API + LSTM + BERT + NLP 활용한 통합 주식 예측 서비스

아트로포스 – AI 를 활용한 차세대 주식 예측 서비스

SUMMARY OF CONTENTS

01. 프로젝트 개요

■ 아트로포스란?

더 복잡해진 시장과 다양한 정보를 분석을 하기 위해
기존의 로보어드바이저, 퀀트 투자 보다 향상된

통합 예측 모델의 필요성 인식

01. 프로젝트 개요

주요 주식 투자 모델 비교

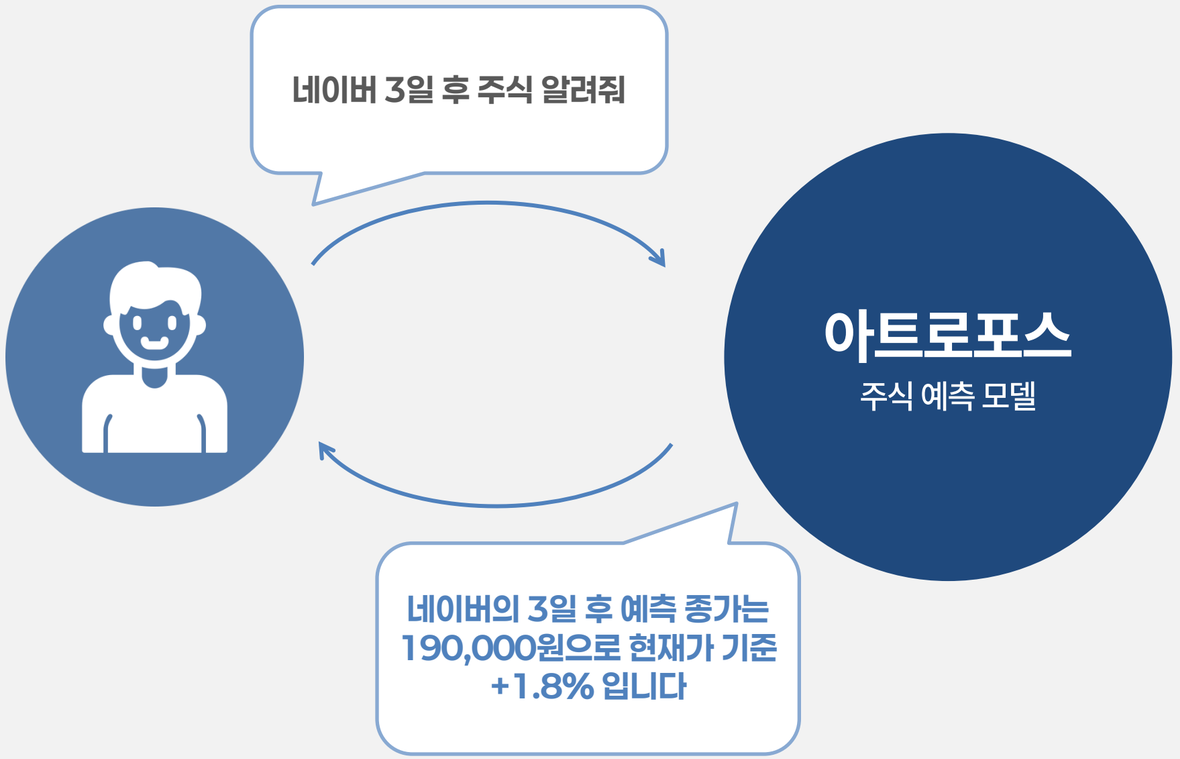
구분	로보 어드바이저(Robo-Advisor)	퀀트 투자(Quantitative Investing)	아트로포스(Athropos)
정의	자동화된 포트폴리오 추천/관리 서비스	수학적/통계적 모델로 투자 판단	뉴스,공시,주가를 융합한 AI 투자 플랫폼
기반 기술	규칙 기반 + 간단한 알고리즘	수리 통계 + 백테스트 전략	딥러닝(LSTM, BERT 등) + 멀티모달 융합 전략
주요 특징	위험 성향 설문 기반 ETF 중심 자산 배분 저비용 자동 리밸런싱	수학, 코딩 기반 주로 펀더멘털/테크니컬 지표 사용 백테스트로 전략 검증	뉴스 + 공시 + 주가 자동 결합 감성 분석 + 시계열 예측 종목별 딥러닝 기반 5일 예측
목표 사용자	일반 투자자 초보자 중심	수학적 사고 기반의 트레이더 펀드매니저	AI 기반 분석을 원하는 리서치 B2B 분석 기업 일반 투자자, 재태크 입문자
접근성	높음(앱, 웹 플랫폼)	중간 (코딩, 지식 필요)	낮음(자체 구축, 차후 웹, 앱 플랫폼 확장)
예측 방식	포트폴리오 최적화 기반 리밸런싱	인과/상관 분석 기반 전략 선택	AI 학습 기반 뉴스,공시,주가 통합 예측
데이터 소스	주가, ETF성과, 경제 지표	주가, 재무제표, 기술적 지표	뉴스 기사, DART 공시, 주가 캔들차트, 감성 분석 점수
기술 난이도	★☆☆☆☆ GUI 위주	★★★★☆ 파이썬, 수학 필요	★★★★★ ML, DL, NLP, 시계열, DB결합
투자 방식	장기 분산 투자 지향	단기~중기 전략 혼합	초단기(5일 단위) 시계열 예측 기반 전략
예시 서비스	Toss, 뱅가드, 한국투자 RAA 등	인베스팅닷컴 전략, 알파퀀트, 팩터기반 모델	BigEpoch – Athropos(자체 모델링 플랫폼)

01. 프로젝트 개요

구분	내용
개발언어	Python 3.12, Javascript, HTML5, CSS,
개발환경	VS Code, Cursor, WSL(Ubuntu), Colab Pro, Jupyter Lab, AWS EC2, Github
웹프레임워크	Django 5.2.1
라이브러리	FinanceDataReader, Transformer, Kobert, Cuda, LSTM, Keras, TensorFlow, sci-kit learn, torch, Pandas, Requests, Numpy, LAG, BeatifulSoup4, Selenium, Tqdm,
데이터베이스	MySql
외부 API	OpenAI API, y-finance, 네이버 API, Dart API,
기타	ChatGPT, Gemini, Grok, Copilot, 뤼튼, Notion, Google Drive

01. 프로젝트 개요

프로젝트 구조도



Ubuntu + Django + MySQL

데이터 수집 API/크롤링	API 한국 투자 증권	API YahooFinance	API DART 공시 데이터	웹크롤링 종목별 뉴스 기사
API Key연동, 웹 크롤링 모델 .py 실행(콘솔)				
데이터 전처리 컬럼/DB저장	Kospi 5년치 데이터	Kosdaq 5년치 데이터	DART 공시 내용 5년치 데이터	종목별 뉴스 본문 5년치 데이터
파일 저장/MySQL DB 저장				
모델 학습 머신러닝/딥러닝	딥러닝	시계열 분석 LSTM	규칙 기반 학습 Rule	감성 분석 KoBert
모델 학습 – 베스트 모델 저장, 분석 가중치 저장				
모델 예측 API/DB	API/DB 한국 투자 증권	API/DB YahooFinance	API/DB DART 공시	웹크롤링/DB 종목별 뉴스 기사

