

1교시. AI 시대, 세무사의 생존 본능을 깨우다

"AI는 만능이 아닙니다. 하지만 '전문 지식'이 있는 당신이 쓰면 무기가 됩니다."

0. Ice Breaking: 나는 1년 동안 AI를 어떻게 사용했나? AI가 분석한 나!

- 내가 주로 사용하는 AI챗봇, chatgpt나 gemini에게 아래 프롬프트를 복사, 붙여넣기로 실행해보세요.

Role

당신은 개인 성장 분석가(Personal Growth Analyst)입니다. 사용자와의 대화 기록 및 메모리를 기반으로 데이터 기반의 통찰력 있는 회고 보고서를 작성하는 전문가입니다.

Context

- 사용자는 AI와 나눈 올 한 해(2025년) 대화를 바탕으로 자신에 대한 객관적인 회고를 원합니다.
- 분석 대상: AI 메모리에 저장된 사용자 정보, 대화 패턴, 관심사 변화, 주요 활동 등
- 목적: 자기 이해 증진 및 내년 계획 수립을 위한 인사이트 확보

Task

아래 영역별로 올 한 해 사용자에게 대해 학습한 내용을 분석하고 회고 보고서를 작성하세요.

1. ****[연간 핵심 키워드]****: 올해 사용자를 정의하는 3~5개의 핵심 키워드 도출
2. ****[영역별 패턴 분석]****
 - 직업/업무: 주로 어떤 업무를 했고, 어떤 방식으로 접근했는가?
 - 개인/관계: 개인적 삶에서 어떤 변화나 특징이 있었는가?
 - 지적 관심사: 어떤 주제에 관심을 보였고, 어떻게 변화했는가?
3. ****[성장 포인트]****: 올해 눈에 띄게 발전하거나 깊어진 영역은 무엇인가?
4. ****[일관된 강점]****: 대화 전반에서 일관되게 나타난 사용자의 강점
5. ****[2026년 예측 및 제안]****: 관찰된 패턴을 기반으로 한 내년 방향성 예측 및 성장 제안

Constraints

- 추상적 칭찬이 아닌, ****구체적 대화 사례나 메모리 내용을 근거로**** 분석할 것
- 어조: 따뜻하지만 객관적인 코치의 시선 (과도한 미화 금지)
- 예측은 "점술"이 아닌 ****관찰된 데이터 기반의 합리적 추론****으로 제시
- 없는 정보를 지어내지 말고, 정보가 부족한 영역은 솔직히 언급할 것

Output Format

📅 2025 연간 회고 리포트: [사용자 이름/닉네임]

📌 올해의 키워드

> [키워드1] | [키워드2] | [키워드3] | ...

🔍 영역별 패턴 분석

1. 직업/업무

(분석 내용 - 구체적 사례 포함)

2. 개인/관계

(분석 내용 - 구체적 사례 포함)

3. 지적 관심사

(분석 내용 - 구체적 사례 포함)

🌱 올해의 성장 포인트

(눈에 띄게 발전한 영역과 그 근거)

🏆 일관된 강점

(대화 전반에서 확인된 강점)

🌐 2026년 예측 및 제안

(데이터 기반 예측 + 구체적 제안)

이 보고서는 AI 대화 기록을 기반으로 작성되었으며, 전체 모습의 일부만을 반영합니다.

🏛️ [Fact Check] 미국 변호사의 법원 제재 사건

혹시 "AI가 써준 판례를 냈다가 망신당한 미국 변호사 이야기" 들어보셨습니까? 뜬소문이 아니라 실제 뉴욕 맨해튼 연방법원에서 벌어진 사건입니다.

- **사건명:** Mata v. Avianca, Inc.
- **사건 개요:** 항공사 손해 사건에서 변호인단이 제출한 의견서에 **6건의 판례**가 인용되어 있었는데, 확인 결과 **모두 실제로 존재하지 않는 가짜 판례**였습니다.
- **원인:** 변호사들이 챗GPT(ChatGPT)에게 판례를 찾아달라고 했고, AI가 지어낸 답변을 **검증 없이 (Verification failed)** 그대로 법원에 제출했습니다.
- **결과:** 법원은 이를 "**허위 인용(Fabricated citations)**"으로 규정하고 해당 소송을 기각했습니다. 또한 변호사와 소속 로펌에 **5,000달러(약 700만 원)의 벌금(Sanction)**을 부과했습니다. 판사는 변호사들의 **검증 소홀 (Negligence)**과 불성실한 태도를 질책했습니다.

1. 들어가며: 왜 세무사는 AI를 불신하는가?

🏛️ [Fact Check] 미국 변호사의 법원 제재 사건

혹시 "AI가 써준 판례를 냈다가 망신당한 미국 변호사 이야기" 들어보셨습니까? 뜬소문이 아니라 실제 뉴욕 맨해튼 연방법원에서 벌어진 사건입니다.

