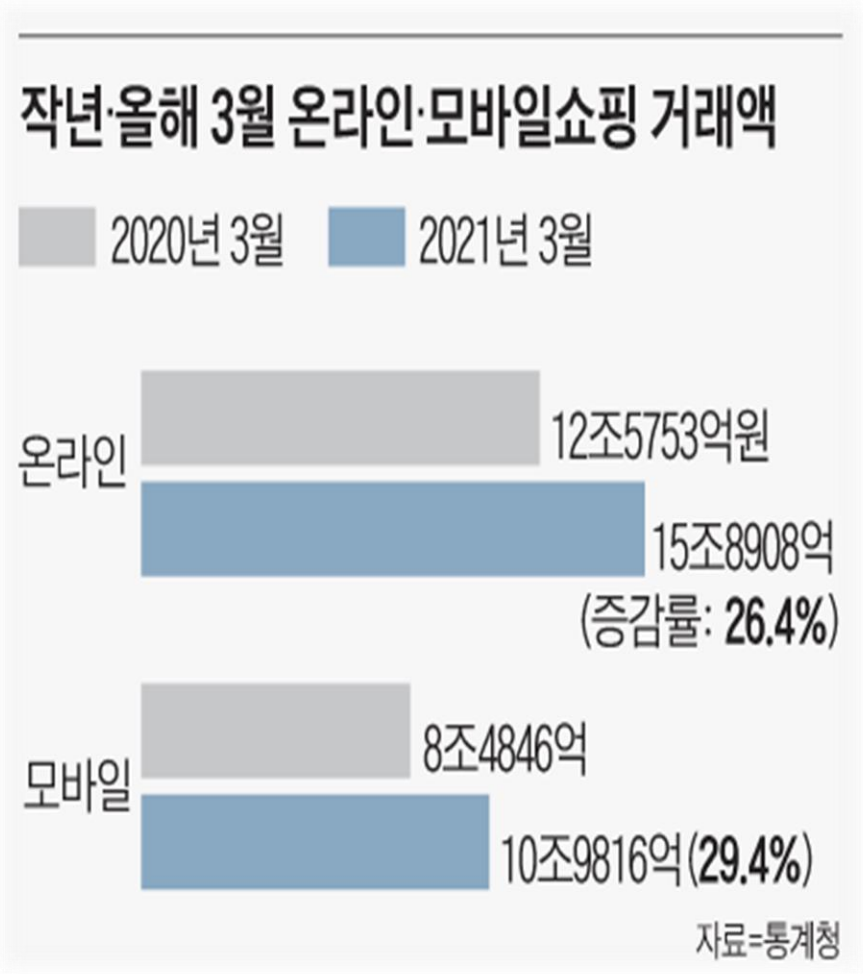
RNN 기반 주가 예측 모델을 통해 OTT, 이커머스 분야 대상으로 투자 기업 선정

