Stock Price Prediction

Do Duc Quan
Deep Learning for Artificial Intelligence
Instructor: Prof. Dang Huynh

Agenda

Overview

1. Objectives

Our objectives is to, given a stock dataset, create a model that can handle the time series features of stock price.

And having the trained model, we use it to make forecast on the future, give recommendation on trading, and optimize stock portfolio.

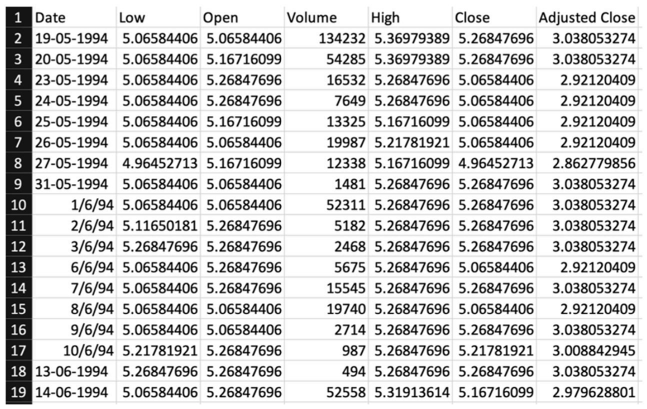
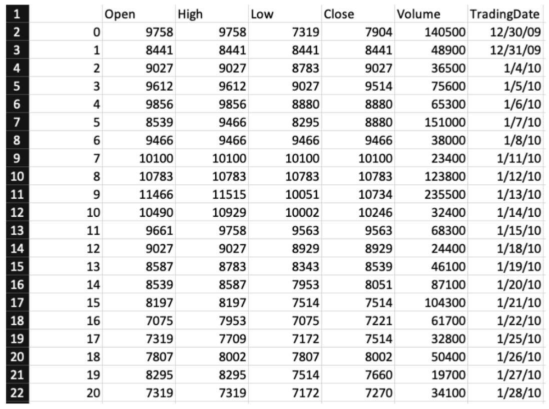
2. Dataset

In this project, we are given the stock price of several companies in Vietnam and global market.

- For Vietnam market, the dataset includes but not limitted to:
    - UPCOM *(Unlisted Public Company Market)*
    - HNX *(Hanoi Stock Exchange)*
    - and HOSE *(HoChiMinh Stock Exchange)*.

- For international market (particularly US market), we have stock data of NASDAQ *(National Association of Securities Dealers Automated Quotation System)*.

Below are the example of a company in the NASDAQ and Vietnam dataset.

NASDAQ	Vietnam (UPCOM, HNX, and HOSE)
	

1	Date	Low	Open	Volume	High	Close	Adjusted Close
2	19-05-1994	5.06584406	5.06584406	134232	5.36979389	5.26847696	3.038053274
3	20-05-1994	5.06584406	5.16716099	54285	5.36979389	5.26847696	3.038053274
4	23-05-1994	5.06584406	5.26847696	16532	5.26847696	5.06584406	2.92120409
5	24-05-1994	5.06584406	5.26847696	7649	5.26847696	5.06584406	2.92120409
6	25-05-1994	5.06584406	5.16716099	13325	5.16716099	5.06584406	2.92120409
7	26-05-1994	5.06584406	5.06584406	19987	5.21781921	5.06584406	2.92120409
8	27-05-1994	4.96452713	5.16716099	12338	5.16716099	4.96452713	2.862779856
9	31-05-1994	5.06584406	5.06584406	1481	5.26847696	5.26847696	3.038053274
10	1/6/94	5.06584406	5.06584406	52311	5.26847696	5.26847696	3.038053274
11	2/6/94	5.11650181	5.26847696	5182	5.26847696	5.26847696	3.038053274
12	3/6/94	5.26847696	5.26847696	2468	5.26847696	5.26847696	3.038053274
13	6/6/94	5.06584406	5.26847696	5675	5.26847696	5.06584406	2.92120409
14	7/6/94	5.06584406	5.26847696	15545	5.26847696	5.26847696	3.038053274
15	8/6/94	5.06584406	5.06584406	19740	5.26847696	5.06584406	2.92120409
16	9/6/94	5.06584406	5.06584406	2714	5.26847696	5.26847696	3.038053274
17	10/6/94	5.21781921	5.26847696	987	5.26847696	5.21781921	3.008842945
18	13-06-1994	5.26847696	5.26847696	494	5.26847696	5.26847696	3.038053274
19	14-06-1994	5.06584406	5.26847696	52558	5.31913614	5.16716099	2.979628801

1		Open	High	Low	Close	Volume	TradingDate
2	0	9758	9758	7319	7904	140500	12/30/09
3	1	8441	8441	8441	8441	48900	12/31/09
4	2	9027	9027	8783	9027	36500	1/4/10
5	3	9612	9612	9027	9514	75600	1/5/10
6	4	9856	9856	8880	8880	65300	1/6/10
7	5	8539	9466	8295	8880	151000	1/7/10
8	6	9466	9466	9466	9466	38000	1/8/10
9	7	10100	10100	10100	10100	23400	1/11/10
10	8	10783	10783	10783	10783	123800	1/12/10
11	9	11466	11515	10051	10734	235500	1/13/10
12	10	10490	10929	10002	10246	32400	1/14/10
13	11	9661	9758	9563	9563	68300	1/15/10
14	12	9027	9027	8929	8929	24400	1/18/10
15	13	8587	8783	8343	8539	46100	1/19/10
16	14	8539	8587	7953	8051	87100	1/20/10
17	15	8197	8197	7514	7514	104300	1/21/10
18	16	7075	7953	7075	7221	61700	1/22/10
19	17	7319	7709	7172	7514	32800	1/25/10
20	18	7807	8002	7807	8002	50400	1/26/10
21	19	8295	8295	7514	7660	19700	1/27/10
22	20	7319	7319	7172	7270	34100	1/28/10