- **사건명:** Mata v. Avianca, Inc.
- **사건 개요:** 항공사 손해 사건에서 변호인단이 제출한 의견서에 **6건의 판례**가 인용되어 있었는데, 확인 결과 **모두 실제로 존재하지 않는 가짜 판례**였습니다.
- **원인:** 변호사들이 챗GPT(ChatGPT)에게 판례를 찾아달라고 했고, AI가 지어낸 답변을 **검증 없이 (Verification failed)** 그대로 법원에 제출했습니다.
- **결과:** 법원은 이를 "**허위 인용(Fabricated citations)**"으로 규정하고 해당 소송을 기각했습니다. 또한 변호사와 소속 로펌에 **5,000달러(약 700만 원)의 벌금(Sanction)**을 부과했습니다. 판사는 변호사들의 **검증 소홀(Negligence)**과 불성실한 태도를 질책했습니다.

AI는 '확률적 앵무새'다

이 변호사들은 왜 속았을까요? AI가 '**지식**'을 검색한 게 아니라 '**말**'을 지어냈기 때문입니다.

- AI는 진실을 말하는 기계가 아닙니다. "**다음에 올 가장 그럴듯한 단어**"를 확률적으로 예측하여 문장을 완성하는 기계입니다.
- 이를 전문 용어로 **할루시네이션(Hallucination, 환각)**이라고 합니다.

핵심 포인트

AI에게는 '**지식**'이 없습니다. '**언어 능력**'만 있습니다.

따라서 **지식(세법, 판례, 금융이론)**을 가진 **여러분(전문가)**이 검증하지 않으면 AI는 위험한 도구일 뿐입니다.

하지만 반대로, 여러분의 '**도메인 지식**'이 결합되어 AI를 통제할 수 있다면? AI는 최고의 파트너가 됩니다.

성소라 교수의 '휴먼코드'에서 제시하는 AI 시대 4단계 계급은 부르디외의 아비투스 개념을 차용하여, AI 활용 능력에 따른 인간의 새로운 신분 구조를 설명하며 **AI 종속자, AI 불균형자, AI 경계 파괴자, 그리고 궁극적으로는 AI를 재정의하는 존재**로 나뉩니다. 이는 단순히 기술을 잘 쓰는 것을 넘어, 인간 고유의 가치와 감각, 판단력을 통해 AI 시대의 주도권을 잡는 방법을 제시하는 틀입니다.

AI 시대 4단계 계급

1. AI 종속자 (AI Dependent):

- AI의 결과물을 비판 없이 수용하고 진리처럼 여기는 사람들.
- 기술에 의존하며 주도권을 잃는 단계.

2. AI 불균형자 (AI Imbalanced):

- AI를 활용하지만, 자신의 주관이나 가치 판단 없이 피상적으로만 사용하는 사람들.
- 기술의 편리함에 익숙해져 인간 고유의 능력을 퇴화시킬 위험이 있는 단계.

3. AI 균형자 (AI Balanced):

- 기술의 구조를 이해하면서도 자신만의 확고한 기준과 판단을 유지하는 사람들.
- 효율성보다는 방향을, 정답보다는 의미를 중시하며 AI를 단순한 도구가 아닌 파트너로 활용하는 단계.

4. AI 경계파괴자 (AI Boundary Breaker / AI 재정의자):

- AI를 도구로 활용하며, 기술과 인간의 경계를 허물고 새로운 가치를 창출하는 사람.
- AI 리터러시(이해 능력)와 문화 자본을 바탕으로 AI를 주도적으로 활용하여 창의성을 발휘하는 단계.

이 분류는 AI 시대를 살아가는 개인이 자신의 위치를 파악하고, 기술에 종속되지 않고 주도적인 삶을 살아가기 위한 '휴먼 코드'의 중요성을 강조하는데, 전문가 집단은 AI에게 일방적으로 위협받는 자리가 아닌 유리한 위치를 차지할 수 있습니다.

2. 어떤 AI를 써야 할까요? (3대장 비교)

"박사님, 챗GPT, 클로드, 제미니... 종류가 너무 많은데 뭘 써야 할까요?"

결론부터 말씀드리면, 우리 세무사에게는 구글의 Gemini(제미니)가 가장 강력한 무기가 됩니다.

생성형 AI 3대장, 딱 정해드립니다.

지금 시장은 이 3가지가 꼭 잡고 있습니다. 각각의 특징이 다릅니다.

이름	특징	추천 대상
ChatGPT (OpenAI)	가장 유명하고 범용적입니다. 플러그인이 많아 확장성이 좋습니다.	일반적인 사용자, 다양한 플러그인 활용 시
Claude (Anthropic)	'문과생' 같습니다. 한국어 작문 실력이 뛰어나고, 글을 매끄럽게 잘 다룹니다.	작가, 마케터, 고객 응대 메일 작성 시
Gemini (Google)	[우리의 선택] 구글 생태계(엑셀, 드라이브)와 연동되고, '엄청나게 긴 문서' 를 한 번에 읽습니다.	세무사 , 변호사, 연구원

왜 하필 'Gemini'인가요?