비즈니스 인포매틱스 석사 1기 이현태

Table of Contents

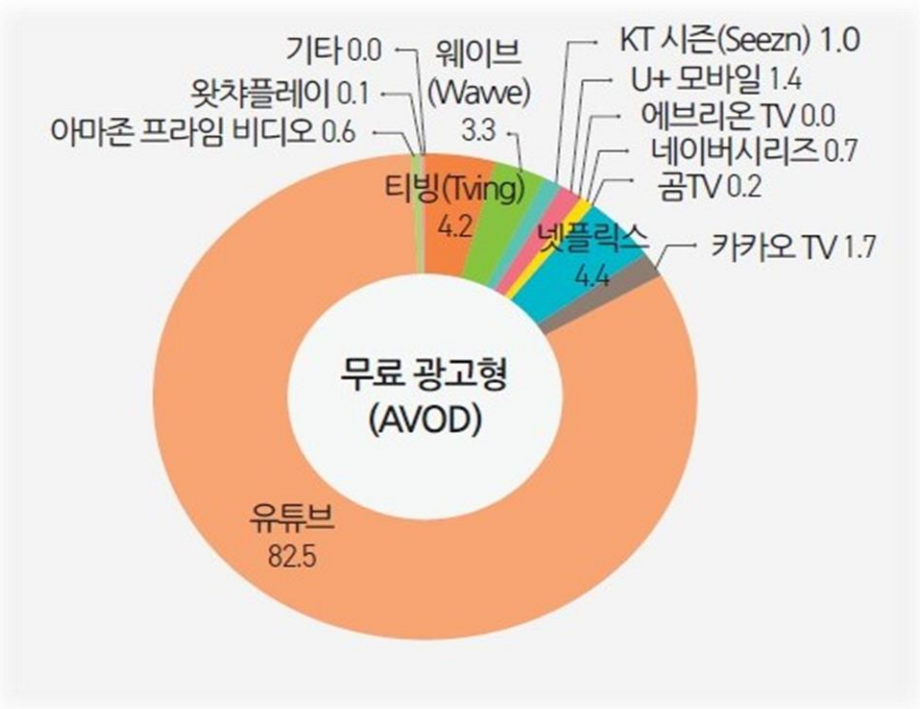
Introduction

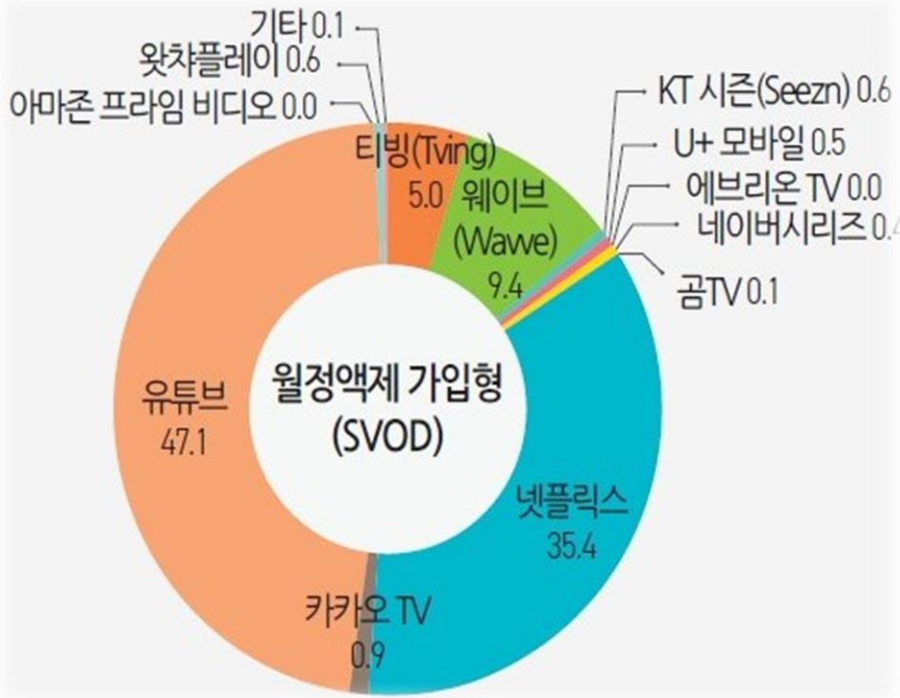
- 최근 코로나 상황으로 인해 언택트 사회로 급변하게 되면서 비대면 사업이 주목 받게 되었다
- 특히 디지털 미디어 사업 산업인 OTT와 전자상거래 이용률이 크게 증가했다. (정보통신정책연구원, 2021) (통계청, 2021)



Introduction

- 따라서 많은 국내 디지털 미디어 기업들이 OTT와 이커머스에 뛰어들고 있다.
- 하지만 무료 OTT (AVOD) 같은 경우에는 여전히 유튜브가 절대다수를 차지 (정보통신정책연구원, 2021)
- 유료 OTT(SVOD)도 마찬가지로 유튜브와 넷플릭스가 대부분을 차지 (정보통신정책연구원, 2021)
- 따라서 국내 OTT 기업은 여전히 성장하고 있는 단계





Introduction

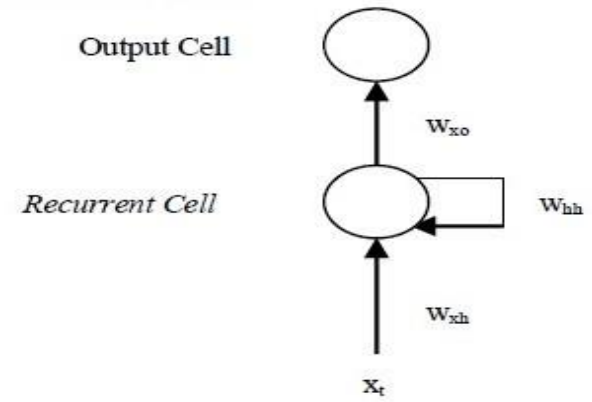
- OTT, 이커머스 분야에 뛰어드는 국내 기업의 주가 시계열 데이터를 분석해 성장 가능성 있는 기업 상대로 투자 대상 선정
- 종가의 시퀀스를 활용해 향후 주가 예측하기 위해 LSTM 레이어를 활용
- 분석 기업 목록
  - CJ ENM (티빙, cj오쇼핑), 카카오 (카카오TV, 카카오커머스), 네이버(티빙, 네이버쇼핑), SKT(웨이브, 11번가), KT(Seezn, 쇼핑라이브), LG유플러스(넷플릭스, 디즈니 플러스 제휴, 디버), 아프리카TV(아프리카TV샵)
- Research Question
  - OTT, 이커머스 분야로 사업을 확장하는 미디어, IT, 통신사 중 과연 어느 업종이 코로나 이후 좋은 성과를 보일 것인가?

Background and related works

- 순환 신경망 (RNN)
  - 내부의 순환 구조를 활용하기 때문에 순차적인 데이터에 많이 활용됨
  - 현재의 출력 값을 만들 때 이전 타임 스텝에 계산된 출력 값이 가중치와 곱해져 다시 활용됨
  - 활성화 함수는 보통 하이퍼볼릭 탄젠트(tanh) 함수가 사용됨

(빅데이터 공부 한 걸음: RNN(순환 신경망)이란?)

