수익률 예측

학번 : 2020147051
이름 : 장건

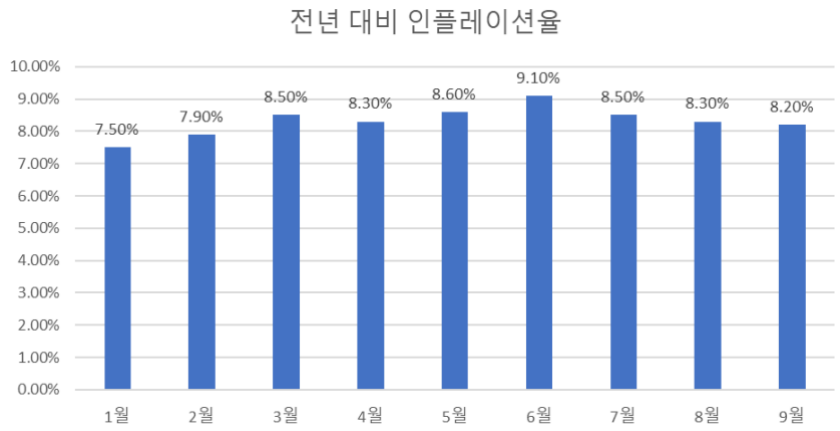
Contents

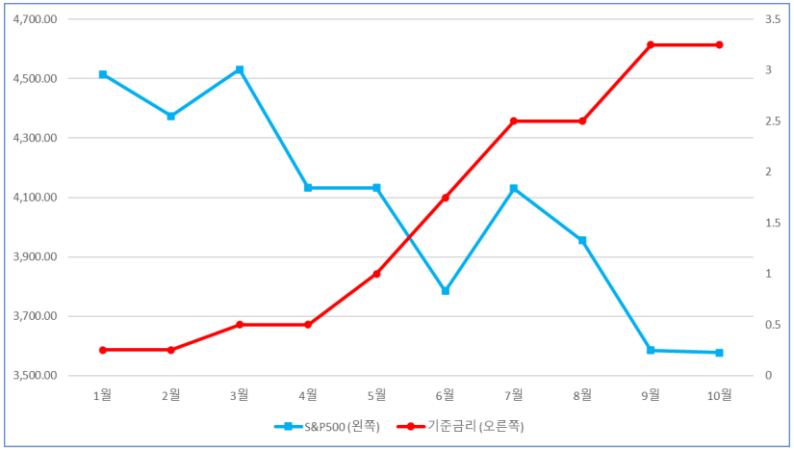
데이터 선택(종목 선택)

분석 종목

1. 코스피200(KS200)
2. 신한지주(055550)-KOSPI
3. 삼성화재(000810) -KOSPI
4. 매일홀딩스(005990) -KOSDAQ
5. 풍국주정(023900) -KOSDAQ

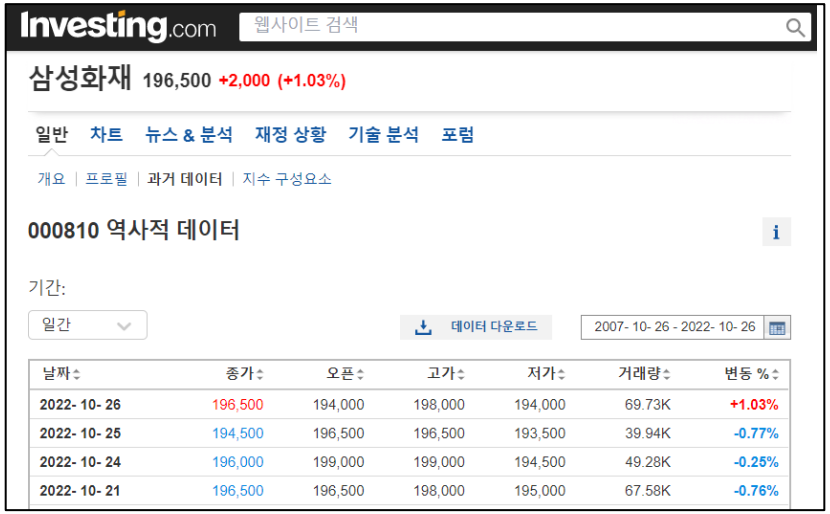
데이터 선택(종목 선택 근거)





치솟는 인플레이션과 연준의 지속적인 금리 인상으로 주식시장에 큰 타격을 입었다. 따라서 인플레이션과 금리인상에 가격방어를 할 수 있거나, 수혜를 받는 금융업, 음식료업에서 종목을 선정하였다.
또한, 최근에 설립 또는 인수/합병으로 변동이 심하고, 꾸준한 데이터가 없는 종목들은 시가총액의 크기와 업종에 불문하고 종목 선정 대상에서 제외하였다

