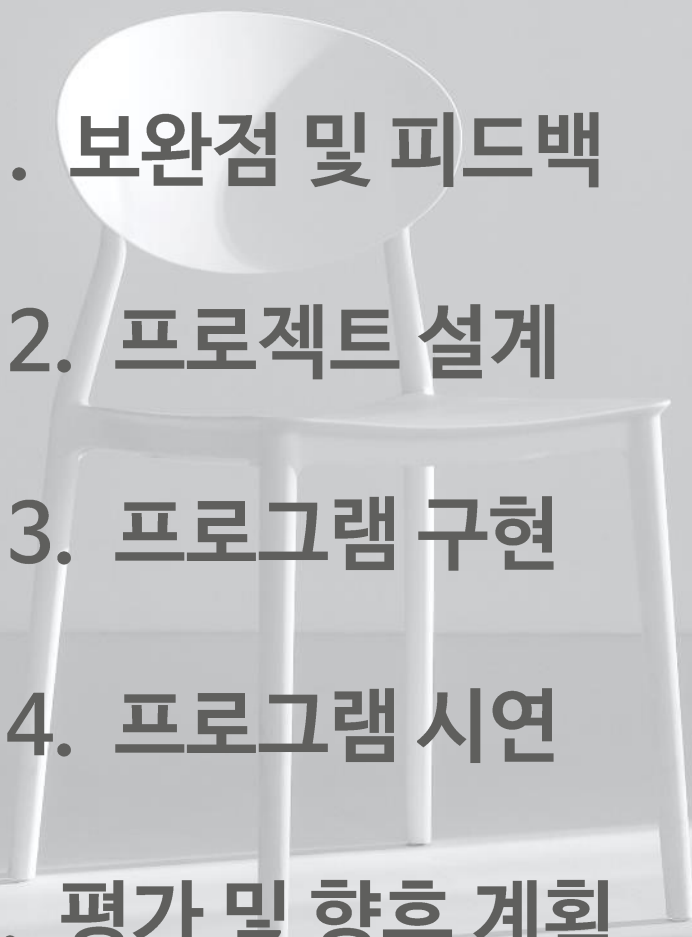
| AI를 이용한 Trading |



3조

정연비, 김덕준, 이시현, 임수빈

Contents

- 
1. 보완점 및 피드백
 2. 프로젝트 설계
 3. 프로그램 구현
 4. 프로그램 시연
 5. 평가 및 향후 계획

Part 1

보완점 및 피드백



보완점 반영

- 모델의 학습 데이터 추가
- 주식에 영향을 주는 지표값 추가
- 데이터들의 가중치 비중을 개별 설정
- 코드간 상호작용 설명 보완



피드백 반영

- 데이터들의 가중치 비중 설정(① PER, ② PBR, ③ ROE, ④ EPS)
- 22년까지 실제 값과 예측 값 비교 (오차 측정)
- 상용 주가예측 서비스와의 주가예측 성능비교



Part 2

프로젝트 설계



개발자 지원도구

개발 언어



주요 라이브러리 및 모듈



- 수치 계산과 머신러닝을 도와주는 오픈소스 라이브러리



- 딥러닝 모델을 간편하게 만들고 훈련시킬 수 있는 딥러닝 프레임워크



- 벡터, 행렬 등 수치 연산을 수행하는 선형대수 라이브러리



- 데이터 처리와 분석을 위한 라이브러리



- 데이터 추출하고 파싱할 때 간편하게 해주는 오픈소스 라이브러리

urllib

- URL을 가져오기 위한 파이썬 모듈

개발 소통 도구



- 프로젝트 공동 작업 및 관리



- 소스코드 호스팅

소프트웨어 개발 방법론 선택

애자일 방법론 : 프로젝트를 시작한 후 끊임없이 개선 노력



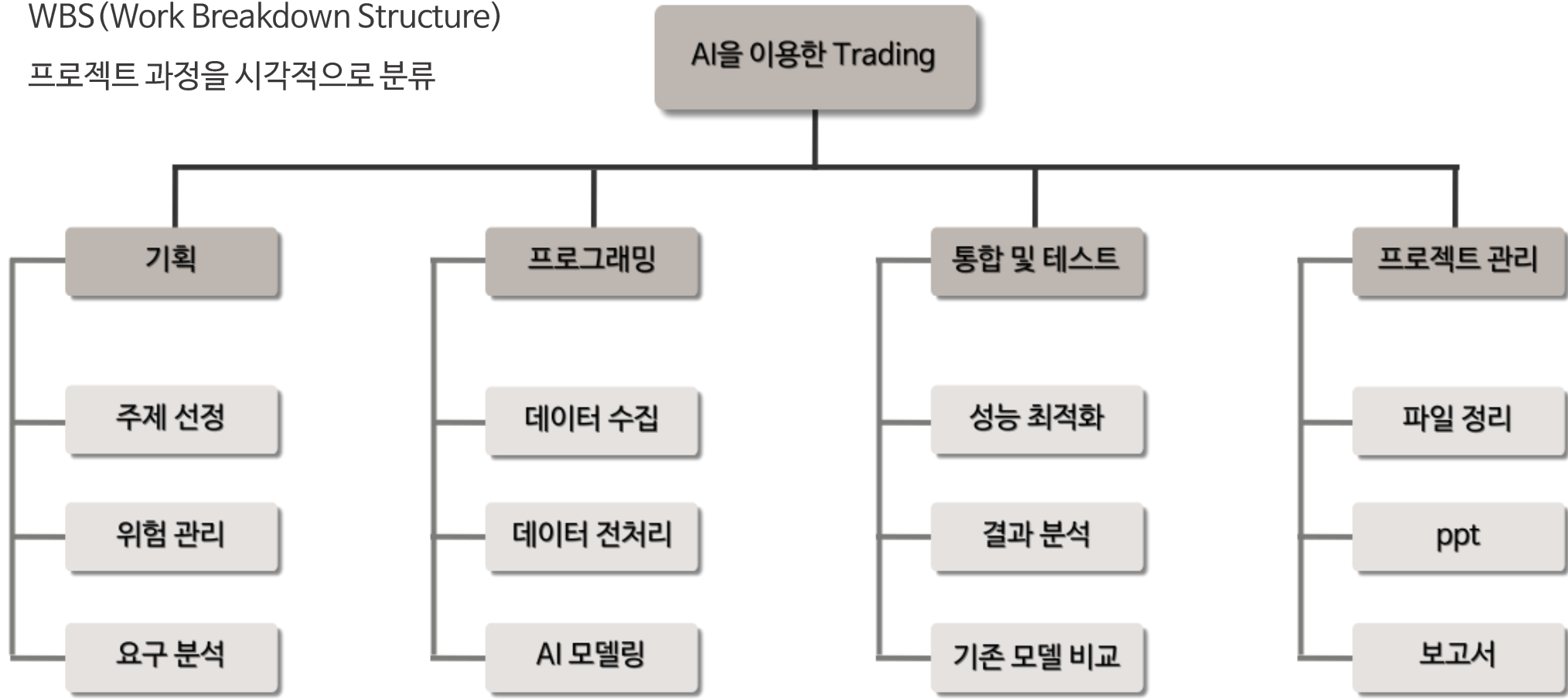
선택이유

- 점진적으로 테스트 할 수 있어 버그를 쉽고 빠르게 발견
- 계획 혹은 기능에 대한 수정과 변경에 유연
- 즉각적인 피드백에 유연

프로젝트 업무 분담

WBS (Work Breakdown Structure)

