Python

주식 예측 프로그램 파이썬으로 만들어보기

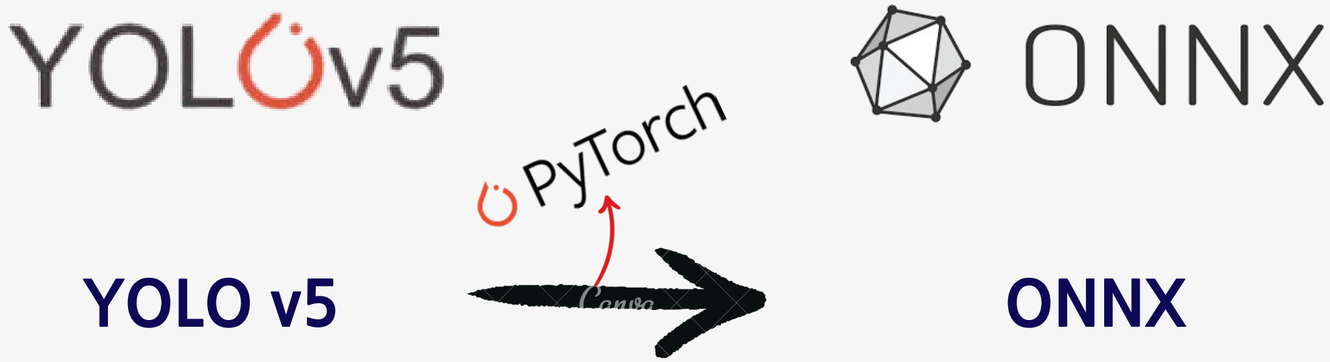
“처음에 계획했던 방법“

처음 계획 (작동 원리)

YOLO v5

“You Only Look Once”
CNN 딥러닝 모델을 이용한
이미지 기반 실시간 객체 검출 시스템

처음 계획 (작동 원리)



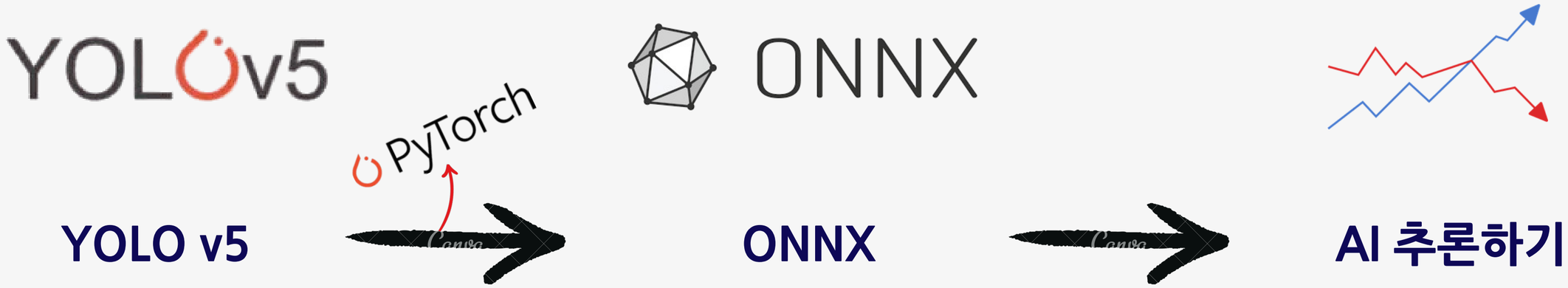
YOLO v5

“You Only Look Once”
CNN 딥러닝 모델을 이용한
이미지 기반 실시간 객체 검출 시스템

ONNX

“Open Neural Network Exchange”
다양한 딥러닝 프레임워크 간에 모델
을 호환할 수 있게 해주는 표준 형식

처음 계획 (작동 원리)



YOLO v5 → **ONNX** → **AI 추론하기**

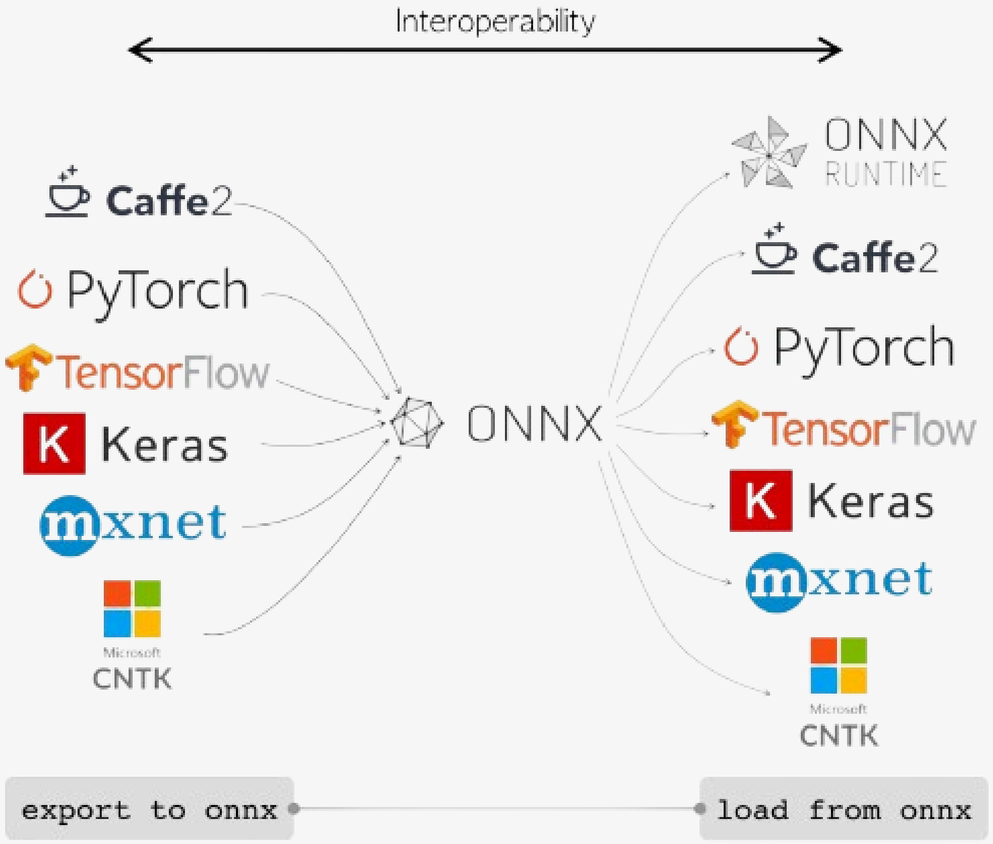
“You Only Look Once”
CNN 딥러닝 모델을 이용한
이미지 기반 실시간 객체 검출 시스템

“Open Neural Network Exchange”
다양한 딥러닝 프레임워크 간에 모델
을 호환할 수 있게 해주는 표준 형식

주가 그래프 이미지 분석해
시각적 패턴에서 규칙을 찾기

왜 ONNX 파일로 변환할까?

왜 ONNX 파일로 변환할까?

