🤖 모델링 담당: **재환, 민서**

🎯 주요 목표

- 전처리된 데이터를 기반으로 **XGBoost** 및 **LSTM** 모델을 개발하여 주가를 예측
- 두 모델의 성능을 비교하여 최적 모델을 선정하고 리포트로 작성

---

1. 데이터 준비

- `processed/` 폴더에 있는 전처리 완료 CSV 파일 불러오기
- 필요한 컬럼만 추출 (예: Date, Open, High, Low, Close, Volume)
- 날짜 기준으로 오름차순 정렬된 시계열 데이터 확인
- 종목별로 데이터를 분리하며, 예측 기준일은 **해당 주의 일요일**로 통일함 (예: AAPL.csv, MSFT.csv 등)

---

2. XGBoost 모델 개발

☑ 입력 데이터 구성

- 과거 n일치 데이터를 바탕으로 **해당 주 일요일의 종가(Close)**를 예측하는 supervised 학습 구조로 변환
- Feature 예시: 이동평균, 수익률, 거래량 변화율, 변동성 등 파생 변수 생성
- 입력값은 날짜별 수치 정보이며, 모델이 해당 날짜의 정보를 보고 그 다음 날짜의 종가를 예측하는 방식임

☑ 모델 학습 및 검증

- `train_test_split()`을 이용해 학습/검증 데이터 분리 (최근 데이터를 검증용으로 사용)
- `xgboost.XGBRegressor()`로 학습 수행
- 평가 지표: MAE, RMSE 등

☑ 하이퍼파라미터 튜닝

- GridSearchCV 또는 Optuna 등을 활용하여 성능 최적화
- 하이퍼파라미터란 모델의 학습 방식이나 구조를 설정하는 값들 (예: 학습률 `learning_rate`, 트리 수 `n_estimators`, 최대 깊이 `max_depth` 등)
- 튜닝을 통해 과적합을 방지하고 일반화 성능을 향상시킴

☑ 예측 결과 시각화 및 저장

- 날짜별 예측값 vs 실제값 그래프 시각화
- 20개 종목에 대해 예측 결과 저장
- 저장 경로: `target/XGB/<YYYY-MM-DD>_prediction.csv`

---

3. LSTM 모델 개발 (딥러닝)

☑ 데이터 구조 변환

- 시계열 데이터를 LSTM 입력 구조인 `(samples, time_steps, features)`로 변환
  - 예: 최근 10일의 데이터를 사용해 다음 날 예측 시 → `(n_samples, 10, feature 수)` 형태
- 모든 feature는 **MinMaxScaler**를 사용하여 0~1 사이로 정규화
  - LSTM은 정규화된 데이터에서 더 안정적인 성능을 보임

☑ 모델 구성

- Keras 또는 PyTorch 기반으로 구성
- 예시 구조: `LSTM(64) → Dropout(0.2) → Dense(1)`

☑ 학습 및 검증

- 손실 함수: MSE (Mean Squared Error)
- 옵티마이저: Adam
- 조기 종료(EarlyStopping)를 적용하여 과적합 방지
- Epoch 수와 batch size는 실험적으로 설정

☑ 결과 출력 및 저장

- 예측 결과 시각화 그래프 생성
- 예측값은 CSV 형식으로 저장: `target/LSTM/<YYYY-MM-DD>_prediction.csv`

---

4. 모델 성능 비교 및 리포트 작성

- XGBoost와 LSTM 모델 간의 성능 비교 (MAE, RMSE, 예측 정확도 등)
- 종목별로 어떤 모델이 더 효과적인지 평가
- 시각자료를 포함한 요약 리포트 작성
- 보고서 저장 예: `target/model_comparison_summary.pdf`

☑ 모델링팀 체크리스트

항목	완료 유무 [O / X]
전처리 데이터 불러오기 및 정리	O / X
XGBoost 모델 개발	O / X
LSTM 모델 개발	O / X
예측 결과 저장 및 시각화	O / X
모델 성능 비교 및 리포트 작성	O / X