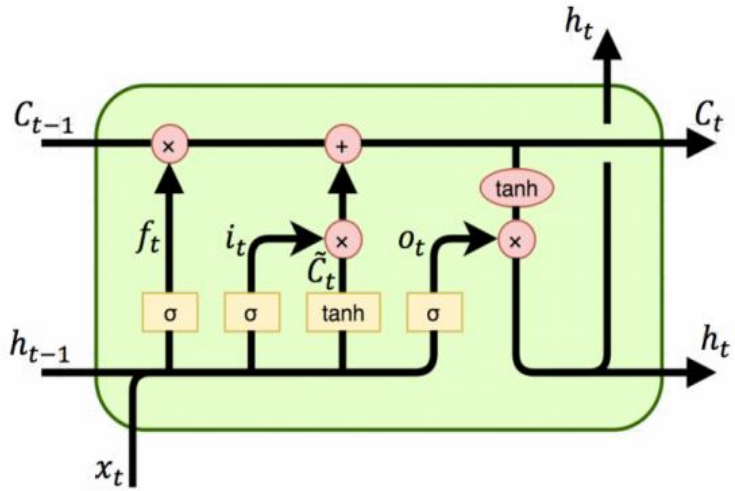
$$h_t = tanh\,(h_{t-1}W_h + x_tW_x + b)$$



- LSTM (Long-Short Term Memory)
  - LSTM 셀에서는 셀 상태인 C_t를 통해 셀 상태를 보존하면서 장기의존성 문제를 해결

(김환희 2020, 183)

  - Memory cell, Input gate, Forget gate, Output gate로 구성 (Hochreiter et al 1997, 1743–1748)



Background and related works

- 주가 예측 모델을 만들기 위해 시퀀스 데이터에 많이 쓰이는 LSTM이 많이 쓰여지고 있음
  - RNN, LSTM, CNN 모델을 활용해 IT 분야에 있는 두 회사와 제약 분야의 한 회사에 대해서 분 단위로 주가 예측 (Selvin et al, 2017)
  - Selvin et al은 단기 미래 예측을 위해 sliding window approach를 활용
  - Window size는 100분으로 설정해 향후 10분 주가를 예측
  - 이 Term-paper의 실험도 비슷하게 window size를 20일로 정하고 향후 하루 종가를 예측
  - RNN 모델로 주가 예측을 하기 위해 쓰이는 최적의 학습 데이터 주기 (look-back period)를 분석 (Saud et al 2020)
  - Saud et al은 LSTM의 적합한 look-back period는 5일 이하라고 주장
  - 이 실험에서는 20일이 제일 정확한 예측을 보여줌

Methodology

- 데이터 수집 및 전처리
  - 각 기업 별로 2010년 1월 1일부터 2021년 6월 18일까지의 종가 데이터를 FnGuide를 통해 수집
  - 종가 값을 Scikit-learn의 MinMaxScaler를 활용해 종가 데이터를 정규화 (0 ~1 사이의 값을 가지도록 함)

In [11]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scale_cols = ['시가', '고가', '저가', '종가', '거래량']
df_scaled = scaler.fit_transform(kakao[scale_cols])

df_scaled = pd.DataFrame(df_scaled)
df_scaled.columns = scale_cols

print(df_scaled)
```

```
            시가        고가        저가        종가       거래량
0     0.011371  0.015344  0.009925  0.009746  0.015265
1     0.011371  0.015344  0.009925  0.009746  0.015265
2     0.011371  0.015344  0.009925  0.009746  0.015265
3     0.009746  0.017161  0.009723  0.016041  0.009289
4     0.015228  0.019382  0.016002  0.017259  0.007254
...        ...       ...       ...       ...       ...
4182  0.143147  0.156067  0.145230  0.156345  0.364615
4183  0.158376  0.159095  0.154345  0.160406  0.289239
4184  0.159391  0.162124  0.158396  0.157360  0.291923
4185  0.155330  0.169190  0.154345  0.167513  0.316032
4186  0.171574  0.185342  0.172574  0.181726  0.450268

[4187 rows x 5 columns]
```

Methodology

- 데이터 준비 및 전처리
  - Scikit-learn을 활용해 데이터를 훈련 세트와 테스트 세트를 각각 8:2 비율로 나눔

In [14]:
```
from sklearn.model_selection import train_test_split
```

In [15]:
```
feature_cols = ['시가', '고가', '저가', '종가', '거래량']
label_cols = ['종가']

feature = df[feature_cols]
label = df[label_cols]
x_train, x_test, y_train, y_test = train_test_split(feature, label, test_size=0.2, random_state=0, shuffle=False)
```

Methodology

- 데이터 준비 및 전처리
  - Window size를 정하여 학습 데이터를 생성할 함수를 사용 (몇일 종가 데이터를 기반으로 다음날 종가 예측할지 지정)
  - 20일 기반으로 하루를 예측하기 때문에 window_size=20으로 설정
  - 학습 때 사용될 batch size는 32로 지정

In [20]:
```
def windowed_dataset(series, window_size, batch_size, shuffle):
    series = tf.expand_dims(series, axis=-1)
    ds = tf.data.Dataset.from_tensor_slices(series)
    ds = ds.window(window_size + 1, shift=1, drop_remainder=True)
    ds = ds.flat_map(lambda w: w.batch(window_size + 1))
    if shuffle:
        ds = ds.shuffle(1000)
    ds = ds.map(lambda w: (w[:-1], w[-1]))
    return ds.batch(batch_size).prefetch(1)
```

In [21]:
```
WINDOW_SIZE=20
BATCH_SIZE=32
```

In [22]:
```
train_data = windowed_dataset(y_train, WINDOW_SIZE, BATCH_SIZE, True)
test_data = windowed_dataset(y_test, WINDOW_SIZE, BATCH_SIZE, False)
```

Methodology

- 모델 생성
  - Conv1D을 활용해 1차원 시계열 데이터에 대한 feature map 생성
  - Input shape은 window_size에 따라 설정 (20, 1)
  - 그 다음에는 LSTM 알고리즘 적용

In [25]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Conv1D, Lambda
from tensorflow.keras.losses import Huber
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint


model = Sequential([
    # 1차원 feature map 생성
    Conv1D(filters=32, kernel_size=5,
           padding="causal",
           activation="relu",
           input_shape=[WINDOW_SIZE, 1]),
    # LSTM
    LSTM(16, activation='tanh'),
    Dense(16, activation="relu"),
    Dense(1),
])
```