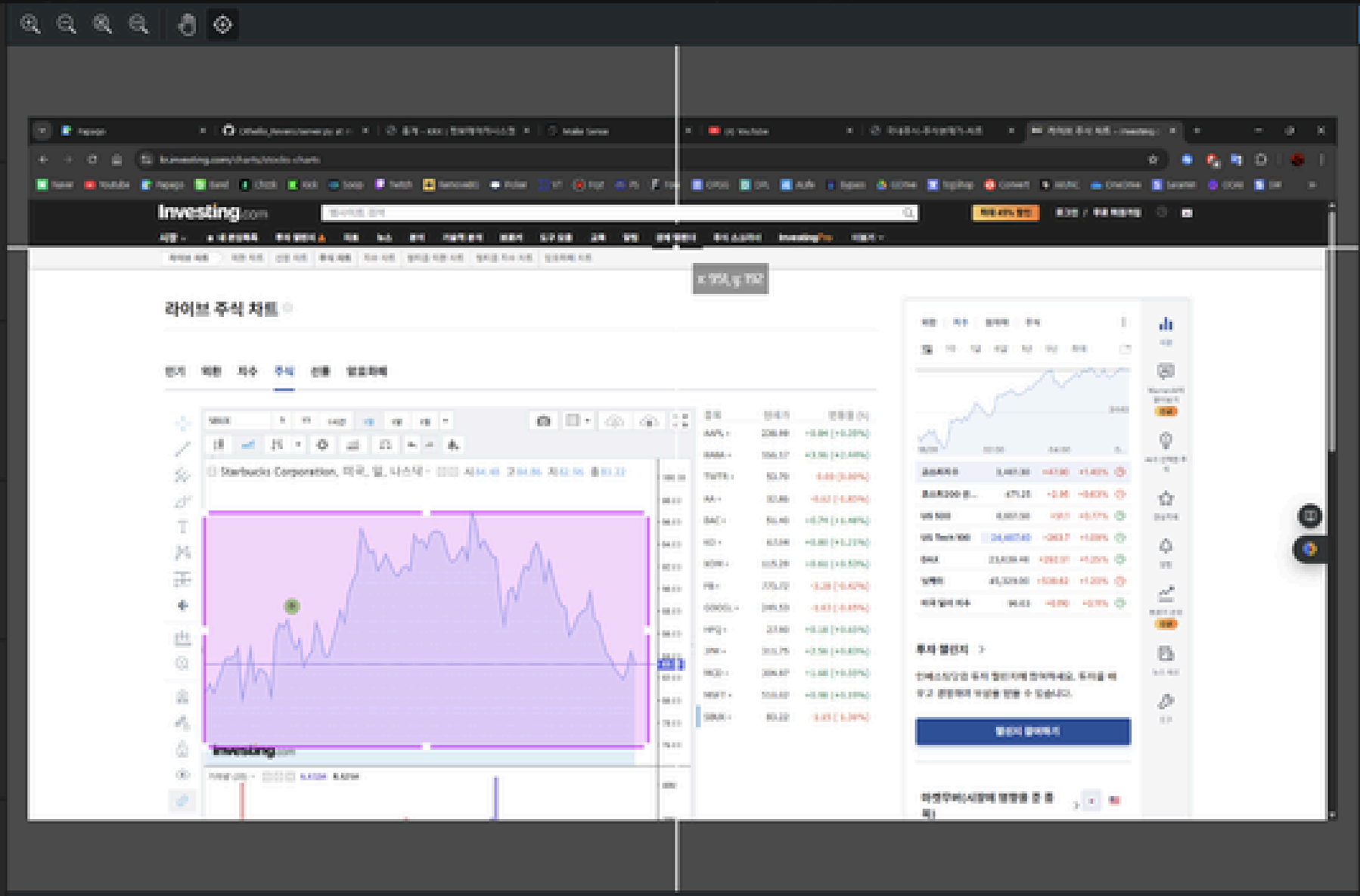


- 어디서 학습했든 다른 환경에서도 쉽게 사용할 수 있음
- 배포 및 통합 관리 용이성

Make Sense

makesense.ai

Make Sense | Actions | Community | Project Name: my-project-name



Rect

graph

Point

Line

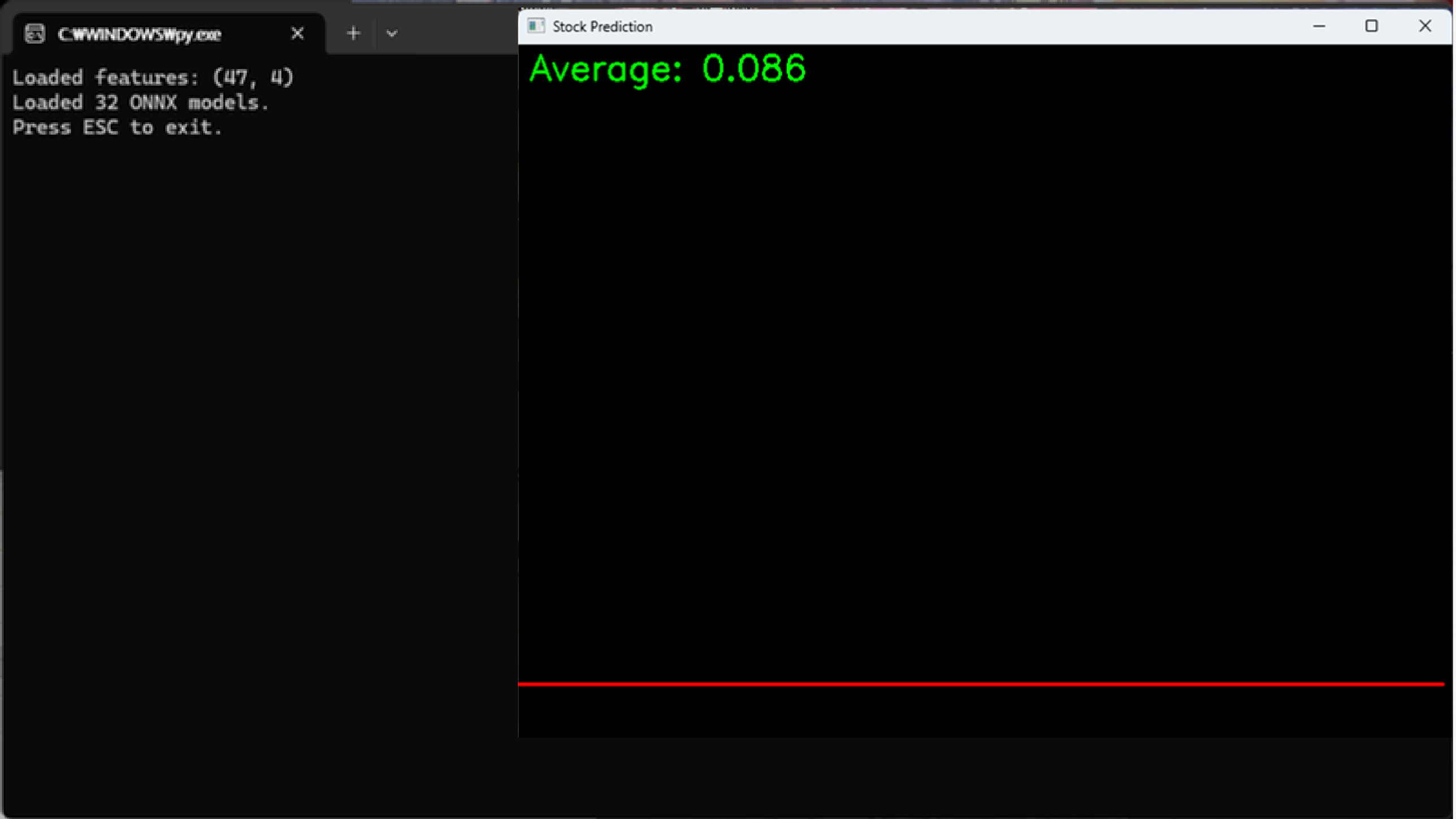
Polygon

스크린샷 2025-09-18 203824.png

```python
import sys
import numpy as np
import onnxruntime as ort
import cv2
from pathlib import Path
import time
import os
import math

labels_folder = Path("labels")
models_folder = Path("models")

# 폴더 존재 여부 확인
if not labels_folder.exists():
    sys.exit(f"Labels folder not found: {labels_folder}")
if not models_folder.exists():
    sys.exit(f"Models folder not found: {models_folder}")

features_list = []

for file in labels_folder.iterdir():
    if file.suffix == ".txt":
        with open(file, "r") as f:
            for line in f:
                parts = list(map(float, line.strip().split()))
                features = parts[1:]  # 첫번째 같은 라벨, 나머지가 feature
                features_list.append(features)

features_array = np.array(features_list, dtype=np.float32)  # shape (N, feature_size)
batch_size, feature_size = features_array.shape
print(f"Loaded features: {features_array.shape}")

onnx_files = [f for f in models_folder.iterdir() if f.suffix == ".onnx"]
if not onnx_files:
    sys.exit("No ONNX models found in the models folder.")

onnx_sessions = [ort.InferenceSession(f, providers=['CPUExecutionProvider']) for f in onnx_files]
print(f"Loaded {len(onnx_sessions)} ONNX models.")

def pulsar_avg(values, cut_ratio=0.15):
    sorted_values = np.sort(values)
    n = len(sorted_values)
    k = int(n * cut_ratio)
    if n - 2 * k <= 0:
        return np.mean(sorted_values)
    return np.mean(sorted_values[k:n - k])

prediction_table = []
max_table = 100
print("Press ESC to exit.")

while True:
    if cv2.waitKey(100) & 0xFF == 27:  # ESC 키
        break

    model_preds = []
    for session in onnx_sessions:
        input_name = session.get_inputs()[0].name
        pred = session.run(None, {input_name: features_array})[0]  # shape (N,1)
        model_preds.append(pred.flatten())

    # 모델별 평균
    model_preds = np.array(model_preds)
    avg_pred = model_preds.mean(axis=0)  # shape (N,)

    # 펄사평균
    smooth_pred = pulsar_avg(avg_pred)
    prediction_table.append(smooth_pred)
    if len(prediction_table) > max_table:
        prediction_table.pop(0)

    img = np.zeros((600, 800, 3), dtype=np.uint8)