프로젝트 과정을 시각적으로 분류



간트 차트 (종합 프로젝트 계획)

WBS 바탕으로 프로젝트 전체적으로 계획

WBS 번호	작업 제목	작업 완료 (%)	1주					2주					3주				
			월	화	수	목	금	월	화	수	목	금	월	화	수	목	금
1	프로젝트 주제선정 및 구상수립																
1.1	프로젝트 주제선정	100%															
1.2	프로젝트개발 구상수립	100%															
2	프로젝트 업무분담 및 개발일정																
2.1	범위 및 목표 설정	100%															
2.2	개인별 업무 지정	100%															
2.3	커뮤니케이션 방법 설정	100%															
2.4	위험 관리	100%															
2.5	프로젝트 개발일정 조율	100%															
3	프로젝트 구상 및 착수																
3.1	지표데이터 크롤링	100%															
3.2	환율데이터 크롤링	100%															
3.3	kospi 및 경제심리지수 크롤링	100%															
3.3	주식가격데이터 크롤링	100%															
3.4	데이터 정규화	100%															
3.5	딥러닝 모델 인풋데이터 제작	100%															
3.6	인풋 label 데이터 제작	100%															
3.7	모델 신경망 제작 및 학습	100%															
3.8	모델 예측 및 성능 최적화	100%															
4	PPT 발표자료 준비																
4.1	ppt 발표흐름 구상	100%															
4.2	자료 및 정보 취합	100%															
4.3	ppt 자료 배치	100%															
4.4	ppt 디자인	100%															
5	보고서 작성																
5.1	보고서 작성흐름 구상	100%															
5.2	보고서 자료 및 정보 취합	100%															
5.3	보고서 작성	100%															
6	프로젝트 모니터링 및 시연																
6.1	프로젝트 시연	100%															
6.2	결과물 품질 검증	100%															
6.4	프로젝트 최종 성과 및 성능 정리	100%															

Part 3

프로젝트 구현



주가 예측 참고 지표

주식투자는 크게 **단기투자**와 **장기투자**로 나눌 수 있다.

소요가 큰 실시간 이슈데이터 대신 **장기투자지표**를 채택하여 프로젝트에 적용시켰다.