데이터 선택(입력 데이터)


Investing.com 웹사이트 검색

삼성화재 196,500 +2,000 (+1.03%)

일반 차트 뉴스 & 분석 재정 상황 기술 분석 포럼

개요 | 프로필 | 과거 데이터 | 지수 구성요소

000810 역사적 데이터

기간:

일간 데이터 다운로드 2007- 10- 26 - 2022- 10- 26

날짜	종가	오픈	고가	저가	거래량	변동 %
2022- 10- 26	196,500	194,000	198,000	194,000	69.73K	+1.03%
2022- 10- 25	194,500	196,500	196,500	193,500	39.94K	-0.77%
2022- 10- 24	196,000	199,000	199,000	194,500	49.28K	-0.25%
2022- 10- 21	196,500	196,500	198,000	195,000	67.58K	-0.76%

- Investing.com에서 각 종목의 2007.10.26~2022.10.26 기간동안 일별 데이터 엑셀 형식으로 다운로드.
  ->2007년까지 데이터가 없을 경우 최대기간 데이터로 다운로드.

- 여러 column중 input data로는 '종가'와 '거래량 ' 을 입력 데이터로 선정.

데이터 선택(입력 데이터 근거)

1. 구간설정

다우존스의 차트처럼 국내 증시도 글로벌 경제에 영향을 받아 큰 상승 이후 하락 중이다. 지금 시점과 비슷한 '큰 상승 이후 하락' 구간을 학습시키고자 10년 이상의 장기간 일별 데이터를 input data로 설정하였다.

데이터 선택(입력 데이터 근거)

2. 입력데이터(종가, 거래량)

본 프로젝트의 목표인 각 종목의 종가 예측을 위해 종가는 필연적으로 필요한 데이터이다.

거래량은 종목의 매매양을 뜻하고 이는 시장 참여자들의 관심 정도라고 파악할 수 있다. 주가의 향방에 유의미한 영향을 미치는 데이터로 입력데이터로 적절하다고 판단하였다.

데이터 탐색(데이터 형태)

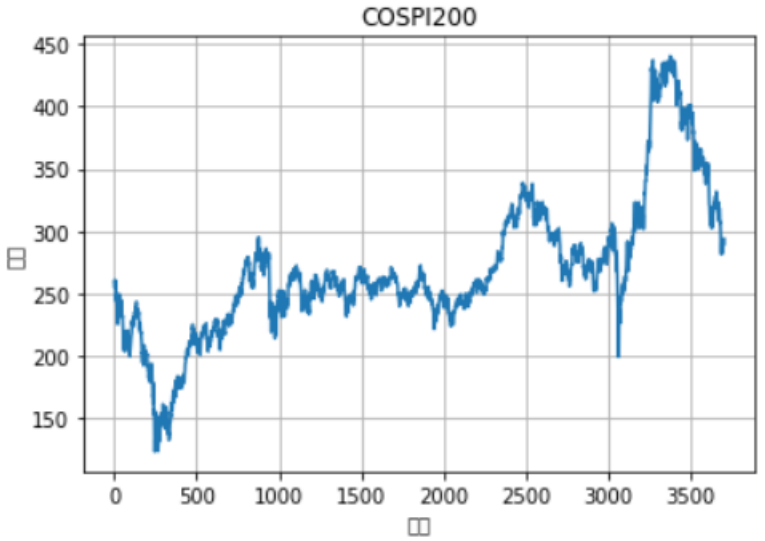
```
path='/content/sample_data/코스피 200 내역.csv'
raw_df = pd.read_csv(path)  # investing.com 로부터 데이터 다운로드

raw_df.head()
```

	날짜	종가	오픈	고가	저가	거래량	변동 %
0	2007-10-26	255.65	249.95	255.65	247.67	136.89M	2.75%
1	2007-10-29	260.40	258.38	261.45	257.75	135.22M	1.86%
2	2007-10-30	258.87	259.51	260.54	256.85	122.93M	-0.59%
3	2007-10-31	260.42	258.78	260.60	257.82	119.11M	0.60%
4	2007-11-01	260.76	263.06	263.15	258.71	122.33M	0.13%

날짜, 종가, 오픈, 고가, 저가, 거래량, 변동 % column들을 가진 데이터 형태이다.

