

C A P S T O N E   P R O J E C T

# STOCK PRICE PREDICTION SYSTEM



TEAM H

Donghun Jung  
Chanyoung Lee  
Yujin Seo

# Contents

---

**01**

Project  
Objectives

**02**

AI Models  
& Programs

**03**

Challenges  
& Progress

**04**

Demo

**05**

Result  
& Discussion

# Project Objectives

---


## Stock Price Prediction System

- ✓ **Using AI Models – LSTM, GRU, Transformer**
  - Specialized for sequential data(time series data) analysis
- ✓ **Front-end Implementation**
  - Visualize real and predictive graphs through website development
- ✓ **Not Real-Time**

# Project Objectives

---

What is the difference with existing services?



**키우GO**

투자목표를 키워주는 인공지능(AI) 로보어드바이저「키우GO」  
키움증권의 온라인 자산관리 서비스입니다.

#목표기반투자 #로보어드바이저 #글로벌 분산투자

SK C&C, '마켓캐스터 AIST' 활용한 실제 'AI 투자 전략 서비스' 개발... 상용화 위  
한 검증도 진행

- ✓ **There are already many stock investment services using AI**
  - But they care about the stable distribution and management of assets of the users

# Project Objectives

So, we decided to focus only on improving the predictive performance...

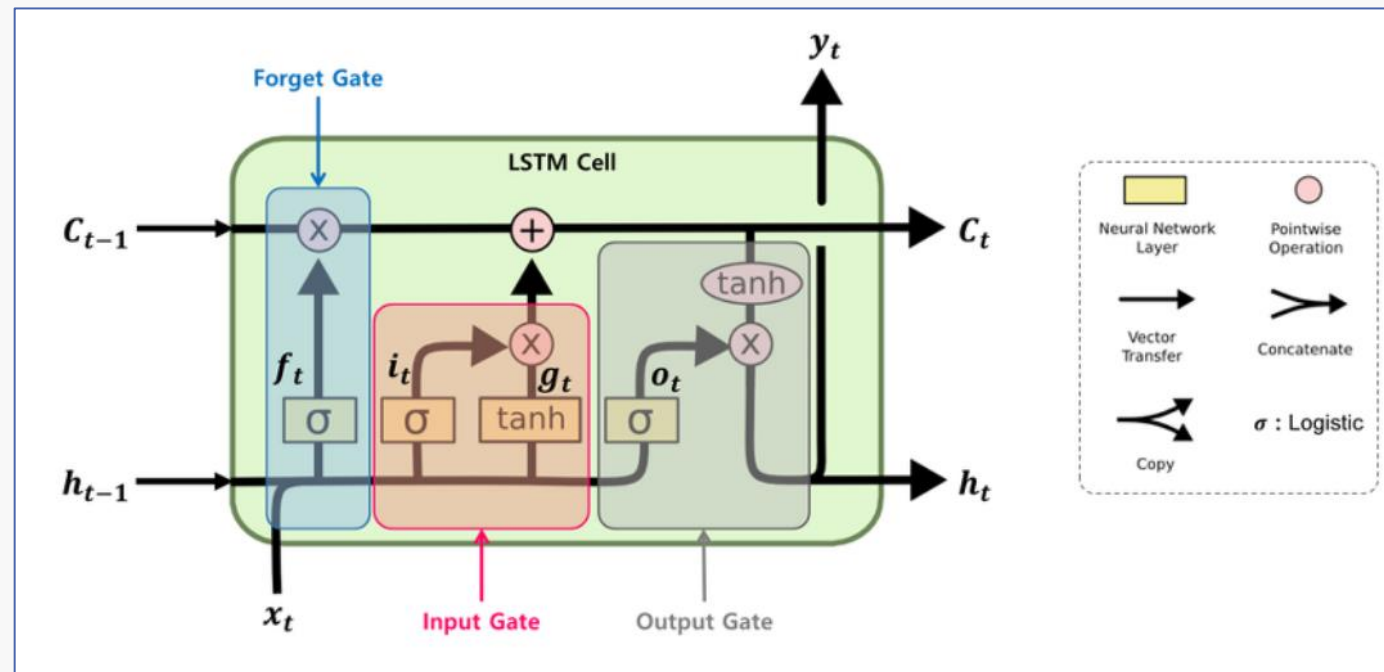


- ✓ Because it is hard to implement many functions due to the limited cost and time given

# AI Models & Programs

## LSTM (Long Short Term Memory)

- ✓ Solves gradient vanishing problem of existing RNN when input data is too long



# AI Models & Programs

## LSTM (Long Short Term Memory)

- ✓ Solves gradient vanishing problem of existing RNN when input data is too long

