ARIMA, LSTM 알고리즘을 이용한 주가예측 분석

2021.08.27

C조 (금융)

팀장 : 정승연

팀원 : 허상범

최윤석

1. 주제
   1.1 주제 명 : 시계열 분석을 통한 주가예측
   1.2 분석 목표 KOSPI, KOSDAQ, NASDAQ,SOX 지수 및 삼성전자 주가 데이터를 통해 ARIMA, RNN LSTM 모델 분석을 수행하여 주가예측 모델을 구현함.

2. 분석개요
   2.1 분석 개요

- KOSPI, KOSDAQ의 종가, DEXKOUS(원달러환율), DGS3(3년물국채), WTI(국제유가 - 서부 텍사스 유), NASDAQCO(나스닥종합주가지수), SP500(S&P종합주가지수) 데이터를 활용한 ARIMA 분석을 통해 주가 예측 모델 구현
- 가격데이터, 기술지표데이터, 시장데이터를 이용하여 다음날 수정종가의 상승, 하락을 예측하는 LSTM 모델 구현

2.2 분석 데이터 개요

- ARIMA : KRX에서 한국 대표 주가지수인 KOSPI, KOSDAQ의 종가를 수집 및 FRED에서 DEXKOUS(원달러환율), DGS3(3년물국채), WTI(국제유가 - 서부 텍사스 유), NASDAQCO(나스닥종합주가지수), SP500(S&P종합주가지수)를 수집했으며, 데이터 수집 기간은 2021년 1월 4일부터 8월 11일까지의 데이터를 사용한다.
- LSTM : Yahoo finance에서 삼성전자(005930.KS), KOSPI(^KS11), 반도체섹터지수(^SOX)를 수집했으며, 기간은 2021년 1월 4일부터 8월 20일까지의 데이터를 사용한다.

3. ARIMA
(https://github.com/hersheythings/Data-Analysis/blob/main/ML/ARIMA_ML.ipynb)
3.1. 분석 개요
RNN 기반의 LSTM 모델을 사용하기 전에, 시계열 데이터를 예측하는 전통적인 머신러닝 방법론인 **ARIMA 모델**을 활용하여 다양한 경제/금융 변수들의 데이터 추이를 예측하는 것을 목표로 함

3.2. 활용 데이터

변수명	설명	집계 국가	전처리 이전 데이터 개수
일자	2021년 1월부터 7월까지 총 160일간의 시계열 index	공통	160
원달러환율	KRW/USD 매매기준율	한국, 미국	150
3년물국채 이자율	3-Year US Treasury Bill 만기 이자율	미국	152

나스닥	나스닥 지수	미국	6
국제유가	Crude Oil Prices (Brent - Europe)	유럽	152
S&P500	S&P500 지수	미국	152
코스피	코스피 지수	한국	153
코스닥	코스닥 지수	한국	153

3.3. 분석 과정 및 주요 방법론

3.3.1. 데이터 수집

- FRED, 한국은행의 경제통계 DB를 통해 7가지 변수(날짜 인덱스 제외)의 csv 파일을 다운로드 받아 원본 데이터를 수집 (dtype은 전부 float64)