01. 프로젝트 개요

- 아트로포스 사이트 개요

01. 프로젝트 개요

- 모바일 #01 메인페이지

main.html

화면 설명

- 영역별 상세설명

01. 헤더 영역
  a - 로고
  b – 알림 아이콘
  c – 로그인 아이콘

02. 오늘자 지수 / 예측 지수

03. 메인 슬라이드 이미지
  d – 메인슬라이드 이미지 여러 개
  e – 이미지 번호(현재번호 / 총 번호)

04. 예측 콘텐츠(메인 슬라이드 연동)
  f – 오리지널 콘텐츠 A
  g – 오리지널 콘텐츠 B
  h – 오리지널 콘텐츠 C

05. 커뮤니티 랭킹
  - 실시간 뉴스 게시물 Top 5

06. 환율 정보
  - 주요 통화 환율 정보

07. FAQ, 제휴문의, 광고문의
  - 기타 링크

08. AI챗봇 (플로팅 메뉴)
  - AI 챗봇 페이지 링크

09. 상단가기 (플로팅 버튼)
  - 상단 가기 버튼

10. GNB
  i – 홈(메인 페이지)
  j – 차트
  k – 예측정보
  l – 커뮤니티
  m – 마이페이지

기능 설명

a. 로고 이미지 : 클릭시 -> 메인 페이지
b. 알림 아이콘 : 로그인시 보이기 - 알림
c. 로그인 아이콘 : 클릭시 -> 로그인 페이지
d. 메인 슬라이드 : 오리지널 콘텐츠 f,g,h와 연동
e. 이미지 번호 : 이미지 번호 (현재 / 전체)
5. 실시간 뉴스 – 최대 5개 실시간 뉴스 api 노출
6. 환율 정보 : 주요 통화 환율 시세 정보 노출
7. 기타 버튼 : FAQ / 제휴문의 / 광고 문의 페이지 이동
8. 챗봇 : 챗봇 페이지 링크
9. 상단가기 : 클릭시 페이지 상단 이동
10. GNB: 하단 고정
  i. 홈 : 클릭시 링크
  j. 차트 : 클릭스 링크 -> 차트 페이지
  k. 예측정보 : 클릭스 링크 -> 예측정보 페이지
  l. 커뮤니티 : 클릭시 링크 -> 커뮤니티 페이지
  m. 마이페이지 : 클릭시 링크 -> 마이페이지

- 모바일 #02 차트페이지

chart.html

기능 설명

1. 주식 종목 검색 입력 창
   a. 종목 입력시 드랍 다운으로 종목명자동완성 알림
2. 입력 종목 정보 영역 출력
3. 입력 종목 시세 그래프(종가 기준)
4. 52주 최고가, 최저가 금액 출력
5. 코스피/코스닥 급등락 TOP5 정보
   - 리스트의 종목 명을 누르면 자동으로 1번에 입력
6. 시가총액 TOP 30
7. 주요 및 인근 주식 시장 지수 출력

화면 설명

- 영역별 상세설명

01. 주식 종목 검색 창
   a – 주식 종목 검색 창
   b – 기간 조절 버튼

02. 종목 정보 영역

03. 입력 종목 시세 그래프 렌더링
   h – 그래프 렌더링 출력

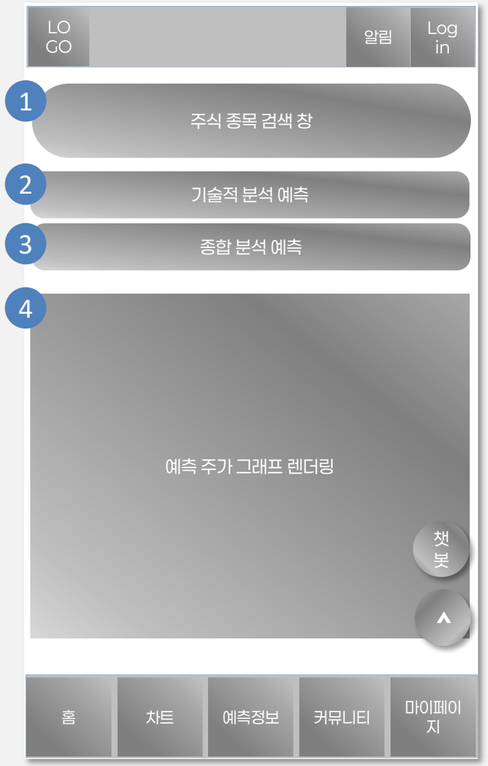
04. 52주 최고가/최저가 출력

05. 급등락 주식 정보 제공
   - 코스피 급등 TOP5
   - 코스닥 급등 TOP5
   - 코스닥 급락 TOP5
   - 코스닥 극락 TOP5

06. 시가총액 TOP 30 리스트

07. 주요 시장/인근 시장 주요 지수

- 모바일 #03 예측 정보 페이지



predict_info.html

화면 설명

- 영역별 상세설명

01. 주식 종목 검색 창
 - 입력시 종목명/종목코드 자동완성 드랍리스트

02. 기술적 분석 예측 버튼
 - 퀀트 투자 방식 예측 실행 요청 시

03. 종합 분석 예측 버튼
 - 종합 분석 예측 실행 요청 시

04. 예측 주가 그래프 렌더링
 - 5일 예측 주가 종가 그래프 렌더링

기능 설명

1. 주식 종목 검색 창 : 입력시 드랍 리스트로 종목명/종목코드 자동 완성 리스트 노출
2. 기술적 분석 예측 : 클릭시 주식 데이터만 활용한 분석 모델 실행
3. 종합 분석 예측 : 클릭시 통합 예측 모델 실행
4. 2~3번 모델 실행 후 DB에서 날짜기준 5일 전 데이터와 차후 5일 후 예측 데이터 로드 후 합쳐 동시에 하나의 그래프선으로 렌더링(실선 + 점선)