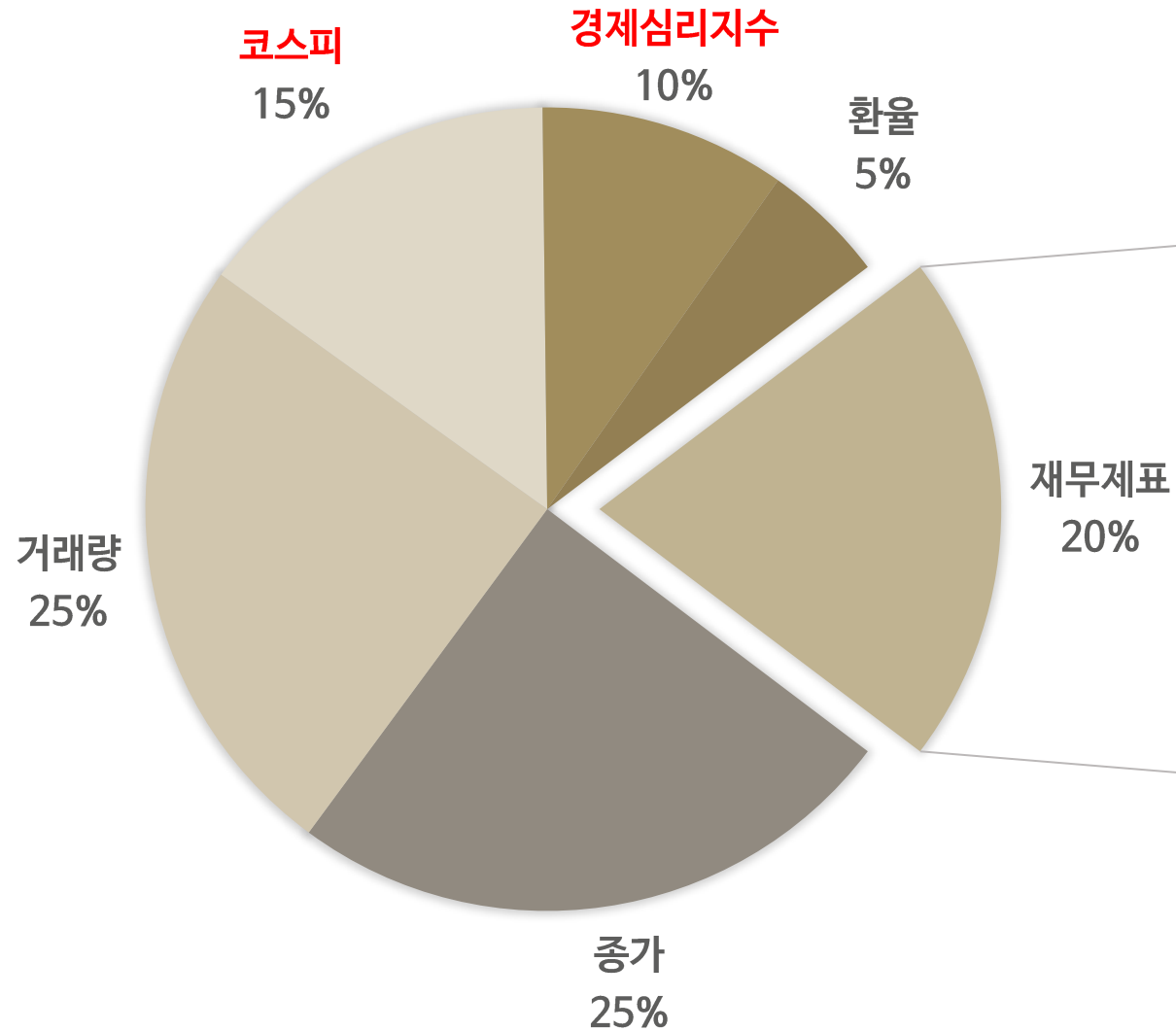
단기 투자지표

- 기업관련 기사 검색
- 트렌드
- 뉴스 기사 (자연어 처리)
- 공매도 비율 외국인 투자경향
- 기관투자 경향
- 코스피 지표

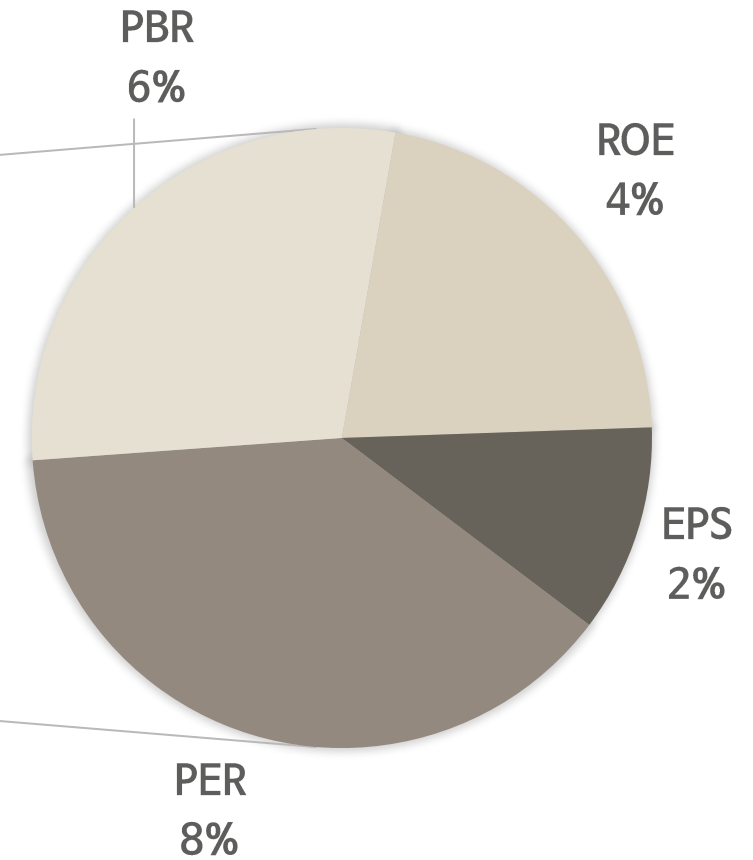
장기 투자지표

- 국가별 경제종합지표
- 국제경제지표
- GDP
- 동일종목 종합지수
- 외국인 투자경향
- 기관투자 경향
- 코스피 지표
- 소비자 심리지수

데이터 추가 및 차등 선정



코스피 지수, 경제심리지수 추가
데이터 우선순위 차등 설정



주가에 영향을 미치는 재무제표 수치

PER | 주가수익비율
Price Earning Ratio

1주당 수익이 몇 배가 되는지 나타내는 지표
※ PER이 낮다면, 회사 이익에 비해 주가가 낮다는 의미로 저평가 되었다는 뜻

PER

PBR | 주가수익비율
Price to Book-value Ratio

기업의 순자산에 대해 1주당 몇 배 거래하는지 측정하는 지표
※ PBR이 1보다 낮은 경우, 시장가치가 장부가치에도 못 미치는 상황으로
시장에서 주가가 저평가 되었다는 뜻

PBR

ROE | 자기 자본 이익률
Return On Equity

투입(보유)자본 대비 얼마나 이익을 냈는지 나타내는 지표
※ 대표적 수익성지표로, ROE가 낮다면 경영진이 무능하거나 업종이 불황일 수 있음

ROE

EPS | 주당 순 이익
Earning Per Share

1주당 이익을 얼마나 창출했는지 나타내는 지표
※ EPS가 높다면 배당어력이 크다는 의미로 주가에 긍정적 영향

EPS

재무제표 데이터 크롤링

재무제표 데이터

ROE, EPS, PER, PBR 데이터 추출

ROA	14.96	13.83	6.28	7.23	9.92	8.32	5.01	7.13
ROE	21.01	19.63	8.69	9.99	13.92	11.62	6.87	9.78
EPS(원)	5421	6024	3166	3841	5777	5352	3406	5130
BPS(원)	28971	35342	37528	39406	43611	48504	50577	54368
DPS(원)	850	1416	1416	2994	1444	1521	1533	1526
PER	9.40	6.42	17.63	21.09	13.55	10.35	16.27	10.80
PBR	1.76	1.10	1.49	2.06	1.80	1.14	1.10	1.02

출처: compinguide.com

주가, 거래량 데이터 크롤링

기업별로 2018년 2월부터 현재까지 휴장일을 제외한 주식가격 데이터

날짜, 종가, 전일비, 시가, 고가, 저가, 거래량 순

날짜	종가	전일비	시가	고가	저가	거래량
2023.01.12	60,500	0	61,100	61,200	59,900	16,034,555
2023.01.11	60,500	▲ 100	61,000	61,200	60,300	12,310,751
2023.01.10	60,400	▼ 300	60,200	61,100	59,900	14,859,797
2023.01.09	60,700	▲ 1,700	59,700	60,700	59,600	18,640,107
2023.01.06	59,000	▲ 800	58,300	59,400	57,900	17,334,989
2023.01.05	58,200	▲ 400	58,200	58,800	57,600	15,682,826
2023.01.04	57,800	▲ 2,400	55,700	58,000	55,600	20,188,071
2023.01.03	55,400	▼ 100	55,400	56,000	54,500	13,547,030
2023.01.02	55,500	▲ 200	55,500	56,100	55,200	10,031,448
2022.12.29	55,300	▼ 1,300	56,000	56,200	55,300	11,295,935

출처 : https://finance.naver.com/item/sise_day.nhn?code=%s



```
2023.01.03,55500,0,55400,55700,54500,9178500
2023.01.02,55500,200,55500,56100,55200,10031448
2022.12.29,55300,-1300,56000,56200,55300,11295935
2022.12.28,56600,-1500,57600,57600,56400,14665410
2022.12.27,58100,200,58000,58400,57900,10667027
2022.12.26,57900,-200,58000,58100,57700,6756411
2022.12.23,58100,-1000,58200,58400,57700,9829407
2022.12.22,59100,1100,58100,59100,58100,10720630
2022.12.21,58000,-600,58700,59100,58000,10356971
```


