"""

module\_03.py ──────────────────────────────────────────────────────────────

3단원: 통합 전략 판단 시스템

---------------------------------------------------------------------------

• 핵심 기능

1) 밸류에이션 점수 (3.1)

2) 기술적 지표 점수 (3.2)

3) 뉴스·감성 점수 (3.3)

4) 통합 스코어 + 추천 (3.4)

5) 매크로·리스크·시나리오 분기 등 고급 기능 (3.5, advanced=True)

• 외부 모듈 의존: module\_01, module\_02가 전달하는 fundamental\_df / meta\_info

• 최종 출력: dict {

'scores': {...각 세부 점수...},

'composite\_score': float,

'recommendation': str,

'explanation': str,

'risk\_metrics': {...},

'scenario': {...}

}

"""

from \_\_future\_\_ import annotations

import datetime as \_dt

import math

import warnings

from dataclasses import dataclass

from typing import Dict, List, Optional, Tuple

import numpy as np

import pandas as pd

import pandas\_ta as ta

import requests

import yfinance as yf

from bs4 import BeautifulSoup

from scipy.stats import norm

from sklearn.preprocessing import MinMaxScaler

# ----------------------------- 선택 라이브러리 ---------------------------- #

# (설치가 안 되어 있으면 graceful-degrade)

try:

from fredapi import Fred # 매크로 지표

except ImportError:

Fred = None # type: ignore

try:

from transformers import pipeline # 감성 분석

except ImportError:

pipeline = None # type: ignore

warnings.filterwarnings("ignore")

# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ UTILITIES ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ #

def \_percent\_rank(series: pd.Series, value: float, higher\_is\_better: bool) -> float:

"""value가 series에서 차지하는 백분위 (0~1)."""

if series.empty or series.nunique() == 1:

return 0.5 # 정보가 없으면 중간값

if not higher\_is\_better:

value = -value

series = -series

rank = (series < value).sum() / len(series)

return float(rank)

def \_latest\_quarter(date: \_dt.date) -> str:

"""YYYY-QQ 형식(예: 2025-Q2) 반환."""

quarter = (date.month - 1) // 3 + 1

return f"{date.year}-Q{quarter}"

def \_safe\_mean(values: List[float]) -> float:

arr = [v for v in values if not math.isnan(v)]

return float(np.mean(arr)) if arr else float("nan")

# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ CORE CLASS ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ #

@dataclass

class StrategyAnalyzer:

ticker: str

fundamental\_df: pd.DataFrame # from module\_02 → PER, PBR, ROE 등 + 'sector'

meta\_info: Dict[str, any] # from module\_01 → 상장일, 시가총액 등

period: str = "1y" # 가격데이터 기간

advanced: bool = True # 3.5 고급 기능 on/off

fred\_api\_key: Optional[str] = None

# ──────────────────────────────────────────────────────────── #

def \_\_post\_init\_\_(self):

self.\_fetch\_price\_data()

self.\_init\_macro\_sources()

self.today = \_dt.date.today()

self.latest\_q = \_latest\_quarter(self.today)

# ----------------------------- 데이터 수집 ----------------------------- #

def \_fetch\_price\_data(self):

"""yfinance 로 가격, 거래량 수집 + 기술지표 계산용 DF 생성."""

data = yf.download(self.ticker, period=self.period, auto\_adjust=True, progress=False)

if data.empty:

raise ValueError(f"Price data not found for {self.ticker}")

data.dropna(inplace=True)

# 기술 지표 추가

data["RSI14"] = ta.rsi(data["Close"], length=14)

macd = ta.macd(data["Close"])

data["MACD"], data["MACD\_SIGNAL"] = macd["MACD\_12\_26\_9"], macd["MACDs\_12\_26\_9"]

data["OBV"] = ta.obv(data["Close"], data["Volume"])

data["SMA50"] = ta.sma(data["Close"], length=50)

data["SMA200"] = ta.sma(data["Close"], length=200)

self.price\_df = data

def \_init\_macro\_sources(self):

self.fred = None

if self.advanced and Fred and self.fred\_api\_key:

self.fred = Fred(api\_key=self.fred\_api\_key)

# ------------------------- 3.1 밸류에이션 점수 ------------------------- #

def \_valuation\_score(self) -> Tuple[float, Dict[str, float]]:

"""

업종 내 PER/PBR Percentile 기반 점수 (높을수록 매력).

fundamental\_df에는 최소 ['PER', 'PBR', 'sector', 'ticker'] 필드가 필요.