Methodology

- 모델 생성
  - 손실함수로 Huber 손실함수 사용하고 최적화 함수로는 Adam optimizer 사용

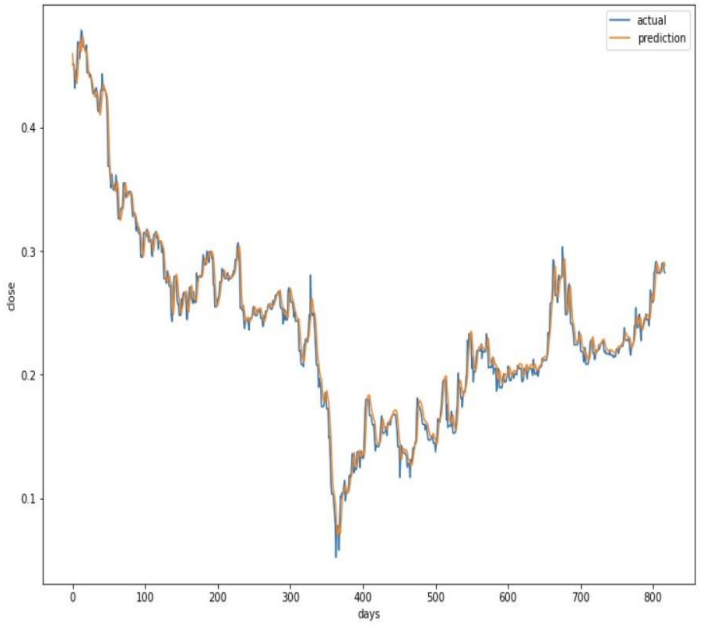
In [26]:

```
loss = Huber()
optimizer = Adam(0.0005)
model.compile(loss=Huber(), optimizer=optimizer, metrics=['mse'])
model.summary()
```