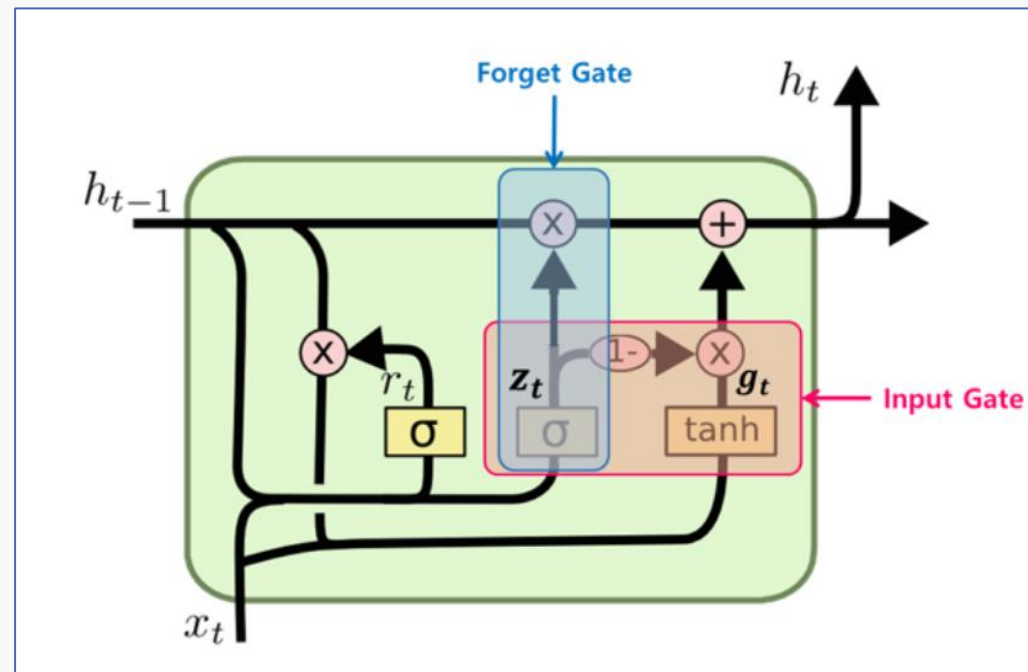
```
class FinanceLSTM(nn.Module):  
  
    def __init__(self, model_args):  
  
        super(FinanceLSTM, self).__init__()  
  
        self.output_length = model_args.output_length  
        self.num_layers = model_args.num_layers  
        self.input_size = model_args.input_size  
        self.hidden_size = model_args.hidden_size  
        self.fc_hidden_size = model_args.fc_hidden_size  
        self.device = model_args.device  
  
        self.lstm = nn.LSTM(input_size = self.input_size, hidden_size = self.hidden_size,  
                             num_layers = self.num_layers, batch_first = True)  
        self.fc1 = nn.Linear(self.hidden_size, self.fc_hidden_size)  
        self.fc2 = nn.Linear(self.fc_hidden_size, self.output_length)  
        self.act = nn.ELU()
```

```
def forward(self, x):  
  
    h_0 = torch.Tensor(torch.zeros(self.num_layers, x.size(0), self.hidden_size)).to(self.device)  
    c_0 = torch.Tensor(torch.zeros(self.num_layers, x.size(0), self.hidden_size)).to(self.device)  
  
    output, (hn, cn) = self.lstm(x, (h_0, c_0))  
    hn = hn.view(-1, self.hidden_size)  
    logits = self.act(hn)  
    logits = self.fc1(logits)  
    logits = self.act(logits)  
    logits = self.fc2(logits)  
    return logits
```

# AI Models & Programs

## GRU (Gated Recurrent Unit)

- ✓ A version of simplifying the time-step cell that makes up the LSTM





# AI Models & Programs

## GRU (Gated Recurrent Unit)

- ✓ A version of simplifying the time-step cell that makes up the LSTM

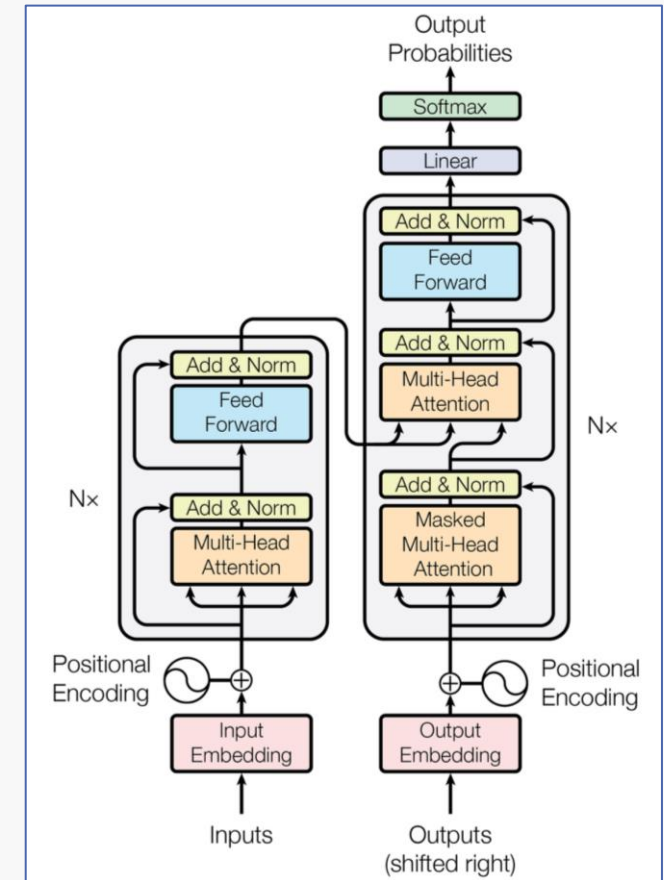
```
class FinanceGRU(nn.Module):  
  
    def __init__(self, model_args):  
  
        super(FinanceGRU, self).__init__()  
  
        self.output_length = model_args.output_length  
        self.num_layers = model_args.num_layers  
        self.input_size = model_args.input_size  
        self.hidden_size = model_args.hidden_size  
        self.fc_hidden_size = model_args.fc_hidden_size  
        self.device = model_args.device  
  
        self.gru = nn.GRU(input_size = self.input_size, hidden_size = self.hidden_size,  
                           num_layers = self.num_layers, batch_first = True)  
        self.fc1 = nn.Linear(self.hidden_size, self.fc_hidden_size)  
        self.fc2 = nn.Linear(self.fc_hidden_size, self.output_length)  
        self.relu = nn.ReLU()
```

```
def forward(self, x):  
    h_0 = torch.Tensor(torch.zeros(self.num_layers, x.size(0), self.hidden_size)).to(self.device)  
    output, (hn) = self.gru(x, (h_0))  
    hn = hn.view(-1, self.hidden_size)  
    logits = self.relu(hn)  
    logits = self.fc1(logits)  
    logits = self.relu(logits)  
    logits = self.fc2(logits)  
  
    return logits
```