"""

row = self.fundamental\_df[self.fundamental\_df["ticker"] == self.ticker]

if row.empty:

return (np.nan, {})

row = row.iloc[0]

sector = row["sector"]

sector\_df = self.fundamental\_df[self.fundamental\_df["sector"] == sector]

per\_score = 1 - \_percent\_rank(sector\_df["PER"], row["PER"], higher\_is\_better=False)

pbr\_score = 1 - \_percent\_rank(sector\_df["PBR"], row["PBR"], higher\_is\_better=False)

roa\_score = \_percent\_rank(sector\_df["ROE"], row["ROE"], higher\_is\_better=True)

# 단순 평균 → 0~100 변환

raw = np.array([per\_score, pbr\_score, roa\_score])

val\_score = float(np.nanmean(raw)) \* 100

details = {

"per\_score": per\_score \* 100,

"pbr\_score": pbr\_score \* 100,

"roe\_score": roa\_score \* 100,

}

return val\_score, details

# ------------------------- 3.2 기술적 지표 점수 ------------------------ #

def \_technical\_score(self) -> Tuple[float, Dict[str, float]]:

df = self.price\_df.copy()

latest = df.iloc[-1]

# 1) 추세: SMA50 vs SMA200, SMA50 기울기

trend\_up = 1 if latest["SMA50"] > latest["SMA200"] else 0

sma\_slope = (df["SMA50"].pct\_change().iloc[-20:]).mean()

slope\_score = np.clip((sma\_slope + 0.002) / 0.004, 0, 1) # -0.2%~+0.2% 범위 정규화

# 2) 오실레이터: RSI

if latest["RSI14"] < 30:

rsi\_score = 1.0 # 과매도 → 긍정

elif latest["RSI14"] > 70:

rsi\_score = 0.0 # 과매수 → 부정

else:

rsi\_score = 1 - abs(latest["RSI14"] - 50) / 20 # 50 중심

# 3) 모멘텀: MACD 히스토그램

macd\_hist = latest["MACD"] - latest["MACD\_SIGNAL"]

macd\_score = np.tanh(macd\_hist \* 5) \* 0.5 + 0.5 # -∞~∞ → 0~1

tech\_score = float(np.mean([trend\_up, slope\_score, rsi\_score, macd\_score])) \* 100

details = {

"trend\_score": trend\_up \* 100,

"sma\_slope\_score": slope\_score \* 100,

"rsi\_score": rsi\_score \* 100,

"macd\_score": macd\_score \* 100,

}

return tech\_score, details

# ----------------------- 3.3 뉴스·감성 점수 --------------------------- #

def \_sentiment\_score(self, max\_articles: int = 20) -> Tuple[float, Dict[str, any]]:

"""

기본형: 간단한 뉴스 헤드라인 감성 분석 (영문·국문 혼합 가능).

advanced=False 또는 transformers 미설치 시 -> 0.5(중립) 고정.

"""

if not pipeline:

return 50.0, {"articles": [], "method": "neutral\_default"}

nlp = pipeline("sentiment-analysis", model="distilbert-base-uncased-finetuned-sst-2-english")

# 간단히 구글 뉴스 RSS → BeautifulSoup 파싱

query = self.meta\_info.get("kor\_name", self.ticker)

url = f"https://news.google.com/rss/search?q={query}"

res = requests.get(url, timeout=10)

soup = BeautifulSoup(res.content, "xml")

items = soup.find\_all("item")[:max\_articles]

headlines = [it.title.text for it in items]

if not headlines:

return 50.0, {"articles": [], "method": "no\_news"}

results = nlp(headlines)

pos\_ratio = sum(1 for r in results if r["label"] == "POSITIVE") / len(results)

score = pos\_ratio \* 100

return score, {"articles": headlines, "method": "transformers\_distilbert"}

# --------------------------- 3.5 고급 기능 ---------------------------- #

def \_macro\_adjustment(self) -> float:

"""금리·CPI 기반 보정값 (-10 ~ +10). 값이 높을수록 상승장 유리."""

if not (self.advanced and self.fred):

return 0.0

try:

# 10y 금리, CPI 전년동월비

dgs10 = self.fred.get\_series\_latest\_release("DGS10")[-1]

cpi\_yoy = self.fred.get\_series\_latest\_release("CPIAUCSL")[-1]

# 예시: 금리↑ → 부정, CPI↓ → 긍정

rate\_adj = np.clip((3.0 - dgs10) / 3.0, -1, 1) # 기준치 3%

cpi\_adj = np.clip((cpi\_yoy - 260) / 20, -1, 1) # CPI 260 기준

return float((rate\_adj - cpi\_adj) \* 10) # -10~+10

except Exception:

return 0.0

def \_risk\_metrics(self, horizon: int = 60) -> Dict[str, float]:

"""VaR, CVaR, MDD 계산."""

closes = self.price\_df["Close"]

returns = closes.pct\_change().dropna()

last\_horizon = returns[-horizon:]

if last\_horizon.empty:

return {}

# Historical VaR 95%

var\_level = np.percentile(last\_horizon, 5)

cvar\_level = last\_horizon[last\_horizon <= var\_level].mean()

running\_max = closes.cummax()

drawdowns = (closes - running\_max) / running\_max

mdd = drawdowns.min()

return {

"VaR\_95": float(var\_level),

"CVaR\_95": float(cvar\_level),

"MDD": float(mdd),

}

def \_scenario\_branch(self, composite: float) -> Dict[str, any]:

"""

상승/보합/하락 3분기 시나리오 + 전략 제안 (예시).

단순 확률: composite\_score → 상승 확률 선형 매핑.

"""

up\_p = np.clip((composite - 40) / 60, 0, 1)

flat\_p = 1 - abs(composite - 50) / 50

down\_p = 1 - up\_p

# 정규화

s = up\_p + flat\_p + down\_p

up\_p, flat\_p, down\_p = up\_p / s, flat\_p / s, down\_p / s

strat = "공격형 성장" if up\_p > 0.5 else ("방어/가치" if down\_p > 0.4 else "균형형")

return {

"P(up)": round(up\_p, 2),

"P(flat)": round(flat\_p, 2),

"P(down)": round(down\_p, 2),

"recommended\_strategy": strat,

}

# -------------------------------------------------------------------- #

# PUBLIC MAIN INTERFACE #

# -------------------------------------------------------------------- #

def run(self, weights: Optional[Dict[str, float]] = None) -> Dict[str, any]:

"""

• weights = {'valuation':0.35, 'technical':0.35, 'sentiment':0.15, 'macro':0.15}

합계 1.0, None이면 기본값 사용.

• 반환: dict (점수·추천·설명·리스크·시나리오)

"""

default\_w = {"valuation": 0.35, "technical": 0.35, "sentiment": 0.15, "macro": 0.15}

if weights:

default\_w.update(weights)

w = default\_w

val\_score, val\_det = self.\_valuation\_score()

tech\_score, tech\_det = self.\_technical\_score()

sent\_score, sent\_det = self.\_sentiment\_score()

macro\_adj = self.\_macro\_adjustment()

# 가중 합계 → 0~100

base\_comp = (

val\_score \* w["valuation"]

+ tech\_score \* w["technical"]

+ sent\_score \* w["sentiment"]

+ (50 + macro\_adj) \* w["macro"]

)

comp = np.clip(base\_comp, 0, 100)

# 추천 등급

if comp >= 75:

reco = "Strong Buy"

elif comp >= 60:

reco = "Buy / Positive"

elif comp >= 45:

reco = "Neutral / Hold"

else:

reco = "Avoid / Risky"

explanation = (

f"Valuation {val\_score:0.1f}, Technical {tech\_score:0.1f}, "

f"Sentiment {sent\_score:0.1f}, MacroAdj {macro\_adj:+0.1f} → "

f"Composite {comp:0.1f} → {reco}"

)

risk = self.\_risk\_metrics()

scenario = self.\_scenario\_branch(comp) if self.advanced else {}

return {

"scores": {

"valuation": val\_score,

"technical": tech\_score,

"sentiment": sent\_score,

"macro\_adjust": macro\_adj,

\*\*val\_det,

\*\*tech\_det,

},

"composite\_score": comp,

"recommendation": reco,

"explanation": explanation,

"risk\_metrics": risk,

"scenario": scenario,

"latest\_quarter": self.latest\_q,

"timestamp": str(\_dt.datetime.now()),

}

# ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ STAND-ALONE EXECUTION TEST ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ #

if \_\_name\_\_ == "\_\_main\_\_":

"""

$ python module\_03.py 로 직접 실행해볼 때 간단 테스트.

Streamlit 앱에서는 StrategyAnalyzer 객체만 import 해서 run() 호출하면 됩니다.

"""

import json

sample\_fundamental = pd.DataFrame(

{

"ticker": ["005930.KS", "000660.KS", "051910.KS"],

"sector": ["Semiconductor", "Semiconductor", "Battery"],

"PER": [12.5, 18.2, 55.3],

"PBR": [1.6, 2.3, 8.1],

"ROE": [11.4, 15.2, 18.0],

}

)

sample\_meta = {"kor\_name": "삼성전자"}

analyzer = StrategyAnalyzer("005930.KS", sample\_fundamental, sample\_meta, period="1y", advanced=True)

result = analyzer.run()

print(json.dumps(result, indent=2, ensure\_ascii=False))