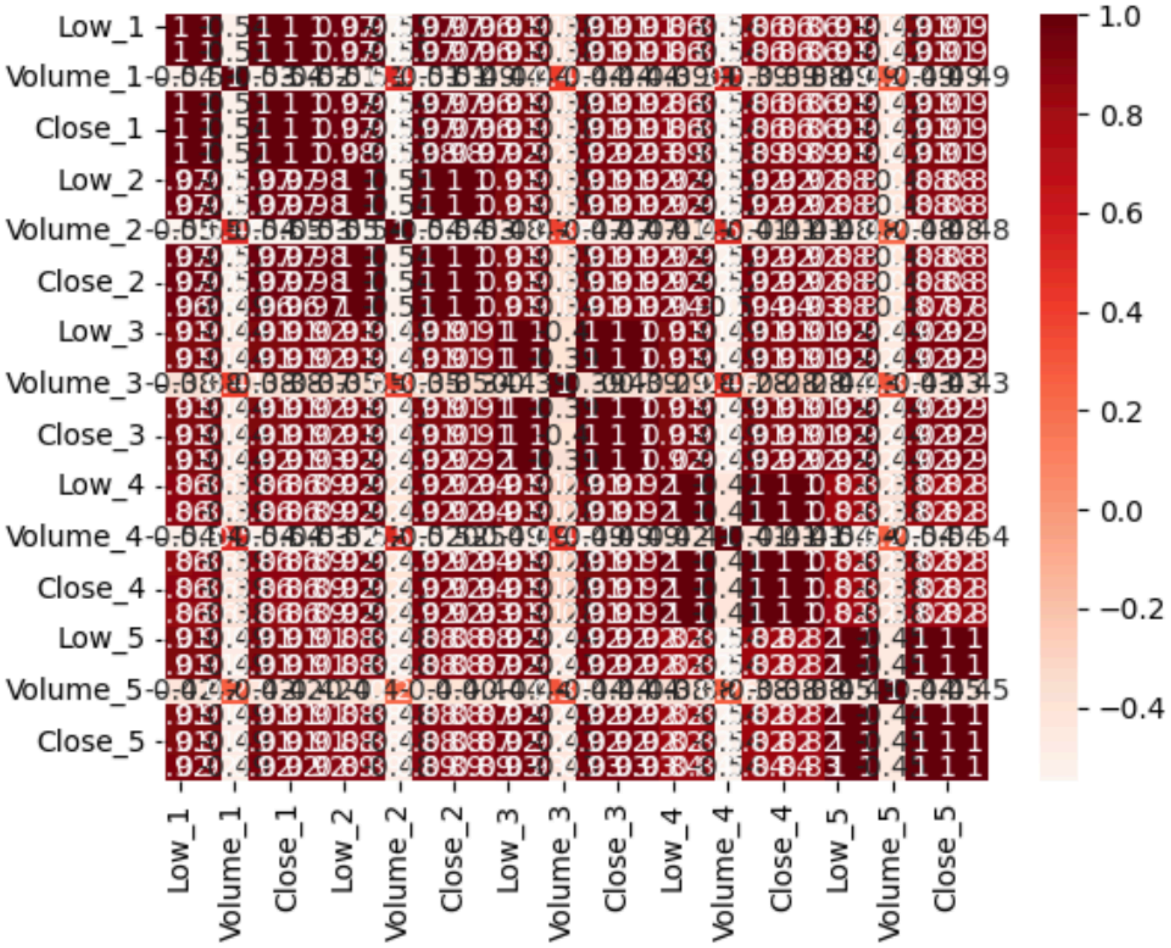
In addition, for Vietnam stock exchanges there are other datasets such as:

- Dividend history: historical dividends of companies.
- Financial ratio: financial health of companies.
- Industry analysis: information about companies in the same industry.

Feature Engineering

1. Data Features

First, we plot a correlation heat map to see the correlation between features of the dataset. Below are the heatmap of the NASDAQ dataset:



It is obtained that most features are correlated to one another, except for `Volume`. Therefore, we will use all the attributes except `Volume`.

Moreover, it is clear that two datasets do not share the same format. For example, NASDAQ has a `Adjusted Close` column that refers to the cost/share at the end of the day with consideration to other factors like dividends, stock splits, and new offerings, yet Vietnam market does not have this attributes.

Since we want to build a model that can handle both Vietnam and NASDAQ dataset, we must choose whether to use `Close` or `Adjusted Close` for NASDAQ. Although we have tested and found no significant different between them, the latter is shown to be a more accurate attribute in the dataset instruction. Therefore, `Adjusted Close` will be chosen instead of `Close`.

Furthermore, the stock prices are only in different currency. While NASDAQ'S currency is in US Dollar (USD), Vietnam is in Vietnam Dong (VND). However, since one must perform trading in $ for NASDAQ and VND for VN markets, we do not perform currency conversion. Instead, we will annotate the currency next to the stock price so that the user can acknowledge the difference in currency and perform conversion themselves if they want to.

The current exchange rate between USD and VND is as follow:

USD	VND
1.00	23,442.00
0.000043	1.00

Having said, below are all the attributes that we will use for the model:

Attributes	Description	Sources
Date	Data collection date	- NASDAQ: Date - VN: TradingDate
Low	Refers to the lowest price in that day	Low
High	Refers to the highest price at which a stock is traded in that day	High
Open	Refers to the price at which a stock started trading when the opening bell rang	Open
Close	Refers to cost of shares at the end of that day	- NASDAQ: Adjusted Close - VN: Close

2. Data Labels

With those data features, it is obviously we want to make prediction on all of them in the next 7 days. However, after building the model, we see that is unreasonable. Moreover, since they are correlated to one another, we decided to only use 2 of them, which are `Open` and `Close`. This is because in practice, `Close` price of the previous day can be used to predict the `Open` price of the next day. Moreover, it can also tell us the stock trend: whether it increases/decrease day by day, or it has a strong "hype" at every openings then decreasing throughout the day.