# AI Models & Programs

## Transformer

- ✓ Parallel processing is possible by putting the sequence at once, and location information can be reflected



# AI Models & Programs

## Transformer

- ✓ Parallel processing is possible by putting the sequence at once, and location information can be reflected

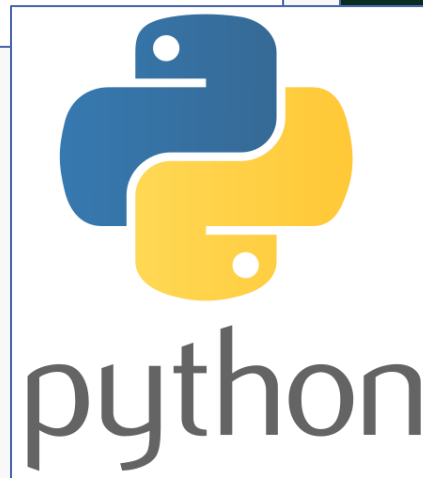
```
class FinanceTransformer(nn.Module):  
  
    def __init__(self, model_args):  
  
        super(FinanceTransformer, self).__init__()  
  
        self.embed_dim = model_args.embed_dim  
        self.resolution = model_args.resolution  
        self.n_head = model_args.n_head  
        self.n_layer = model_args.n_layer  
        self.fc_hidden_size = model_args.tf_fc_hidden_size  
        self.seq_length = model_args.seq_length  
        self.output_length = model_args.output_length  
        self.input_size = model_args.tf_input_size  
        self.device = model_args.device  
  
        embed_args = EmbeddingArgs(embed_dim=self.embed_dim, resolution=self.resolution)  
        self.embed = FinanceEmbedding(embed_args)  
  
        self.encoder_layer = nn.TransformerEncoderLayer(d_model=self.embed_dim*(self.input_size-1), nhead=self.n_head, batch_first=True)  
        self.encoder = nn.TransformerEncoder(encoder_layer = self.encoder_layer, num_layers=self.n_layer)  
        self.fc1 = nn.Linear(self.embed_dim*self.seq_length*(self.input_size-1), self.fc_hidden_size)  
        self.fc2 = nn.Linear(self.fc_hidden_size, self.output_length)  
        self.act = nn.ELU()  
        self.flat = nn.Flatten()
```

```
def forward(self, x):  
    x = self.resolution_map(x)  
    embed_x = self.embed(x)  
    enc_x = self.flat(self.encoder(embed_x))  
    logits = self.fc1(enc_x)  
    logits = self.act(logits)  
    logits = self.fc2(logits)  
    return logits
```