환율 데이터 크롤링

환율데이터 날짜, 종가 시가, 고가, 저가, 거래량, 변동률

날짜	종가	오픈	고가	저가	거래량	변동 %
2023- 01- 01	1,269.97	1,261.91	1,280.82	1,261.90		0.72%
2022- 12- 01	1,260.92	1,301.72	1,326.55	1,253.26		-3.13%
2022- 11- 01	1,301.65	1,427.03	1,429.63	1,300.60		-8.71%
2022- 10- 01	1,425.83	1,440.23	1,446.62	1,397.24		-0.98%
2022- 09- 01	1,439.96	1,341.76	1,445.64	1,341.10		7.36%

출처 : www.investing.com

KOSPI 및 경제심리지수

KOSIS 국가통계포털 데이터 코스피 & 경제 심리지수 월별로 추출

지수별	2018.05	2018.06	2018.07	...	2022.05	2022.06	2022.07	2022.08	2022.09 p)	2022.10 p)	2022.11 p)
선행종합지수(2015=100)	112.1	112.3	112.5	...	128.8	129.2	129.4	129.6	130	130.3	130.4
선행종합지수 전월비(%)	0.3	0.2	0.2	...	0.4	0.3	0.2	0.2	0.3	0.2	0.1
재고순환지표(전월차)(%p)	2.6	2	0.9	...	0.5	-2.2	-1.1	-0.1	4.1	4	0.3
경제심리지수(전월차)(p)	0.2	0.8	-1.4	...	0.3	-0.3	-2.6	-2.5	-1.5	-0.8	-2.6
기계류내수출하지수(선박제외)(전월비)(%)	-0.6	-1.4	0	...	0	0.4	3.4	3.7	1.7	2.7	0.7
건설수주액(전월비)(%)	2.1	-4.6	7.7	...	1.5	2.2	11.1	-0.2	2	-20.8	-9.7
수출입물가비율(전월비)(%)	-1.3	-0.8	-0.1	...	0	0.6	0	0	-0.2	-0.4	-0.2
코스피(전월비)(%)	0.3	-0.8	-2.4	...	-1.2	-2.8	-4.4	-2	-1.9	-1.8	-0.9

출처 : <https://kosis.kr/>

구현: Import

데이터 가공과 연산에 필요한 라이브러리

```
import pandas as pd
import numpy as np
import tensorflow as tf
```

크롤링에 필요한 라이브러리

```
import urllib
from urllib import parse
from bs4 import BeautifulSoup as bs
```

모델 학습, 그래프 작성에 필요한 라이브러리

```
from tensorflow.keras import models
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt
```

구현한 파일(모듈) 선언

```
from Stock import Stock
from returnInputs import *
from plotGraph import plotGraph
from predict import predictNextDay
```

프로그램 도식화

train.py

returnInput

①

exchangeRate

①

returnPrices

monthlydata

②

returnIdx

③

kospidx

④

consumIdx

회사 개수
X 36

...



train



Save
dmodel.h5

프로그램 도식화

Main.py

codeName

회사이름
예측할 기간



PredictNextDay × 예측 기간만큼 반복

ReturnInput

회사와 예측 기간에 대한
학습데이터



Road

dmodel.h5

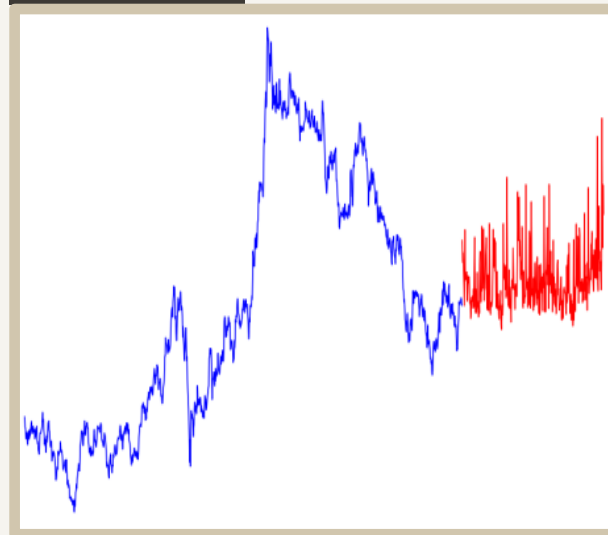
Predict



예측된 종가데이터 배열



PlotGraph



Part 4

프로그램 시연



프로젝트 모델 시연

The screenshot displays an IDE interface with a Python script on the left and a plot on the right. The script, located at `C:\Users\Waddu\Work\SW-main\Worhon\main.py`, imports `numpy`, `Stock`, `predictNextDay`, and `plotGraph`. It prompts the user for a stock name and the number of days to predict. The script then fetches historical stock prices, predicts the next day's price using `predictNextDay`, and appends the prediction to the price list. Finally, it calls `plotGraph` to visualize the data.

```
1 import numpy as np
2 from Stock import Stock
3 from predict import predictNextDay
4 from plotGraph import plotGraph
5 import warnings
6 warnings.filterwarnings("ignore", category=np.VisibleDeprecationWarning)
7
8 coname = input("종목명 : ")
9 coid = Stock.codeName(coname)
10 days = int(input("예측 일 수 : "))
11 CO = Stock(coid)
12
13 prices, date = CO.returnPrices()
14 idx = CO.returnIdx()
15
16 prices = prices[-1127:,:]
17 dates = date[-1127:0].tolist()
18
19 for i in range(days):
20     predictedPrice = predictNextDay(prices, idx)[0,0]
21     prices = np.concatenate((prices, np.array([predictedPrice, prices[-1,1]]).reshape(1,2)))
22     dates.append(f"D+{i+1}")
23
24 plotGraph(prices, dates)
25
26
```

The plot on the right shows a line graph of stock prices over time. The x-axis represents dates, and the y-axis represents prices. The graph shows a blue line for historical prices and a red line for predicted prices. The plot is titled "57 %".