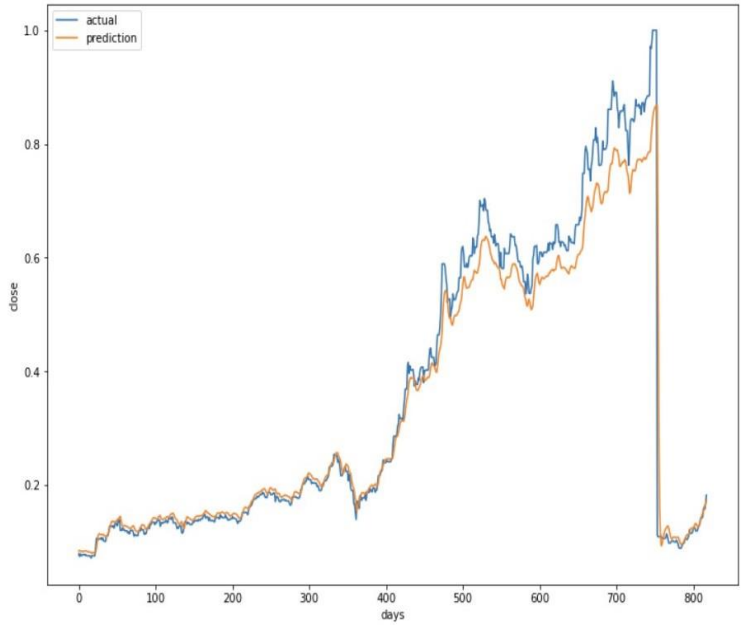
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 20, 32)            192
_________________________________________________________________
lstm (LSTM)                  (None, 16)                3136
_________________________________________________________________
dense (Dense)                (None, 16)                272
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 17
=================================================================
Total params: 3,617
Trainable params: 3,617
Non-trainable params: 0
_________________________________________________________________
```

Results