# AI Models & Programs

---

## Backend Programs



# AI Models & Programs

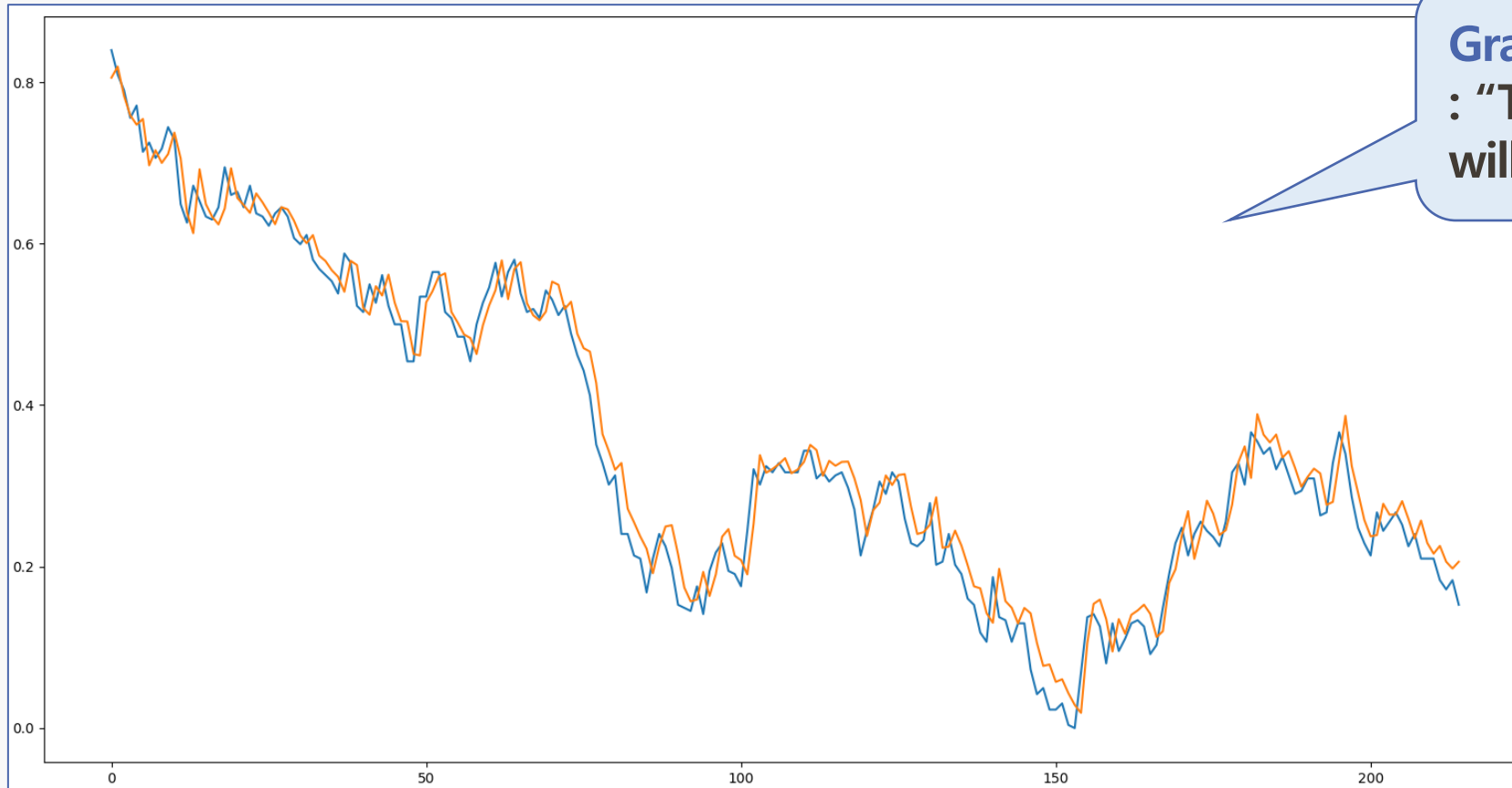
---

## Frontend Programs



# Challenges & Progress

---



## Graph Shifting

: "Tomorrow's stock price will be Today's stock price"

# Challenges & Progress

---

## Problem: Graph Shifting

💡 **Solution:** Apply more diverse figures to predictions

👉 **Progress:** Apply **VADER score** of stock market articles

```
[5] import pandas as pd
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

[6] nltk.download('vader_lexicon')
sentiment = SentimentIntensityAnalyzer()

[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data] Package vader_lexicon is already up-to-date!

[7] sentiment.polarity_scores('good good good')
{'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': 0.8271}

▶ sentiment.polarity_scores('Memory Market: It is judged to enter the low-demand phase such as PC and mobile. Demand for some builds appears.')
{'neg': 0.073, 'neu': 0.927, 'pos': 0.0, 'compound': -0.128}
```

# Challenges & Progress

## Problem: Graph Shifting

- 💡 **Solution:** Train models with another stock data
- 👉 **Progress:** Increase the dataset and divide it into 4 categories

Korea Stable	Korea unstable	US Nasdaq Stable	US Nasdaq unstable
Samsung Electronics Co Ltd Samsung SDI Co Ltd KT&G Corp Yuhan Corp SK Hynix Inc DB Insurance Co Ltd Korea Zinc Inc Fila Holdings Corp Meritz Financial Group Inc Cheil Worldwide Inc LG Innotek Co Ltd Shinsegae Inc IBK NCSOFT Corp Lotte Chemical Corp Leeno Industrial Inc Nongshim Co Ltd S1 Corp Hyundai Marine & Fire Insurance Co. Ltd.	Posco Holdings Inc Naver Corp Hyundai Motor Co LG Chem Ltd Kia Corp Celltrion Inc KB Financial Group Inc Shinhan Financial Group Co Ltd LG Energy Solution Ltd Kakao Corp HYUNDAI MOBIS CO., LTD. Samsung C&T Corp Samsung Biologics Co Ltd LG Electronics Inc Hana Financial Group Inc	Coca-Cola Co McDonald's Corp Waste Management, Inc. Republic Services Inc PepsiCo, Inc. Colgate-Palmolive Company Walmart Inc Cboe Global Markets Inc General Dynamics Corp Kimberly-Clark Corp Procter & Gamble Co Cencora Inc IBM Common Stock	Apple Inc Microsoft Corp Amazon.com Inc NVIDIA Corp Meta Platforms Inc Broadcom Inc Alphabet Inc Class A Alphabet Inc Class C Tesla Inc Adobe Inc Costco Wholesale Corporation Cisco Systems Inc Netflix Inc Advanced Micro Devices, Inc.