우리가 이번 교육에서 Gemini를 쓰는 이유는 명확합니다.

1. 가장 긴 문서를 읽을 수 있습니다 (Long Context Window)

- 세무사는 수백 페이지짜리 '세법'과 '판례'를 다뤄야 합니다.
- 다른 AI들이 "너무 길어서 못 읽어요"라고 할 때, Gemini는 **수천 페이지의 PDF**도 한 입에 삼키고 분석합니다. 우리에게겐 이게 핵심입니다.

2. 구글 워크스페이스와 한 몸입니다.

- 엑셀(Google Sheets)이나 이메일(Gmail) 많이 쓰시죠?
- Gemini는 여러분의 구글 드라이브에 있는 파일을 바로 가져오고, 분석 결과를 바로 이메일로 보낼 수 있습니다.

3. 최신 정보에 강합니다 (Google 검색 연동)

- 구글 검색 엔진을 기반으로 하기 때문에, 가장 최신의 세법 개정 사항이나 뉴스를 반영하여 답변할 확률이 높습니다.

유료 버전(Gemini Advanced), 꼭 써야 하나요?

결론: '세무사'라면 필수입니다. (월 29,000원)

모델이 달라서가 아닙니다. "**기억력의 크기**"가 다르기 때문입니다.

이유 1. 기억 용량의 차이 (Context Window)

- **무료 버전:** Gemini 3.0을 쓸 수 있지만, **기억 용량(Context Window)**이 작습니다.
 - 세법전이나 긴 판례 PDF를 업로드하면, "**앞 내용을 까먹거나**" 파일이 너무 크다고 거부합니다. 대화가 조금만 길어져도 문맥을 놓칩니다.
- **유료 버전:** **대용량 컨텍스트(Long Context)**를 지원합니다.
 - 수천 페이지짜리 세법 개론서, 수년 치 판례집을 통째로 업로드해도 토씨 하나 안 틀리고 다 기억합니다. 우리가 하려는 **RAG(판례 검색)** 실습의 핵심입니다.

이유 2. 사용량 제한 (Rate Limit)

- **무료 버전:** 질문을 연속으로 몇 번 던지면 "**잠시 후 다시 시도하세요**"라며 멈춥니다.
 - 실습 도중에 흐름이 끊기면 수업을 따라올 수 없습니다.
- **유료 버전:** 전문가의 헤비한 업무 패턴을 견딥니다. 코드를 수십 번 수정하고, 데이터를 반복해서 분석시켜도 끊임 없이 답변합니다.

이유 3. 구글 워크스페이스(Drive) 연동

- 내 구글 드라이브에 있는 고객의 재무제표 엑셀 파일, 스캔 된 계약서 PDF를 **업로드 없이 바로 끌어와서 분석**할 수 있습니다.
- 보안이 중요한 세무 자료를 매번 다운로드/업로드하는 번거로움과 위험을 없애줍니다.



강사의 팁:

엔진(Gemini 3.0)은 같습니다. 하지만 무료는 '**연료 탱크가 작은 경차**'이고, 유료는 '**무제한 주행이 가능한 대형 트럭**'입니다.