01. 프로젝트 개요

- 모바일 #04 커뮤니티 페이지

community.html

게시판 형태

화면 설명

- 영역별 상세설명

01. 커뮤니티 게시판 선택 메뉴
  - 뉴스/공시 게시판
  - 커뮤니티 게시판

02. 실시간 공시
  - 실시간 공시 뉴스

03. 게시판 글쓰기 버튼

a. 게시판 정렬 필터 버튼
  - 최신순 정렬(디폴트)
  - 조회수 순 정렬
  - 좋아요 순 정렬
  - 걱정돼요 순 정렬

04. 게시판 리스트

05. 페이지네이션 넘버

06. 게시글 검색 창

b. 작성자 프로필 이미지

c. 회원 관련 정보
  - 닉네임
  - 회원 등급
  - 등록 게시판
  - 등록 날짜

d. 게시글 제목

e. 게시글 본문 미리보기

f. 조회수, 추천 수, 댓글 수

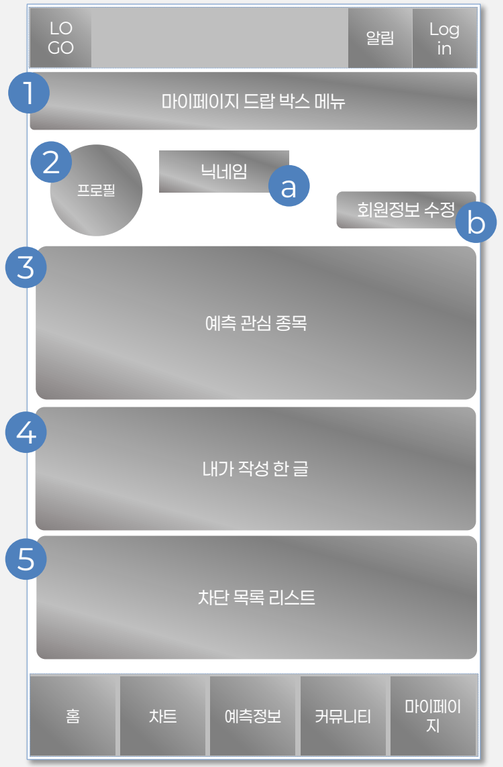
기능 설명

1. 메뉴 바 : 메뉴 선택시 하단 게시판 목록 변경
  - 뉴스/공시 게시판 리스트 불러오기
  - 커뮤니티 게시판 리스트 불러오기
2. 실시간 공시 : api를 활용한 실시간 공시 제목 불러오기
3. 글쓰기 버튼 : 로그인 성공한 사용자만 사용가능

a. 필터 버튼 : 드랍다운 버튼 클릭시 최신/조회수/좋아요/걱정대요 순으로 게시물 정렬 선택
4. 게시판 리스트
5. 페이지 네이션 넘버링
6. 게시글 검색창

b. 프로필 : 사용자가 설정한 프로필 이미지
c. 작성 정보 : 닉네임, 회원 등급, 등록게시판, 등록 날짜 노출
d. 게시판 제목(타이틀)
e. 본문 미리보기
f. 조회수, 추천수, 댓글 수 해당 게시글에 대한 정보

- 모바일 #05 마이페이지



mypage.html

화면 설명

- 영역별 상세설명

01. 마이페이지 드랍 박스 메뉴
  - 마이 페이지
  - 관심 종목
  - 내가 쓴 글
  - 차단 계정 목록

02. 프로필
  a. 닉네임
  b. 회원정보 수정 버튼

03. 예측 관심 종목

04. 내가 작성한 글

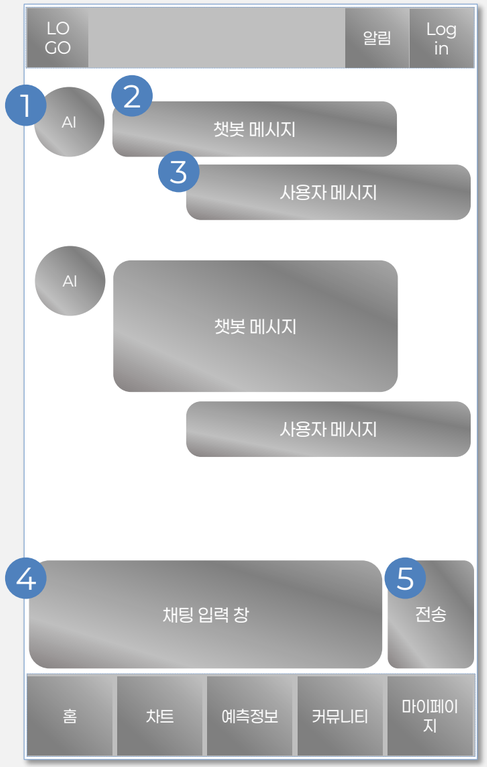
05. 차단 목록 리스트

기능 설명

1. 메뉴 바 : 메뉴 선택 시 하단 영역으로 스크롤 바로 가기
  - 최상단 / 예측 관심 종목 / 내가 쓴 글 / 차단 목록
2. 사용자 정보 영역 : 프로필 클릭시 이미지 등록 가능
  a : 사용중인 닉네임
  b : 회원 정보 수정 페이지
3. 예측 정보 페이지에서 관심 등록한 종목 리스트
4. 커뮤니티 페이지에서 내가 작성한 글 목록 노출
  - 모니터링
  - 수정/삭제
5. 커뮤니티 페이지에서 내가 차단한 사용자 목록
  - 모니터링
  - 수정/삭제

01. 프로젝트 개요

- 모바일 #06 AI 챗봇 페이지



chatbot.html

화면 설명

- 영역별 상세설명

01. 아트로포스 채팅 영역
  - 로고 이미지

02. 챗봇 메시지

03. 사용자 메시지

04. 채팅 입력 창(프롬프트)

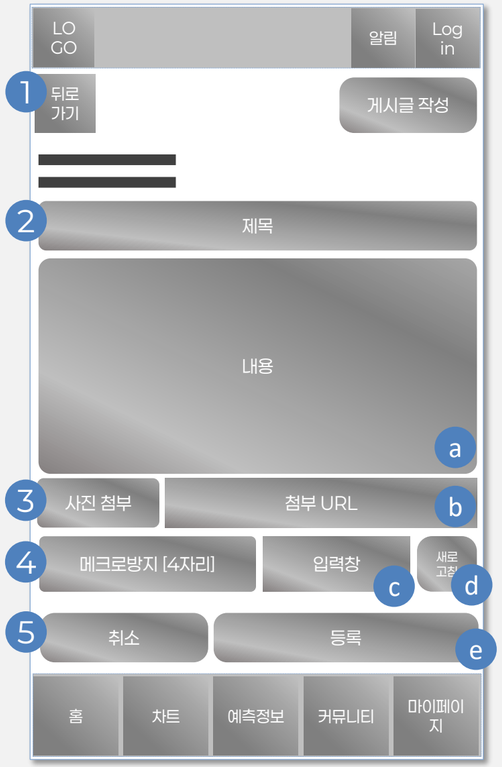
05. 전송 버튼

기능 설명

2. 처음 랜딩시 챗봇 메시지의 기본 인사말 메시지 전달
3. 사용자 메시지는 챗봇 메시지 한 칸 아래로 메시지가 출력된다.
4. 입력창에서 사용자는 보낼 메시지를 미리 확인 가능
5. 전송 버튼 클릭 시 ai 챗봇에게 메시지 전달

공통.
- 챗봇 메시지는 왼쪽, 사용자 메시지는 오른쪽 정렬
- 사용자가 비속어/욕설 작성 시 메시지에서 ***으로 치환되어 출력

- 모바일 #07 게시판 글 작성 페이지



community_write.html

화면 설명

- 영역별 상세설명

01. 뒤로가기 버튼

02. 제목 입력창
a. 내용 입력창

03. 사진첨부 버튼
b. 첨부된 파일 URL 주소

04. 매크로방지 영역
c. 입력창
d. 새로고침 버튼

05. 취소버튼
e. 등록 버튼

기능 설명

1. 뒤로가기 버튼 클릭시 이전 페이지로 이동
2. 제목 타이틀 입력 창
a. 본문 내용 입력 창
3. 이미지 사진 첨부 버튼 클릭시 파일 선택
b. 파일 선택이 완료되면 해당 url이 출력 됨