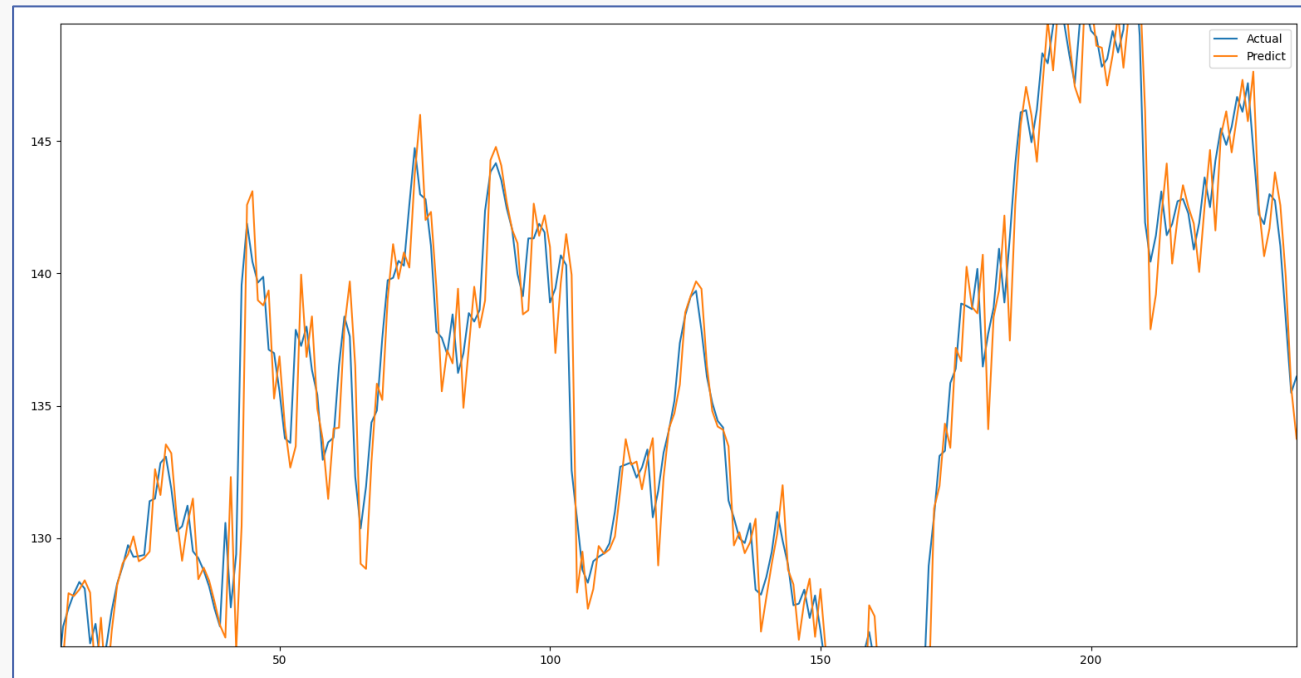
# Challenges & Progress

---

## Problem: Graph Shifting

💡 **Solution:** Make some changes in the learning rate of the models

👉 **Progress:** Apply learning rate scheduler to prediction system



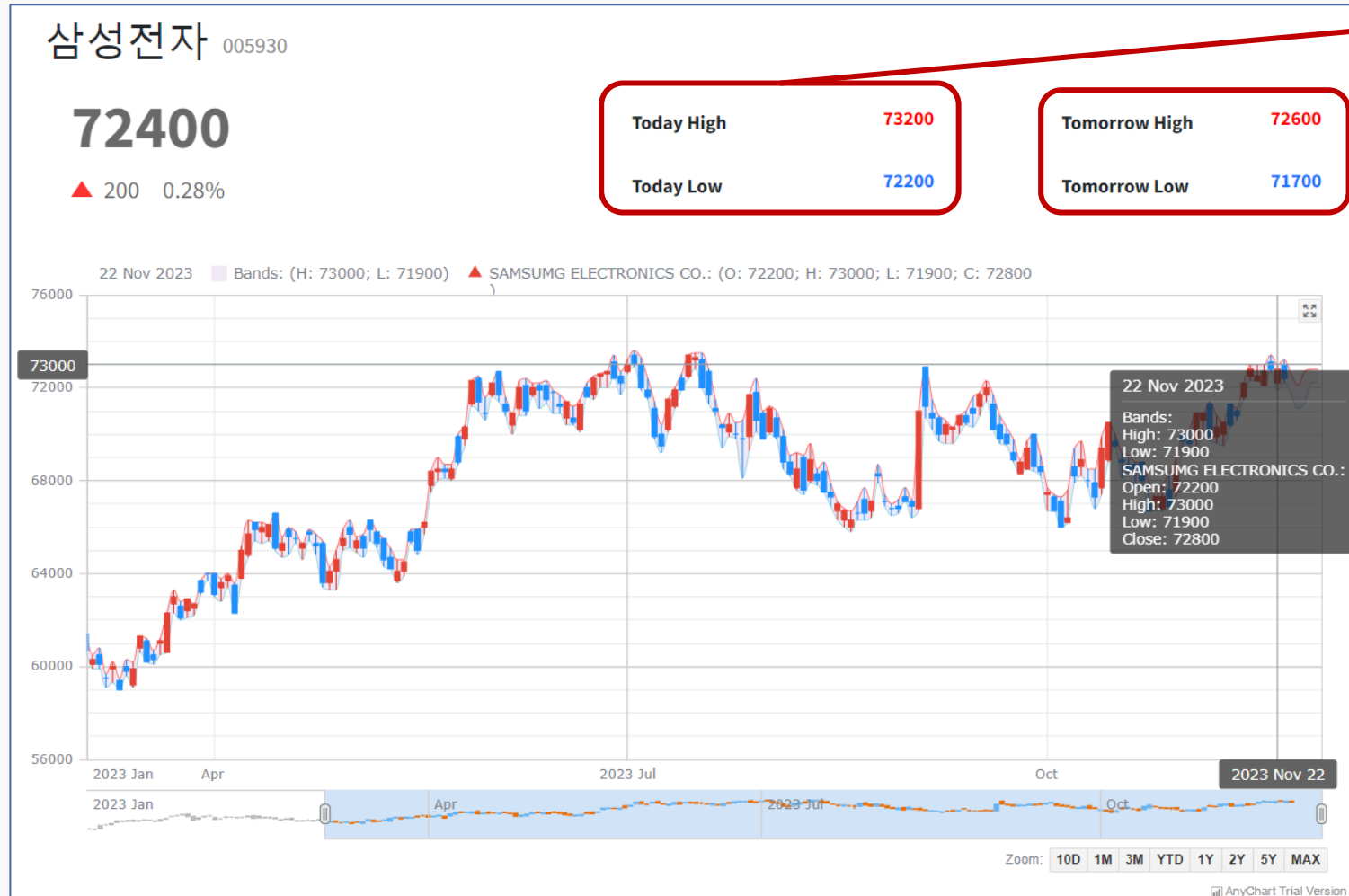
# Demo



Stock Name

Stock Price (Real)