    for i in range(1, len(prediction_table)):
        x1, y1 = int((i-1)*(800/max_table)), int(600 - prediction_table[i-1]*600)
        x2, y2 = int(i*(800/max_table)), int(600 - prediction_table[i]*600)
        cv2.line(img, (x1, y1), (x2, y2), (0, 0, 255), 2)
    cv2.putText(img, f"Average: {smooth_pred:.3f}", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

    cv2.imshow("Stock Prediction", img)

cv2.destroyAllWindows()
```


```python
import sys
import numpy as np
import onnxruntime as ort
import cv2
from pathlib import Path
import time
import os
import math

labels_folder = Path("labels")
models_folder = Path("models")

# 폴더 존재 여부 확인
if not labels_folder.exists():
    sys.exit(f"Labels folder not found: {labels_folder}")
if not models_folder.exists():
    sys.exit(f"Models folder not found: {models_folder}")

features_list = []

for file in labels_folder.iterdir():
    if file.suffix == ".txt":
        with open(file, "r") as f:
            for line in f:
                parts = list(map(float, line.strip().split()))
                features = parts[1:]  # 첫번째 같은 라벨, 나머지가 feature
                features_list.append(features)

features_array = np.array(features_list, dtype=np.float32)  # shape (N, feature_size)
batch_size, feature_size = features_array.shape
print(f"Loaded features: {features_array.shape}")

onnx_files = [f for f in models_folder.iterdir() if f.suffix == ".onnx"]
if not onnx_files:
    sys.exit("No ONNX models found in the models folder.")

onnx_sessions = [ort.InferenceSession(f, providers=['CPUExecutionProvider']) for f in onnx_files]
print(f"Loaded {len(onnx_sessions)} ONNX models.")

def pulsar_avg(values, cut_ratio=0.15):
    sorted_values = np.sort(values)
    n = len(sorted_values)
    k = int(n * cut_ratio)
    if n - 2 * k <= 0:
        return np.mean(sorted_values)
    return np.mean(sorted_values[k:n - k])

prediction_table = []
```


Discord RPC Connected to Discord


C:\WINDOWS\py.exe
Loaded features: (47, 4)
Loaded 32 ONNX models.
Press ESC to exit.
Stock Prediction
Average: 0.086

왜 잘 작동하지 않았을까?

데이터 특성과 모델 불일치

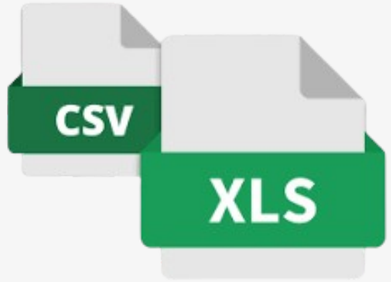
- YOLO는 이미지 객체 탐지에 최적화되어 있음
- 주식 데이터는 시계열 데이터로, 연속적 변화와 패턴의 값을 구해야함
- 이미지 기반 접근은 시간적 흐름을 반영할 수 없음 → 정확한 예측 어려움

정보 손실 문제

- 작은 변동이나 추세를 모델이 제대로 학습하기 어려움
- 그래프를 이미지로 변환하면, 숫자 값 자체의 정밀한 변화가 손실됨

“개선한 방법“

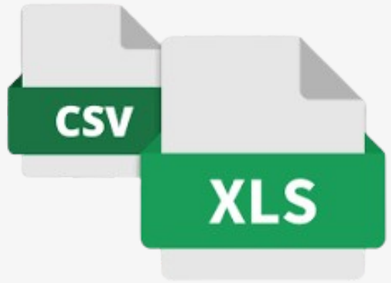
개선된 방법 (작동 원리)