4. 매크로방지 영역
   - 랜덤 숫자 4자리 노출
   - 4에서 생성된 4자리 숫자를 c에 입력한다
   - 새로고침 버튼 클릭시 초기화 – 새로 생성

- 모바일 #08 회원 가입 페이지

signup.html

화면 설명

- 영역별 상세설명

01. 프로필 이미지

02. 프로필 사진 선택 버튼

03. 회원가입 정보 입력창
[3-1]아이디는 최소 4자 이상
[3-2]이름은 최소 2자 이상
[3-3]닉네임은 최소 2자 이상
[3-4]비밀번호는 최소 8자 이상
[3-5]이메일은 user1@naver.com 형식

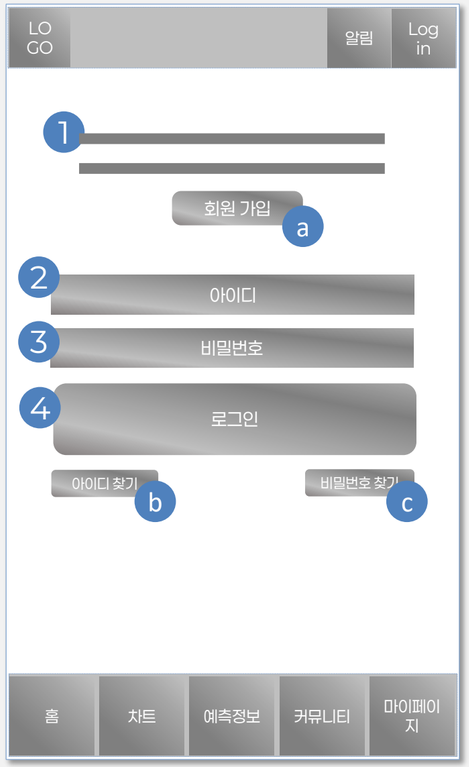
04. 가입하기 버튼

기능 설명

1. 2번에서 선택한 이미지를 출력해주는 영역
3. 회원 정보 입력 창
  - 아이디 : 프라이머리값, 유니크?
  - 이름 : 한글? 영어?
  - 닉네임 : 한글 영어 유니크?
  - 비밀번호 / 비밀번호 확인 : 최소 8자 이상 영대소/숫자 혼합
  - 이메일 주소 : 이메일 주소 입력

4. 가입하기 버튼
a. 안내 텍스트
  - “이미 계정이 있으신가요? 로그인하기” 출력문에서 로그인에 링크 표시

- 모바일 #09 로그인 페이지



login.html

화면 설명

- 영역별 상세설명

01. 안내 텍스트 출력
  - '차세대 인공지능 주식 예측 서비스'
  - '아트로포스에 오신걸 환영합니다'

a. 회원가입 버튼

02. 아이디 입력 창

03. 비밀번호 입력 창

04. 로그인 버튼

b. 아이디 찾기 버튼
C. 비밀번호 찾기 버튼

기능 설명

a. 버튼 클릭시 회원 가입 페이지로 이동
2. 아이디 입력 : 입력 값이 틀릴 경우 경고 메시지 출력
  - '입력한 아이디/비밀번호가 틀립니다'
3. 비밀번호 입력 : 5회 이상 틀릴 시 계정 차단
  - 비밀번호 찾기 활용하여 계정 활성화

4. 로그인 버튼 : 정상 입력시 사용자 로그인 상태 부여
b. 아이디 찾기 페이지 이동
c. 비밀번호 찾기 페이지 이동

감정적 의사결정 최소화

통계적 알고리즘에 따라 매매 시점
자동 판단으로 과잉 매매, 공포 매도 방지

시장 변화에 대한 신속 대응

시계열 데이터를 활용해 5일치 주가를
예측함으로 단기 변동에도 자동으로
포지션 조정 가능

운용 효율성 증대

초기 모델, 포트폴리오 설정 후 일상적
관리 불필요, 개인 투자자의 시간,
노력 절감

02. 프로젝트 팀 - BigEpoch1984

팀 구성원 및 역할 분담

김가람 팀장

- GitHub 버전 관리
- Django 및 MySQL DB 설계
- 회원가입/로그인 프로세스 구현
- 비속어 필터링 머신러닝 모델 구현

정태영 팀원

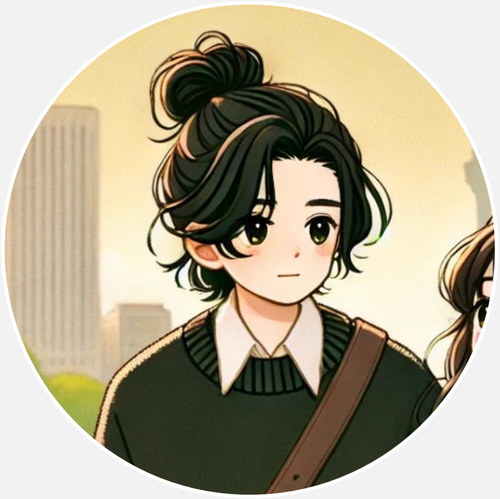
- 프로젝트 기획서, UI 스토리보드 작성
- 기능별 모델 기획
- 뉴스 기사 크롤링 모델 구현
- 공시 데이터 크롤링 모델 구현
- Kobert 감성분석 모델 구현

문성준 팀원

- 알고리즘 모델 구현 테스팅
- OpenAI API 챗봇 구현
- 차트 페이지 UI 구현
- 주식 그래프 렌더링 구현

02. 프로젝트 팀 - BigEpoch1984

팀 구성원 및 역할 분담



이승배 팀원

- 예측 페이지 프론트엔드 구현
- 파이낸스 데이터로더 데이터수집
- 주식 데이터 MySQL DB테이블 설정
- 예측 모델 구축/학습/예측 그래프 렌더링

조지형 팀원

- 커뮤니티, 마이페이지 Django 앱 구현
- 뉴스탭내 네이버/다트 뉴스 api 연동
- 페이지네이션, 마이페이지 프론트엔드
- 메인사이트 UI 프론트엔드 디자인

조준행 팀원

- 팀과제 진행 계획 수립
- 모델 설계 자문 활동
- PDF prompt 엔지니어링 패턴 개발

03. 수행 절차 및 방법

기간별 활동 내용

구분	기간	활동	비고
계획	2025.04.21 ~ 2025.04.25	- 팀 과제 진행 계획 수립 - 개인 별 역할 분배	
사전 준비	2025.04.28 ~ 2025.05.02	- 모델 확인 및 모델별 기획 - 데이터 수집처 서칭 및 API 가입 - 사이트 UI 스토리보드 및 디자인 컨셉 기획	
설계	2025.05.06 ~ 2025.05.12	- 데이터 수집(데이터파일, 웹크롤링, API) - OpenAI API 연동 - 페이지별 UI 영역 구분	
모델 구현	2025.05.13 ~ 2025.05.23	- 머신러닝, 딥러닝 모델 학습, 예측 모델 구현 - OpenAI API 연동 챗봇 구현 - 페이지별 UI영역 구분 및 프론트엔드 기본 기능 구현	
시험 및 피드백	2025.05.26 ~ 2025.05.29	- 모델 테스트/피드백/모델 수정	