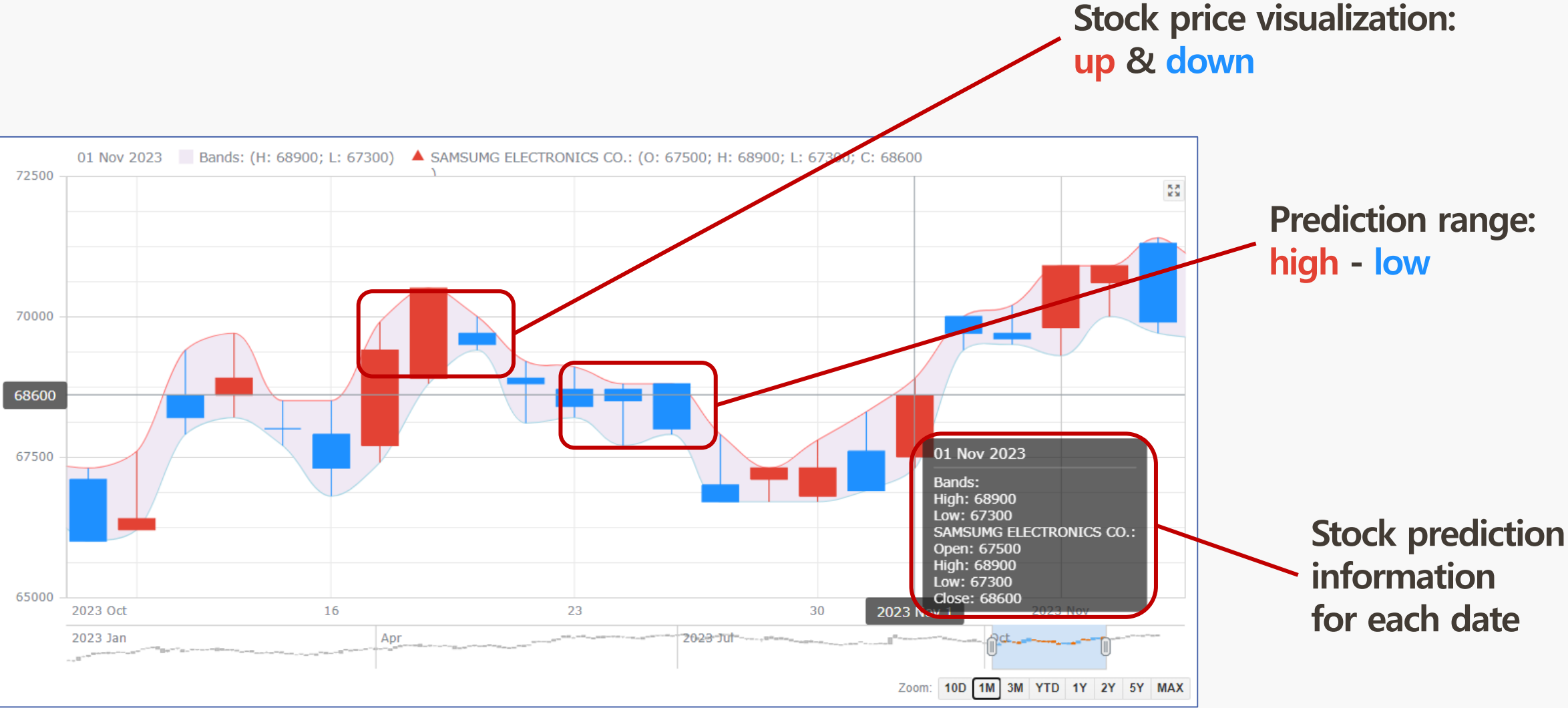
# Demo



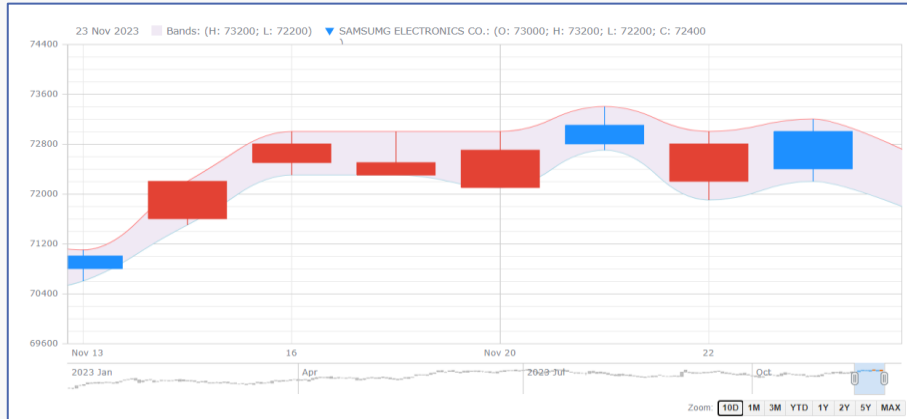
Today Stock Price  
(Prediction)

Tomorrow Stock Price  
(Prediction)