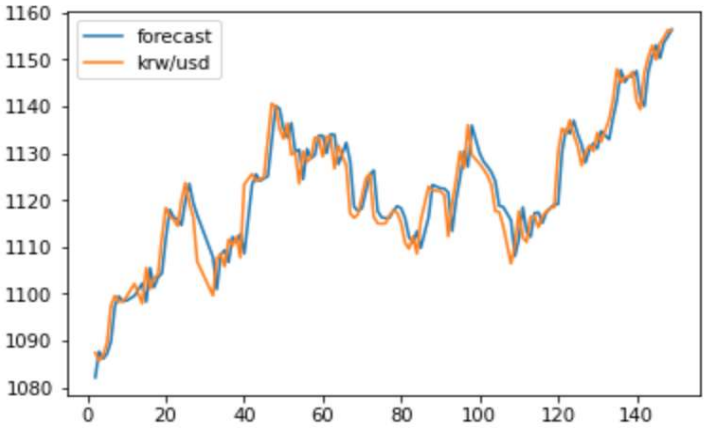
3.3.2. 데이터 전처리 및 EDA

- 시계열 데이터의 길이를 동일하게 정의하기 위해서 총 7가지의 변수 가운데 최소 데이터 길이를 가진 변수를 기준으로 데이터프레임을 정의하고자 함
  - 이 과정에서, 국제 유가 데이터의 길이가 6으로 매우 짧아 타 변수 대비 분석의 유의미성이 매우 낮을 것으로 판단하여 해당 변수를 drop함 ⇒ 6개의 변수만을 가지고 모델링을 진행하게 됨
  - 이에 따라, 2번째로 최소 길이(150개)를 가진 변수인 “원달러환율” 변수의 길이에 맞게 데이터프레임을 정의함
- EDA의 경우, 시계열 데이터이기 때문에 plot 차트를 통해 개별 변수의 추이를 먼저 살펴보았음. 이후, 데이터 시각화 라이브러리 matplotlib을 사용하여 scatterplot, pairplot, 상관관계 Heatmap 차트를 그려 변수간 상관성을 살펴보고, 상관관계가 낮은 변수들은 무엇인지, 높은 변수들은 무엇인지 확인 과정을 거침.
  - 특히, 상관관계 분석 과정에서 변수 간 상관관계가 높음 또는 낮음을 정의하려면 어느정도의 상관계수가 도출되어야 하는지 난해한 부분이 많았음.

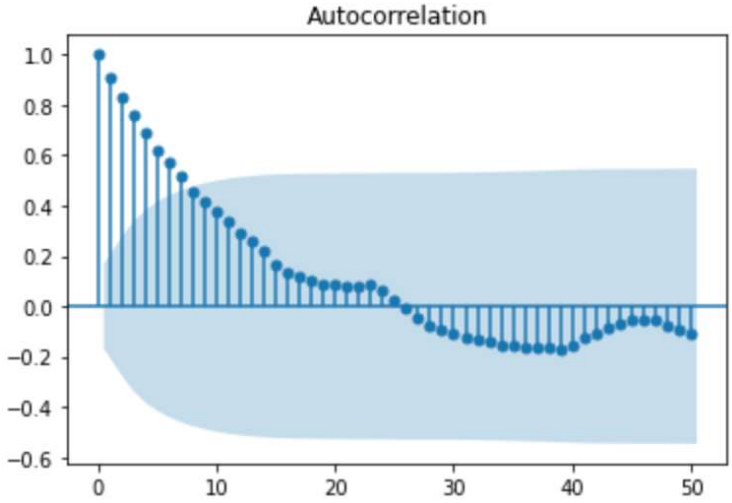
3.3.3. 시계열 데이터 분석용 Python 라이브러리 : statsmodel.tsa

- ARIMA 모델을 Python으로 구현함에 있어 매우 유용한 라이브러리인 statsmodel.tsa를 사용하여 해당 라이브러리에 내장되어 있는 ARIMA 패키지를 사용했고, ARIMA 모델에서 매우 중요한 함수인 자기상관함수, 부분 자기상관함수까지 시각화해보는 것을 목표로 함
  - ARIMA 모델은 단어 그대로 AR, MA 모델이 결합되어있는 예측 모델이기 때문에 모델에 입력되는 파라미터(모수) 역시 AR(p), MA(q) 모델을 그대로 추종하며, 여기에 ARIMA만의 모수인 차분 횟수(d)를 추가로 입력받음.

- 실제 파라미터를 입력할 때에는 변수 6개에 전부 (1,1,1)을 입력하여 총 6개의 예측 값을 도출했고, 원달러환율을 예시로 하여 차트를 그려보면 아래와 같은 차트가 도출됨 ⇒ 원본 데이터와 거의 동일한 추세로 예측되고 있음



- 자기상관성함수의 경우, lag이 어느 정도로 변하는지에 따라 이전 시퀀스 데이터로부터 어느 정도의 상관계수(자기상관계수)를 갖는지 평가하였음.
  - 이 과정에서는 자기상관성함수 시각화 패키지인 plot_acf을 사용하여 lags의 값을 변화시켜보면서 자기상관계수의 변동을 살펴봄. lags=50일로 측정할 경우 아래와 같은 차트가 도출됨



4. RNN - LSTM

(https://www.kaggle.com/jesy0412/lstm-stock-price-prediction?scriptVersionId=73217959)

4.1. 분석개요 - RNN을 활용한 주가 방향성 분류 예측

참고한 기존 논문(Application of Deep Learning to Algorithm Trading , 2017)은 미국 주식시장에 상장된 intel 종목의 주가데이터, 기술지표데이터, 시장데이터를 활용하여 LSTM 알고리즘을 이용해 다음날 수정 종가를 예측하는 회귀문제를 분석하는 목적이었다. 실제 투자에서는 상승, 하락에 대한 정보를 바탕으로 포지셔닝을 해야하기 때문에 분류문제가

타당하다고 판단하였다. 따라서, 상승, 하락을 예측하는 분류문제로 변환한 깃허브 코드 (https://github.com/quant4junior/algoTrade)를 참고하여, KOSPI에 상장된 삼성전자의 주가 방향성 분류를 예측하는 분류문제 분석을 진행하였다.