The console output shows the execution of the script, displaying the predicted price for each day and the time taken for each prediction step.

```
1/1 [=====] - 0s 78ms/step
1/1 [=====] - 0s 77ms/step
1/1 [=====] - 0s 79ms/step
1/1 [=====] - 0s 78ms/step
1/1 [=====] - 0s 76ms/step
1/1 [=====] - 0s 77ms/step
1/1 [=====] - 0s 78ms/step
1/1 [=====] - 0s 76ms/step
1/1 [=====] - 0s 78ms/step
1/1 [=====] - 0s 79ms/step
1/1 [=====] - 0s 78ms/step
1/1 [=====] - 0s 77ms/step
1/1 [=====] - 0s 79ms/step
1/1 [=====] - 0s 81ms/step
1/1 [=====] - 0s 78ms/step
1/1 [=====] - 0s 76ms/step

In [101]:
```

The status bar at the bottom indicates the environment is `conda (Python 3.9.13)`, the language server is `LSP: Python`, and the file encoding is `UTF-8`.

Part 5

평가 및 향후 계획



2022년 실제 값과 성능 비교

다음주식

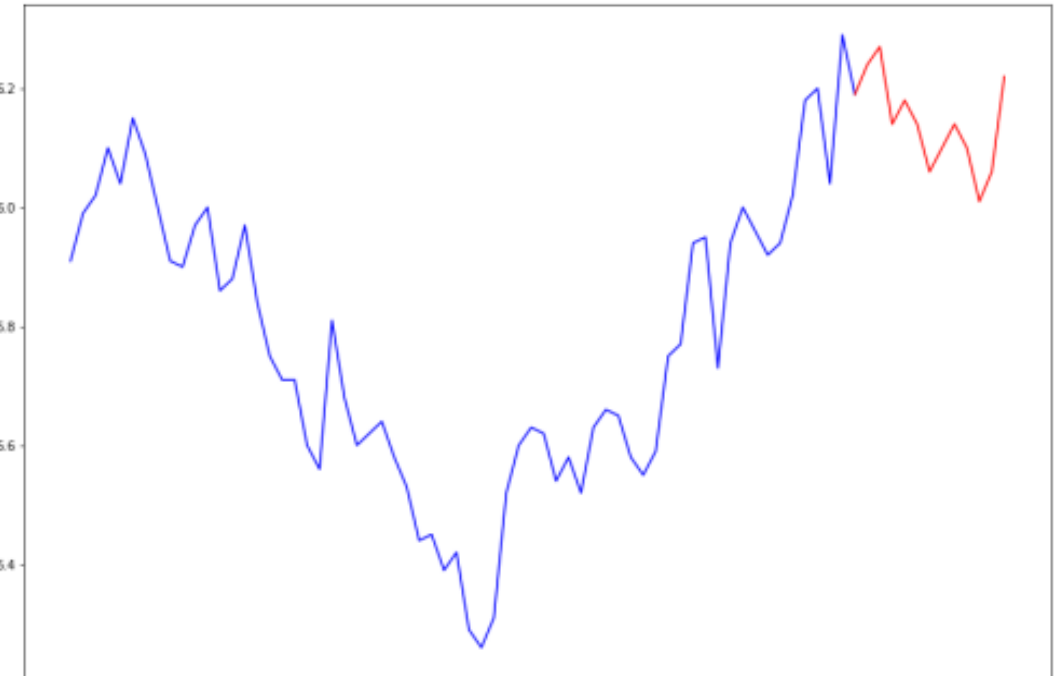
삼성전자(005930) 12월 1일 주식값 62,600원



출처 : <https://finance.daum.net/quotes/A005930#chart>

주가예측프로그램

삼성전자(005930) 12월 1일 주식값 예측 : 62,200원



VS

오차값 400원

0.64%

$$\text{오차} = \frac{\text{실제값} - \text{예측값}}{\text{예측값}} \times 100$$

기존 예측 사이트와의 성능 비교

한경닷컴

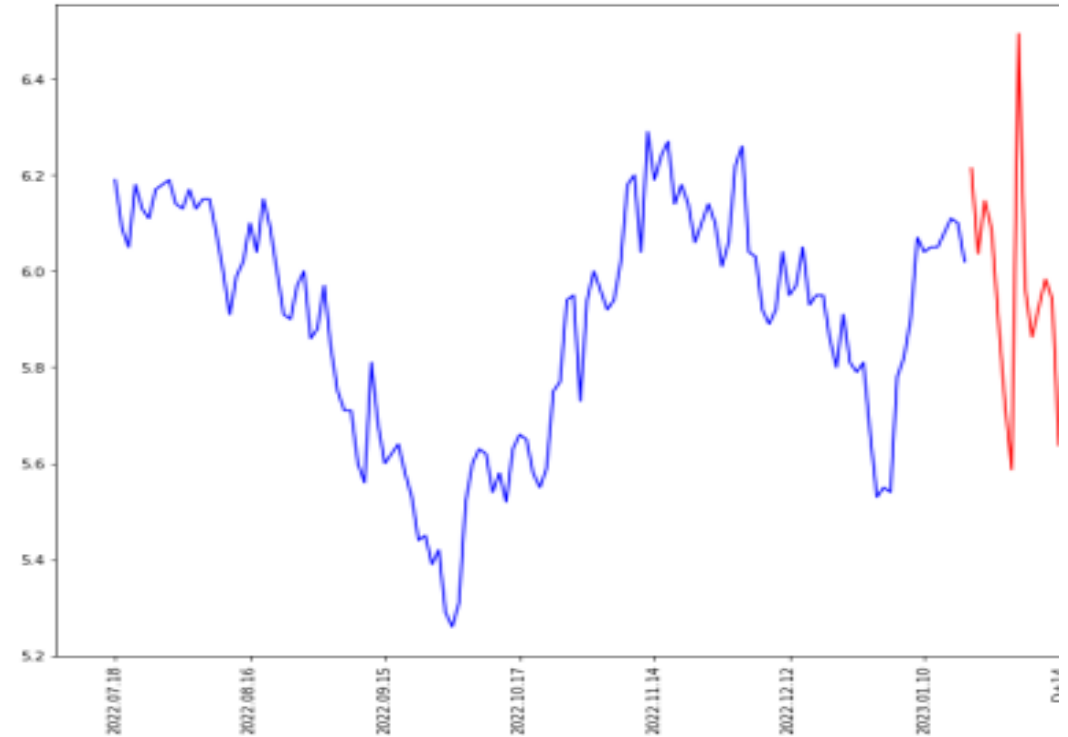
삼성전자(005930) 5일 뒤 주식값 예측 : 62,400원



출처 : <https://www.wowtv.co.kr>

주가예측 프로그램

삼성전자(005930) 5일 뒤 주식값 예측 : 61,100원



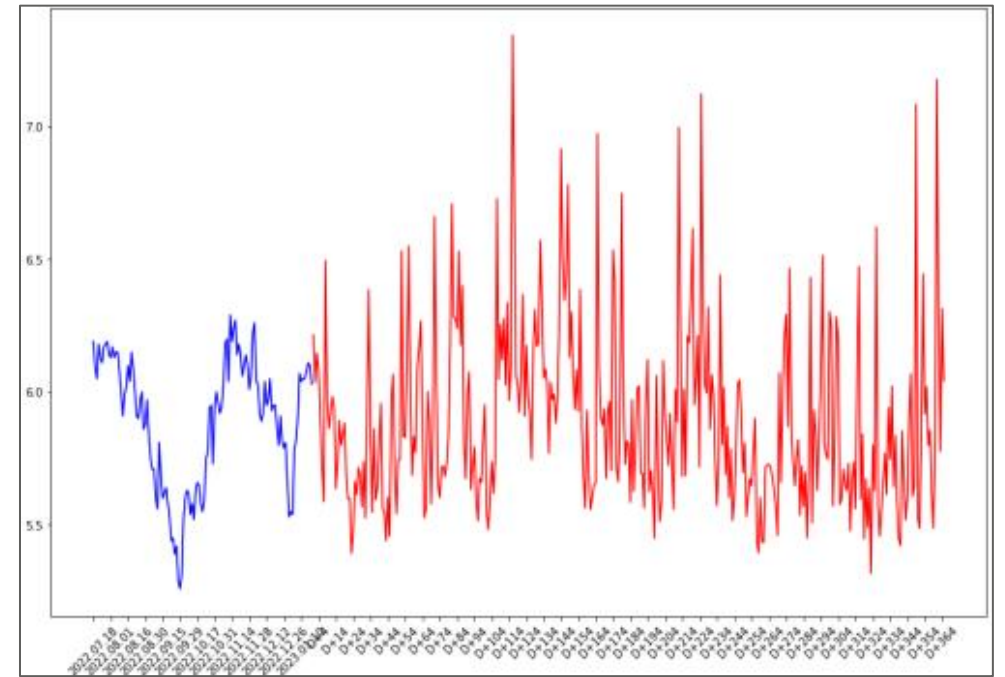
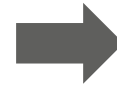
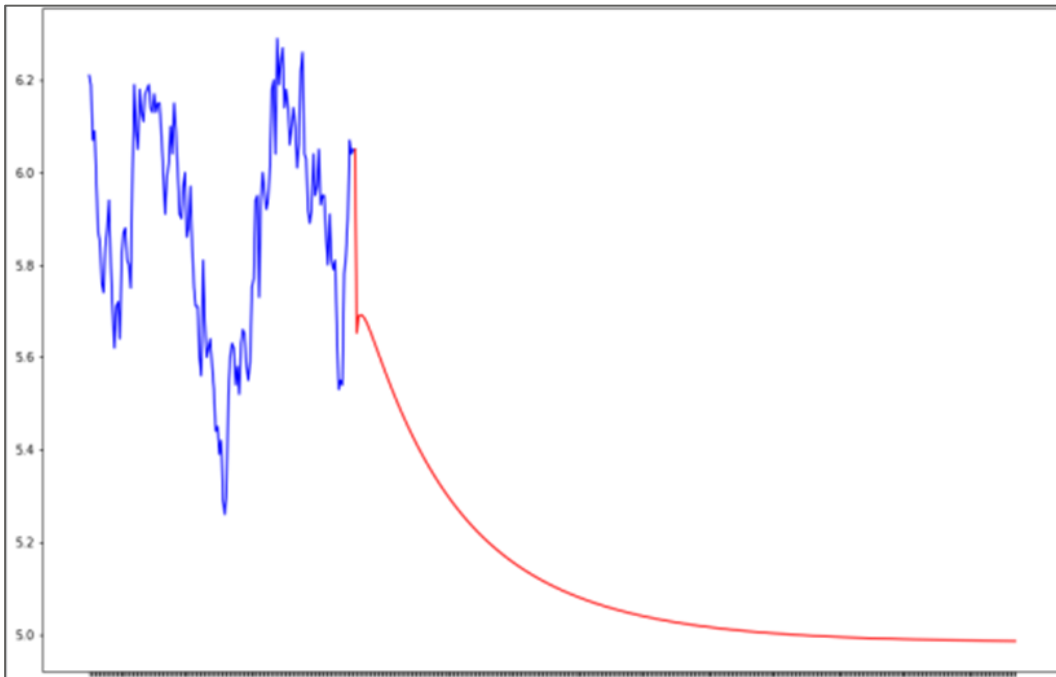
VS

오차값 1300원

2.12%

$$\text{오차} = \frac{\text{이론값} - \text{예측값}}{\text{예측값}} \times 100$$

중간발표와 성능 비교 (1년 예측)