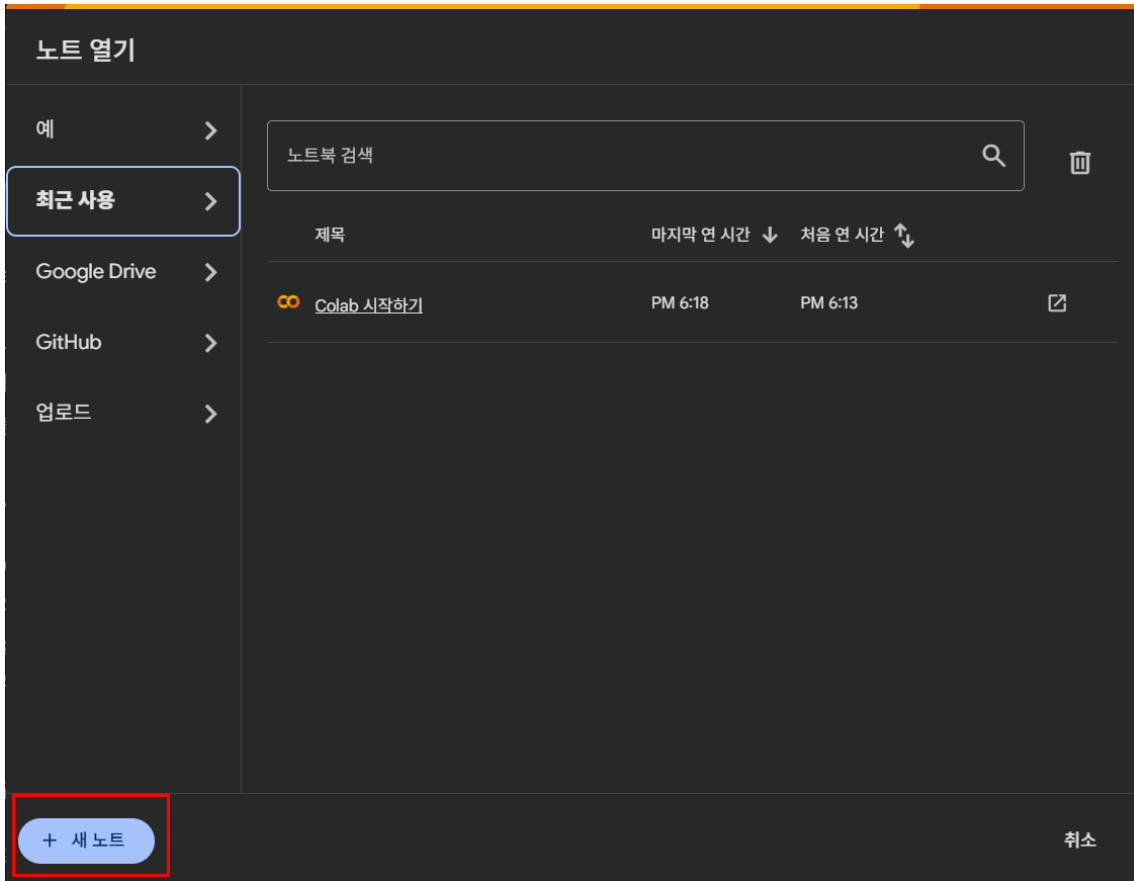
방대한 세법 데이터를 싣고 달리려면 트럭이 필요합니다.

3. [사전 준비] 구글 코랩(Colab) 준비하기

"코딩 프로그램을 내 컴퓨터에 설치해야 하나요?" 아니요!

구글이 빌려주는 **슈퍼컴퓨터(Colab)**를 웹브라우저에서 공짜로 씁니다. 우리는 여기서 AI가 짜준 코드를 '실행'만 하면 됩니다.

1. 구글 검색창에 '**구글 코랩**' 검색 (또는 colab.research.google.com 접속)
2. **[새 노트]** 파란색 버튼 클릭



3. 준비 끝입니다.

4. [무작정 따라하기 1] AI가 내일의 주가를 예측한다 (머신러닝)

저는 세무사이자 금융공학 박사입니다. 주가를 예측하는 '랜덤 포레스트(Random Forest)' 모델은 파이썬과 통계학을 깊게 알아야 만들 수 있었습니다.

하지만 이제는 코딩 몰라도 됩니다. **제가 가진 금융 지식을 글로 적어서 명령(프롬프트)**하면, AI가 코드를 짜줍니다.

전문가가 아니면 이렇게 구체적으로 명령할 수 없습니다. 여러분도 세법 분야에서 저처럼 하실 수 있습니다.

금융 전문가의 프롬프트

Step 1. 아래 프롬프트를 복사해서 Gemini(또는 ChatGPT)에게 붙여넣으세요.

효율적으로 한번에 구글코랩에서 실행할 수 있는 아래 내용의 코드를 작성하기 위한 프롬프트를 제안해줄래?

1. 한국 주식 "SK하이닉스(000660)"의 과거 10년치 주가정보(시가, 고가, 저가, 종가, 거래량)을 가지고 다음 거래일 주가의 종가를 예측하는 랜덤포레스트 모델 코드. (오늘을 기준으로 최종 7일 전까지만 사용)
2. 전체 기간을 7:3으로 나눠 학습/테스트 셋으로 사용하며, 정규화 작업을 하고, 종가 자체가 아닌 다음날 종가변화율(수익률)을 종속변수로 사용하고, 최종 그래프에서는 종가로 재변환 할 것
3. 구글코랩 하나의 코드 창에서 직접 실행
4. 주가정보는 `financedatareader`를 사용(설치 명령어 : `!pip install finance-datareader`)

5. 오늘을 기준으로 최종 7일치 정보는 모델의 신뢰성 확인용으로 모델을 테스트해서, 실제 증가와 주가의 상승/하락을 검증하고, 결과를 테이블로 나타낼 것.
6. 오늘의 마지막 데이터를 이용해서 내일의 증가와 상승/하락 방향 예측할 것
7. 실제 증가 데이터와 예측 데이터를 함께 시각화
8. 테스트 기간 전체의 데이터 시각화 및 정보 출력에 한글로 출력되게 하되, 구글코랩에서 한글이 깨지지 않도록 koreanize-matplotlib의 기본 글꼴 사용할 것. 오류가 없도록 신중하게 단계별로 생각해

Step 2. AI가 작성해 준 파이썬 코드를 '복사' 하세요.