4.2. 데이터

구분	내용
가격데이터	시가, 고가, 저가, 수정 종가, 거래량, 로그수익률
기술지표 데이터	이동평균선 (5일, 10일) 이동 표준 편차(5일, 10일) 볼린저 밴드(이동 평균선 상위 2표준 편차선, 하위 2표준 편차선) 주가의 변동성 측정값(ATR) 1개월 모멘텀:한 달 전 가격과 현재 가격의 차이 CCI (Commodity Channel Index) 사이클 트렌드 오실레이터 3개월 모멘텀 변동비율 MACD(Moving Average Convergence Divergence) : 모멘텀 트렌드 지표 Williams percent range : 매수/매도 스트레스 측정
시장 데이터	KOSPI 지수 : 한국 시장 대표 지수 변동성지수(VIX) : KOSPI 지수 옵션 가격의 향후 30일 동안의 변동성 SOX : 반도체 지수

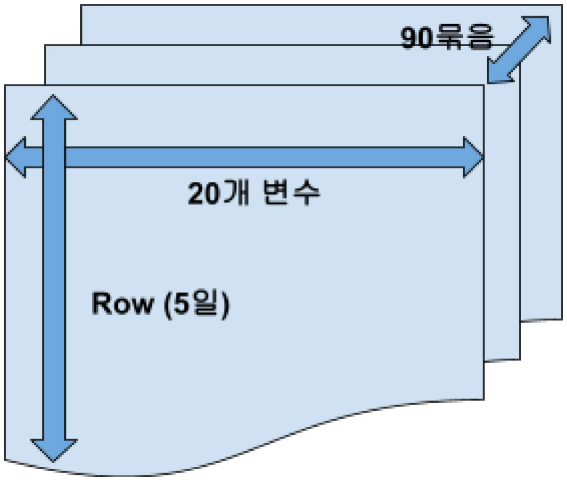
Yahoo finance를 통해 KOSPI(KS11), 삼성전자(005930), SOX(반도체 섹터 지수)를 사용하였다. 소비자가전 섹터 지수(DWCCSE)를 추가하고 싶었지만, 매일 갱신되며 수집이 불가능한 데이터라서 반도체 섹터만 고려하였다.
데이터 가공 절차는 다음과 같다.

1) Raw 데이터 수집 (yahoo finance)
2) 데이터 가공 및 결합
3) Test, validation, train 데이터셋 분리
4) 데이터 정규화(min-max normalization)
5) Feature, label data 나누기

Date	Open	High	Low	Close	Adj Close	Volume	next_price	next_rtn	log_return	CCI	...	ATR	ub	middle	lb	MTM1	MTM3	ROC	WPR	KOSPI	SOX
2021-01-04	81000	84400	80200	83000	82268.57031	38655276	83160.64844	0.024691	0.000000	0.000000	...	0.000000	0.00000	0	0.00000	0	0	0.000000	0.000000	2944.449951	2783.209961
2021-01-05	81600	83900	81600	83900	83160.64844	35335669	81475.62500	0.028186	0.010785	0.000000	...	0.000000	0.00000	0	0.00000	900	0	0.000000	0.000000	2990.570068	2837.169922
2021-01-06	83300	84500	82100	82200	81475.62500	42089013	82169.46094	-0.013205	-0.020470	0.000000	...	0.000000	0.00000	0	0.00000	-1700	0	0.000000	0.000000	2968.209961	2827.959961
2021-01-07	82800	84200	82700	82900	82169.46094	32644642	88017.46094	0.001208	0.008480	0.000000	...	0.000000	0.00000	0	0.00000	700	-100	0.000000	0.000000	3031.679932	2937.000000
2021-01-08	83300	90000	83000	88800	88017.46094	59013307	90198.07813	0.066026	0.068752	0.000000	...	0.000000	0.00000	0	0.00000	5900	4900	0.000000	0.000000	3152.179932	2936.469971
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2021-08-17	74000	75100	74000	74200	74200.00000	30944647	73900.00000	0.002703	-0.002692	-167.844768	...	1314.244468	83507.60891	79175	74842.39109	-200	-4300	-6.900878	-97.849462	3143.090088	3256.820068
2021-08-18	73900	74600	73100	73900	73900.00000	29192631	73100.00000	0.000000	-0.004051	-149.767764	...	1327.512720	83826.15939	78920	74013.84061	-300	-3100	-7.509387	-92.156863	3158.929932	3208.830078
2021-08-19	73500	74400	73100	73100	73100.00000	22166298	72700.00000	-0.005442	-0.010884	-124.448323	...	1325.547526	84174.30991	78650	73125.69009	-800	-1300	-8.395990	-100.000000	3097.830078	3235.870117
2021-08-20	73500	73900	72500	72700	72700.00000	22364803	73500.00000	-0.010884	-0.005487	-108.716086	...	1330.865560	84373.54921	78300	72226.45079	-400	-1500	-8.668342	-98.148148	3060.510010	3256.500000
2021-08-23	73300	73700	73000	73500	73500.00000	7312479	73500.00000	0.002729	0.010944	-85.006878	...	1307.232305	84409.96875	78010	71610.03125	800	-400	-8.239700	-90.740741	3094.060078	3256.500000