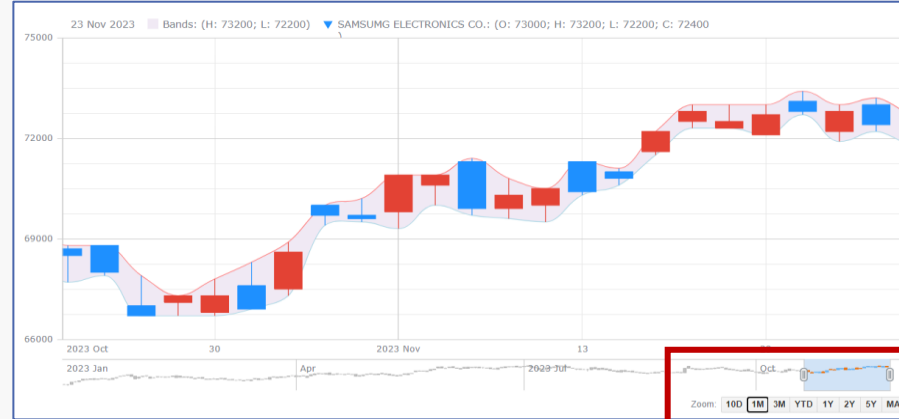
# Demo



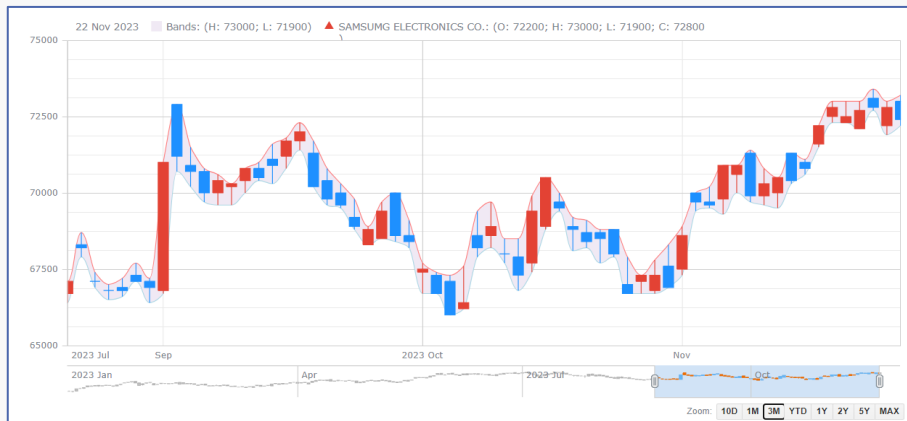
# Demo



Period: 10 days



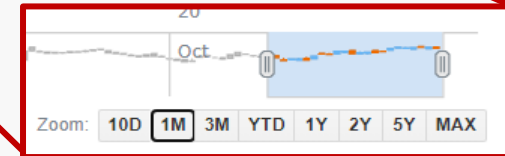
Period: 1 month



Period: 3 months



Period: 2 years




User can select the period with buttons or time scrollbar

# Demo

Hello, This is

# Stock Price Prediction System

Team H, SWE3028



<h2>Description</h2> <p>Our project has focused on predicting the short-term movements of</p>	<h2>Our Model</h2> <p>LSTM</p>	<h2>Transformer</h2> <p>The Transformer model represents a significant shift from traditional RNNs, as it completely eschews recurrence and instead utilizes an attention mechanism. This mechanism allows the model to process different parts of the input sequence in parallel and weigh them based on</p>
---	--------------------------------	---

# Result & Discussion

---

$$\begin{aligned} \text{MPA}_t &= 1 - \frac{1}{N} \sum_{i=1}^N \frac{|\hat{y}_i - y_i|}{y_i} \\ \text{MAE}_t &= \frac{1}{N} \sum_{i=1}^N |\hat{y}_i - y_i| \\ \text{TA}_t &= \frac{1}{N} \sum_{i=1}^N \mathbb{1}(\text{sign}(\hat{y}_i - \hat{y}_{t-1}) = \text{sign}(y_i - y_{t-1})) \\ \text{Accuracy}_t &= \frac{1}{N} \sum_{i=1}^N \frac{\text{length}(\{y_{\min} < y < y_{\max} \cap \hat{y}_{\min} < y < \hat{y}_{\max}\})}{\max(\text{length}(\hat{y}_{\min} < y < \hat{y}_{\max}), \text{length}(y_{\min} < y < y_{\max}))} \end{aligned}$$

## Accuracy measurement method

1. **MPA & MAE:** Evaluate the effectiveness of our methods
2. **TAC:** Focus on short-term stock price trends
3. **Accuracy:** Compare the similarity between the predicted and actual price intervals

# Result & Discussion

		STABLE				UNSTABLE			
Model		MPA	MAE	TAC	ACC	MPA	MAE	TAC	ACC
LSTM	High	0.9922	1.2179	0.5640	0.4183	0.9856	4.2027	0.5872	0.4163
	Low	0.9921	1.1985	0.5702		0.9851	4.1493	0.5711	
GRU	High	0.9929	1.1118	0.5640	0.4340	0.9864	3.9909	0.5930	0.4199
	Low	0.9926	1.1193	0.5659		0.9862	3.8426	0.5747	
Transformer	High	0.9906	1.4721	0.4743	0.4295	0.9827	5.2046	0.4731	0.3374
	Low	0.9892	1.6502	0.5361		0.9830	4.8531	0.4616	
BASELINE		0.9815				0.9815			

- ✓ Performance of our models on the test set is quantified.