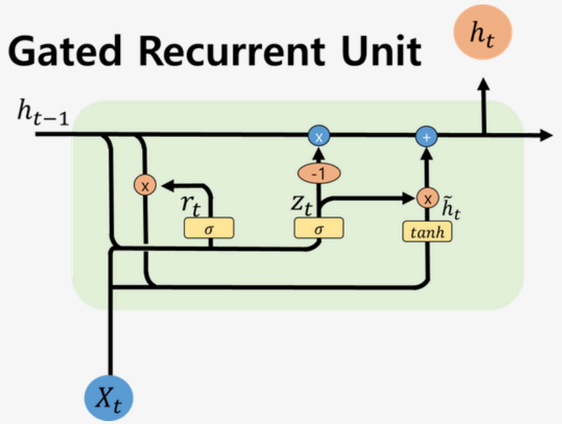


엑셀(CSV) 파일

주식 사이트에서 제공하는 주가
데이터를 담은 엑셀 파일에서
날짜별 시세, 거래량 등의 정보 추출

개선된 방법 (작동 원리)



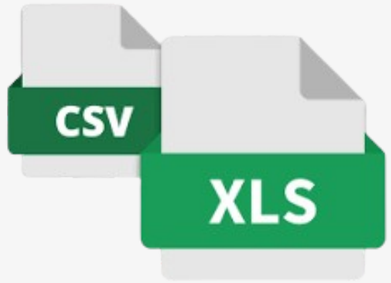


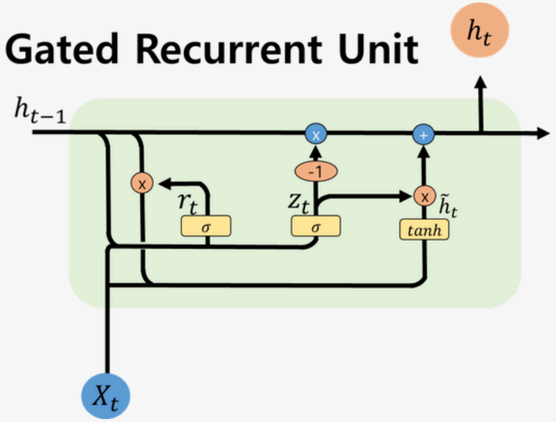
엑셀(CSV) 파일 → **GRU**

주식 사이트에서 제공하는 주가
데이터를 담은 엑셀 파일에서
날짜별 시세, 거래량 등의 정보 추출

“Gated Recurrent Unit”
시계열 데이터 예측에 강한 순환
신경망(RNN) 모델

개선된 방법 (작동 원리)





엑셀(CSV) 파일

주식 사이트에서 제공하는 주가 데이터를 담은 엑셀 파일에서 날짜별 시세, 거래량 등의 정보 추출

GRU

“Gated Recurrent Unit”
시계열 데이터 예측에 강한 순환 신경망(RNN) 모델

AI 추론하기

Numpy, Pandas, GRU를 이용해 주가 데이터로 미래 시세 추론

```python
import sys
import msvcrt
import math
import numpy as np
import onnxruntime as ort
import cv2
from PIL import Image, ImageGrab
import os
from pathlib import Path
import time
import torch
import torch.nn as nn
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import pandas as pd
import plotext as plot
import json

stock_ai_ascii_art = r"""
  /$$$$$$   /$$                          /$$             /$$$$$$  /$$$$$$
 /$$__  $$ | $$                         | $$            /$$__  $$|_  $$_/
| $$  \__//$$$$$$    /$$$$$$   /$$$$$$$| $$   /$$     | $$  \ $$  | $$
|  $$$$$$|_  $$_/   /$$__  $$ /$$_____/| $$  /$$/     | $$$$$$$$  | $$
 \____  $$ | $$    | $$  \ $$| $$      | $$$$$$/      | $$__  $$  | $$
 /$$  \ $$ | $$ /$$| $$  | $$| $$      | $$_  $$      | $$  | $$  | $$
|  $$$$$$/ |  $$$$/|  $$$$$$/|  $$$$$$$| $$ \  $$     | $$  | $$ /$$$$$$
 \______/   \___/   \______/  \_______/|__/  \__/     |__/  |__/|______/
"""

o_ascii_art = r"""
  /$$$$$$
 /$$__  $$
| $$  \ $$
| $$  | $$
| $$  | $$
| $$  | $$
|  $$$$$$/
 \______/
"""

x_ascii_art = r"""
 /$$   /$$
| $$  / $$
|  $$/ $$/
 \  $$$$/
  >$$  $$
 /$$/\  $$
| $$  \ $$
```

```python
sys.stdout.write(stock_ai_ascii_art + "\n")

# exe나 일반 파이썬으로 실행할때 오류 나지 않게 하기
def resource_path(relative_path):
    # exe로 실행할때
    if getattr(sys, 'frozen', False):
        base_path = Path(sys.executable).parent
    else:
        # 일반 Python
        base_path = Path(__file__).parent
    return base_path / relative_path

# 폴더 경로 설정
csv_folder_parent_folder_path = resource_path("csv")
models_folder_path = resource_path("models")
config_folder_path = resource_path("config")
settings_file_path = config_folder_path / "settings.json"

# 'settings.json' 기본 설정 파일
settings_data = {
    "graph": True,
    "graph_visualization": "matplotlib.pyplot",
    "trimmed_mean_percentage": 1,
    "matplotlib_pyplot_asynchronous": False,
    "br": "br",
    "kor_comment_graph": "예상 그래프를 보여줄지",
    "kor_comment_graph_visualization": "plotext: cmd 안에서 점이나 선으로, matplotlib.pyplot: 외부 창에서",
    "kor_comment_trimmed_mean_percentage": "절사평균값 (10이면 상위/하위 10% 제거)",
    "kor_comment_matplotlib.pyplot_asynchronous": "비동기 사용여부(사용안함: false/ 사용함: (몇 분 후에 창이 꺼질지))",
    "bl": "bl",
    "eng_comment_graph": "Whether to show the predicted graph",
    "eng_comment_graph_visualization": "plotext: dots and lines in the CMD, matplotlib.pyplot: in an anoter external window",
    "eng_comment_trimmed_mean_percentage": "Trimmed mean value (10 means removing the top and bottom 10%)",
    "eng_comment_matplotlib.pyplot_asynchronous": "Use asynchronous mode (False: disabled / [minutes]: enabled, window will close after
}

# 'config_folder', 'settings.json' 없을시 생성, 있으면 계속하기
if not config_folder_path.exists():
    sys.stdout.write("'config folder' does not exist.\n")
    sys.stdout.write(f"Creating 'config folder' into: {config_folder_path}\n")
    config_folder_path.mkdir(parents=True)
    with open(settings_file_path, "w", encoding="utf-8") as f:
        json.dump(settings_data, f, ensure_ascii=False, indent=4)
    sys.stdout.write(f"Automatically creating 'settings folder' into: {settings_file_path}\n")
elif not settings_file_path.exists():
    sys.stdout.write("'config folder' found!\n")
    sys.stdout.write(f"'config folder' location: {config_folder_path}\n")
    with open(settings_file_path, "w", encoding="utf-8") as f:
```