04. 프로젝트 수행 결과

일요일	월요일	화요일	수요일	목요일	금요일	토요일
20	21 조 편성 주제 정하기 WBS/ERD 작성	22 데이터 수집 기획서/화면설계서 작성 정책/정의서 작성	23 기획서 완료	24 알고리즘 구현 초안 크롤링/DB 테스트 페이지 네이션	25 회원가입/로그인	26
27	28 알고리즘 1차 모델 선별 페이지 디자인 시안 작업	29	30 모델 1차 학습 Django 페이지 생성	5/1	2 모델 1차 예측/평가 모델 2차 학습 게시판	3
4	5 어린이날 부처님 오신날	6 어린이날(대체)	7 모델 2차 예측/평가 모델 3차 학습 페이지 디자인 적용	8	9 모델 3차 예측/평가	10
11	12	13	14	15	16	17
18	19 기능 테스트	20	21	22	23 최종 테스트	24
25	26	27	28 마감	29	30	31

04. 프로젝트 수행 결과

주요 모델 구조도

모델명	모델 형태	주요 기능	수집처	학습 형태	상세 내용	비고
뉴스 크롤링	웹 크롤링	뉴스 기사 웹 크롤링	조선일보 (조선일보/비즈조선/it조선/tv조선)	Kobert + 딥러닝 감성 분석	CSV / DB	A
공시 크롤링	웹 크롤링	다트 공시 데이터 크롤링	Dart api	Kobert + Rule 기반 머신러닝	CSV / DB	B
주식 데이터 학습	LSTM	주식 데이터 학습	- 파이낸스 데이터로더 - 한국 투자 증권 api	딥러닝 + LSTM	DB	C
통합 데이터 학습	ML	통합 데이터 학습	A,B,C 데이터로 학습	딥러닝 + LSTM	.keras 모델	D
통합 데이터 예측	DL	통합 데이터 예측	사전 학습된 D모델에 사용자 입력값 예측	딥러닝 + LSTM	통합 예측 활용	
비속어 필터링	ML + Rule기반	비속어/욕설 필터링	CSV + 머신러닝	Kobert + 머신러닝	커뮤니티/챗봇 활용	
AI 챗봇	API + Rule 기반	AI 챗봇 어시스턴스	OpenAI API + Rule 기반	Kobert + 머신러닝	챗봇 페이지 활용	

04. 프로젝트 수행 결과

- 모델 #01. 뉴스 기사 크롤링 모델

알고리즘 기능 개요

정적 페이지 크롤링을 위한 라이브러리 뷰티플수프(BeatifulSoup4)와 비동기 페이지 크롤링을 위한 셀레니움(Selenium) 라이브러리 동시 사용

조선일보의 경우 실제 기사 본문의 경우 계열사별(조선일보, 비즈조선, it조선, tv조선 등)로 클래스 명이 상이하여 서브도메인을 확인하는 절차를 추가 – 파싱 – 저장함

상세 내용

1. 야후파이낸스에서 kospi, kosdaq 종목별 시장분류
2. 크롤링 도메인이 단일이 아닌 복합 도메인으로 If문으로 반복 확인
3. 파싱 - 종목명, 제목, 요약, 본문, 날짜, URL, 언론사
4. 2020.01.01 ~ 2025.05.xx (코드 실행한 1일전까지)
5. 파싱 후 csv파일로 주식 시장별 2개 파일로 저장

알고리즘 모델 코드

```
# 서브 도메인별 파서 (조선일보, 비즈조선, it조선, tv조선)
def parse_chosun(driver):
    soup = BeautifulSoup(driver.page_source, "html.parser")
    paragraphs = soup.select("p.article-body__content-text")
    return " ".join(p.get_text(strip=True) for p in
paragraphs)

def parse_biz(driver):
    soup = BeautifulSoup(driver.page_source, "html.parser")
    paragraphs = soup.select("p.article-
body__content.article-body__content-text")
    return " ".join(p.get_text(strip=True) for p in
paragraphs)

def parse_it(driver):
    soup = BeautifulSoup(driver.page_source, "html.parser")
    body = soup.select_one("div.article-body")
    return " ".join(p.get_text(strip=True) for p in
body.find_all("p")) if body else ""

def parse_tv(driver):
    soup = BeautifulSoup(driver.page_source, "html.parser")
    box = soup.select_one("div.text-box")
    return " ".join(p.get_text(strip=True) for p in
box.find_all("p")) if box else ""

def get_parser_by_url(url):
    if "biz.chosun.com" in url:
        return parse_biz, "비즈조선"
    elif "it.chosun.com" in url:
        return parse_it, "IT조선"
    elif "tvchosun.com" in url:
        return parse_tv, "TV조선"
    else:
        return parse_chosun, "조선일보"
```

```
# 크롤링 및 파일 저장
if __name__ == "__main__":
    TODAY = datetime.today().strftime("%Y%m%d")
    start_time = perf_counter()
    print("FDR에서 KOSPI 종목 불러오는 중...")
    kospi_list = fdr.StockListing('KOSPI')['Name'].dropna().unique().tolist()
    combined_list = [("KOSPI", stock) for stock in kospi_list]
    all_kospi = []
    total_kospi = 0

    def threaded_crawl(args):
        market, stock = args
        try:
            articles, count = crawl_chosun_news(stock)
            return (market, stock, articles, count)
        except Exception as e:
            tqdm.write(f"[⚠] 실패: {stock} ({e})")
            return (market, stock, [], 0)
    try:
        with ThreadPoolExecutor(max_workers=4) as executor:
            futures = [executor.submit(threaded_crawl, arg) for arg in
combined_list]
    finally:
        if all_kospi:
            cols = ["종목명", "날짜", "제목", "요약", "본문", "URL", "언론사"]
            pd.DataFrame(all_kospi)[cols].to_csv(
                os.path.join(SAVE_DIR, f"chosun_kospi_{TODAY}.csv"),
                index=False, encoding="utf-8-sig")
            print(f"KOSPI 저장 완료: {len(all_kospi)}건")

        elapsed = perf_counter() - start_time
        print(f"\n✅ 전체 기사 크롤링 완료 ¦ 소요 시간: {elapsed:.2f}초 ≈
{elapsed/60:.1f}분")
```

04. 프로젝트 수행 결과

- 모델 #02. 다트 공시 데이터 수집 모델

알고리즘 기능 개요

다트 오픈 api를 이용해서 작업. 다트에서 전 종목 코드 불러온 후 종목 별로 5년치 다트공시데이터를 받아서 저장함. 전체 공시 중에서 중요도 순으로 선별해서 선별된 공시 수집(주요사항보고서, 영업실정공시, 유,무상증자결정 등).
공시 수집후 kobert 모델을 이용해서 자연어 처리를 하고 긍 부정 점수를 종목과, 시장, 날짜 별로 저장함