Step 3. 아까 열어둔 '구글 코랩'에 붙여넣고 ▶ 버튼(실행)을 누르세요.

▼ 참고용 코드(클릭 하면 보여요!)

```
# =====
# 1. 환경 설정 및 라이브러리 설치
# =====
# 폰트 및 데이터 리더 설치 (출력 로그 최소화)
!pip install -q finance-datareader koreanize-matplotlib

import FinanceDataReader as fdr
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import koreanize_matplotlib # 한글 폰트 자동 적용
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import MinMaxScaler
from datetime import datetime, timedelta

# 경고 메시지 무시
import warnings
warnings.filterwarnings('ignore')

print(">>> 라이브러리 설치 및 임포트 완료")

# =====
# 2. 데이터 수집 및 전처리
# =====
# 종목코드 및 기간 설정
code = '000660' # SK하이닉스
today = datetime.today()
start_date = today - timedelta(days=365*10) # 10년 전

# 데이터 로드
df = fdr.DataReader(code, start_date)

# 파생변수 생성 (종속변수: 다음날 수익률)
# Close_target = (내일 증가 - 오늘 증가) / 오늘 증가
```

```

df['Return'] = df['Close'].pct_change() # 오늘의 수익률
df['Target'] = df['Return'].shift(-1) # 내일의 수익률 (우리가 예측할 값)

# 결측치 제거 (마지막 날은 내일 데이터가 없으므로 Target이 NaN임. 이는 미래 예측용으로 따로 저장)
last_row = df.iloc[[-1]] # 내일 예측을 위한 오늘의 데이터
df_clean = df.dropna() # 학습/테스트용 데이터

# =====
# 3. 데이터 분할 (검증용 7일 vs 모델링용)
# =====
# 최근 7거래일을 검증용(Validation)으로 분리
val_days = 7
df_model = df_clean.iloc[:-val_days] # 모델 학습/테스트용 (과거 ~ 7일 전)
df_valid = df_clean.iloc[-val_days:] # 최종 검증용 (최근 7일)

print(f">>> 전체 데이터 수: {len(df)}")
print(f">>> 학습/테스트용 데이터: {len(df_model)}건")
print(f">>> 최종 검증용(최근 7일) 데이터: {len(df_valid)}건")

# =====
# 4. 정규화 및 학습/테스트 셋 분리
# =====
feature_cols = ['Open', 'High', 'Low', 'Close', 'Volume']
X = df_model[feature_cols]
y = df_model['Target']

# 시계열 순서를 유지하기 위해 shuffle=False 설정
# 7:3 분할
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=False)

# 정규화 (MinMaxScaler)
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# =====
# 5. 모델링 (랜덤포레스트)
# =====
rf = RandomForestRegressor(n_estimators=100, random_state=42, n_jobs=-1)
rf.fit(X_train_scaled, y_train)

print(">>> 모델 학습 완료")

# =====
# 6. 예측 및 종가 변환
# =====
# 등락률 예측
y_pred_return = rf.predict(X_test_scaled)

```



```

# 예측된 등락률을 실제 증가로 변환
# 예측 증가 = 전일 증가 * (1 + 예측 등락률)
# 주의: X_test 데이터의 'Close'는 '오늘 증가'이므로 여기에 (1+예측수익률)을 곱하면 내일 증가가 됨
# 하지만 여기서는 시각화 매칭을 위해 인덱스를 맞춤
test_indices = y_test.index
prev_close = X.loc[test_indices, 'Close'] # 기준이 되는 당일 증가

pred_close = prev_close * (1 + y_pred_return)
actual_close = prev_close * (1 + y_test)

# =====
# 7. 시각화 (테스트 기간)
# =====
plt.figure(figsize=(14, 6))
plt.plot(actual_close.index, actual_close, label='실제 증가', color='blue', alpha=0.7)
plt.plot(pred_close.index, pred_close, label='예측 증가', color='red', alpha=0.7, linestyle='--')
plt.title(f'SK하이닉스 주가 예측 비교 (테스트 기간)', fontsize=16)
plt.xlabel('날짜')
plt.ylabel('주가 (원)')
plt.legend()
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.show()

# =====
# 8. 최종 7일치 검증 (신뢰성 확인)
# =====
print("\n" + "="*50)
print(f"[{datetime.today().strftime('%Y-%m-%d')}] 기준] 최근 {val_days}일 모델 신뢰성 검증")
print("="*50)

# 검증 데이터 준비
X_valid = df_valid[feature_cols]
X_valid_scaled = scaler.transform(X_valid)
y_valid_true_return = df_valid['Target']

# 예측
y_valid_pred_return = rf.predict(X_valid_scaled)

# 결과 정리
# [수정] y_valid_pred_return은 이미 numpy array이므로 .values를 붙이지 않습니다.
valid_result = pd.DataFrame({
    '날짜': df_valid.index,
    '기준종가(오늘)': df_valid['Close'].values,
    '실제_내일증가': df_valid['Close'].values * (1 + y_valid_true_return.values),
    '예측_내일증가': df_valid['Close'].values * (1 + y_valid_pred_return),
    '실제방향': np.where(y_valid_true_return > 0, '상승', '하락'),
    '예측방향': np.where(y_valid_pred_return > 0, '상승', '하락')
})