# Result & Discussion

		STABLE				UNSTABLE			
Model		MPA	MAE	TAC	ACC	MPA	MAE	TAC	ACC
LSTM	High	0.9922	1.2179	0.5640	0.4183	0.9856	4.2027	0.5872	0.4163
	Low	0.9921	1.1985	0.5702		0.9851	4.1493	0.5711	
GRU	High	0.9929	1.1118	0.5640	0.4340	0.9864	3.9909	0.5930	0.4199
	Low	0.9926	1.1193	0.5659		0.9862	3.8426	0.5747	
Transformer	High	0.9906	1.4721	0.4743	0.4295	0.9827	5.2046	0.4731	0.3374
	Low	0.9892	1.6502	0.5361		0.9830	4.8531	0.4616	
BASELINE		0.9815				0.9815			

- ✓ Incorporating VADER has resulted in MPA surpassing that of the baseline model (DP-LSTM).
- ✓ **It seems that applying VADER score has the effect of increasing prediction accuracy.**

# Result & Discussion

		STABLE				UNSTABLE			
Model		MPA	MAE	TAC	ACC	MPA	MAE	TAC	ACC
LSTM	High	0.9922	1.2179	0.5640	0.4183	0.9856	4.2027	0.5872	0.4163
	Low	0.9921	1.1985	0.5702		0.9851	4.1493	0.5711	
GRU	High	0.9929	1.1118	0.5640	0.4340	0.9864	3.9909	0.5930	0.4199
	Low	0.9926	1.1193	0.5659		0.9862	3.8426	0.5747	
Transformer	High	0.9906	1.4721	0.4743	0.4295	0.9827	5.2046	0.4731	0.3374
	Low	0.9892	1.6502	0.5361		0.9830	4.8531	0.4616	
BASELINE		0.9815				0.9815			

- ✓ The model's performance varies based on the stability of the stocks. For unstable stocks, the MAE significantly increases, and ACC decreases.

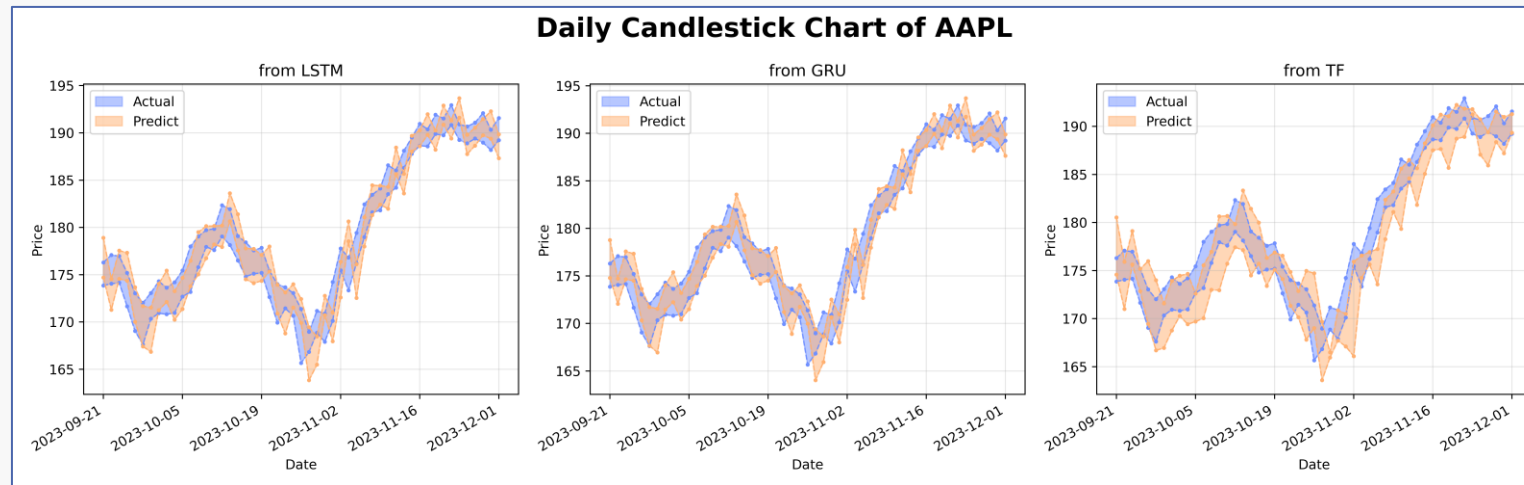
# Result & Discussion

		STABLE				UNSTABLE			
Model		MPA	MAE	TAC	ACC	MPA	MAE	TAC	ACC
LSTM	High	0.9922	1.2179	0.5640	0.4183	0.9856	4.2027	0.5872	0.4163
	Low	0.9921	1.1985	0.5702		0.9851	4.1493	0.5711	
GRU	High	0.9929	1.1118	0.5640	0.4340	0.9864	3.9909	0.5930	0.4199
	Low	0.9926	1.1193	0.5659		0.9862	3.8426	0.5747	
Transformer	High	0.9906	1.4721	0.4743	0.4295	0.9827	5.2046	0.4731	0.3374
	Low	0.9892	1.6502	0.5361		0.9830	4.8531	0.4616	
BASELINE		0.9815				0.9815			

- ✓ Both LSTM and GRU models exhibit similar performance metrics, whereas the Transformer model has relatively lower performance.

# Result & Discussion

## Limitations: Prediction Quality



- ✓ **Increased inaccuracies for specific cases**
  - Problem arises when the stock price remains relatively constant with minor fluctuations (2023-11-16 to 2023-12-01)

# Result & Discussion

---

## Limitations: Lack of Information

- ✓ **Imbalance of news articles**
  - Famous stocks have a lot of related information, but obscure stocks update less and slower
- ✓ **Articles written in a limited period of time**
  - The referenced news site's articles are limited to maximum 1,000 per stocks

# Roles

---

## Donghun Jung

- ✓ Creating the initial UI/UX design

## Chanyoung Lee

- ✓ Implementing GRU, CNN, Transformer

## Yujin Seo

- ✓ Implementing GRU and collecting news information



**Any Question?**