Launchpad Run Testcases 0 ⚠ 0 Discord RPC Connected to Discord

server.py

C: > Users > Unknown User > Desktop > Stock_AI > server.py > ...

```python
# 'config_folder', 'settings.json' 없을시 경로, 모드만 체크하기
if not config_folder_path.exists():
    sys.stdout.write("'config folder' does not exist.\n")
    sys.stdout.write(f"Creating 'config folder' into: {config_folder_path}\n")
    config_folder_path.mkdir(parents=True)
    with open(settings_file_path, "w", encoding="utf-8") as f:
        json.dump(settings_data, f, ensure_ascii=False, indent=4)
    sys.stdout.write(f"Automatically creating 'settings folder' into: {settings_file_path}\n")
elif not settings_file_path.exists():
    sys.stdout.write("'config folder' found!\n")
    sys.stdout.write(f"'config folder' location: {config_folder_path}\n")
    with open(settings_file_path, "w", encoding="utf-8") as f:
        json.dump(settings_data, f, ensure_ascii=False, indent=4)
    sys.stdout.write("'settings.json' does not exist.\n")
    sys.stdout.write(f"Creating 'settings.json' into: {settings_file_path}")
else:
    sys.stdout.write("'config folder' found!\n")
    sys.stdout.write(f"'config folder' location: {config_folder_path}\n")
    sys.stdout.write("'settings.json' found!\n")
    sys.stdout.write(f"'settings.json' location: {settings_file_path}\n")
    sys.stdout.write("You can change settings in the 'settings.json'.\n")

# 'settings.json' 안에 설정된 값을 불러오기
with open(settings_file_path, "r", encoding="utf-8") as f:
    settings = json.load(f)
graph = settings["graph"]
graph_visualization = settings["graph_visualization"]
trimmed_mean_percentage = settings["trimmed_mean_percentage"]
asynchronous = settings["matplotlib_pyplot_asynchronous"]
sys.stdout.write(f"Graph is set to '{graph}'.\n")
sys.stdout.write(f"Graph visualization is set to '{graph_visualization}'.\n")
sys.stdout.write(f"Trimmed mean value is set to '{trimmed_mean_percentage}%'.\n")
sys.stdout.write(f"Asynchronous is set to '{asynchronous}'\n")

# 사용자로부터 받은 'graph' 값이 이상한 값인지 확인하고, 이상한 값이면 자동으로 기본값으로 설정
if not isinstance(graph, bool):
    sys.stdout.write("Invalid graph value!\n")
    graph = True
    sys.stdout.write(f"Automatically set 'graph' value to '{graph}'!\n")
    sys.stdout.write("Please open 'settings.json' and set the values as described in the comments.\n")

# 사용자로부터 받은 'graph_visualization' 값이 이상한 값인지 확인하고, 이상한 값이면 자동으로 기본값으로 설정
if graph_visualization not in ["plotext", "matplotlib.pyplot"]:
    sys.stdout.write("Invalid graph_visualization value!\n")
    graph_visualization = "matplotlib.pyplot"
    sys.stdout.write(f"Automatically set 'graph_visualization' value to '{graph_visualization}'!\n")
    sys.stdout.write("Please open 'settings.json' and set the values as described in the comments.\n")
```

server.py

C: > Users > Unknown User > Desktop > Stock_AI > server.py > ...

```python
# 사용자로부터 받은 'trimmed_mean_percentage' 값이 이상한 값인지 확인하고, 이상한 값이면 자동으로 기본값으로 설정
if not isinstance(trimmed_mean_percentage, int) or trimmed_mean_percentage <= 0:
    sys.stdout.write("Invalled trimmed_mean_percentage value!\n")
    trimmed_mean_percentage = 1
    sys.stdout.write(f"Automatically set 'trimed_mean_percentage' value to '{trimmed_mean_percentage}'!\n")
    sys.stdout.write("Please open 'settings.json' and set the values as described in the comments.\n")

# 사용자로부터 받은 'matplotlib_pyplot_asynchronous' 값이 이상한 값인지 확인하고, 이상한 값이면 자동으로 기본값으로 설정
if not isinstance(asynchronous, (int, bool)):
    sys.stdout.write("Invalled matplotlib_pyplot_asynchronous value!\n")
    asynchronous = False
    sys.stdout.write(f"Automatically set 'matplotlib_pyplot_asynchronous' value to '{asynchronous}'!\n")
    sys.stdout.write("Please open 'settings.json' and set the values as described in the comments.\n")
elif isinstance(asynchronous, int) and asynchronous <= 0:
    sys.stdout.write("Invalled matplotlib_pyplot_asynchronous value!\n")
    asynchronous = 5
    sys.stdout.write(f"Automatically set 'matplotlib_pyplot_asynchronous' value to '{asynchronous}'!\n")
    sys.stdout.write("Please open 'settings.json' and set the values as described in the comments.\n")

# 'csv_folder_parent_folder' 폴더 있는지 확인, 없으면 생성하기
if not csv_folder_parent_folder_path.exists():
    sys.stdout.write("'csv folder's parent folder' does not exist.\n")
    sys.stdout.write(f"Creating 'csv folder's parent folder' into: {csv_folder_parent_folder_path}\n")
    csv_folder_parent_folder_path.mkdir(parents=True)
else:
    sys.stdout.write("'csv folder's parent folder' found!\n")
sys.stdout.write(f"'csv folder's parent folder' location: {csv_folder_parent_folder_path}\n")

# 'csv_folder' 폴더들을 리스트화
csv_folder = [i for i in csv_folder_parent_folder_path.iterdir() if i.is_dir()]

if csv_folder:
    # 'csv_folder_parent_folder' 안에 파일 있으면 사용가능한 파일 출력하기
    sys.stdout.write("'csv folders' found!\n")
    sys.stdout.write(" \n")
    sys.stdout.write("Availavle 'csv folders':\n")
    count = 1
    for idk in csv_folder:
        sys.stdout.write(f"{count}: {idk.name}\n")
        count += 1
    sys.stdout.write(" \n")
    sys.stdout.write("Select a 'csv folder' u want to load: \n")

    #사용자로부터 사용할 'csv_folder' 입력받기
    while True:
        if len(csv_folder) == 1:
            sys.stdout.write("Only one folder available, automatically selected.\n")
            useringput = 1
```

Launchpad | Run Testcases | 0 ⚠ 0 | Discord RPC | Connected to Discord

```python
    #사용자로부터 사용할 'csv_folder' 입력받기
    while True:
        if len(csv_folder) == 1:
            sys.stdout.write("Only one folder available, automatically selected.\n")
            userinput = 1
            break

        userinput = sys.stdin.readline().strip()
        if not userinput.isdigit():
            sys.stdout.write("Please enter a number!\n")
            continue

        userinput = int(userinput)
        if userinput not in range(1, count):
            sys.stdout.write("Invalid number. Try again: \n")
            continue

        break
else:
    sys.stdout.write("'csv folders' does not exist.\n")
    sys.stdout.write("Please make at least one 'csv folder' to proceed!\n")
    sys.stdout.write("The program will exit in 5 seconds...\n")
    time.sleep(5)
    sys.exit()

# 'csv_folder' 경로 설정
csv_folder_path = csv_folder[userinput - 1]

# 'csv_file' 전부 다 리스트화
csv_file = [i for i in csv_folder_path.iterdir() if i.is_file() and i.suffix == ".csv"]

# 'csv_folder'안에 'csv_file'이 있는지 확인, 없으면 종료
if csv_file:
    sys.stdout.write(f"Selected {userinput}th csv file!\n")
    sys.stdout.write(f"'csv file' location: {csv_folder_path}\n")
else:
    sys.stdout.write("No 'csv files' found in the selected folder!\n")
    sys.stdout.write("The program will exit in 5 seconds...\n")
    time.sleep(5)
    sys.exit()

# 절사: 상위/하위 trimmed_mean_percentage% 제거
def trimmed_array(arr, percentage):
    n = len(arr)
    k = int(n * percentage / 100)
    if k >= n // 2:  # 데이터가 너무 적으면 그냥 원본 사용
        return arr
```