상세 내용

1. API 키 관리 & 재시도 로직
2. 기업 고유번호(CORPCODE.xml)파싱
3. 공시 목록(list.json) 조회
4. 공시 문서 (document.xml)다운로드 -> ZIP/PDF -> 텍스트 추출
5. 분류(classify_report) -> 임베딩 요약 -> 일별 집계
6. 병렬화(ThreadPoolExcutor) +체크포인트

알고리즘 모델 코드

```
# API 키 관리 & 재시도
def get_current_api_key():
    with api_key_lock:
        return DART_API_KEYS[g_current_api_key_index]
def rotate_next_api_key():
    with api_key_lock:
        g_current_api_key_index += 1
        return g_current_api_key_index < len(DART_API_KEYS)
def requests_get_with_retry(url, params, is_api_call=False, retries=3, delay=15):
    for attempt in range(retries):
        if is_api_call:
            params['crtfc_key'] = get_current_api_key()
        resp = requests.get(url, params=params, timeout=...)
        if resp.status_code == 200:
            return resp
        # 키 제한 오류 감지 시 키 교체
        if is_api_call and resp.json().get('status') in ['010','011','020']:
            rotate_next_api_key()
        time.sleep(delay)
    raise Exception("요청 최종 실패")
# 기업 고유번호 파싱
def generate_corp_df():
    res = requests_get_with_retry("https://opendart.fss.or.kr/api/corpCode.xml",
                                  params={}, is_api_call=True)
    # ZIP/비압축 XML 판별 → Bytes → ET.parse → <list> 반복 추출
    items = [{"corp_name":n, "corp_code":c, "stock_code":s}
             for each <list> node]
    df = pd.DataFrame(items)
    # FDR StockListing 병합 → KOSPI/KOSDAQ 필터
    return pd.merge(df, fdr.StockListing("KRX")[['Code','Market','Name']], …)
```

```
# 공시 목록 조회
def fetch_dart_disclosure_list(corp_code, start, end):
    params = {"corp_code":corp_code, "bgn_de":start, "end_de":end, …}
    all_reports = []
    while True:
        res = requests_get_with_retry("…/list.json", params, is_api_call=True)
        data = res.json()
        if data['status']!='000': break
        all_reports += data['list']
        if params['page_no'] >= data['total_page']: break
        params['page_no'] += 1
    return all_reports
#감성, 임팩트 분류
def classify_report(title, text):
    # ① 제목 기반 스코어링
    for kw,(s,w) in negative_keywords.items():
        if kw in title: …  # 음수 가중치 누적
    for kw,(s,w) in positive_keywords.items():
        …


    # ② 본문 기반 스코어링 (키워드 카운트)
    text_score = sum(…)/sum(weights)

    # ③ 정기공시 vs 이벤트공시 분기
    if any(rt in title for rt in ["분기보고서","사업보고서"]):
        …
    elif abs(title_score)>0.4:
        …
    return label, sentiment, impact_score, weight
```

04. 프로젝트 수행 결과

- 모델 #03. 주식 데이터 수집 모델

알고리즘 기능 개요

FinanaceDataReader 모듈로 주식데이터 수집을 위해 pandas, numpy ,pandas_ta 등의 라이브러리를 설치 적용 코스피/코스닥 전 종목 1일 데이터를 5년동안 불러와서 저장. 기타 모듈에 포함되있지 않은 지표데이터들은 가져온 데이터와 pandas_ta 를 이용해서 직접 계산처리해서 데이터에 포함시킴 불러왔지지 않아서 비어있는 데이터들은 NaN 값이 있으면 학습을 위해서 0이나 앞데로 채우는 방식이 아닌 모델링학습시 NaN으로 인식하도록 구현함. 모든 처리를 마치고 csv 파일로 종목, 시장, 날짜별로 저장함

상세 내용

1. 외부 데이터 (코스피. 환율, 투자자 동향) 사전로드
2. 개별 종목 OHLCV+Fundamental 수집, 표준화
3. 기술지표 (TA) 계산
4. 5년 데이터 필터링 후 저장
5. ProcessPoolExecutor로 병렬 처리

알고리즘 모델 코드

```python
# 외부 데이터 사전 로드 (fetch_with_retry)
def fetch_with_retry(fn, *args, stock_code_for_log="", **kw):
    for i in range(RETRY):
        try:
            df = fn(*args, **kw)
            if not df.empty: return df
        except Exception as e:
            log_message(f"{stock_code_for_log} 시도{i+1} 오류: {e}")
        time.sleep(DELAY)
    log_message(f"{stock_code_for_log} 최종 실패")
    return pd.DataFrame()

#데이터 표준화 함수
def standardize_pykrx_data(df_raw, is_index=False):
    if df_raw.empty: return pd.DataFrame()
    df = df_raw.reset_index().rename(columns={df_raw.index.name:'Date'})
    df['Date'] = pd.to_datetime(df['Date'])
    mapping = {'시가':'Open','고가':'High','저가':'Low','종가':'Close'}
    if not is_index:
        mapping.update({'거래량':'Volume','등락률':'Change'})
    return df.rename(columns=mapping)
})
```

```python
# 기술지표(TA) 계산 핵심
def calculate_technical_indicators(df):
    # ATR
    df['ATR_14'] = ta.atr(df.High, df.Low, df.Close, length=14)
    # 볼린저, RSI, MACD, Stochastic, OBV, ADX
    df.ta.bbands(20,2,append=True)
    df.ta.rsi(14,append=True)
    df.ta.macd(12,26,9,append=True)
    df.ta.stoch(14,3,3,append=True)
    df.ta.obv(append=True)
    df.ta.adx(14,append=True)
    return df
# 종목별 업데이트 & 저장
def update_stock_data_file(...):
    # 1) PyKRX로 OHLCV + Fundamental 수집
    df = fetch_with_retry(stock.get_market_ohlcv, ...)
    df = standardize_pykrx_data(df)
    # 2) 시가총액·PER·PBR 병합
    df = merge_fundamentals(df)
    # 3) 외부(KOSPI, 환율, 투자자 동향) 병합 + ffill/bfill
    df = merge_external(df, kospi, fx, investor_trend)
    # 4) 기술지표 계산
    df = calculate_technical_indicators(df)
    # 5) 5년 필터링 & CSV 저장
    df_final = df[(df.Date >= cutoff) & (df.Date <= today)]
    df_final.to_csv(new_filename, index=False)
    return new_filename
```

04. 프로젝트 수행 결과

- 모델 #04. 통합 데이터 학습 모델

알고리즘 기능 개요

주식 시장의 과거 시계열 데이터(날짜, 종가, 거래대금 등 총 34개 데이터)와 DART 공시라는 비정형 EVENT 데이터를 통합적으로 분석하여 미래 주가를 예측하는 LSTM 기반 딥러닝 모델입니다. 2개의 다른 성격의 데이터를 융합하여 보다 정교한 예측을 목표로 합니다.

