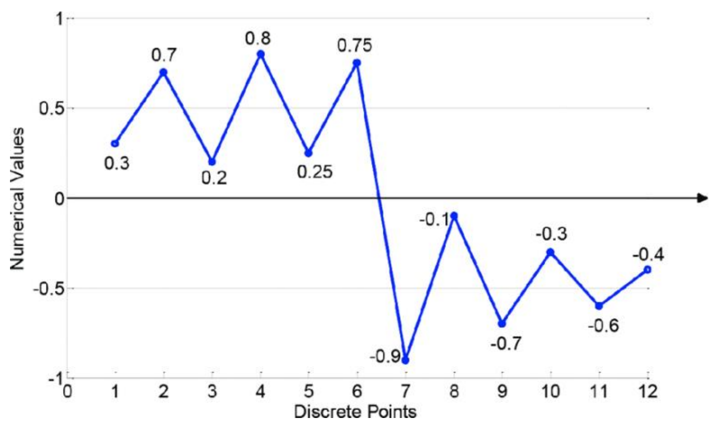
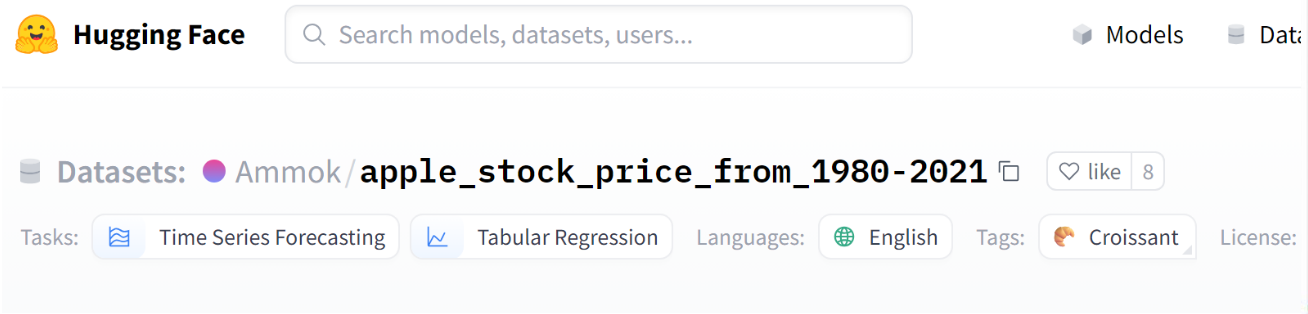
LSTM을 이용한 Apple 주가 예측 모델

목차

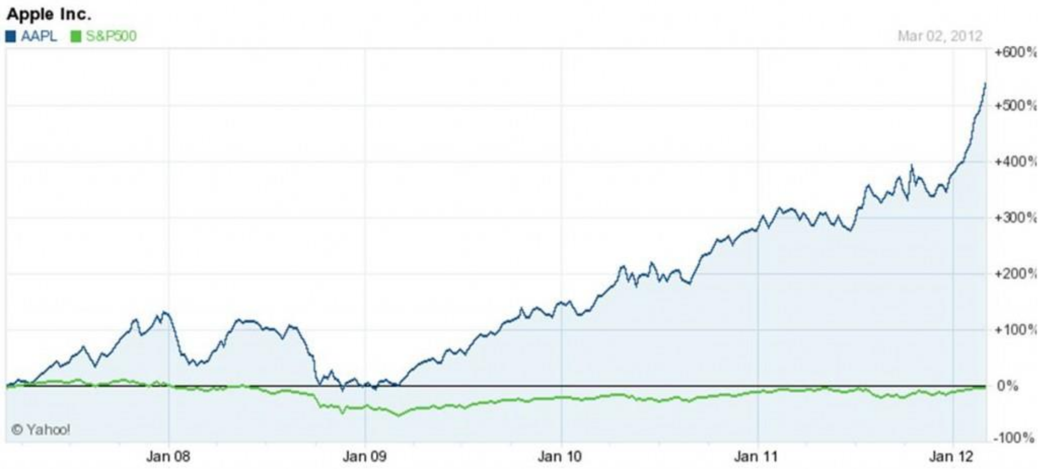
- 모델링 배경
- Apple_Stock_Price Data
- 모델 구성