데이터 탐색(데이터 형태)

```
plt.title('COSPI200')
plt.ylabel('종가')
plt.xlabel('기간')
plt.grid()

plt.plot(raw_df['종가'], label='종가')

plt.show()
```



기간과 종가로 시각화를 진행하면 흔히 HTS에서 볼 수 있는 주식 차트 모양이다.
이를 통해 시계열 데이터인 것을 다시 한번 확인할 수 있다.

데이터 탐색(데이터 형태)

```
# 통계정보 확인

raw_df.describe()
```

	종가	오픈	고가	저가
count	3708.000000	3708.000000	3708.000000	3708.000000
mean	269.458778	269.591060	271.137009	267.702985
std	57.424371	57.473796	57.651752	57.308319
min	123.270000	121.310000	127.660000	117.910000
25%	241.267500	241.245000	242.730000	239.622500
50%	260.090000	259.980000	261.225000	258.765000
75%	291.282500	292.060000	293.462500	289.845000
max	440.400000	440.780000	449.040000	438.750000

Pandas 라이브러리의 describe()함수를 통해 데이터의 통계값들을 볼 수 있다.
총 3708일의 데이터이며, 종가 기준 123.27~440.40 가격까지 범위를 가지는 것을 알 수 있다.

전처리 방법

1. 데이터 탐색 결과에 따른 필요한 전처리 방법들 소개
   1) 사용할 전처리 방법 소개
   2) 전처리 방법 적용할 데이터와 그 이유 (전처리 했을 때의 이점 등)
   3) 적용 방법 소개 (library, 함수 등)

   +) 다수의 전처리 방법 사용 가능

전처리 방법(결측치 제거)

```
# Missing Data 확인

raw_df.isnull().sum()
```

```
날짜       0
종가       0
오픈       0
고가       0
저가       0
거래량      0
변동 %     0
dtype: int64
```

```
# Missing Data 확인

raw_df.isnull().sum()
```

```
날짜       0
종가       0
오픈       0
고가       0
저가       0
거래량    228
변동 %     0
dtype: int64
```

왼쪽 사진은 코스피200의 결측치 데이터이고, 오른쪽 사진은 신한지주의 결측치 데이터이다. 코스피 200은 결측치가 없지만 신한지주와 같은 개별종목들은 다수의 결측치가 발견되었다.
결측치 해당 날짜에 가격변동이 없는것으로 보았을때, 거래정지, 휴일, 데이터 유실 등으로 파악이 되므로 결측치에 해당하는 열은 모두 삭제하기로 하였다.

전처리 방법(결측치 제거)

```
# missing data 처리

raw_df = raw_df.dropna()

raw_df.isnull().sum()
```

```
날짜      0
종가      0
오픈      0
고가      0
저가      0
거래량     0
변동 %    0
dtype: int64
```

dropna()를 이용하여 결측치를 제거

전처리 방법(데이터타입 변경)

```
# 데이터 타입 확인

raw_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2801 entries, 0 to 3028
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   날짜      2801 non-null   object
 1   종가      2801 non-null   object
 2   오픈      2801 non-null   object
 3   고가      2801 non-null   object
 4   저가      2801 non-null   object
 5   거래량     2801 non-null   object
 6   변동 %    2801 non-null   object
dtypes: object(7)
memory usage: 175.1+ KB
```

데이터 타입을 확인시 모든열의 데이터가 문자열 함수인것을 알 수 있다.

문자열은 연산시 컴퓨터가 읽지 못하므로 float형식으로 변환하였다.

전처리 방법(데이터타입 변경)

```
raw_df['거래량']=raw_df['거래량'].str.replace('M','0000')
raw_df['거래량']=raw_df['거래량'].str.replace('K','0')
raw_df['거래량']=raw_df['거래량'].str.replace('.','')
raw_df['거래량']=raw_df['거래량'].astype(float)
raw_df['종가']=raw_df['종가'].str.replace(',','')
raw_df['종가']=raw_df['종가'].astype(float)
```

이번 프로젝트에서 input으로 넣을 거래량과 종가 데이터만 float형식으로 변환하였다.
문자열을 replace함수로 제거하거나 대체한 뒤 astype 함수를 통해 float타입으로 변환하였다.