With that being said, we believe it is a good way to output both `Open` and `Close` price.

3. Period

We will use 4 attributes (`Low`, `High`, `Open`, `Close`) of **30 days** to make prediction on `Open` and `Close` price of the next **7 days**.

This is because we want to know the price of the next 7 days to perform trading (to buy new stocks or to sell some stocks). After trails and errors, we find that no period of training is efficient for 7 days forecasting. Our model is not capable of learning the functunation of stock price and produce the prediction themselves. It is just basically repeated the pattern in the training period and apply that to the predicting period.

First, I thought this is the model's mistake and my failure to train that prediction model. However, I realized that stocks does not follow any trend. It is, definitely, follows social, political and economical trends. But, by just using the stock price itself, even human cannot tell anything. Therefore, we cannot blame the model.

However, we still have to choose the trainign period. So, we choose 30 days as it is the best period for our model to repeat the trend most correctly.

4. Stock Selection Criteria

We have _ stocks data in the Vietnam dataset and _ stocks data in the NASDAQ dataset. This is enormous and we cannot run the model for this large number of stocks. Therefore, we downscale to, first, using only companies in NASDAQ, HNX, and UPCOM. The reason for NASDAQ is quite obvious as we dont have another dataset for international market. For Vietnam market, UPCOM is a sub-market of HNX for unlisted companies. Therefore, I thought that using HNX and UPCOM will be good as they will share some characteristics.

With that, we defined a criteria for companies as below to make them confront to the same requirements.

- NASDAQ:
  - In NASDAQ-100
  - Has data from January 4th, 2007 towards.
- HNX and UPCOM:
  - Has data from January 11th, 2007 towards.
    The reason we choose January 2007 is that we want to have a 5 years modeling (2007-2022).

Notwithstanding, the number of stocks is still large. So, we choose to only consider companies in these sectors:

- NASDAQ: Consumer Non-Durables, Consumer Services, Health Technology, Retail Trade, and Technology Services
- HNX: Basic Resources, Chemicals, Media
- UPCOM: Food & Beverage, Health Care, Technology

The reason for choosing these sector is based on number of companies in each sector. As our resources is limited, we chose those that *contains less than 15 companies*.

With that, we now have **56** companies to work with (*39 for NASDAQ and 17 for VN*). The reason why VN is so smaller in number comparing to NASDAQ is because Vietnam only opens its economics quite a few years ago. Therefore, there is not much company having stock data from 2007.

Model

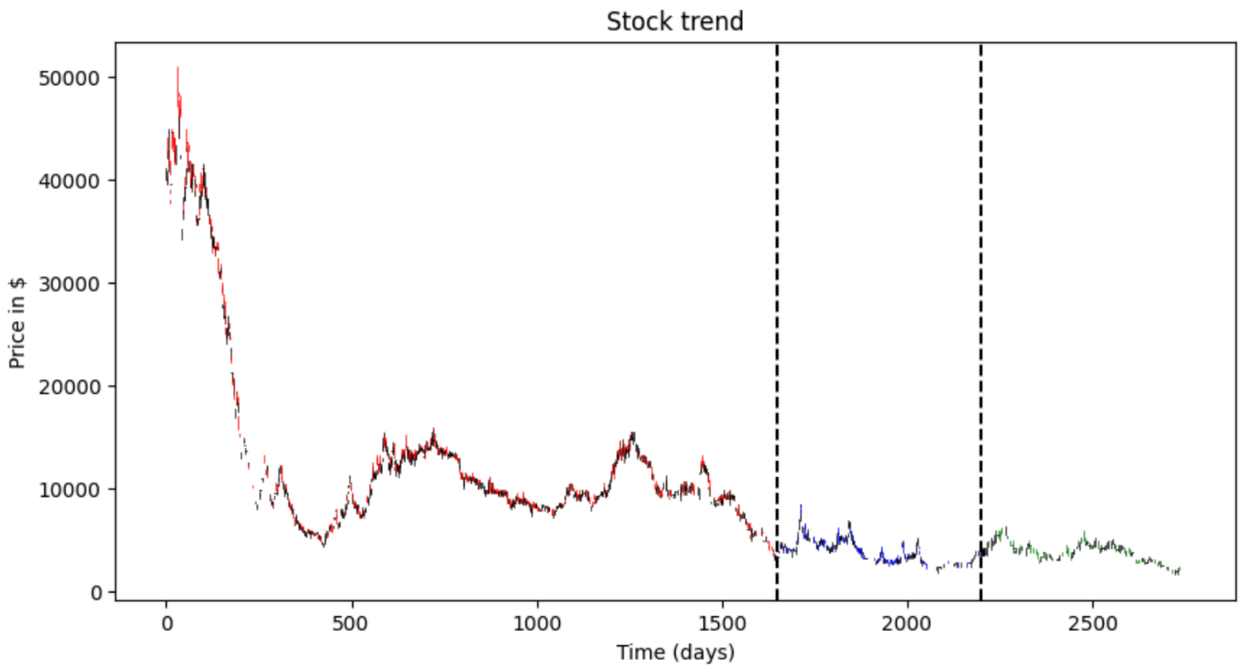
The full code of the model is located at models.py.

1. Dataset Split

First, we divide the stock into 4 parts as below.

- The last 30 days are used for forecasting. This is excluded from training, validation, and testing.