모델링 배경





https://huggingface.co/datasets/Ammok/apple_stock_price_from_1980-2021



- **시계열 데이터(Time Series Data)**

시간의 흐름에 따라 일정 시간 동안 수집되며, 순서대로 관측되는 데이터셋

- **연속 시계열(Continuous Time Series)**

연속적으로 생성되는 시계열 자료로서, 관측 값들이 연속적으로 연결된 형태의 자료

Apple_Stock_Price Data

Datasets: Ammok / apple_stock_price_from_1980-2021 like 8

Dataset card Viewer Files Community

Split (1)
train · 10.5k rows

API

Search this dataset

Date	Open	High	Low	Close	Adj Close	Volume
string · *lengths*	float64	float64	float64	float64	float64	int64
10 10	0.05 183	0.05 183	0.05 179	0.05 182	0.04 182	0 7.42B
1980-12-12	0.128348	0.128906	0.128348	0.128348	0.100178	469,033,600
1980-12-15	0.12221	0.12221	0.121652	0.121652	0.094952	175,884,800
1980-12-16	0.113281	0.113281	0.112723	0.112723	0.087983	105,728,000
1980-12-17	0.115513	0.116071	0.115513	0.115513	0.09016	86,441,600
1980-12-18	0.118862	0.11942	0.118862	0.118862	0.092774	73,449,600
2022-06-08	148.580002	149.869995	147.460007	147.960007	147.960007	53,950,200
2022-06-09	147.080002	147.949997	142.529999	142.639999	142.639999	69,473,000
2022-06-10	140.279999	140.759995	137.059998	137.130005	137.130005	91,437,900
2022-06-13	132.869995	135.199997	131.440002	131.880005	131.880005	122,207,100
2022-06-14	133.130005	133.889999	131.479996	132.759995	132.759995	84,784,300
2022-06-15	134.289993	137.339996	132.160004	135.429993	135.429993	91,533,000
2022-06-16	132.080002	132.389999	129.039993	130.059998	130.059998	108,123,900
2022-06-17	130.070007	133.080002	129.809998	131.559998	131.559998	134,118,500

Apple_Stock_Price Data

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10468 entries, 0 to 10467
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       10468 non-null  object 
 1   Open       10468 non-null  float64
 2   High       10468 non-null  float64
 3   Low        10468 non-null  float64
 4   Close      10468 non-null  float64
 5   Adj Close  10468 non-null  float64
 6   Volume     10468 non-null  int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 572.6+ KB
```

- 10468개의 데이터
- Date column dtype : object
- Open(시가) : 주식 시장이 개장할 때 주식 가격
- High(고가) : 주식이 특정 기간 동안 거래된 최고 가격
- Low(저가) : 주식이 특정 기간 동안 거래된 최저 가격
- Close(종가) : 주식 시장이 폐장할 때 주식 가격
- Adj Close(조정 종가) : 배당, 주식 분할 등 이벤트를 반영하여 조정된 종가
- Volume(거래량) : 특정 기간 동안 거래된 주식의 총 수량

모델 구성

```
[1] pip install datasets
```

숨겨진 출력 표시

```
from datasets import load_dataset

dataset = load_dataset("Ammok/apple_stock_price_from_1980-2021")
```

숨겨진 출력 표시

```
[3] import pandas as pd
    import numpy as np
    from tensorflow import keras
    from tensorflow.keras.optimizers import Adam, SGD
```

```
[4] dataset
```

```
DatasetDict({
    train: Dataset({
        features: ['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'],
        num_rows: 10468
    })
})
```