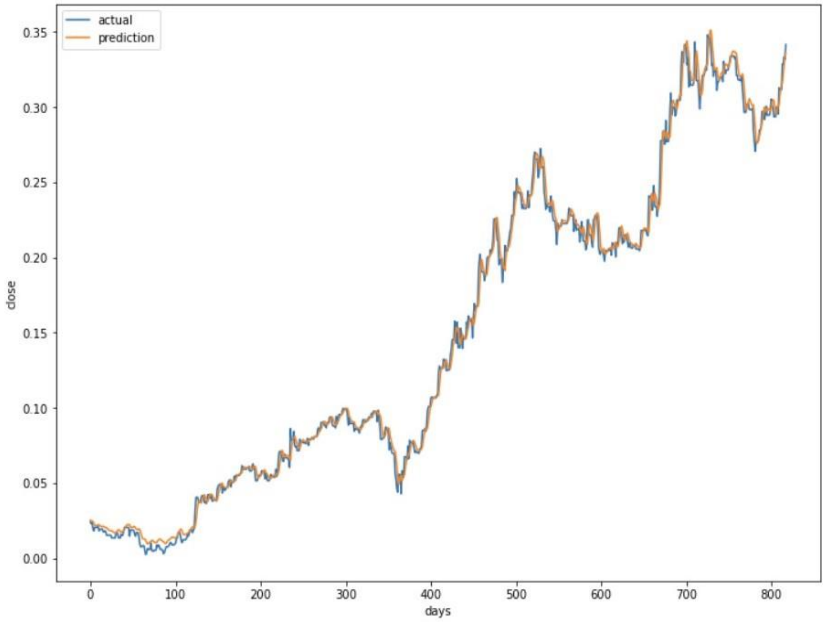
- Epoch는 50으로 설정하고 학습을 실행한 결과, 테스트 데이터에서 모두 실제 데이터와 비슷한 추세로 맞춘 결과를 보여줌



CJ ENM test data 예측

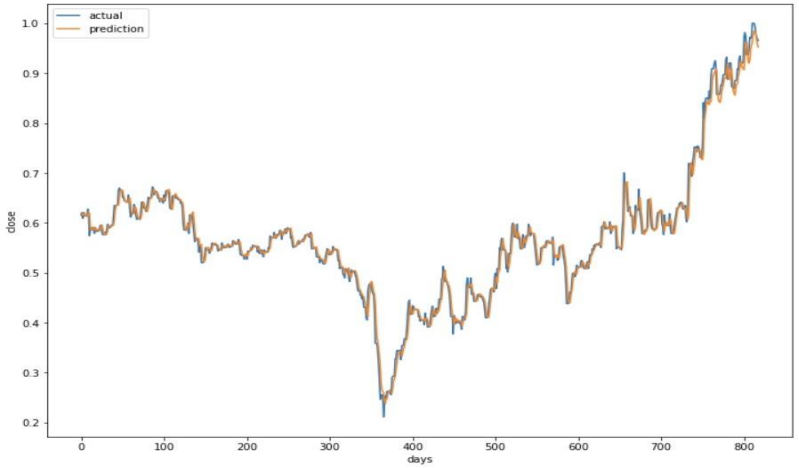


카카오 test data 예측

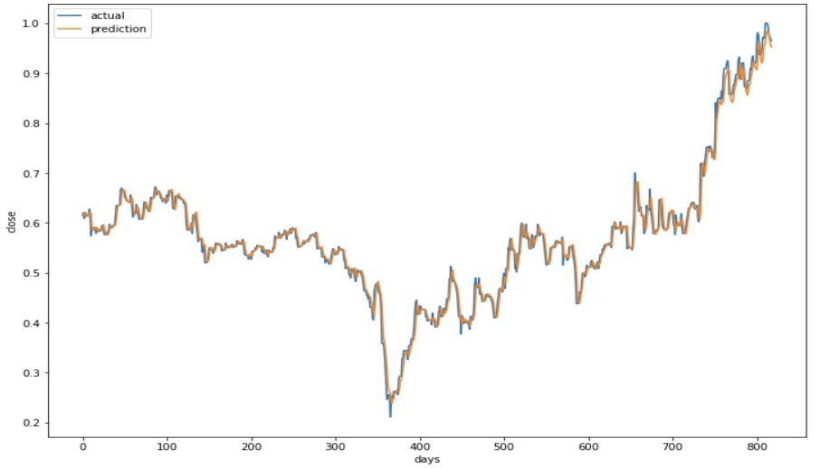


네이버 test data 예측

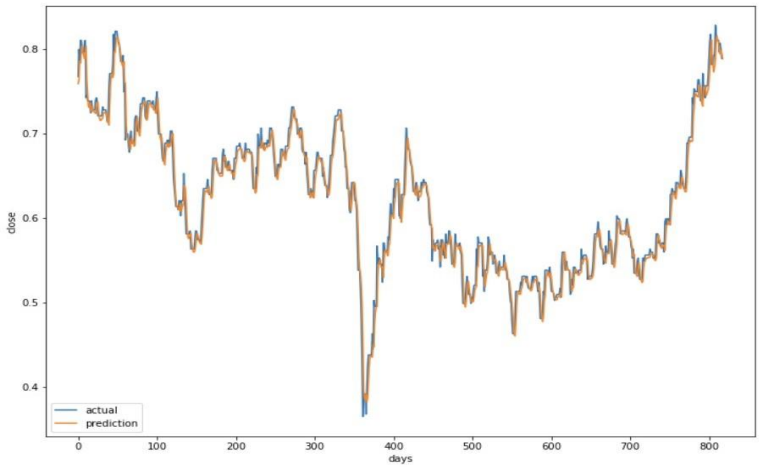
Results



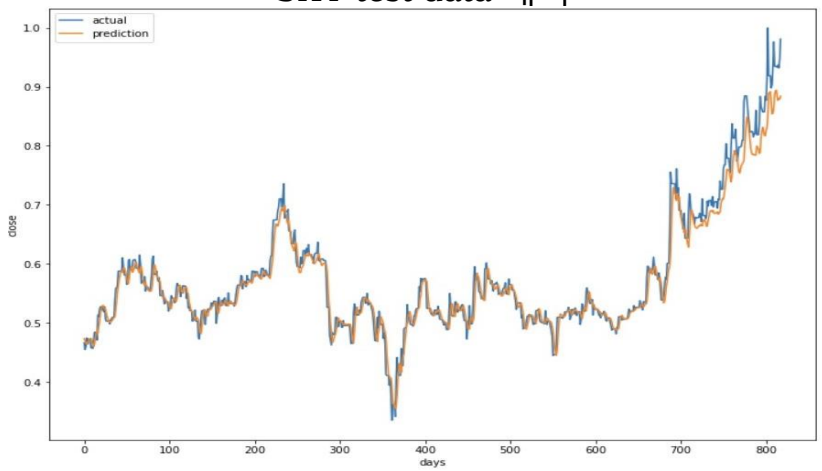
KT test data 예측



SKT test data 예측



LG U+ test data 예측



아프리카TV test data 예측

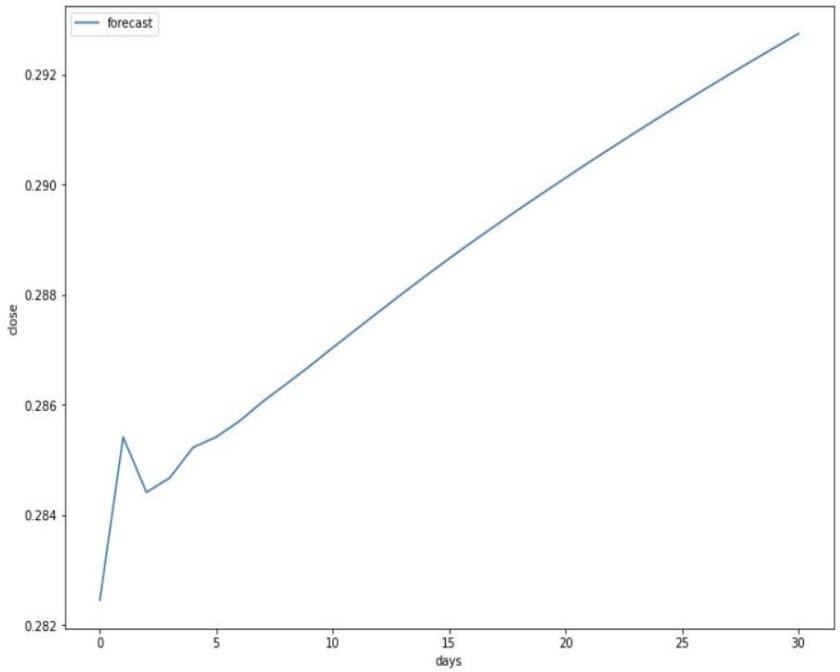
Results

- 이렇게 학습된 모델을 토대로 미래 주가 예측
- 다음날 종가 예측을 위해 마지막 20일 종가 데이터를 model.predict() 함수에 넣어준다
- 그 다음날 종가 예측을 위해 19일 데이터와 전날 예측된 종가 데이터를 넣어주는 식으로 1달 주가 예측

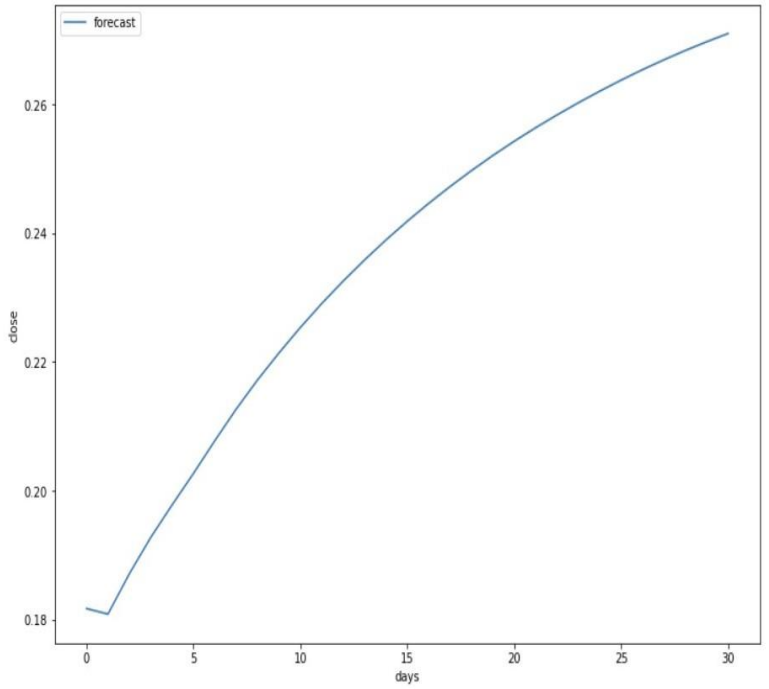
In [40]:

```
close_data = close_data.reshape((-1))

look_back = 20

def predict(num_prediction, model):
    prediction_list = close_data[-look_back:]

    for _ in range(num_prediction):
        x = prediction_list[-look_back:]
        x = x.reshape((1, look_back, 1))
        out = model.predict(x)[0][0]
        prediction_list = np.append(prediction_list, out)
    prediction_list = prediction_list[look_back-1:]

    return prediction_list

def predict_dates(num_prediction):
    last_date = df['일자'].values[-1]
    prediction_dates = pd.date_range(last_date, periods=num_prediction+1).tolist()
    return prediction_dates

num_prediction = 30
forecast = predict(num_prediction, model)
forecast_dates = predict_dates(num_prediction)
```

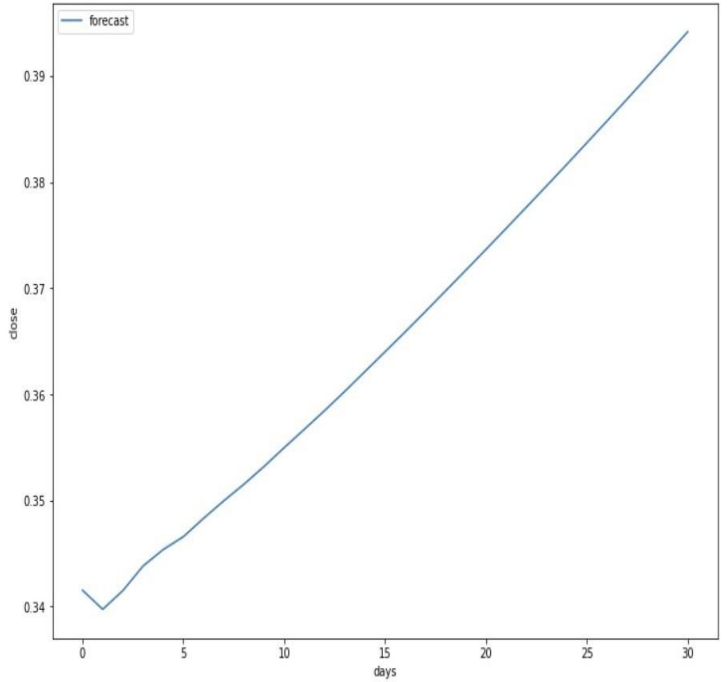
Results



CJ ENM 1달 종가 예측

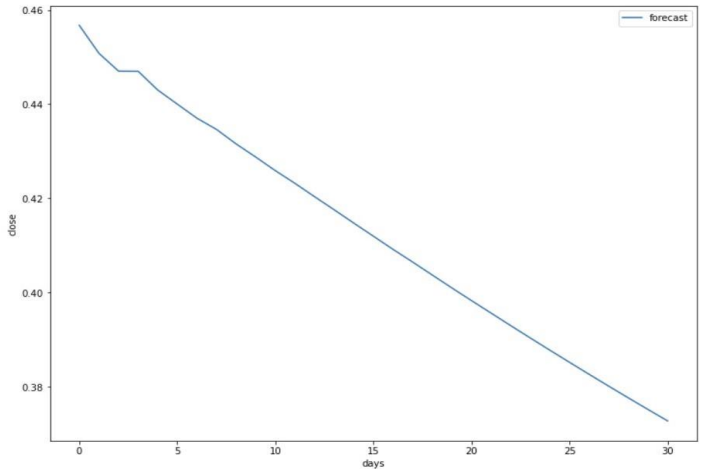


카카오 1달 종가 예측

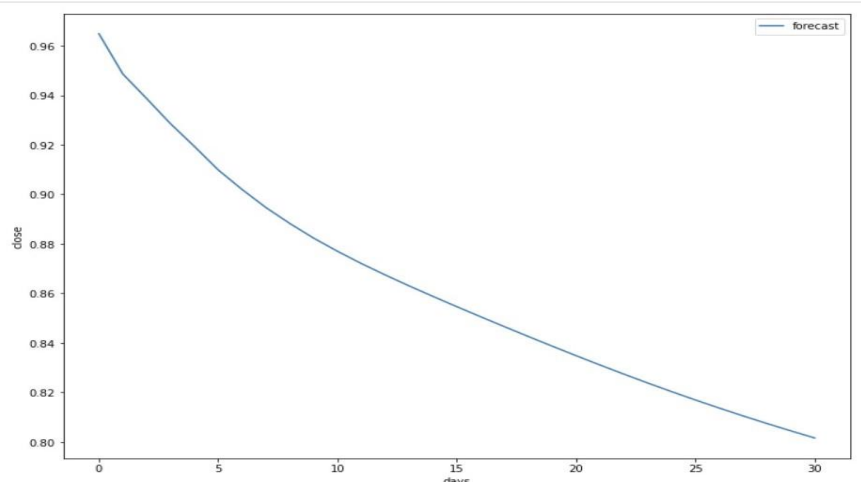


네이버 1달 종가 예측

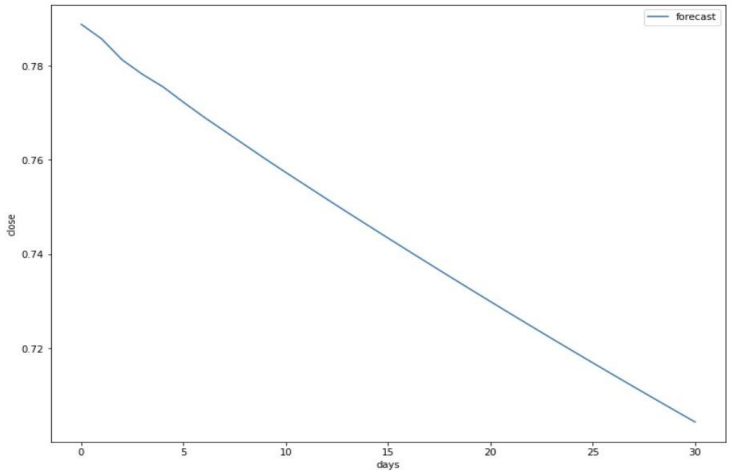
Results



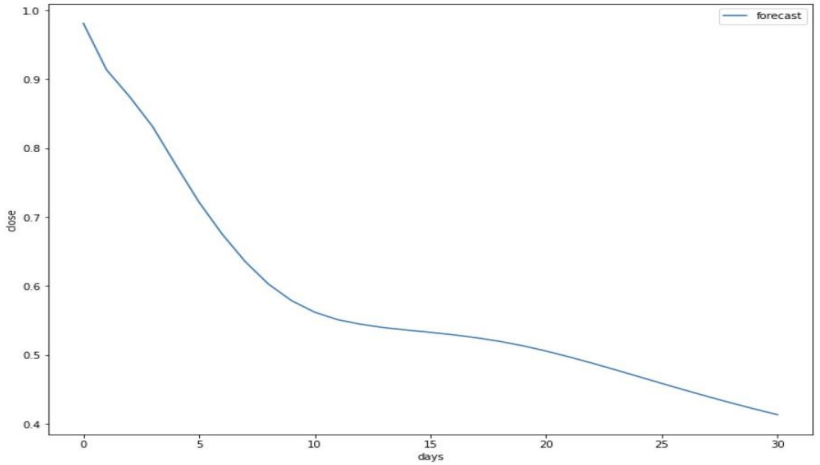
KT 1달 종가 예측



SKT 1달 종가 예측



LG U+ 1달 종가 예측



아프리카TV 1달 종가 예측

Conclusions and limitations

- Conclusions
  - 예측한 데이터를 토대로 분석한다면 IT 회사 (네이버, 카카오)와 미디어 회사 (CJ ENM)가 상승 추세를 보여주고 있음
  - 네이버와 CJ ENM은 티빙이라는 플랫폼을 통해 OTT 사업 제휴를 맺고 있어 강세를 보인다고 분석할 수 있음
  - 카카오, 네이버는 다른 기업보다 IT 인프라를 갖추고 있고 CJ ENM은 방대한 미디어 콘텐츠를 소유하고 있음
  - 따라서 향후 OTT , 이커머스 분야에서 성공하기 위해서는 IT 인프라와 소유된 콘텐츠가 중요한 요소라고 추측할 수 있음
- Limitations
  - LSTM 레이어를 통해 향후 5년을 예측하는데 한계가 있기 때문에 향후 1달만 종가 예측 실시