- The rest are divided into 3 parts with 6-2-2 ratio where:
  - The first 6 parts are for training
  - The next 2 parts are for validation
  - The last 2 parts are for testing

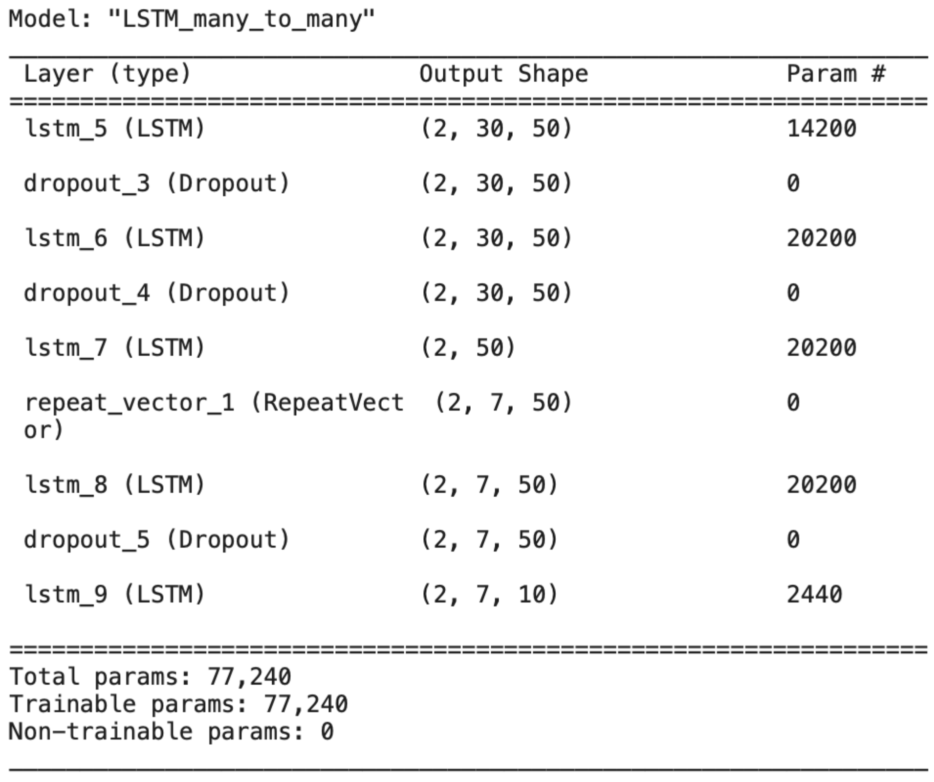


2. Architecture

We built a autoencoder LSTM model with 4 LSTM layers having `50` units alongside 3 Dropout layers having a `0.5` probability. The reason we chose this network architecture is because we want to have a many-to-many model where the dimension (30 days) of the input differs that of the output (7 days).

```
autoencoder = Sequential([
    # Encoder
    LSTM(50, return_sequences = True, stateful = True,
batch_input_shape = (2, 30, dim_feature)),   # Many to many
    Dropout(0.5),
    LSTM(50, return_sequences = True, stateful = True),    # Many to
many
    Dropout(0.5),
    LSTM(50, stateful = True),                             # Many to
one

    # Decoder
    RepeatVector(7),                      # One to many
    LSTM(50, return_sequences = True, stateful = True),    # Many to
many
    Dropout(0.5),
    LSTM(dim_label, return_sequences = True, stateful = True)      #
Classifiers
], name = "LSTM_many_to_many")
```

Below are the model summary.



```
Model: "LSTM_many_to_many"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_5 (LSTM)               (2, 30, 50)               14200

 dropout_3 (Dropout)         (2, 30, 50)               0

 lstm_6 (LSTM)               (2, 30, 50)               20200

 dropout_4 (Dropout)         (2, 30, 50)               0

 lstm_7 (LSTM)               (2, 50)                   20200

 repeat_vector_1 (RepeatVect  (2, 7, 50)               0
 or)

 lstm_8 (LSTM)               (2, 7, 50)                20200

 dropout_5 (Dropout)         (2, 7, 50)                0

 lstm_9 (LSTM)               (2, 7, 10)                2440

=================================================================
Total params: 77,240
Trainable params: 77,240
Non-trainable params: 0
_________________________________________________________________
```

There are 77M parameters in this model, which is reasonable for training quite a huge dataset.

3. Training

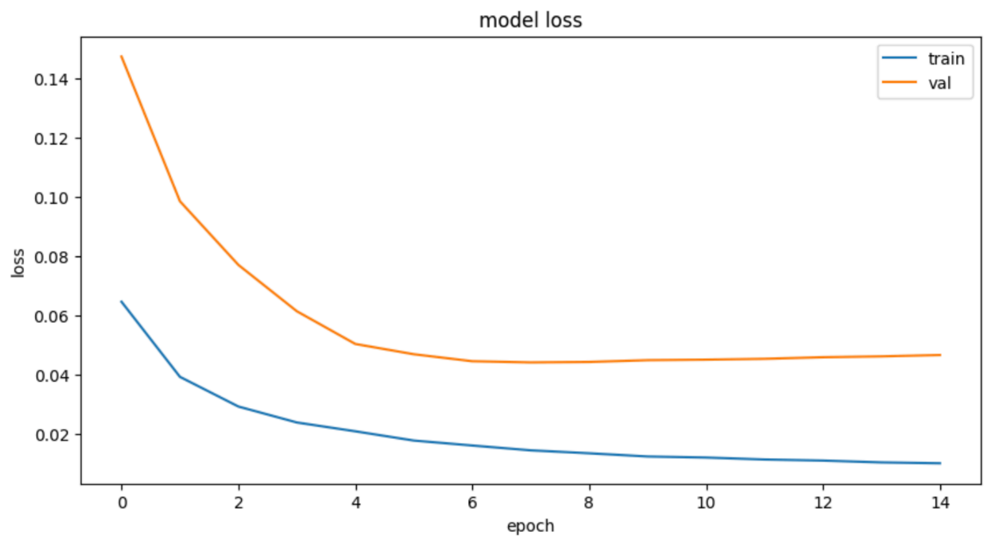
For training, we use `Adam` optimizer with small learning rate of `0.00005` and *mean squared error* as loss function (since our labels are stock prices).