전처리 방법(정규화)

```
# 정규화

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scale_cols = ['종가','거래량']

scaled_df = scaler.fit_transform(raw_df[scale_cols])

scaled_df = pd.DataFrame(scaled_df, columns=scale_cols)

print(scaled_df)
```

```
            종가       거래량
0     0.417431  0.219699
1     0.432409  0.217018
2     0.427585  0.197288
3     0.432472  0.191155
4     0.433545  0.196324
...        ...       ...
3703  0.521363  0.219715
3704  0.521237  0.164281
3705  0.530382  0.186452
3706  0.530729  0.186724
3707  0.537887  0.193146

[3708 rows x 2 columns]
```

주가 특성상 편향이 자주 일어날 수 있기에 예측 학습이 과적합될 수 있다.
따라서 변수의 편향성을 줄이기 위하여 위와 같이 정규화를 진행하였다.

변수로만 사용하는 종가와 거래량의 정규화를 진행하였고, sklean라이브러리에서 MinMaxScaler를 이용하여 정규화를 진행하였다.

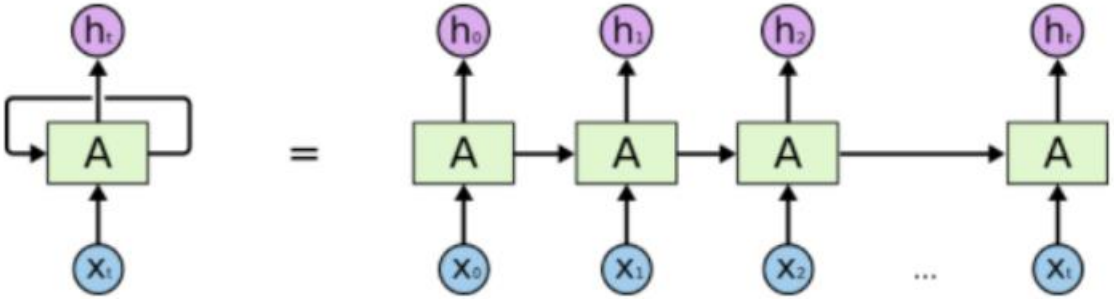
전처리 방법(데이터 분할)

```
# train, test 분리

split = -1100 #train:test = 7:3

x_train = X[0:split]
y_train = Y[0:split]

x_test = X[split:]
y_test = Y[split:]

print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

Train set과 test set을 7:3으로 지정하여 분할하였다.

8:2로 학습 결과 정확도가 떨어지는 경향이 나타나 7:3으로 분할하였다.

모델 소개(RNN)



순환 신경망에서는 뉴런을 셀(Cell)이라고 부르는데, 순환 신경망에서 순환(recurrent)한다는 것은 위 그림에서 보듯이 이전의 데이터를 통해 학습된 셀의 상태 정보가 다음 데이터를 이용하여 학습시킬 때 다시 사용된다는 의미다.
따라서, 순환 신경망은 시계열 데이터를 처리할 때 적합하며, 위 그림처럼 펼쳐서 나타낼 수 있습니다.

모델 소개(RNN)

2006년 7월부터 2015년 7월까지의 데이터를 학습 집합으로 사용하였고 2015년 8월부터 2016년 7월까지의 데이터를 테스트 집합으로 사용하였다. 학습 집합으로 신경망 학습 후, 테스트 집합으로 결과로 정확도를 측정한 결과 MLP는 51.11%, CNN은 51.19%, RNN은 52.09%를 기록하였다.

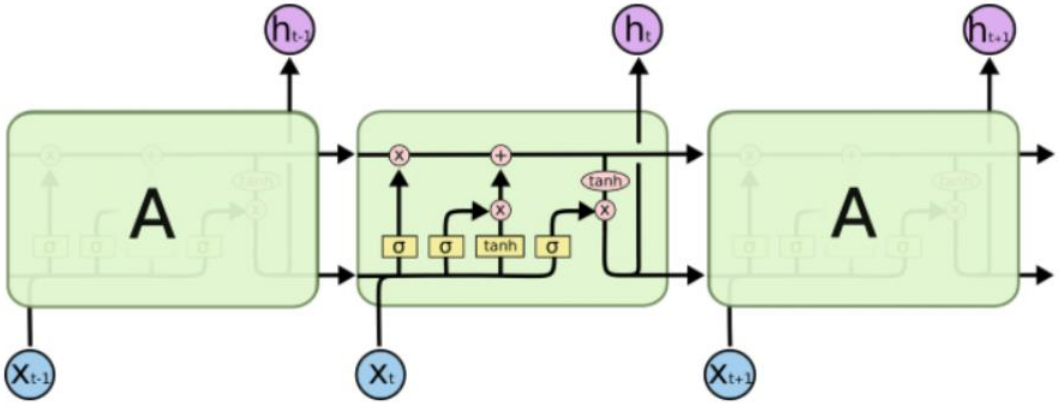
위 논문에서 알 수 있듯이 주가예측이라는 시계열 분석에서 RNN의 효율이 다른 딥러닝 모델보다 정확도가 높은 것을 알 수 있다.

참고문헌: 이지훈 "Stock price prediction model using deep learning"

모델 소개(LSTM)

그러나 RNN은 만약 데이터가 너무 길어져 이를 표현하는 신경망이 깊어져야만 할 경우 문제가 발생합니다. RNN은 역전파를 통해 학습하는데, 층이 깊어지면 그래디언트가 너무 작아져 학습이 잘 안 되는 문제(Vanishing Gradient Problem)가 발생한다. 이 문제를 해결하기 위해 LSTM이 만들어졌다.

LSTM은 셀 스테이트(cell state)라는 개념을 도입하여 그 내부에 있는 게이트(gate)들을 통해 어떤 정보를 기억하고 어떤 정보를 버릴지 추가적인 학습이 가능하다. 이를 통해 RNN이 가진 문제(Vanishing Gradient Problem)를 해결할 수 있다.



모델 소개(LSTM)