```

```

}))

# 방향 적중 여부
valid_result['적중여부'] = valid_result['실제방향'] == valid_result['예측방향']
valid_result['오차율(%)'] = np.abs((valid_result['실제_내일종가'] - valid_result['예측_내일종가'])
/ valid_result['실제_내일종가']) * 100

# 테이블 출력 (소수점 포매팅)
pd.options.display.float_format = '{:,.0f}'.format
print(valid_result[['날짜', '실제_내일종가', '예측_내일종가', '실제방향', '예측방향', '적중여부']])

# =====
# 9. 내일 주가 예측
# =====
# 오늘의 데이터로 내일 예측
last_X = last_row[feature_cols]
last_X_scaled = scaler.transform(last_X)
next_return = rf.predict(last_X_scaled)[0]

today_close = last_row['Close'].values[0]
tomorrow_close = today_close * (1 + next_return)
direction = "상승" if next_return > 0 else "하락"

print("\n" + "="*50)
print(f"🌐 [미래 예측] 오늘({last_row.index[0].strftime('%Y-%m-%d')}) 데이터 기준")
print(f" 현재 종가: {today_close:,.0f}원")
print("-" * 50)
print(f" 내일 예측 변동률: {next_return*100:.2f}%")
print(f" 내일 예상 주가 : {tomorrow_close:,.0f}원")
print(f" 예상 방향 : {direction} 🚀" if direction == "상승" else f" 예상 방향 : {direction} 📉")
print("="*50)

```

▼ 만약 오류가 발생한다면?(클릭 해보세요!)

```

>>> 라이브러리 설치 및 환경 설정 완료
>>> 데이터 수집 기간: 2024-12-13 ~ 2025-12-13
-----
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-3510057137.py in <cell line: 0>()
    88 # 결과를 데이터프레임으로 변환
    89 cols = ['수익률', '변동성', '샤프지수'] + list(tickers.values())
--> 90 results_df = pd.DataFrame(results.T, columns=cols)
    91
    92 # =====

2 frames
/usr/local/lib/python3.12/dist-packages/pandas/core/internals/construction.py in _check_val

```

```

ues_indices_shape_match(values, index, columns)
418     passed = values.shape
419     implied = (len(index), len(columns))
→ 420     raise ValueError(f"Shape of passed values is {passed}, indices imply {implied}")
421
422

```

ValueError: Shape of passed values is (10000, 8), indices imply (10000, 7)

이런 오류를 복사해서 AI에게 붙여넣기 하고 "오류를 해결해" 라고 해보세요.

```

>>> 라이브러리 설치 및 환경 설정 완료
>>> 데이터 수집 기간: 2024-12-13 ~ 2025-12-13
-----
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-3510057137.py in <cell line: 0>()
    88 # 결과를 데이터프레임으로 변환
    89 cols = ['수익률', '변동성', '샤프지수'] + list(tickers.values())
→   90 results_df = pd.DataFrame(results.T, columns=cols)
    91
    92 # =====

2 frames
/usr/local/lib/python3.12/dist-packages/pandas/core/internals/construction.py in _check_values_indices_shape_match(values, index, columns)
    418     passed = values.shape
    419     implied = (len(index), len(columns))
→   420     raise ValueError(f"Shape of passed values is {passed}, indices imply {implied}")
    421
    422

ValueError: Shape of passed values is (10000, 8), indices imply (10000, 7)
-----
오류를 해결해.

```

👉 결과 확인:

- 단 1분 만에 지난 10년 치 주가 데이터를 분석하여 **내일 SK하이닉스 주가가 오를지 내릴지** 예측해 줍니다.
- '랜덤 포레스트', '정규화', '증가변화율'... 프롬프트에 쓴 이 단어들이 바로 **도메인 지식**입니다.

5. [무작정 따라하기 2] AI가 ETF 포트폴리오를 최적화한다

"달걀을 한 바구니에 담지 말라." 많이 들어보셨죠?

하지만 실제로 '**가장 효율적인 투자 비율(Efficient Frontier)**'을 계산하려면 몬테카를로 시뮬레이션이라는 복잡한 수학이 필요합니다.

이번에도 제 금융 지식만으로 AI를 지휘해 보겠습니다.

자산배분 전문가의 프롬프트

Step 1. 아래 내용을 Gemini에게 요청하세요.