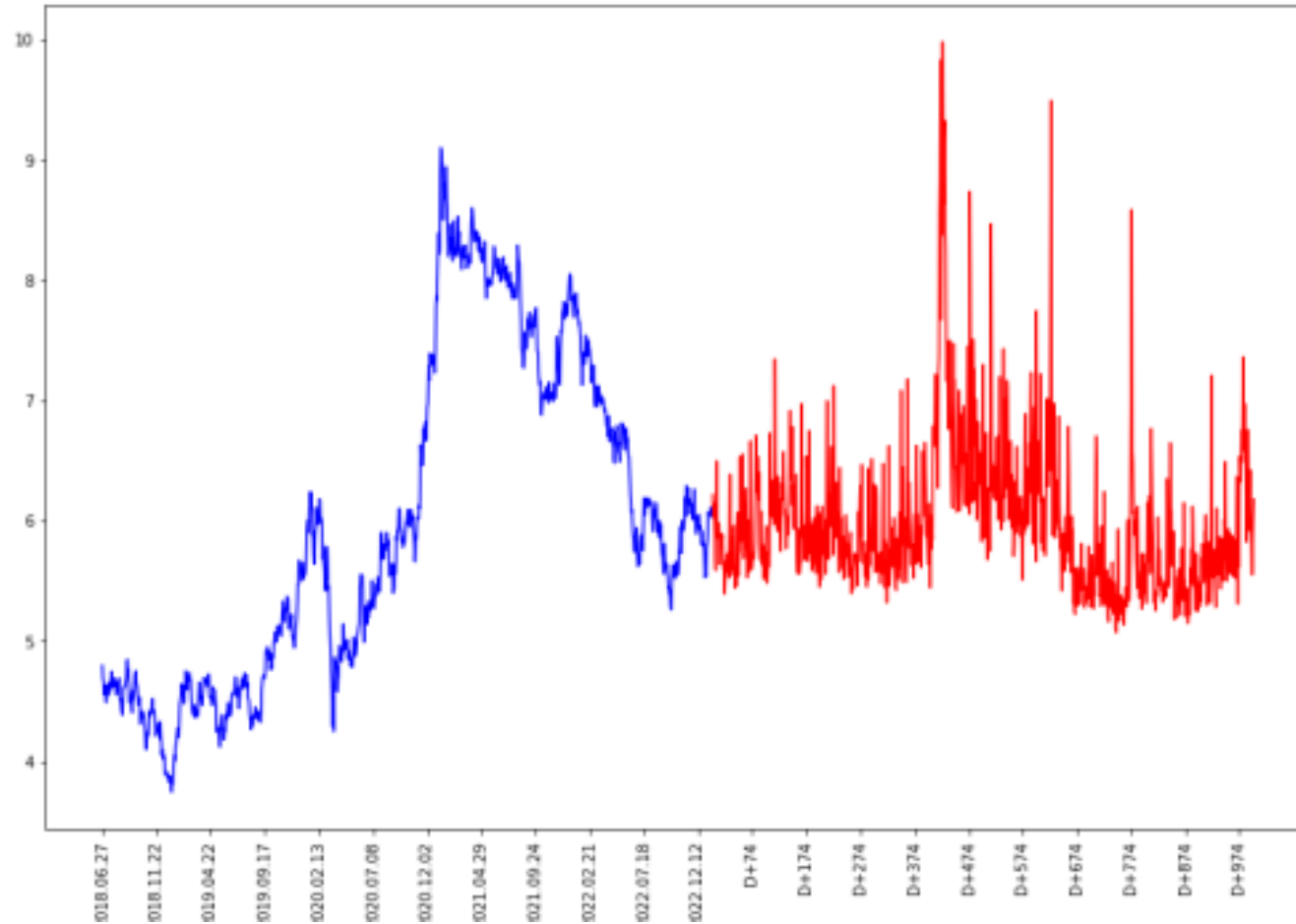


중간발표 결과물과 비교하여 장기 주가 예측 성능이 높아졌다.

평가 및 보완점

장기투자 3년 예측

3년의 투자 예측까지 성능 향상



- 각 회사별 데이터 개수 세분화 부족
- 종목별 지표지수와 같이 새로운 지표를 더 추가하지 못한 점
- 웹 구현을 못함 → 서버 성능에 비해 학습 연산량이 많아 부하로 인한 서버 오류
- 검증에 날짜 값 하나만을 가지고 오차율을 정한 점

향후 발전시킬 기능들

- 외부 프로세스를 자바에서 실행 되도록 BUILDER 클래스를 이용, 주식 예측 프로그램을 자바 기반 웹 페이지 서비스
- 가중치 조절 법을 정확하게 알아보고 더 나은 예측 값 도출
- 기사내용 크롤링 후, 자연어 처리하여 인풋데이터에 포함

Q&A



감사합니다!



codeName 함수

[BACK](#)

```
def codeName(name):  
    csv = pd.read_csv("./Code.csv", encoding='cp949')  
    csv = np.array(csv)  
    data=csv[:, :]  
    for i in range(len(data)):  
        if(data[i][3] == name):  
            return data[i][1];
```

exchangeRate 함수

```
def exchangeRate():  ##1달간격의 환율의 증가데이터 1행배열 57요소( 18.05.01~ 23.01.01) 배열로 출력 SIZE(1,57)
    csv = pd.read_csv(Stock.Exchange,encoding='utf8')
    #print(csv) 환율 전체데이터 확인하기
    csv = np.array(csv)
    data=[]
    for i in range(len(csv)):
        fnum=""
        for j in csv[i][1]:
            if(j == ','):
                continue
            fnum+=j
        data.append(float(fnum))
    data = np.array(data)
    data = data.astype('float32')
    data = normalize_scale(data)
    return data
```

consumIdx 함수

```
@staticmethod
def consumIdx():
    csv = pd.read_csv("./totalIdx.csv", encoding='cp949')
    csv = np.array(csv)
    csv = csv[0][1:]
    csv = csv.reshape(-1,1)
    return csv
```

kospidx 함수

[BACK](#)

```
@staticmethod
def kospidx():
    csv = pd.read_csv("./totalIdx.csv", encoding='cp949')
    csv = np.array(csv)
    csv = csv[1][1:]
    csv = csv.reshape(-1, 1)
    return csv
```


returnIdx 함수

```
def returnIdx(self):##ROA, ROE, EPS, BPS, DPS, PER, PBR의 각 항목당 (2018.12 ~ 2023.12)의 데이터 6개 값을 SIZE(7,8)
    get_param = {
        'pGB':1,
        'gicode':'A%s'%(self.code),
        'cID':'',
        'MenuYn':'Y',
        'ReportGB':'',
        'NewMenuID':101,
        'stkGb':701,
    }
    get_param = parse.urlencode(get_param)
    url="http://comp.fnguide.com/SV02/ASP/SVD_Main.asp?%s"%(get_param)
    tables = pd.read_html(url, header=0,encoding='utf-8')
    sit=np.array(tables[11])
```

returnIdx 함수

```
data=[]#정규화 시작
for i in range(6):
    crw=[]
    if (i == 2 or i==3): continue
    for j in range(7):

        if(sit[18+i][j+1] != sit[18+i][j+1]):
            idx = (float(sit[18+i][j]) + float(sit[18+i][j+2]))/2.0
            crw.append(idx)
            continue
        idx = float(sit[18+i][j+1])
        crw.append(idx)
    data.append(normalize(crw))
data=np.array(data)
return data
```

returnPrices 함수

종가, 거래량 등 주가 데이터 크롤링

```
def returnPrices(self):
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36'+
               '(KHTML, like Gecko)Chrome/86.0.4240.272 Whale/2.9.118.16 Safari/537.36',
               'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
               'Accept-Charset': 'ISO-8859-1,utf-8;q=0.7,*;q=0.3',
               'Accept-Encoding': 'none',
               'Accept-Language': 'en-US,en;q=0.8',
               'Connection': 'keep-alive'}
    url = "https://finance.naver.com/item/sise_day.nhn?code=%s"%(self.code)
    df = pd.DataFrame()
```