server.py


```python
# 절사: 상위/하위 trimmed_mean_percentage% 제거
def trimmed_array(arr, percentage):
    n = len(arr)
    k = int(n * percentage / 100)
    if k >= n // 2:  # 데이터가 너무 적으면 그냥 원본 사용
        return arr
    arr_sorted = np.sort(arr)
    lower = arr_sorted[k]
    upper = arr_sorted[-k-1]
    trimmed = np.clip(arr, lower, upper)
    return trimmed

# csv에서 가져올 열 값들
required_cols = ['종가', '시가', '고가', '저가', '거래량', '변동 %']
all_data = []

# 'csv_file'들을 모두 읽고 원하는 값 추출해내기
for f in csv_file:
    df = None
    #utf-8, cp949 모두에 맞춰서 csv 전처리하기
    for enc in ['utf-8', 'cp949']:
        try:
            df = pd.read_csv(f, encoding=enc, thousands=',')
            break
        except:
            continue
    if df is None:
        sys.stdout.write(f"[Skip] {f.name}: Cannot read\n")
        continue

    df.columns = df.columns.str.strip()
    if not all(col in df.columns for col in required_cols):
        sys.stdout.write(f"[Skip] {f.name}: Missing columns\n")
        continue

    # 전처리
    df['거래량'] = pd.to_numeric(df['거래량'].astype(str).str.replace('M','').str.strip(), errors='coerce').fillna(0)*1e6
    df['변동 %'] = pd.to_numeric(df['변동 %'].astype(str).str.replace('%','').str.strip(), errors='coerce').fillna(0)/100
    for col in ['종가','시가','고가','저가']:
        df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0)

    all_data.append(df[required_cols].values[::-1])

if not all_data:
    sys.stdout.write("No valid CSV data found!\n")
    sys.stdout.write("the program will exit in 5 seconds...\n")
    time.sleep(5)
    sys.exit()
```

```python
if not all_data:
    sys.stdout.write("No valid CSV data found!\n")
    sys.stdout.write("the program will exit in 5 seconds...\n")
    time.sleep(5)
    sys.exit()

all_data = np.concatenate(all_data, axis=0)
idk = all_data.shape[0]
sys.stdout.write(f"Loaded {idk} rows from CSV files.\n")
idk -= idk % 10
if 50 > idk // 10:
    EPOCHS = 50
else:
    EPOCHS = idk // 10
sys.stdout.write(f"Train the AI {EPOCHS} times.\n")

# 데이터 준비
SEQ_LENGTH = 10
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(all_data)

X_list, y_list = [], []
for i in range(len(data_scaled)-SEQ_LENGTH):
    X_list.append(data_scaled[i:i+SEQ_LENGTH])
    y_list.append(data_scaled[i+SEQ_LENGTH][0])  # 종가 예측

X = torch.tensor(np.array(X_list), dtype=torch.float32)
y = torch.tensor(np.array(y_list), dtype=torch.float32).unsqueeze(1)

# GRU 모델 정의
class GRUStock(nn.Module):
    def __init__(self, input_size=6, hidden_size=32, num_layers=1):
        super().__init__()
        self.gru = nn.GRU(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size,1)
    def forward(self,x):
        out,_ = self.gru(x)
        return self.fc(out[:,-1,:])

model = GRUStock()

# 학습하기
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
sys.stdout.write("AI training started...\n")
for epoch in range(EPOCHS):
    model.train()
    optimizer.zero_grad()
    output = model(X)
    loss = criterion(output, y)
    loss.backward()
    optimizer.step()
    if (epoch+1)%10==0:
        sys.stdout.write(f"Success {epoch+1}/{EPOCHS}, Loss: {loss.item():.6f}\n")

# 예측 및 시각화
model.eval()
with torch.no_grad():
    pred = model(X).numpy()
    pred_prices = scaler.inverse_transform(
        np.concatenate([pred, np.zeros((pred.shape[0], all_data.shape[1]-1))], axis=1)
    )[:,0]
    actual_prices = all_data[SEQ_LENGTH:,0]

# 예측값에 절사평균 적용
pred_prices_trimmed = trimmed_array(pred_prices, trimmed_mean_percentage)

# 미래 예측값 생성
future_days = 20
last_known_idx = len(actual_prices) - 1
future_pred = pred_prices_trimmed[-future_days:]
combined_future_pred = np.concatenate([[pred_prices_trimmed[-1]], future_pred])
x_future = range(last_known_idx, last_known_idx + len(combined_future_pred))

# 마지막 실제 값과 미래 예측값 저장하기
last_actual_price = actual_prices[-1]
last_future_price = combined_future_pred[-1]

# 화면에 그래프 띄우기
if graph and graph_visualization == "matplotlib.pyplot":
    sys.stdout.write(f"Draw a graph using {graph_visualization}.\n")
    plt.figure(figsize=(12, 6))
    plt.plot(actual_prices, label='Actual Price', color='black')
    plt.plot(pred_prices_trimmed, color='green', label=f'Predicted Price (Trimmed Mean)')
    plt.axvline(x=last_known_idx, color='red', linestyle='--', label='Future Prediction Start')
    plt.plot(x_future, combined_future_pred,
             '--', color='blue', label='Future Prediction')
    plt.title('Stock Price Prediction with Future Forecast')
    plt.xlabel('Days')
```

C: > Users > Unknown User > Desktop > Stock_AI > server.py > ...