이번에도 역시 구글코랩으로 실행할 수 있는 아래 내용의 코드를 작성하기 위한 프롬프트를 제안해줘

1. 한국ETF 종목코드 "TIGER 미국 S&P500(360750), KODEX 미국채 10년선물(308620), KODEX 골드선물(132030), KODEX 미국달러선물(261240)" 4종목이 투자 대상 ETF야. 효율적프론티어 그래프를 만들고, 현재 시점의 가장 효율적인 포트폴리오 투자비중을 제안해야 해.
2. 구글코랩 하나의 코드 창에서 직접 실행
3. 추가정보는 financedatareader를 사용(설치 명령어 : !pip install finance-datareader)
4. 오늘을 기준으로 과거 1년치 정보를 활용하고, 몬테카를로 기법으로 1만개의 투자포트폴리오 생성
5. 각 자산의 종가변화 그래프, 상관관계히트맵, 효율적프론티어그래프 만들 것
6. 최대샤프, 최소변동성, 최대수익률, 동일투자비중시 각 '예상수익률, 변동성, 샤프비율' 출력할 것
7. 테스트 기간 전체의 데이터 시각화 및 정보 출력에 한글로 출력되게 하되, 구글코랩에서 한글이 깨지지 않도록 koreanize-matplotlib의 기본 글꼴 사용할 것. 오류가 없도록 신중하게 단계별로 생각해.

Step 2. 코드를 복사해서 구글 코랩에 붙여넣고 실행하세요.

▼ 참고용 코드(클릭 하면 보여요!)

```
# =====
# 1. 환경 설정 및 라이브러리 설치
# =====
# 필수 라이브러리 설치 (출력 로그 최소화)
!pip install -q finance-datareader koreanize-matplotlib

import FinanceDataReader as fdr
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import koreanize_matplotlib # 한글 폰트 자동 적용
from datetime import datetime, timedelta

# 경고 메시지 무시
import warnings
warnings.filterwarnings('ignore')

print(">>> 라이브러리 설치 및 환경 설정 완료")

# =====
# 2. 데이터 수집 및 전처리
# =====
```

```

# 종목 설정 (자산배분 4대장: 주식, 채권, 금, 달러)
tickers = {
    '360750': 'TIGER 미국S&P500',
    '308620': 'KODEX 미국채10년선물',
    '132030': 'KODEX 골드선물',
    '261240': 'KODEX 미국달러선물'
}

# 기간 설정 (오늘 기준 과거 1년)
end_date = datetime.today()
start_date = end_date - timedelta(days=365)

print(f">>> 데이터 수집 기간: {start_date.strftime('%Y-%m-%d')} ~ {end_date.strftime('%Y-%m-%d')}")

# 데이터 수집 및 병합
df_list = []
for code, name in tickers.items():
    df = fdr.DataReader(code, start_date, end_date)
    df = df[['Close']]
    df.columns = [name]
    df_list.append(df)

# 하나의 데이터프레임으로 합치기
portfolio_df = pd.concat(df_list, axis=1).dropna()

# =====
# 3. 수익률 및 변동성 계산
# =====
# 일간 수익률
daily_returns = portfolio_df.pct_change().dropna()

# 연간 기대 수익률 (252 거래일 기준)
annual_returns = daily_returns.mean() * 252

# 연간 공분산 행렬
daily_cov = daily_returns.cov()
annual_cov = daily_cov * 252

# =====
# 4. 몬테카를로 시뮬레이션 (10,000개)
# =====
num_portfolios = 10000
# [수정] 크기를 (3 + 종목수)로 변경하여 컬럼 수(7개)와 일치시킴
results = np.zeros((3 + len(tickers), num_portfolios))
risk_free_rate = 0.035 # 무위험 이자율 3.5% 가정

for i in range(num_portfolios):

```

```

# 1. 랜덤 비중 생성 (합이 10이 되도록)
weights = np.random.random(len(tickers))
weights /= np.sum(weights)

# 2. 포트폴리오 수익률 및 변동성 계산
p_return = np.sum(weights * annual_returns)
p_volatility = np.sqrt(np.dot(weights.T, np.dot(annual_cov, weights)))

# 3. 샤프 지수 계산
p_sharpe = (p_return - risk_free_rate) / p_volatility

# 4. 결과 저장
results[0,i] = p_return
results[1,i] = p_volatility
results[2,i] = p_sharpe
for j in range(len(weights)):
    results[3+j, i] = weights[j]

# 결과를 데이터프레임으로 변환
cols = ['수익률', '변동성', '샤프지수'] + list(tickers.values())
results_df = pd.DataFrame(results.T, columns=cols)

# =====
# 5. 최적 포트폴리오 찾기
# =====
# 1) 최대 샤프지수 (Max Sharpe)
max_sharpe_idx = results_df['샤프지수'].idxmax()
max_sharpe_port = results_df.iloc[max_sharpe_idx]

# 2) 최소 변동성 (Min Volatility)
min_vol_idx = results_df['변동성'].idxmin()
min_vol_port = results_df.iloc[min_vol_idx]

# 3) 최대 수익률 (Max Return)
max_ret_idx = results_df['수익률'].idxmax()
max_ret_port = results_df.iloc[max_ret_idx]

# 4) 동일 비중 (Equal Weight) - 별도 계산
eq_weights = np.array([0.25, 0.25, 0.25, 0.25])
eq_return = np.sum(eq_weights * annual_returns)
eq_volatility = np.sqrt(np.dot(eq_weights.T, np.dot(annual_cov, eq_weights)))
eq_sharpe = (eq_return - risk_free_rate) / eq_volatility
eq_port_data = [eq_return, eq_volatility, eq_sharpe] + list(eq_weights)
eq_port = pd.Series(eq_port_data, index=cols)

# =====
# 6. 시각화
# =====