160 rows × 25 columns

위와 같은 방법으로 구성된 데이터는 160 x 25 사이즈로, x_train.shape는 (90, 5, 20)이다. 즉, 5일 타임라인에 걸친 20개의 변수를 사용해 다음날 종가를 예측하도록 했으며, 1일씩 윈도우를 슬라이딩해서 90번의 이동이 진행되는 3차원 데이터이다.



Train data는 1월 4일부터 5월 21일까지, Validation data는 5월 24일부터 7월 5일까지, Test data는 7월 6일부터 8월 20일까지 설정하여, 전체 160개의 데이터 중 Train 95개, Valid 31개 Test데이터를 34개로 대략 6:2:2 의 비율로 나누었다.

4.3. 모델 구조

케라스를 활용해 LSTM모델을 구성하고 모델을 훈련한다. LSTM 신경망은 시퀀스 데이터를 분석하는데 좋은 성능을 보이는 RNN 알고리즘으로, 기존RNN보다 가중치 감쇠

문제를 해결하는데 우수한 성능을 보인다. 논문과 참조코드를 바탕으로 5개의 LSTM층과 200개의 뉴런을 사용하였으며, 각 LSTM층에 200개의 뉴런을 사용할 수 있도록 매개변수를 전달하고 5개의 층을 만들어 L2 정규화를 적용하였다. 각 층에서 출력되는 값을 다음 층의 입력 인풋으로 사용할 수 있도록 설정하였다. 모델의 요약 정보는 다음과 같다.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 5, 20)]           0
_________________________________________________________________
lstm (LSTM)                  (None, 5, 200)            176800
_________________________________________________________________
batch_normalization (BatchNo (None, 5, 200)            800
_________________________________________________________________
lstm_1 (LSTM)                (None, 5, 200)            320800
_________________________________________________________________
dropout (Dropout)            (None, 5, 200)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 5, 200)            320800
_________________________________________________________________
batch_normalization_1 (Batch (None, 5, 200)            800
_________________________________________________________________
lstm_3 (LSTM)                (None, 5, 200)            320800
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 200)            0
_________________________________________________________________
lstm_4 (LSTM)                (None, 200)               320800
_________________________________________________________________
batch_normalization_2 (Batch (None, 200)               800
_________________________________________________________________
dense (Dense)                (None, 2)                 402
=================================================================
Total params: 1,462,802
Trainable params: 1,461,602
Non-trainable params: 1,200
_________________________________________________________________
None
```

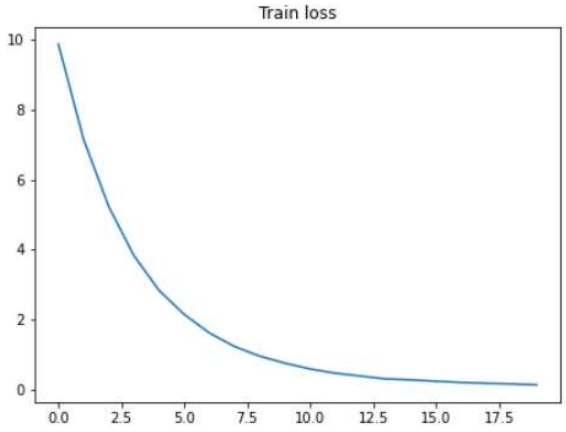
4.4 모델 학습

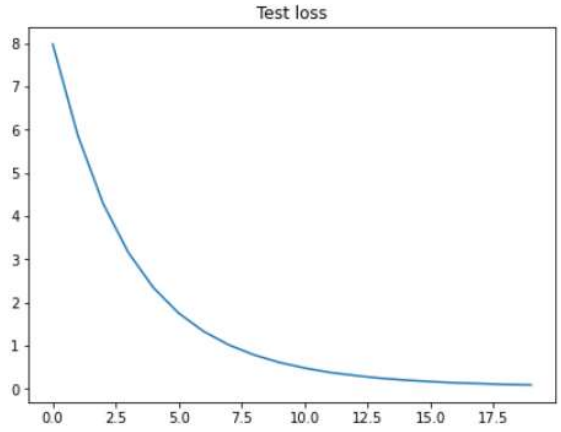
손실함수로 크로스 엔트로피, 옵티마이저로는 Adam 옵티마이저를 사용한다. 작업 효율성을 위해 epochs 수는 20회로 설정한다.