To optimize resources (i.e. running time and GPU memory), we use a batch size of 2 instead of 1 (which means 2 periods of 30 days per step), along with 15 epochs.

```
# Compile and train the model with Mean Squared Error loss function
autoencoder.compile(optimizer = Adam(learning_rate = 1e-5), loss =
'mse', metrics = ['mse'])
lstm_performance = autoencoder.fit(X_train_norm, y_train_norm,
validation_data = (X_val_norm, y_val_norm), shuffle = False, epochs =
15, batch_size = 2, callbacks = [ResetStatesCallback()])
```

```
Epoch 1/15
825/825 [==============================] - 21s 13ms/step - loss: 0.0646 - mse: 0.0646 - val_loss: 0.1472 - val_mse: 0.1472
Epoch 2/15
825/825 [==============================] - 10s 12ms/step - loss: 0.0393 - mse: 0.0393 - val_loss: 0.0985 - val_mse: 0.0985
Epoch 3/15
825/825 [==============================] - 10s 12ms/step - loss: 0.0293 - mse: 0.0293 - val_loss: 0.0770 - val_mse: 0.0770
Epoch 4/15
825/825 [==============================] - 11s 13ms/step - loss: 0.0239 - mse: 0.0239 - val_loss: 0.0614 - val_mse: 0.0614
Epoch 5/15
825/825 [==============================] - 11s 13ms/step - loss: 0.0209 - mse: 0.0209 - val_loss: 0.0504 - val_mse: 0.0504
Epoch 6/15
825/825 [==============================] - 9s 11ms/step - loss: 0.0178 - mse: 0.0178 - val_loss: 0.0469 - val_mse: 0.0469
Epoch 7/15
825/825 [==============================] - 9s 11ms/step - loss: 0.0161 - mse: 0.0161 - val_loss: 0.0446 - val_mse: 0.0446
Epoch 8/15
825/825 [==============================] - 10s 12ms/step - loss: 0.0145 - mse: 0.0145 - val_loss: 0.0441 - val_mse: 0.0441
Epoch 9/15
825/825 [==============================] - 9s 11ms/step - loss: 0.0135 - mse: 0.0135 - val_loss: 0.0443 - val_mse: 0.0443
Epoch 10/15
825/825 [==============================] - 9s 11ms/step - loss: 0.0124 - mse: 0.0124 - val_loss: 0.0449 - val_mse: 0.0449
Epoch 11/15
825/825 [==============================] - 9s 11ms/step - loss: 0.0121 - mse: 0.0121 - val_loss: 0.0451 - val_mse: 0.0451
Epoch 12/15
825/825 [==============================] - 9s 11ms/step - loss: 0.0114 - mse: 0.0114 - val_loss: 0.0454 - val_mse: 0.0454
Epoch 13/15
825/825 [==============================] - 9s 11ms/step - loss: 0.0111 - mse: 0.0111 - val_loss: 0.0459 - val_mse: 0.0459
Epoch 14/15
825/825 [==============================] - 10s 12ms/step - loss: 0.0104 - mse: 0.0104 - val_loss: 0.0462 - val_mse: 0.0462
Epoch 15/15
825/825 [==============================] - 9s 11ms/step - loss: 0.0102 - mse: 0.0102 - val_loss: 0.0466 - val_mse: 0.0466
```

Let us see the training and validation performance in terms of loss and accuracy.



Looking at the chart, it is clear that our model is quite overfit. However, this can be due to large fluctuation and difference among stocks. However, as the difference between training and validation is `~0.08`, we believe the model is acceptable,

4. Testing

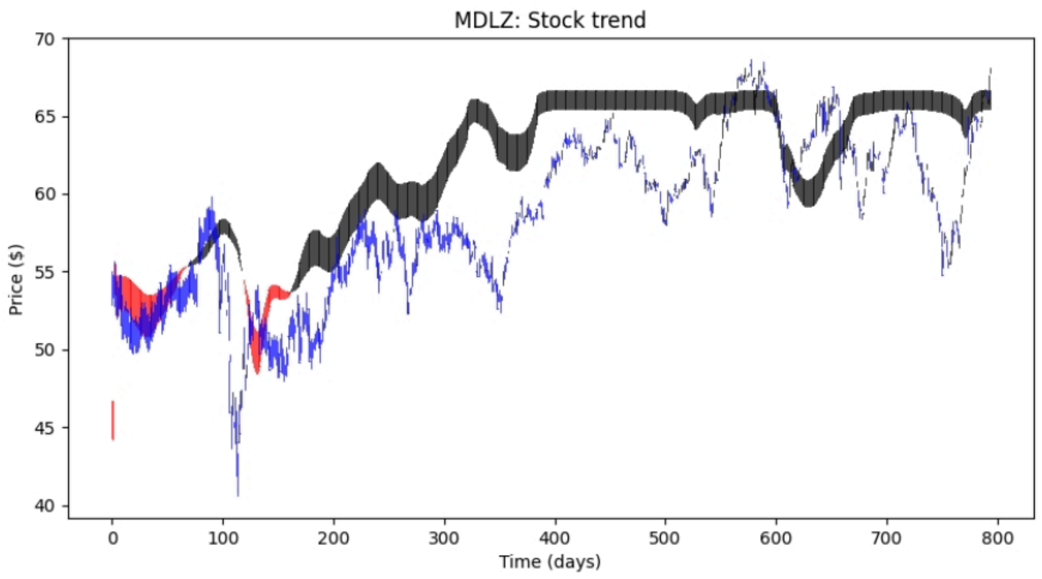
We performed the prediction on the test set and obtain the following MSE.

```
# Get prediction on the test data
y_pred_norm = autoencoder.predict(X_test_norm, batch_size = 2)

# Calculate the average MSE
average_mse = mean_squared_error(y_pred_norm.flatten(),
y_test_norm.flatten())
print("MSE on the test set: ", average_mse)
```

```
275/275 [==============================] - 3s 4ms/step
MSE on the test set:  0.04327697519078901
```