상세 내용

1. 날짜, 시장, 종목 기준으로 정제된 주식 데이터와 수치화된 공시 데이터를 하나의 시퀀스 데이터로 구성.
2. LSTM 레이어를 포함한 딥러닝 모델을 구성하고, Adam 옵티마이저와 MSE 손실 함수를 사용하여 학습을 진행
3. 모델은 이 시퀀스를 바탕으로 향후 특정 기간(예: 5 거래일)의 종가를 예측하도록 학습

알고리즘 모델 코드

```
--- 데이터 준비 ---
# DART 공시 감성 데이터 경로 (사용자님의 DART 스크립트 실행 결과물 기준)
TODAY_STR_FOR_DART_FILE = datetime.today() # 오늘 날짜로 기본 설정.
SAVE_DIR_DART = "./dart_nlp_aggregated_data"
DART_SENTIMENT_KOSPI_PATH = os.path.join(SAVE_DIR_DART,
f"dart_kospi_aggregated_{TODAY_STR_FOR_DART_FILE}.csv")

# 주가 데이터 기간 설정 (Dart데이터와 동일하게)
END_DATE_DT = datetime.today()
START_DATE_DT = END_DATE_DT - timedelta(days=5*365) # 최근 5년치 데이터
START_DATE_STR_STOCK = START_DATE_DT.strftime("%Y-%m-%d")
END_DATE_STR_STOCK = END_DATE_DT.strftime("%Y-%m-%d")


--- 시퀀스 데이터 생성 ---
def create_sequences(data_df, feature_cols, target_col, sequence_length):
    X_list, y_list = [], []
    unique_stocks = data_df['stock_code'].unique()

    print(f"\n시퀀스 데이터 생성 중 (Sequence Length: {sequence_length})...")
    for stock_idx, stock_code in enumerate(unique_stocks):
        stock_df = data_df[data_df['stock_code'] == stock_code].copy()
        stock_df.sort_values(by='date', inplace=True)

        features = stock_df[feature_cols].values
        target = stock_df[target_col].values
    ]
```

```
  --- 딥러닝 모델 정의 및 학습 ---
if 'X_train' in locals() and X_train.size > 0:
    INPUT_SHAPE = (X_train.shape[1], X_train.shape[2])

    model = Sequential([
        Bidirectional(LSTM(LSTM_UNITS, return_sequences=True,
input_shape=INPUT_SHAPE)),
        Dropout(DROPOUT_RATE),
        LSTM(LSTM_UNITS // 2, return_sequences=False),
        Dropout(DROPOUT_RATE),
        Dense(32, activation='relu'),
        Dropout(DROPOUT_RATE),
        Dense(1, activation='sigmoid')
    ])
    ---  모델 평가 ---
# 최적 가중치 모델 로드 (ModelCheckpoint에 의해 저장된 모델)
    if os.path.exists(MODEL_CHECKPOINT_PATH):
        print(f"  저장된 최적 모델 로드: {MODEL_CHECKPOINT_PATH}")
        model.load_weights(MODEL_CHECKPOINT_PATH) # 또는 model =
tf.keras.models.load_model(MODEL_CHECKPOINT_PATH)

    eval_results = model.evaluate(X_test, y_test, verbose=0)
    test_loss = eval_results[0]
    test_accuracy = eval_results[1]
    test_precision = eval_results[2]
    test_recall = eval_results[3]
    test_f1_score = 2 * (test_precision * test_recall) / (test_precision +
test_recall) if (test_precision + test_recall) > 0 else 0
```

04. 프로젝트 수행 결과

- 모델 #05. 통합 데이터 예측 모델

알고리즘 기능 개요

Colab에서 주식 데이터와 DART 공시 감성 데이터를 이용해 학습 완료된 종합시계열 예측 모델 (.h5 파일)을 장고에 통합합니다. 공시와 주식데이터를 API와 라이브러리로 불러와 MYSQL DB에 저장합니다. 저장되 있는 데이터를 모델에 입력해 예측한 값을 DB에 저장합니다. 페이지에서 예측을 클릭시 DB에 있는 5일치 예측 값을 표와 그래프로 페이지에 표시합니다

상세 내용

1. 학습된 Keras 모델 파일 (.h5 또는 .keras)과 데이터 스케일링에 사용된 MinMaxScaler 객체를 Django 프로젝트에서 접근 가능한 위치에 저장, 로드.
2. 모듈, Api 를 통해 가장 최근까지의 데이터를 불러와서 기술적 분석 지표, dart 감성점수 분석 후 DB에 저장.
3. 준비된 시퀀스를 딥러닝 모델에 투입한 예측 값(미래 5일치 종가)을 Django 를 통해 사용자에게 시각적으로 표시.

알고리즘 모델 코드

```
--- 모델 및 스케일러 로드 (앱 로딩 시 1회 실행) ---

LSTM_MODEL_PATH = 'ml_models/kospi_lstm_v1.h5'
SCALER_OBJECT = load_my_scaler('ml_models/data_scaler.pkl')
try:
  stock_predictor_model = load_model(LSTM_MODEL_PATH)
except OSError:
  stock_predictor_model = None # 로드 실패 시 None 처리

SEQUENCE_LEN = 30 # 학습 시퀀스 길이
NUM_EXPECTED_FEATURES = 15 # 학습에 사용된 특성 개수

--- 최신 데이터 가져오기 (stock_code 기반) ---
raw_stock_data, raw_disclosure_data = get_latest_data_for_stock(stock_code,
days=SEQUENCE_LEN + 10)
if raw_stock_data is None or raw_stock_data.empty:
return Response({"error": "데이터를 가져올 수 없습니다."},
status=status.HTTP_404_NOT_FOUND)

--- LSTM 입력 형태로 데이터 변환 및 스케일링 ---
input_lstm_data = transform_data_for_lstm(raw_stock_data, raw_disclosure_data,
SCALER_OBJECT, SEQUENCE_LEN, NUM_EXPECTED_FEATURES)
 })
```

```
--- 예측 실행 ---
try:
모델이 (1, 5) 형태의 정규화된 5일치 종가를 반환한다고 가정
predicted_prices_scaled = PREDICTION_MODEL.predict(input_sequence_final)[0] # (5,)
형태의 배열
except Exception as e:


--- 결과 해석 및 예측 날짜 생성: 정규화된 예측값을 원래 스케일로 복원 ---
predicted_prices_raw =
PRICE_SCALER.inverse_transform(predicted_prices_scaled.reshape(-1, 1)).flatten()

prediction_dates = pd.date_range(start=pd.Timestamp.now().normalize() +
pd.Timedelta(days=1), periods=5).strftime('%Y-%m-%d').tolist()

return {
     "stock_code": stock_code,
     "last_actual_price": float(last_actual_price_raw) if last_actual_price_raw else
None,
     "predicted_prices_5days": [
         {"date": date, "price": round(float(price), 2)} for date, price in
zip(prediction_dates, predicted_prices_raw)
     ]
--- 예측 값 표시 ---
def predict_stock_5days_view(request, stock_code):
stock_code = request.GET.get('code') #  GET 요청으로 종목 코드 받기

prediction_result = get_stock_prediction_5days(stock_code)
return JsonResponse(prediction_result)
```

04. 프로젝트 수행 결과

- 모델 #06. OpenAI api 챗봇 모델

알고리즘 기능 개요

"금융 특화 챗봇 시스템"
사용자가 입력한 메시지를 욕설 필터링한 후, OpenAI GPT-4 Turbo를 통해 금융 전문 답변을 생성하는 웹 기반 챗봇.