returnPrices 함수

```
for i in range(1, 121):
    page_url = '{}&page={}'.format(url, i)
    req = urllib.request.Request(page_url, headers=headers)
    response = urllib.request.urlopen(req)
    html = bs(response.read(), 'lxml')
    df = pd.concat([df, pd.read_html(str(html), header=0)[0]])
df = df.dropna()
arr = df.to_numpy()
arr1 = np.flipud(arr[:,1].reshape(-1,1))
arr1 /= normalizeAmount(arr1)
arr2 = np.flipud(arr[:,6].reshape(-1,1))
arr2 /= normalizeAmount(arr2)
date = np.flipud(arr[:,0].reshape(-1,1))
return np.concatenate((arr1, arr2), axis=1), date
```

plotGraph 함수

```
def plotGraph(prices, dates):  
    data_df = pd.DataFrame(prices[:,0], dates)  
    total_len = len(data_df.index[0:])  
    plt.figure(figsize=(15,10))  
    plt.plot(data_df.index[0:1127], data_df[0:1127], color='blue')  
    plt.plot(data_df.index[1126:], data_df[1126:], color='red')  
    plt.xticks(np.arange(0, total_len+1, 10), rotation=45)  
    plt.show()
```

predictNextDay 함수

[BACK](#)

```
dmodel = load_model('dmodel.h5')
def predictNextDay(prices, idx):
    pre_inputs = []
    prices=prices[-1127:]
    idx=idx[-1127:]
    pre_inputs.append(returnrInput(prices, idx))
    pre_inputs = np.array(pre_inputs)
    return dmodel.predict(pre_inputs)
```

```
coname = input("회사명 : ")
coid = Stock.codeName(coname)
days = int(input("일 수 : "))
CO = Stock(coid)

prices, date = CO.returnPrices()
idx = CO.returnIdx()

prices=prices[-1127:,:]
dates = date[-1127:,0].tolist()

for i in range(days):
    predictedPrice = predictNextDay(prices, idx)[0,0]
    prices = np.concatenate((prices, np.array([predictedPrice,prices[-1,1]]).reshape(1,-1)), axis=0)
    dates.append("D+%s"%(i+1))

plotGraph(prices, dates)
```


Train.py

[BACK](#)

```
SA = Stock('005930'); sa,_=SA.returnPrices(); sadata=SA.returnIdx();
SAC = Stock('029780'); sac,_=SAC.returnPrices(); sacdata=SAC.returnIdx();
SAL = Stock('032830'); sal,_=SAL.returnPrices(); saldata=SAL.returnIdx();
SAS = Stock('006400'); sas,_=SAS.returnPrices(); sasdata=SAS.returnIdx();
HDC = Stock('005380'); hdc,_=HDC.returnPrices(); hdcdata=HDC.returnIdx();
```

•
•
•

```
train_inputs = []
train_labels = []
train_inputs.append(returnInput(sa, sadata))
train_inputs.append(returnInput(sac, sacdata))
train_inputs.append(returnInput(sal, saldata))
train_inputs.append(returnInput(sas, sasdata))
train_inputs.append(returnInput(hdc, hdcdata))
train_inputs.append(returnInput(hdm, hdmdata))
```

•
•
•

Train.py

```
train_inputs = np.array(train_inputs)
train_labels = train_inputs[:, -1, 0]
train_inputs = np.array(train_inputs)[: , :-1, :]
train_labels = np.array(train_labels)

cp_callback = ModelCheckpoint(
    './dmodel.h5', monitor='loss', verbose=1, save_best_only=True, save_weights_only=False)
dmodel = Sequential([
    layers.LSTM(64, input_shape=(train_inputs.shape[1], train_inputs.shape[2]), return_sequences=True),
    layers.Dense(1, activation = 'linear')
])

dmodel.compile(loss='mse', optimizer='adam')
dmodel.summary()
history = dmodel.fit(train_inputs, train_labels, batch_size=1, epochs=50,
                    callbacks=cp_callback)
```

monthlydata 함수

BACK

```
def monthlydata(er):  
    monthdata = np.concatenate((np.full((23,1), er[-55]),  
                                np.full((19,1), er[-54]),  
                                np.full((20,1), er[-53]),  
                                np.full((22,1), er[-52]),  
                                •  
                                •  
                                •  
                                np.full((22,1), er[-3]),  
                                np.full((19,1), er[-2]),  
                                np.full((17,1), er[-1])), axis=0)  
    return np.array(monthdata)
```

returnInput 함수

```
def returnInput(price, idx):  
    price2018 = price[40:159]  
    price2019 = price[159:405]  
    price2020 = price[405:653]  
    price2021 = price[653:901]  
    price2022 = price[901:1167]  
  
    prices = np.concatenate((price2018, price2019, price2020, price2021, price2022), axis=0)  
    idxArray = np.concatenate((np.full((len(price2018),4), idx[:,0]),  
                                np.full((len(price2019),4), idx[:,1]),  
                                np.full((len(price2020),4), idx[:,2]),  
                                np.full((len(price2021),4), idx[:,3]),  
                                np.full((len(price2022),4), idx[:,4])), axis=0)  
    arr = np.concatenate((prices, idxArray), axis=1)  
    array = np.array(arr)
```

returnInput 함수

```
exc = Stock.exchangeRate()
exc = np.array(exc).reshape(-1, 1)
exchange = monthlydata(exc)
kos = Stock.kospiIdx()
kos = np.array(kos).reshape(-1, 1)
kospi = monthlydata(kos)
con = Stock.consumIdx()
con = np.array(con).reshape(-1, 1)
consume = monthlydata(con)
idxdata = ((array[:,2]*6.0).reshape(-1,1)+(array[:,3]*560.0).reshape(-1,1)
            +(array[:,4]*(-10.0)).reshape(-1,1)+(array[:,5]/(-1.25)).reshape(-1,1))
idxdata /= 4.0
idxdata = np.array(idxdata)
allidx = array[:,1].reshape(-1,1)*5+idxdata+np.flipud(exchange.reshape(-1,1))
        +np.flipud(kospi.reshape(-1,1))+np.flipud((consume/10).reshape(-1,1))
train_inputs = array[:,0].reshape(-1,1)

train_inputs = np.concatenate((train_inputs,
                                allidx),
                                axis=1)
return train_inputs[:,:].astype(float)
```

```
cp_callback = ModelCheckpoint(
    './dmodel.h5', monitor='loss', verbose=1, save_best_only=True, save_weights_only=False)
dmodel = Sequential([
    layers.LSTM(64, input_shape=(train_inputs.shape[1],train_inputs.shape[2]), return_sequences=True),
    layers.Dense(1, activation = 'linear')
])

dmodel.compile(loss='mse',optimizer='adam')
dmodel.summary()
history = dmodel.fit(train_inputs, train_labels, batch_size=1, epochs=150,
                    callbacks=cp_callback)
```