I believe this is good as the MSE on training is `0.010` and on validation is `0.046`. And since users do not use MSE to validate stocks, let's plot the stock price out.

```
# Plot the subset splits
fig, ax = plt.subplots(figsize = (10, 5))
candlestick3D(ax, y_pred, company = 2, colordown = 'blue', full =
False)
candlestick3D(ax, y_test, company = 2, colordown = 'red', full =
False)

ax.set_title("Stock trend")
plt.xlabel('Time (days)'); plt.ylabel('Price in $')
plt.show()
```

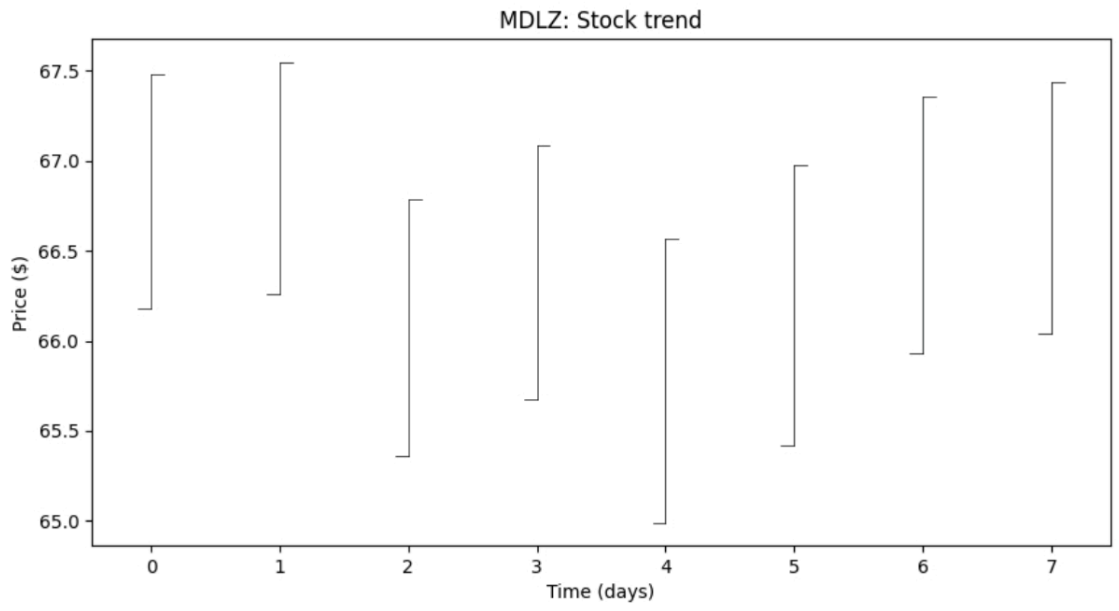


Although there is misalignment on from day 1 to day 100, the rest is quite good. Our model is capable of following the trend with some delays.

5. Forecasting

With this acceptable model, we now use stock price of the last 30 days to make forecast on 7 days.

```
# Plot the subset splits
fig, ax = plt.subplots(figsize = (10, 5))
candlestick2D(ax, y_pred[-1], company = 4, colordown = 'blue', full =
False)
ax.set_title("Stock trend")
plt.xlabel('Time (days)'); plt.ylabel('Price in $')
plt.show()
```



From the above graph, we can make some statements for the next 7 days

- **Current price**: `$66.177`/share. This is the close price of the first day.
- **Lowest price**: `$64.991`/share. This is the lowest close price.
- **Average price**: `$66.443`/share. This is the close price of the all days in average.
- **Highest price**: `$67.551`/share. This is the highest close price.
- **Trend**: decrease `-$0.089`/share (`-0.133%`). This is calculated by taking the close price of last day - close price of first day.

6. Recommendations

With that, we can say that the risk is acceptable, so you can buy some shares. In details, more recommendations can be:

- **Best buying**: $64.991/share on opening of the 4-th day. This is the lowest closing price.
- **Best selling**: $67.551/share on close of the 1-th day. This is the highest closing price.
- **Trading profit**: $2.56/share. This is the difference between buying and selling.
- **Trading risk**: $0/share. This is because we advertise people to sell on 1st day and buy on 4th day. With the decreasing trend, it is safe to sell when high and buy when low.

More on the trading, our full code is as below.

```
# Whether buy first of sell first
price_risk = round(price_sell - price_sell * trend_per, decimal) if
high_day < low_day else round(price_buy * trend_per - price_buy,
decimal)

# Risk can only to negative
price_risk = abs(price_risk) if rice_risk < 0 else 0
```

The idea is that we see whether we advertise user to buy first or sell first.

- If buy first, `risk = price_to_sell - price_when_buy`.
- If sell first, `risk = price_to_buy - price_when_sell`.

And so, with that information, we can calculate the risk and profit of all companies and categorize them into groups as follow:

- **Should hold**: when stock has an increasing trend
- **Should sell**: when stock has a decreasing trend
- **Risk takers**: when risk - profit > 0
- **Safety net**: when risk = 0

Application

Once having the model, we then trained it with our data and save the trained model. We will use these pre-trained model whenever there is an user using our website.

We deployed it and host the web application here on Streamlit.

About the code, we made it open-sourced and you can access on GitHub.