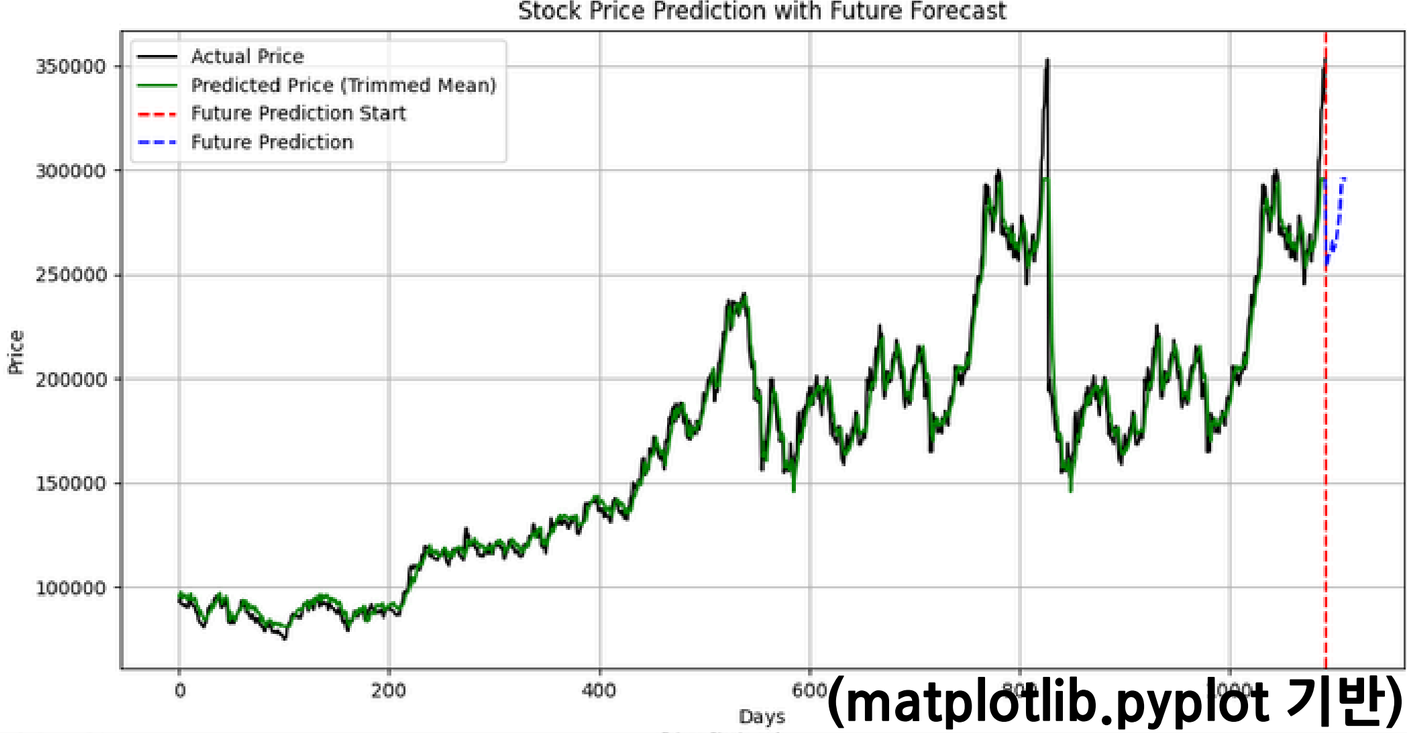
```python
    # 화면에 그래프 띄우기
    if graph and graph_visualization == "matplotlib.pyplot":
        sys.stdout.write(f"Draw a graph using {graph_visualization}.\n")
        plt.figure(figsize=(12, 6))
        plt.plot(actual_prices, label='Actual Price', color='black')
        plt.plot(pred_prices_trimmed, color='green', label=f'Predicted Price (Trimmed Mean)')
        plt.axvline(x=last_known_idx, color='red', linestyle='--', label='Future Prediction Start')
        plt.plot(x_future, combined_future_pred,
                 '--', color='blue', label='Future Prediction')
        plt.title('Stock Price Prediction with Future Forecast')
        plt.xlabel('Days')
        plt.ylabel('Price')
        plt.legend()
        plt.grid(True)
        if asynchronous == False:
            plt.show()
        elif isinstance(asynchronous, int) and asynchronous > 0:
            plt.show(block = False)
            plt.pause(asynchronous)
        else:
            sys.stdout.write("Invalid asynchronous value!\n")
            sys.stdout.write("Please open 'settings.json' and set the values as described in the comments.\n")
            sys.stdout.write("The program will exit in 5 seconds...\n")
            time.sleep(5)
            sys.exit()

    elif graph and graph_visualization == "plotext":
        sys.stdout.write(f"Draw a graph using {graph_visualization}.\n")
        plot.plot(actual_prices, color="white", label="Actual Price")
        plot.plot(pred_prices_trimmed, color="green", label="Predicted Price (Trimmed Mean)")
        plot.vline(last_known_idx, color="red")
        plot.plot(x_future, combined_future_pred, color="blue", marker="--", label="Future Prediction")
        plot.title("Stock Price Prediction with Future Forecast")
        plot.xlabel("Days")
        plot.ylabel("Price")
        plot.grid(True)
        plot.title('Legend:')
        plot.title('Actual Price: black')
        plot.title('Predicted Price: green')
        plot.title('Future Prediction: blue')
        plot.title('Future Start: red')
        plot.show()

    elif not graph:
        sys.stdout.write("Graphs are not drawn according to the settings.\n")

    else:
        sys.stdout.write("Invailed settings values!\n")
```