```
Epoch 1/20
9/9 [==============================] - 13s 232ms/step - loss: 10.5564 - accuracy:
0.4817 - val_loss: 7.9709 - val_accuracy: 1.0000
Epoch 2/20
9/9 [==============================] - 0s 15ms/step - loss: 7.5658 - accuracy:
0.5358 - val_loss: 5.8494 - val_accuracy: 1.0000
Epoch 3/20
9/9 [==============================] - 0s 15ms/step - loss: 5.5270 - accuracy:
0.6978 - val_loss: 4.2665 - val_accuracy: 1.0000
```

```
Epoch 4/20
9/9 [==============================] - 0s 15ms/step - loss: 4.0524 - accuracy:
0.7507 - val_loss: 3.1297 - val_accuracy: 1.0000
Epoch 5/20
9/9 [==============================] - 0s 15ms/step - loss: 2.9697 - accuracy:
0.8656 - val_loss: 2.3152 - val_accuracy: 1.0000
Epoch 6/20
9/9 [==============================] - 0s 15ms/step - loss: 2.1944 - accuracy:
0.9409 - val_loss: 1.7253 - val_accuracy: 1.0000
Epoch 7/20
9/9 [==============================] - 0s 15ms/step - loss: 1.6472 - accuracy:
0.9461 - val_loss: 1.3020 - val_accuracy: 1.0000
Epoch 8/20
9/9 [==============================] - 0s 15ms/step - loss: 1.2476 - accuracy:
0.9720 - val_loss: 0.9906 - val_accuracy: 1.0000
Epoch 9/20
9/9 [==============================] - 0s 15ms/step - loss: 0.9534 - accuracy:
0.9965 - val_loss: 0.7618 - val_accuracy: 1.0000
Epoch 10/20
9/9 [==============================] - 0s 15ms/step - loss: 0.7508 - accuracy:
0.9865 - val_loss: 0.5942 - val_accuracy: 1.0000
Epoch 11/20
9/9 [==============================] - 0s 15ms/step - loss: 0.5839 - accuracy:
0.9934 - val_loss: 0.4676 - val_accuracy: 1.0000
Epoch 12/20
9/9 [==============================] - 0s 15ms/step - loss: 0.4894 - accuracy:
0.9978 - val_loss: 0.3732 - val_accuracy: 1.0000
Epoch 13/20
9/9 [==============================] - 0s 15ms/step - loss: 0.3800 - accuracy:
0.9889 - val_loss: 0.3022 - val_accuracy: 1.0000
Epoch 14/20
9/9 [==============================] - 0s 16ms/step - loss: 0.3327 - accuracy:
0.9856 - val_loss: 0.2441 - val_accuracy: 1.0000
Epoch 15/20
9/9 [==============================] - 0s 15ms/step - loss: 0.2700 - accuracy:
0.9889 - val_loss: 0.2021 - val_accuracy: 1.0000
Epoch 16/20
9/9 [==============================] - 0s 15ms/step - loss: 0.2036 - accuracy:
1.0000 - val_loss: 0.1667 - val_accuracy: 1.0000
Epoch 17/20
9/9 [==============================] - 0s 15ms/step - loss: 0.1706 - accuracy:
0.9965 - val_loss: 0.1392 - val_accuracy: 1.0000
Epoch 18/20
9/9 [==============================] - 0s 15ms/step - loss: 0.1437 - accuracy:
```