1. Technology Stack

This is our technology stack:

Streamlit

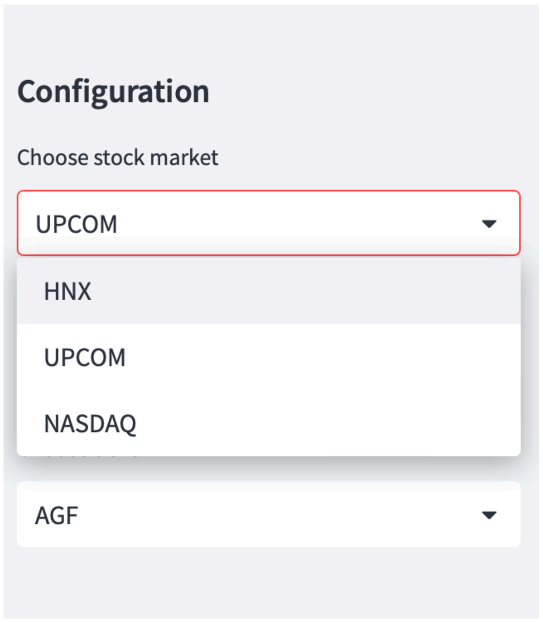
GitHub
TensorFlow
Python
Business Logic

- Streamlit is used for hosting and interacting with users
- GitHub is used for middleware and version management
- TensorFlow is used for interacting with pre-trained model
- Python is used for the programming

2. Instructions

In the `Configuration` tab (on the left side of the screen:

- Choose a stock market from the list



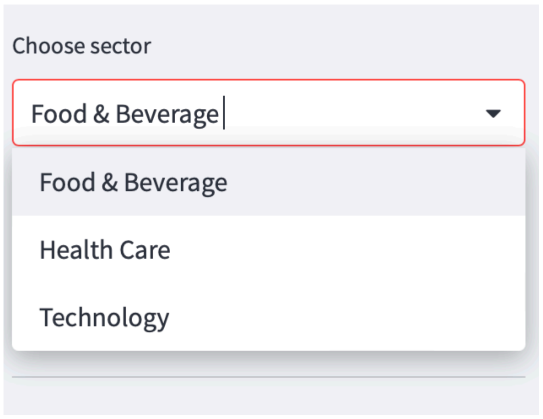
Configuration

Choose stock market

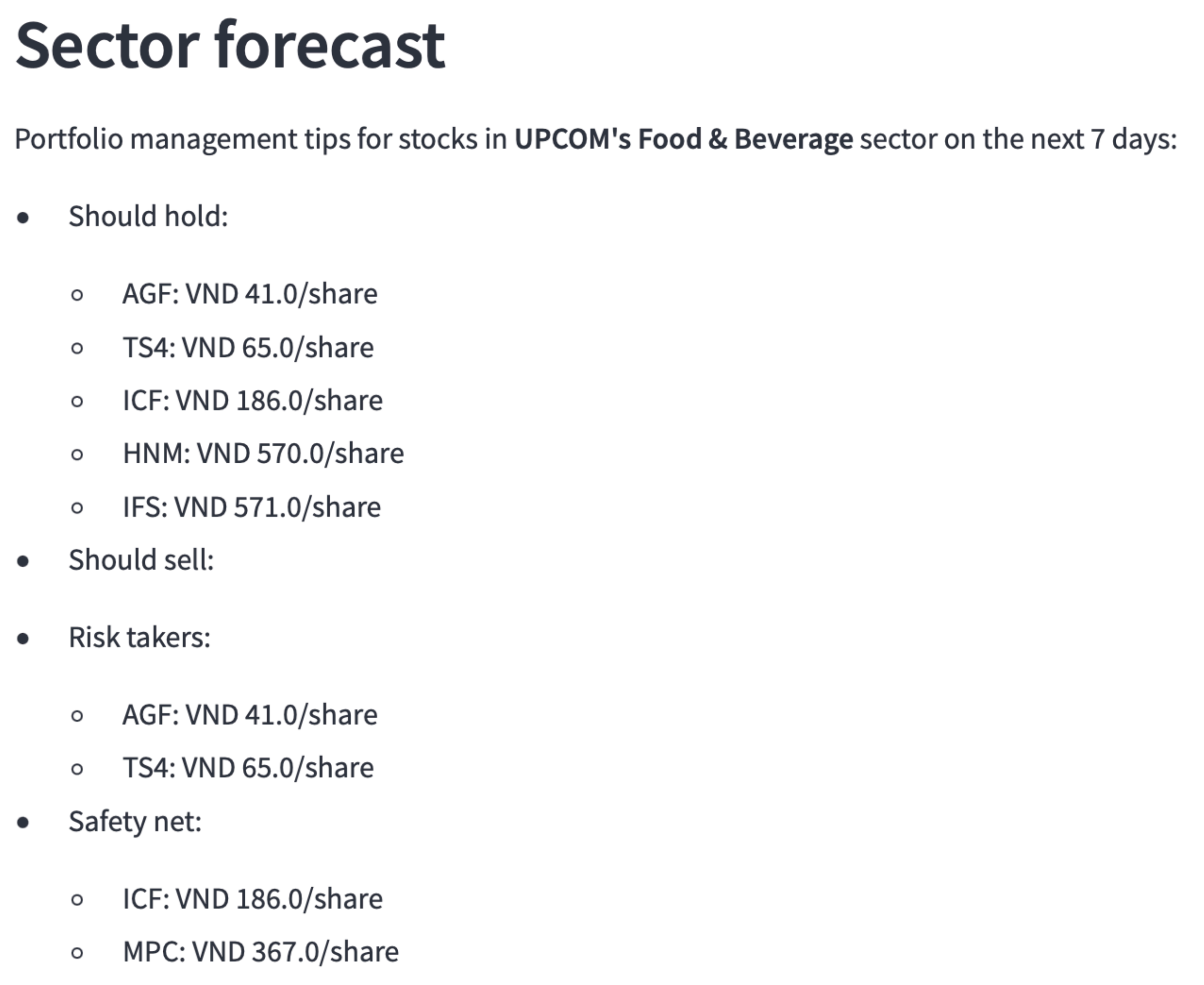
UPCOM

AGF

- Choose a sector from the list:



Then, the first company is automatically selected. Please regard to the main page. You can scroll to the `Sector forecast` section to see the insights for all companies in this industry of that market.