```
# 입력 파라미터 feature, label => numpy type

def make_sequene_dataset(feature, label, window_size):

    feature_list = []       # 생성될 feature list
    label_list = []         # 생성될 label list

    for i in range(len(feature)-window_size):

        feature_list.append(feature[i:i+window_size])
        label_list.append(label[i+window_size])

    return np.array(feature_list), np.array(label_list)
```

얼마동안 기간을 학습하여 다음 데이터를 예측할지 산정하는 window size 를 정의하는 함수에 대한 코드이다.

모델 소개(LSTM)

```
# feature_df, label_df 생성

feature_cols = [ '종가', '거래량' ]
label_cols = [ '종가' ]

feature_df = pd.DataFrame(scaled_df, columns=feature_cols)
label_df = pd.DataFrame(scaled_df, columns=label_cols)
```

```
window_size = 30

X, Y = make_sequene_dataset(feature_np, label_np, window_size)

print(X.shape, Y.shape)
```

Pandas 라이브러리의 dataframe을 통해 input과 ouput의 column들을 지정하였고, 이전 페이지에서 소개한 제작 함수를 통해 window size를 30으로 지정하였다.
30일의 학습을 통하여 30일 이후의 다음날의 종가를 예측한다는 정의이다.

모델 소개(LSTM)

```
# model 생성

model = Sequential()

model.add(LSTM(128, activation='tanh', input_shape=x_train[0].shape))

model.add((Dense(1, activation='linear')))

model.compile(loss='mse', optimizer='adam', metrics=['mae'])
```

Tensorflow 라이브러의 sequential, LSTM, Dense 함수를 통해 LSTM의 모델 구조를 제작하였다. Tanh의 활성화 함수와 adam optimizer로 지정하였다.

모델의 LSTM층은 1층인데, LSTM도 층이 깊어질수록 vanishing gradient 문제가 발생하기에 의도적으로 얇은 층으로 구성하였다.

수익률 예측 – 모델 학습

```
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss', patience=5)

model.fit(x_train, y_train,
          validation_data=(x_test, y_test),
          epochs=100, batch_size=10,
          callbacks=[early_stop])
```

100개의 에폭과 10 배치사이즈를 진행하였고, 과적합 방지를 위해 early stopping 코드를 수행하였다.

수익률 예측 – 모델 학습

결과값이며 예측이 실제와 유사한것을 알 수 있다.

수익률 예측 – 최종 예측

```
date='2022-10-27'  #예측날짜
pred_raw = scaler.inverse_transform(pred)
pred_nextday=pred_raw[-1]
print(date+' 예측종가는 ₩{}입니다.'.format(pred_nextday))
```

```
2022-10-27 예측종가는 ₩[290.57083]입니다.
```

위 코드를 사용하여 예측 종가를 산정하였고, 거래량 또한 학습하여 향후 거래량을 예측하였다. 예측값을 raw data에 업데이트하여 종목당 총 3번 예측 코드를 수행하였다.

수익률 예측 – 최종 예측

구분	종목명	종목코드	10월 27일	10월 28일	10월 31일 (월) 수익률
지수	코스피200	KS200	-0.01116	0.004646	3.77E-04
코스피 종목	신한지주	55550	0.02695	-0.01934	0.023944
코스피 종목	삼성화재	810	-0.00509	0.005115	-0.01323
코스닥 종목	매일홀딩스	5990	-0.09721	0.230148	-0.12582
코스닥 종목	풍국주정	23900	0.008929	-0.02212	0.00905