```
1.0000 - val_loss: 0.1185 - val_accuracy: 1.0000
Epoch 19/20
9/9 [==============================] - 0s 15ms/step - loss: 0.1438 - accuracy:
0.9856 - val_loss: 0.1022 - val_accuracy: 1.0000
Epoch 20/20
9/9 [==============================] - 0s 16ms/step - loss: 0.1145 - accuracy:
0.9965 - val_loss: 0.0878 - val_accuracy: 1.0000
```

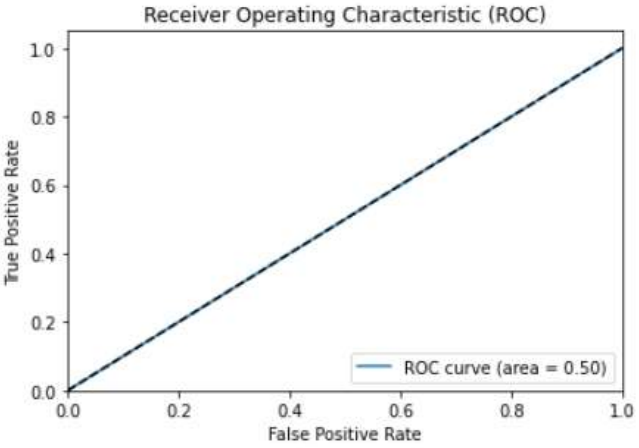


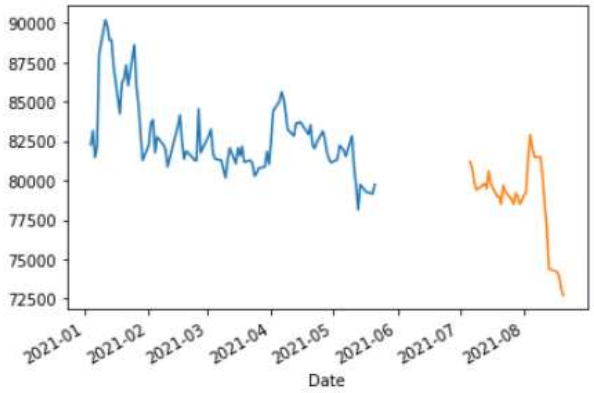


20번 반복 훈련이 시행 되며 loss값이 감소하며 accuracy 또한 일정 수준으로 유지된다. validation의 loss 값도 감소하며 accuracy도 일정 수준으로 유지된다. 샘플수가 적기 때문에 손실과 정확도가 유지됨을 확인 할 수 있다.

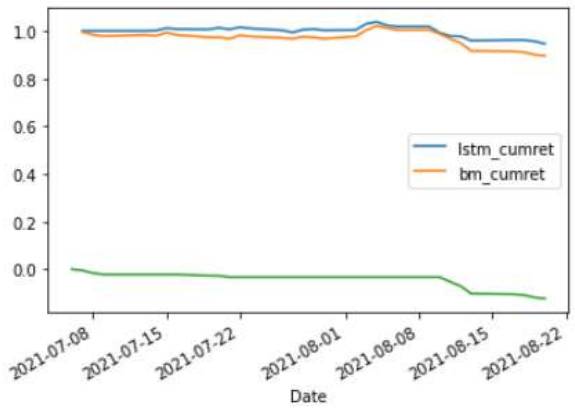
4.5 분석 결과

```
=======
20epochs_10batch
TN: 1
FN: 1
TP: 27
FP: 1
TPR: 0.9642857142857143
FPR: 0.5
accuracy: 0.933
specitivity: 0.5
sensitivity : 0.964
mcc : 0.464
              precision    recall  f1-score   support

           1       0.96      1.00      0.98        27

   micro avg       0.96      1.00      0.98        27
   macro avg       0.96      1.00      0.98        27
weighted avg       0.96      1.00      0.98        27
=======
```