  - 과거 종가 데이터를 통해 하루를 예측하기 때문에 단기적인 예측은 가능하나 그 이상은 실제 데이터가 주어지지 않기 때문에 정확한 예측이 불가능
  - 결국 실제 과거 데이터가 있어야 정확한 미래 예측이 가능
  - 1달 예측도 상승 혹은 하락 추세만 보여주고 있음
  - 종가 외에도 다른 변수가 영향을 줄 수 있기 때문에 정확한 예측은 불가능

Reference

- 김윤화, *OTT(온라인동영상서비스) 유·무료 이용행태 분석*, 정보통신정책연구원, 2021.
- 통계청, *2021년 4월 온라인쇼핑 동향*, 2021.
- “빅데이터 공부 한 걸음: RNN(순환 신경망)이란?”, *골든플래닛*, *http://www.goldenplanet.co.kr/blog/2021/04/27/%EB%B9%85%EB%8D%B0%EC%9D%B4%ED%84%B0-%EA%B3%B5%EB%B6%80-%ED%95%9C-%EA%B1%B8%EC%9D%8C-rnn%EC%88%9C%ED%99%98-%EC%8B%A0%EA%B2%BD%EB%A7%9D%EC%9D%B4%EB%9E%80/*
- 김환희, *시작하세요! 텐서플로 2.0 프로그래밍*. 파주:위키북스, 2020, 183~184.
- Sepp Hochreiter, Jürgen Schmidhuber; Long Short-Term Memory. Neural Comput 1997; 9 (8): 1735–1780. doi: https://doi.org/10.1162/neco.1997.9.8.1735
- S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017, pp. 1643-1647, doi: 10.1109/ICACCI.2017.8126078.
- Arjun Singh Saud, Subarna Shakya, Analysis of look back period for stock price prediction with RNN variants: A case study on banking sector of NEPSE, Procedia Computer Science, Volume 167, 2020, Pages 788-798, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.03.419.
- “LSTM과 FinanceDataReader API를 활용한 삼성전자 주가 예측”,*테디노트*, https://teddylee777.github.io/tensorflow/lstm-stock-forecast
- “Predicting Sequential Data using LSTM: An Introduction”, *towards data science*, https://towardsdatascience.com/time-series-forecasting-with-recurrent-neural-networks-74674e289816

Thank You