```
dataset.set_format(type='pandas')

df = dataset['train'][:]
print(df)
```

```
             Date        Open        High         Low       Close   Adj Close  ₩
0      1980-12-12    0.128348    0.128906    0.128348    0.128348    0.100178
1      1980-12-15    0.122210    0.122210    0.121652    0.121652    0.094952
2      1980-12-16    0.113281    0.113281    0.112723    0.112723    0.087983
3      1980-12-17    0.115513    0.116071    0.115513    0.115513    0.090160
4      1980-12-18    0.118862    0.119420    0.118862    0.118862    0.092774
...           ...         ...         ...         ...         ...         ...
10463  2022-06-13  132.869995  135.199997  131.440002  131.880005  131.880005
10464  2022-06-14  133.130005  133.889999  131.479996  132.759995  132.759995
10465  2022-06-15  134.289993  137.339996  132.160004  135.429993  135.429993
10466  2022-06-16  132.080002  132.389999  129.039993  130.059998  130.059998
10467  2022-06-17  130.070007  133.080002  129.809998  131.559998  131.559998

          Volume
0      469033600
1      175884800
2      105728000
3       86441600
4       73449600
...          ...
10463  122207100
10464   84784300
10465   91533000
10466  108123900
10467  134118500

[10468 rows x 7 columns]
```

모델 구성

```
[6] df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1980-12-12	0.128348	0.128906	0.128348	0.128348	0.100178	469033600
1	1980-12-15	0.122210	0.122210	0.121652	0.121652	0.094952	175884800
2	1980-12-16	0.113281	0.113281	0.112723	0.112723	0.087983	105728000
3	1980-12-17	0.115513	0.116071	0.115513	0.115513	0.090160	86441600
4	1980-12-18	0.118862	0.119420	0.118862	0.118862	0.092774	73449600

모델 구성

[8]
```
def convert_date_format(date_str):
  parts = date_str.split("-")
  return "".join(parts)

df['Date'] = df['Date'].apply(convert_date_format)
```

```
df
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	19801212	0.128348	0.128906	0.128348	0.128348	0.100178	469033600
1	19801215	0.122210	0.122210	0.121652	0.121652	0.094952	175884800
2	19801216	0.113281	0.113281	0.112723	0.112723	0.087983	105728000
3	19801217	0.115513	0.116071	0.115513	0.115513	0.090160	86441600

[13]
```
df['Date'] = pd.to_datetime(df['Date'])
```

[14]
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10468 entries, 0 to 10467
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       10468 non-null  datetime64[ns]
 1   Open       10468 non-null  float64
 2   High       10468 non-null  float64
 3   Low        10468 non-null  float64
 4   Close      10468 non-null  float64
 5   Adj Close  10468 non-null  float64
 6   Volume     10468 non-null  int64
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 572.6 KB
```

모델 구성

```
df
```

	Open	High	Low	Close	Volume	Close2
0	0.128348	0.128906	0.128348	0.128348	469033600	0.128348
1	0.122210	0.122210	0.121652	0.121652	175884800	0.121652
2	0.113281	0.113281	0.112723	0.112723	105728000	0.112723
3	0.115513	0.116071	0.115513	0.115513	86441600	0.115513
4	0.118862	0.119420	0.118862	0.118862	73449600	0.118862
...	...	...	...	...	...	...
10463	132.869995	135.199997	131.440002	131.880005	122207100	131.880005
10464	133.130005	133.889999	131.479996	132.759995	84784300	132.759995
10465	134.289993	137.339996	132.160004	135.429993	91533000	135.429993
10466	132.080002	132.389999	129.039993	130.059998	108123900	130.059998
10467	130.070007	133.080002	129.809998	131.559998	134118500	131.559998

모델 구성

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df.iloc[:, 0:5] = scaler.fit_transform(df.iloc[:, 0:5])
```