```
ROC AUC 값 : 0.5000
28
28
CAGR :  -57.02 %
Sharpe :  -4.58
VOL :  18.6 %
MDD :  12.3 %
CAGR :  -34.68 %
Sharpe :  -2.89
VOL :  14.82 %
MDD :  12.3 %
```



Train, Validation 데이터도 1에 가까운 높은 Accuracy를 보이며, predict model의 Accuracy도 0.93으로 높게 도출되었다. 투자 성과 분석 지표를 확인하면, 분석기간 동안 연평균복리수익률(CAGR)은 -57% 성장했으며, 실제 주가 그래프(5p 참고)를 확인해도 계속해서 떨어지는 것을 볼 수 있다. 최대낙폭(MDD)이 12.3%, 변동성(VOL)이 18.6%, 위험대비 수익성 지표인 샤프지수가 -2.89%로 좋지 않다는 것을 볼 수 있다.

개인투자자들이 8월 19일 기준으로 2021년 삼성전자 주식을 사들인 자금은 32조원에 달하며, 450만명의 주주수를 돌파하며 개인 지분율이 사상 최대인 13%에 달한다. 명실상부 삼성전자는 '국민주'라고 할 수 있다. 올해 주식을 사들인 대부분의 개인 투자자는 손실이 발생했을 것이며, 이 분석을 사용할 경우 다음날 종가의 대략적인 상승, 하락은 예측하여 참고할 수 있을 것이다.

시행한 분석은 상승, 하락의 예측문제를 해결하는 분석이었으므로, 삼성전자 주가 데이터에서 Train, Validation data는 물론이고 특히 Test data시점은 대부분 주가가 하락했기 때문에 Label 데이터의 불균형이 존재했다. 따라서 Accuracy는 높지만, ROC curve는 낮게 나오는 결과가 도출되었다. 이에 대한 더 정확한 검증을 위해 다른 종목의 데이터를 분석해보거나, 더 긴 시계열을 활용한 분석을 시도해 볼 필요성이 있다. 더하여, 반영하지 못한 Consumer Electronic 섹터 데이터와 변동성 데이터의 추가를 고려하고, 모델의 설명력을 높이기 위한 XAI 모델의 적용을 고안할 예정이다.

5. 느낀점

- 정승연
  빅데이터 캠퍼스를 활용한 '융합' 프로젝트에 참여하게 되며 파이썬 프로그래밍을 처음 시도했습니다. 이전에는 주로 R을 사용하여 머신러닝 분석을 수행했었는데, 딥러닝 모델에 대해 배우고 직접 만들어보며 많이 배울 수 있었습니다. 기존에는 주로 R기본 데이터인 iris데이터를 가지고 데이터 분석을 수행하였습니다. 결측치와 이상치가 없으며 이미 주어져있는 작은 사이즈의 데이터를 사용하다가 데이터 분석을 직접 계획하고 데이터의 범위부터 어떤 데이터를 수집해야 할지, 그 데이터의 결측치와 이상치를 어떻게 처리할지 모든것을 하나하나 결정하는 과정이 생각보다 어렵고 많은 시간이 필요했습니다. ARIMA와 LSTM모델에 둘다 적용되는 데이터를 준비하려고 하다 시간이 많이 지나게 되었고, 결국 두 모델에 대한 데이터와 분석과제를 분리해서 수행한 게 수월성을 더했다고 생각합니다.

  관심있는 금융분야의 데이터를 직접 수집하고, 다루어보고 나름대로 분석해보며 많이 배울 수 있었습니다. 아쉬운 점은 LSTM 모델의 설명력을 높이는 작업을 XAI모델을 통해 시도하고 싶었지만, 하지 못했다는 점입니다. 현재 모델의 성능은 우연하게 높게 측정되었으나, 모델 자체의 성능이 최적화 되지 않았고, 추가적인 데이터를 반영한 학습을 고려하고 있습니다. 아직 모델이 미흡하기 때문에 설명력을 더해도 큰 의미가 없다고 생각해 추후 모델의 성능을 높이고 XAI알고리즘을 적용하여 설명력을 높인 모델을 만들어 투자에 활용하고 싶습니다.

- 최윤석
  이번 방학에 빅데이터 분석에 대해 처음 접하게 되었지만, 강사님이 추천해주신 '누구나 파이썬 통계분석' 책과 강의 그리고 개인적으로 '밑바닥부터 시작하는 딥러닝'을 읽으면서 공부함으로써 기본적인 통계지식과 여러 파이썬 라이브러리를 알게 되었고 딥러닝의 기초를 알 수 있었습니다. DNN, CNN, RNN등 다양한 딥러닝 네트워크가 있음을 알았고, 역전파를 통해 가중치를 변화시켜 학습하는 딥러닝의 방식을 이해할 수 있었고, epochs, 그리고 미니배치등의 용어의 의미도 알게 되었습니다.