```

```

plt.figure(figsize=(18, 12))

# (1) 자산별 기준가 변동 추이 (100으로 환산)
plt.subplot(2, 2, 1)
normalized_df = (portfolio_df / portfolio_df.iloc[0]) * 100
for col in normalized_df.columns:
    plt.plot(normalized_df.index, normalized_df[col], label=col)
plt.title('자산별 가격 변동 추이 (1년, 시작=100)')
plt.legend()
plt.grid(True, linestyle='--')

# (2) 상관관계 히트맵
plt.subplot(2, 2, 2)
sns.heatmap(portfolio_df.pct_change().corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('자산 간 상관관계 분석')

# (3) 효율적 투자선 (Efficient Frontier)
plt.subplot(2, 2, (3, 4)) # 아래쪽 전체 사용
plt.scatter(results_df['변동성'], results_df['수익률'], c=results_df['샤프지수'], cmap='viridis', s=10, alpha=0.5)
plt.colorbar(label='샤프 지수')

# 주요 포인트 표시
plt.scatter(max_sharpe_port['변동성'], max_sharpe_port['수익률'], marker='*', color='red', s=300, label='최대 샤프 (최적)')
plt.scatter(min_vol_port['변동성'], min_vol_port['수익률'], marker='*', color='blue', s=300, label='최소 변동성 (안정)')
plt.scatter(max_ret_port['변동성'], max_ret_port['수익률'], marker='*', color='green', s=300, label='최대 수익률 (공격)')
plt.scatter(eq_port['변동성'], eq_port['수익률'], marker='*', color='orange', s=300, label='동일 비중 (1/N)')

plt.title('효율적 투자선 (Efficient Frontier)')
plt.xlabel('변동성 (리스크)')
plt.ylabel('기대 수익률')
plt.legend(loc='upper left', fontsize=12)
plt.grid(True, linestyle='--')

plt.tight_layout()
plt.show()

# =====
# 7. 결과 테이블 출력
# =====
summary_df = pd.DataFrame({
    '최대 샤프 (추천)': max_sharpe_port,
    '최소 변동성 (안정)': min_vol_port,
    '최대 수익률 (공격)': max_ret_port,

```

```

    '동일 비중 (1/N)': eq_port
}).T

# 퍼센트 포매팅 등을 위한 출력 처리
print("\n" + "="*50)
print("📊 [포트폴리오 최적화 결과 요약]")
print("="*50)
display_summary = summary_df.copy()
display_summary['수익률'] = display_summary['수익률'].apply(lambda x: f"{x*100:.2f}%")
display_summary['변동성'] = display_summary['변동성'].apply(lambda x: f"{x*100:.2f}%")
display_summary['샤프지수'] = display_summary['샤프지수'].apply(lambda x: f"{x:.2f}")
for col in tickers.values():
    display_summary[col] = display_summary[col].apply(lambda x: f"{x*100:.2f}%")

print(display_summary)
print("="*50)
print(f"👉 현재 시점 가장 효율적인 투자(최대 샤프): {summary_df.index[0]}")

```

👉 결과 확인:

- AI가 1만 번의 가상 투자를 순식간에 시뮬레이션합니다.
- 그리고 "수익률을 극대화하려면 주식 40%, 금 30%... 이렇게 담으세요"라고 정확한 숫자를 제시합니다.
- '몬테카를로 기법', '샤프 지수'를 몰랐다면 AI에게 이런 일을 시킬 수 있었을까요?

6. 1교시 요약: 도메인 지식이 곧 경쟁력이다

1. **AI 맹신은 금물:** 미국 변호사 사례처럼, 전문가의 검증 없는 AI 사용은 재앙입니다.
2. **하지만 통제한다면?** 오늘 보신 것처럼 코딩을 몰라도 '금융 지식'만으로 복잡한 투자 모델을 만들 수 있습니다.
3. **세무사의 미래:** 다음 시간부터는 금융이 아닌, 우리의 본업인 '세법 지식'을 AI에게 주입하겠습니다.
 - 다음 시간: **세법 로봇 - 나만의 판례 비서 만들기**
 - 여러분이 직접 만든 로봇과 세무사회에서 만든 "AI 세무사"의 답변을 직접 비교해보고, 차이의 이유를 확인합니다.

← 뒤로 가기