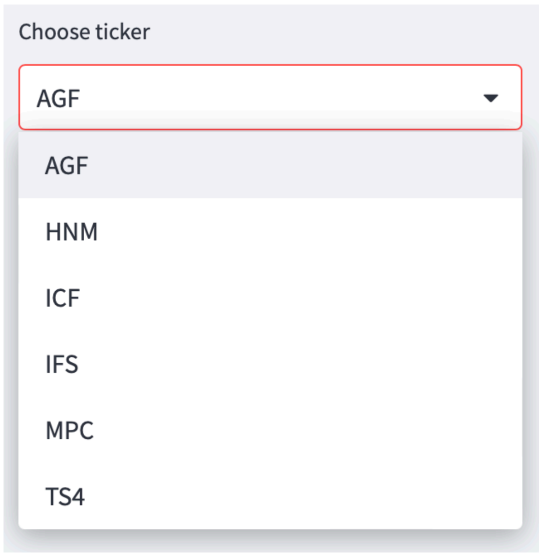


Sector forecast

Portfolio management tips for stocks in **UPCOM's Food & Beverage** sector on the next 7 days:

- Should hold:
  - AGF: VND 41.0/share
  - TS4: VND 65.0/share
  - ICF: VND 186.0/share
  - HNM: VND 570.0/share
  - IFS: VND 571.0/share
- Should sell:
- Risk takers:
  - AGF: VND 41.0/share
  - TS4: VND 65.0/share
- Safety net:
  - ICF: VND 186.0/share
  - MPC: VND 367.0/share

Then, with that information, you can navigate to the `Configuration` tab and select the company that you want to investigate in.

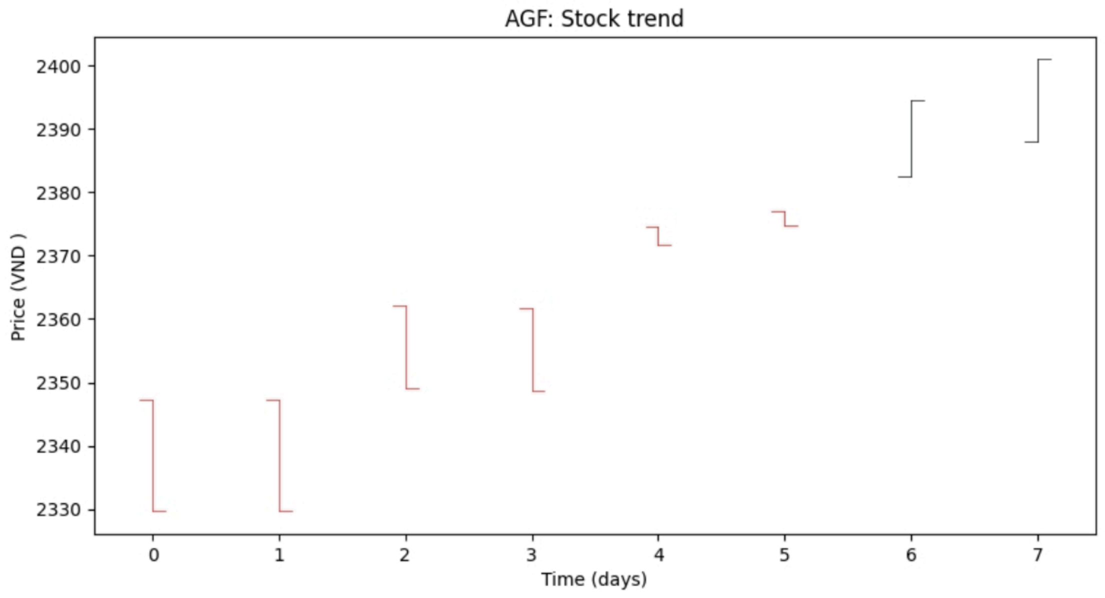


Once you have selected one, you can look at the main page and scroll down to the `Stock forecast` and `Model performance` section.

In the `Stock forecast` section, there are 3 sub-sections: `Stock Prices`, `Stock statistics` and `Trading recommendation` as below. We have discussed about how we come up with those information above.

Stock Prices

Stock prices of *AGF* on the next 7 days:



Stock statistics

Stock statistics for **AGF** in the next 7 days:

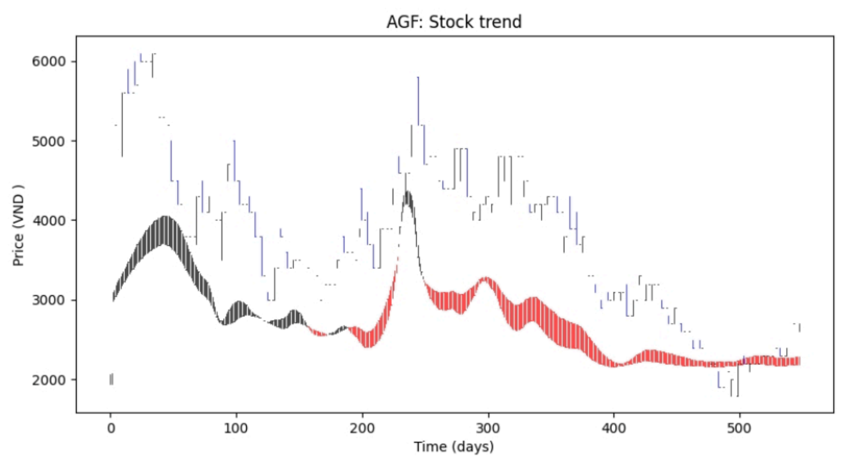
- Current price: VND 2347.0/share
- Lowest price: VND 2330.0/share
- Average price: VND 2365.0/share
- Highest price: VND 2401.0/share
- Trend: Increase VND 56.0/share (2.0%)

Trading recommendation

Trading recommendation for **AGF** in the next 7 days:

- Best buying: VND 2330.0/share on close of the 1-th day
- Best selling: VND 2401.0/share on close of the 7-th day
- Trading profit: VND 71.0/share
- Trading risk: VND 294.0/share
- Conclusion: RISKY!

In the `Model performance` section, we can see the overall stock trend and the MSE value.



Testing's mean squared error: 0.061

Conclusion

So, we conclude that our model is not that good, but at least it can capture the stock price picture and we can interpret its forecast.