  프로젝트에서는 제가 처음 시작하고 공부에 초점을 두고 했기에, 프로젝트에는 거의 기여하지 못하였습니다. 이 부분에 대해선 팀원들에게 정말 죄송하다고 생각합니다.

  하지만 프로젝트의 주제인 ARIMA 모델과 LSTM 모델을 혼자 kaggle에 올라온 코드를 이해하고 한 줄씩 복사-붙여넣기로 돌려보는 과정을 통해 데이터 분석의 흐름을 조금이나마 알 수 있었습니다. 데이터의 전처리부터 모델의 학습 그리고 학습한 모델을 통한 예측과 평가지표를 통해 좋은 모델을 찾는 것 까지의 과정을 천천히 곱씹어 보면서 흐름을 알았습니다. 또한 이런 과정을 통해 데이터 분석에서 자주 쓰이는 파이썬 코드의 의미를 알게 되었습니다. 하나의 LSTM 코드를 제대로 이해해보니 kaggle에 올라온 다른 LSTM 코드도 대부분 비슷한 구조임을 알게 되었습니다.

저는 이번 프로젝트를 통해 빅데이터 분석에 대해 아무것도 모르는 상태에서 kaggle에서 IBM주가를 timestep이 60인 LSTM으로 예측하는 코드의 의미를 이해했습니다. 더 나아가 이 코드를 적절히 바꿔 timestep이 [30,60,90,120,150]일때의 다양한 개별주식의 하루 후의 가격을 예측하는 모델을 만들었고, 정확성을 나타내는 평가지표로 rmse,mape,mae를 사용해 5가지의 timestep에 대해 3*5의 표로 리턴하는 코드로 바꿔보았습니다. 이번 프로젝트를 통해 개인적으로 많이 성장할 수 있었고, 앞으로도 LSTM이나 GRU를 통해 하루 후의 주가가 아닌 7일, 30일, 90일 뒤의 주가를 예측하는 모델을 만들어 실제로 장단기 투자에 사용해보고 싶습니다. 또한 이번에 주가데이터를 다루면서 앞으로 주가데이터를 편하게 다루기 위해 주가데이터가 모여있는 개인 DB의 구축과 관리도 한번 배워보고 싶다고 생각하게 되었습니다.

- 허상범

  평소에 경제, 금융 데이터 분석을 종종 해왔지만, 단순한 기술통계량 분석을 넘어 기계학습 알고리즘을 활용한 예측 분석까지 진행해보고 싶다는 생각을 많이 했었습니다. 그래서 팀장님과 함께 몇차례 논의를 거쳐 심플한 예측 프로젝트를 설정해보았는데, 결과적으로 제가 목표했던 최소한의 학습 및 실습을 경험할 수 있어 매우 좋았습니다. 무엇보다 저는 전체 코스를 잘 따라가진 못했지만, 강사님께서 준비하신 내용의 퀄리티가 매우 높다고 생각했습니다.

  팀 프로젝트의 경우, 개인적으로 특정 콘텐츠를 체화시키는 속도가 빠르지 않은 편이라서, 딥러닝(LSTM)까지 모델링을 해보고자 했지만 ARIMA 모델을 구체적으로 이해하는데에 많은 시간을 쏟았습니다. 이에 따라 딥러닝 모델링은 구체적으로 작업하지 못한 것이 조금 아쉬웠지만, 강사님의 딥러닝 기초 이론 강의와 관련 서적을 틈틈히 읽어보면서, RNN/CNN에 대한 개념적 이해와 인공신경망이 어떻게 동작하게 되는 것인지 등 기본적인 동작 구조나 용어는 익힐 수 있었습니다. 추후에 딥러닝 모델링을 할 때 배경지식이 부족할 걱정은 없을 것이라고 생각합니다!

  제가 집중했던 ARIMA 모델에 대해 학습한 점을 조금 요약하자면, 일반적인 기계학습 프로젝트에서는 N개의 변수를 학습시켜 target을 예측하는 것이 주된 프로세스라고 알고 있었습니다. 그러나 ARIMA의 경우 변수 1개의 추이만을 예측하는 모델임을 명확히 인지하게 되었고, 특정 변수의 시계열 추이만을 예측하고자 할 때 사용하면 유용한 툴이 될 수 있음을 이해했습니다.

  그리고 가장 중요한 것은 ARIMA의 모수인 (p,d,q)의 의미를 수학적으로 명확하게 이해하는 것이었습니다. 파라미터에 대한 이해가 없으면 이 모델로 예측하는 결과의 의미가 무엇인지, 왜 이런 결과가 나온 것인지 제대로 알 수가 없기 때문입니다. 따라서 모델을 단순히 코딩하고 끝내는 것이 아니라, 앞으로도 해당 라이브러리가 동작하는 원리를 최대한 이해해보려고 노력해야겠다는 생각이 강하게 들었습니다. 전반적으로 머신러닝에 대한 자신감을 얻을 수 있었던 좋은 기회였고, 이런 기회를 만들어주신 학교와 팀원분들께 진심으로 감사드립니다 !