Launchpad Run Testcases 0 ⚠ 0 Discord RPC Connected to Discord

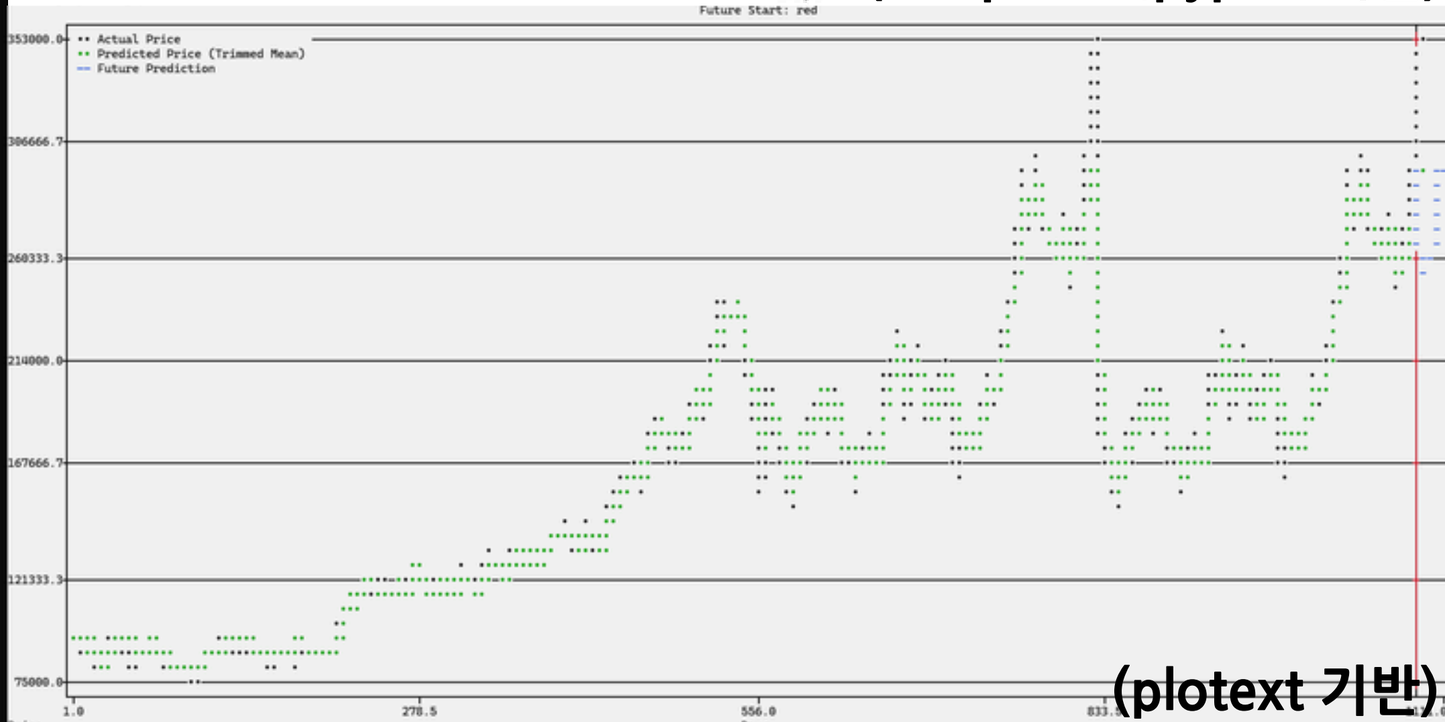
```
{
    "graph": true,
    "graph_visualization": "matplotlib.pyplot",
    "trimmed_mean_percentage": 1,
    "matplotlib_pyplot_asynchronous": false,
    "br": "br",
    "kor_comment_graph": "예상 그래프를 보여줄지",
    "kor_comment_graph_visualization": "plotext: cmd 안에서 점이나 선으로, matplotlib.pyplot: 외부 창에서",
    "kor_comment_trimmed_mean_percentage": "절사평균값 (10이면 상위/하위 10% 제거)",
    "kor_comment_matplotlib.pyplot_asynchronous": "비동기 사용여부(사용안함: False/ 사용함: (몇 분 후에 창이 꺼질지))",
    "bl": "bl",
    "eng_comment_graph": "Whether to show the predicted graph",
    "eng_comment_graph_visualization": "plotext: dots and lines in the CMD, matplotlib.pyplot: in an anoter external window",
    "eng_comment_trimmed_mean_percentage": "Trimmed mean value (10 means removing the top and bottom 10%)",
    "eng_comment_matplotlib.pyplot_asynchronous": "Use asynchronous mode (False: disabled / [minutes]: enabled, window will clo
}
```

```python
        plot.grid(True)
        plot.title('Legend:')
        plot.title('Actual Price: black')
        plot.title('Predicted Price: green')
        plot.title('Future Prediction: blue')
        plot.title('Future Start: red')
        plot.show()

    elif not graph:
        sys.stdout.write("Graphs are not drawn according to the settings.\n")

    else:
        sys.stdout.write("Invailed settings values!\n")
        sys.stdout.write("Please open 'settings.json' and set the values as described in the comments.\n")
        sys.stdout.write("The program will exit in 5 seconds...")
        time.sleep(5)
        sys.exit()

    # 최종적으로 AI 예측 결과 출력하기
    sys.stdout.write(" \n")
    if last_actual_price > last_future_price:
        sys.stdout.write(f"Current value is higher than the value {future_days} days later!\n")
        sys.stdout.write("AI prediction result: You should NOT buy this stock!\n")
        sys.stdout.write(x_ascii_art + "\n")

    elif last_actual_price < last_future_price:
        sys.stdout.write(f"The value {future_days} days later is higher than the current value!\n")
        sys.stdout.write("AI prediction result: You should buy this stock!\n")
        sys.stdout.write(o_ascii_art + "\n")

    else:
        sys.stdout.write(f"The current value and {future_days} days later value are the same!\n")
        sys.stdout.write("AI prediction result: You can buy this stock if you want.\n")
        sys.stdout.write(same_ascii_art + "\n")

    # 아무 키 입력하면 프로그램 종료하기
    sys.stdout.write("Press any key to exit...\n")
    msvcrt.getch() # 아무 키 입력 대기
    sys.exit()
```

```
 /$$$$$$   /$$                          /$$           /$$$$$$  /$$$$$$
 /$$__  $$ | $$                         | $$          /$$__  $$|_  $$_/
| $$  \__//$$$$$$    /$$$$$$   /$$$$$$$| $$   /$$   | $$  \ $$  | $$
|  $$$$$$|_  $$_/   /$$__  $$ /$$_____/| $$  /$$/   | $$$$$$$$  | $$
 \____  $$ | $$    | $$  \ $$| $$      | $$$$$$/    | $$__  $$  | $$
 /$$  \ $$ | $$ /$$| $$  | $$| $$      | $$_  $$    | $$  | $$  | $$
|  $$$$$$/ |  $$$$/|  $$$$$$/|  $$$$$$$| $$ \  $$   | $$  | $$ /$$$$$$
 \______/   \___/   \______/  \_______/|__/  \__/   |__/  |__/|______/

'config folder' found!
'config folder' location: C:\Users\Unknown User\Desktop\Stock_AI\config
'settings.json' found!
'settings.json' location: C:\Users\Unknown User\Desktop\Stock_AI\config\settings.
You can change settings in the 'settings.json'.
Graph is set to 'True'.
Graph visualization is set to 'matplotlib.pyplot'.
Trimmed mean value is set to '1%'.
Asynchronous is set to 'False'
Invailed matplotlib_pyplot_asynchronous value!
Automatically set 'matplotlib_pyplot_asynchronous' value to '5'!
Please open 'settings.json' and set the values as described in the comments.
'csv folder's parent folder' found!
'csv folder's parent folder' location: C:\Users\Unknown User\Desktop\Stock_AI\csv
'csv folders' found!

Availavle 'csv folders':
1: samsung-electronix-co-ltd
2: sk-hynix-inc

Select a 'csv folder' u want to load:
1
Selected 1th csv file!
'csv file' location: C:\Users\Unknown User\Desktop\Stock_AI\csv\samsung-electroni
Loaded 1102 rows from CSV files.
Train the AI 110 times.
AI training started...
Success 10/110, Loss: 0.035480
Success 20/110, Loss: 0.018306
Success 30/110, Loss: 0.012305
Success 40/110, Loss: 0.005810
Success 50/110, Loss: 0.002232
Success 60/110, Loss: 0.001203
Success 70/110, Loss: 0.001170
Success 80/110, Loss: 0.001071
Success 90/110, Loss: 0.000981
Success 100/110, Loss: 0.000958
Success 110/110, Loss: 0.000934
Draw a graph using matplotlib.pyplot.

Current value is higher than the value 20 days later!
AI prediction result: You should NOT buy this stock!

 /$$   /$$
| $$  / $$
|  $$/ $$/
 \  $$$$/
  >$$  $$
 /$$/\  $$
| $$  \ $$
|__/  |__/

Press any key to exit...
```



(matplotlib.pyplot 기반)



(plotext 기반)

차별점

사용자 정의 설정 가능

- 여러가지 방식으로 그래프를 프린트 할 수 있음 (matplotlib.pyplot, plotext)
- 사용자가 입력한 값대로 절사평균값 처리
- 폴더 내에서 원하는 파일 선택 가능, 폴더가 없으면 자동으로 생성
- ‘settings.json’ 파일을 생성해 설정값 저장

절사평균 사용

- 주식 데이터에서 갑작스런 급등, 급락 같은 극단값이 전체평균을 왜곡하는 것을 방지

감사합니다.