```
df.describe()
```

	Open	High	Low	Close	Volume	Close2
count	1.046800e+04	10468.000000	1.046800e+04	10468.000000	1.046800e+04	10468.000000
mean	4.344167e-17	0.000000	4.344167e-17	0.000000	2.172083e-17	14.763533
std	1.000048e+00	1.000048	1.000048e+00	1.000048	1.000048e+00	31.929489
min	-4.608932e-01	-0.460605	-4.611365e-01	-0.460863	-9.764575e-01	0.049107
25%	-4.535664e-01	-0.453183	-4.539184e-01	-0.453523	-6.111464e-01	0.283482
50%	-4.475931e-01	-0.447191	-4.479209e-01	-0.447510	-3.325890e-01	0.475446
75%	6.120328e-03	0.004201	3.110250e-03	0.004336	2.353143e-01	14.901964
max	5.260360e+00	5.203807	5.216003e+00	5.238244	2.092755e+01	182.009995

모델 구성

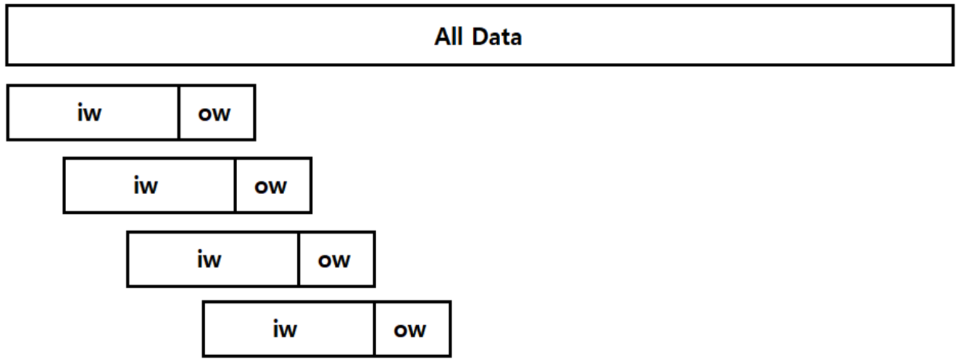
```python
# 20일간의 주가 자료로 그 다음 날 종가를 예측

window_size = 20
x = []
y = []

for i in range(len(df) - window_size):
  temp0 = df.iloc[i:i+window_size, 0:5]
  x.append(temp0)
  temp1 = df.iloc[i+window_size, 5] # 20일 후 종가
  y.append(temp1)
```

```python
x = np.array(x)
y = np.array(y)
```

```python
split_point = begins_2022 - window_size
x_train = x[:split_point]     # (작년말 - 20개)까지의 데이터
x_test  = x[split_point:]     # (작년말 - 20개) 이후의 데이터
y_train = y[:split_point]     # 작년말까지의 주가
y_test  = y[split_point:]     # 올해부터의 주가
```

- **Sliding Window Dataset**

시계열 예측을 위해 데이터의 일정한 길이의 input window, output window를 설정하고, 데이터의 처음 부분부터 끝부분까지 sliding 시켜 데이터셋을 생성



모델 구성

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, shuffle=
```

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

```
(8265, 20, 5) (2067, 20, 5) (116, 20, 5)
(8265,) (2067,) (116,)
```

모델 구성

```
[37] es = keras.callbacks.EarlyStopping(
         patience=10,
         restore_best_weights=True)
```

```
[48] model = keras.Sequential()
     model.add(keras.layers.LSTM(
         48,
         activation='tanh',
         input_shape=(x_train.shape[1],x_train.shape[2])))
     model.add(keras.layers.Dropout(0.4))
     model.add(keras.layers.Dense(
         64,
         activation='relu'))
     model.add(keras.layers.Dropout(0.4))
     model.add(keras.layers.Dense(1))

     model.compile(loss = 'mae',
                   optimizer = Adam(0.01),
                   metrics = 'mae')
     history = model.fit(x_train, y_train,
                         epochs=150,
                         validation_data=(x_val, y_val))
```

```
[49] score = model.evaluate(x_test, y_test, verbose=0)
     print('Test loss:', score[0])
     print('Test accuracy:', score[1])
```

```
Test loss: 5.548529624938965
Test accuracy: 5.548529624938965
```

모델 구성

```
y_pred = model.predict(x_test)
days_2022 = df.index[df.index >= begins_2022]

import matplotlib.pyplot as plt
plt.figure(figsize = (12, 8))
plt.title('Apple Stock 2022')
plt.plot(days_2022, y_test, 'orange', label='ground truth')
plt.plot(days_2022, y_pred, 'skyblue', label='prediction')
plt.ylabel('stock price')
plt.legend(loc='upper right')
```