상세 내용

사용자 입력
•사용자가 #chatInput에 메시지를 입력 → sendBtn 클릭 또는 Enter 키 입력

욕설 필터링 요청 (/chatbot/filter_message/)
•filter_message() 함수에서 filter_curse()로 욕설을 정제함
•욕설 정제 후 클라이언트에 반영 (말풍선 수정)

챗봇 응답 생성 요청 (/chatbot/chat/)
•GPT-4 Turbo 모델 사용
•system prompt를 통해 챗봇 역할을 "금융 전문 어시스턴트"로 한정
•부적절한 질문에는 금융 관련이 아니라는 안내문구 출력

응답 처리
•로딩 중 메시지 표시 → 응답 수신 시 bot_response 출력

알고리즘 모델 코드

```
1. 사용자 메시지 수신
data = json.loads(request.body)
user_message = data.get('message', '')
————————————————————
2. 욕설 필터링
filtered_message = filter_curse(user_message)
filter_curse() 는 외부 모듈에서 import한 욕설/비속어 필터링 함수입니다.
————————————————————
3. OpenAI GPT-4 Turbo 호출
response = client.chat.completions.create(
    model="gpt-4-turbo",
    messages=[
        {"role": "system", "content": "...(금융 전용 지침)..."},
        {"role": "user", "content": filtered_message}
    ],
    temperature=0.7,
    max_tokens=1000
)
system prompt를 통해 챗봇이 **"금융 전문 어시스턴트"**로 제한됩니다.

4. 응답 반환
bot_response = response.choices[0].message.content
return JsonResponse({
    'status': 'success',
    'filtered_message': filtered_message,
    'bot_response': bot_response
})
```

04. 프로젝트 수행 결과

- 모델 #07. 비속어/욕설 필터링 모델

알고리즘 기능 개요

비속어/욕설 관련 git hub의 txt 파일을 csv로 변환 후 Kobert, 토그나이저 등의 기능을 활용하여 해당 git hub의 txt 파일의 문장을 학습한 모델생성 후 Django에 적용하여 utils.py의 소스코드와 연동하여 비속어/욕설 필터링

상세 내용

1. 비속어/욕설 관련 git hub 의 txt 파일을 csv로 변환
2. Kobert, 토그나이저 등의 기능을 활용하여 모델 학습
3. 생성된 모델을 Django 에 적용
4. Django 의 utils.py 의 소스코드와 연동하여 비속어/욕설 필터링

알고리즘 모델 코드

```
i# 3. 데이터 로드 및 전처리
import pandas as pd
import torch
from transformers import BertTokenizer, BertForSequenceClassification,
Trainer, TrainingArguments
from sklearn.model_selection import train_test_split

# dataset.csv 로드
data = pd.read_csv("dataset.csv")

# 레이블 형 변환 (문자열 -> 정수)
data["label"] = data["label"].astype(int)

# 토크나이저 로드
tokenizer = BertTokenizer.from_pretrained("monologg/kobert")

# 데이터 전처리
def preprocess(text):
    return tokenizer(text, padding="max_length", truncation=True,
max_length=128, return_tensors="pt")

data["input"] = data["sentence"].apply(lambda x:
preprocess(x)["input_ids"][0])
data["attention_mask"] = data["sentence"].apply(lambda x:
preprocess(x)["attention_mask"][0])
```

```
# 모델 및 학습 설정
model = BertForSequenceClassification.from_pretrained("monologg/kobert",
num_labels=2)
training_args = TrainingArguments(
    output_dir="./kobert_model",
    num_train_epochs=3,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    eval_strategy="epoch",  # evaluation_strategy -> eval_strategy로
변경
    save_strategy="epoch",
    logging_dir="./logs",
)
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=CurseDataset(train_data),
    eval_dataset=CurseDataset(test_data),
)
# 학습 실행
trainer.train()
# 모델 저장
model.save_pretrained("kobert_curse_model")
tokenizer.save_pretrained("kobert_curse_model")
```

04. 프로젝트 수행 결과

김가람 팀장

휴머로이드 담당

[로그인, 메인, 차트정보, 예측정보] 등 django app 을 나누어 각 app 별로 팀원들 간에 작업 후 git merge 하여 정상동작시키는 과정이 쉽지 않았습니다.
하지만 꾸준한 git 버전 관리, 철저한 mysql DB migration, 회의 및 해당 part에 대한 질문 등을 통해 모두가 협력한 기능을 django 프로젝트에 효과적으로 적용할 수 있게 되었던 것 같습니다.
많은 분들이 하나의 프로젝트 목표를 겨냥하여 진행하였기에 혼자 진행했다면 구현하지 못했을 기능 및 웹 퍼블리싱을 django 프로젝트를 통해 직접 볼 수 있는 좋은 기회였습니다.

정태영 팀원

불가능 기획자 담당

5주라는 길면 길고 짧으면 짧은 프로젝트 기간 동안에 전반적인 단계를 직접 경험해 볼 수 있어서 힘들지만 좋은 시간이라고 생각한다(좋게좋게 생각해야지)
데이터를 활용하는데 있어서 가장 중요한 것이 데이터를 수집하고 처리하는 과정과 솔루션을 해결하기 위해서 모델을 기획 및 설계하는 것임을 알게된거 같다.
예상치 못한 곳에서 갑자기 마주치게 되는 시행착오를 겪게되어 원치않게 고통받고 힘들었지만 다양한 문제를 만나서 여러가지 방법을 찾아보고 적용해보고 실패하고 어떤건 성공하며 하나하나 해결을 하는 과정이 의외로(?) 재미있어서 '아, 이번 생은 망한 쪽인가?' 라는 생각을 하면서 스스로를 응원하게 되었다.
스페셜 땡스 투 AI+X

문성준 팀원

코드 알케미스트 담당

4주간의 팀 프로젝트를 통해, 6명의 팀원과 함께 **퀀트 기반 주식 투자 프로그램**을 개발하는 값진 경험을 하였습니다. 단순한 구현을 넘어, 기존에 존재하던 시스템을 개선하고 확장해 나가는 과정에서 기술적 한계와 마주했고, 이를 뛰어넘는 도전의 연속이었습니다.
야후파이낸스 API와 네이버 금융 데이터를 활용하며 수많은 데이터들을 가공하고 새로운 가치를 창출하는 과정은 무척 흥미로웠습니다.
복잡한 알고리즘과 분석 로직을 구현하면서도 시간 가는 줄 모를 정도로 몰입했던 순간들이 지금도 생생합니다.
이번 프로젝트는 단지 결과물을 만드는 것 이상의 의미를 지녔습니다. 지난 6개월간 파이썬을 배우며 쌓아온 역량이 실제 문제 해결로 이어졌고, 그 과정에서 저는 **기술에 대한 이해와 사고의 깊이**가 한층 더 성장했음을 체감할 수 있었습니다.
이 여정을 통해 개발자로서의 자신감과 방향성을 얻었으며, 앞으로도 이 즐거움과 성장을 계속 이어가기 위해 **더 치열하게 정진할 것**입니다.

04. 프로젝트 수행 결과

이승배 팀원

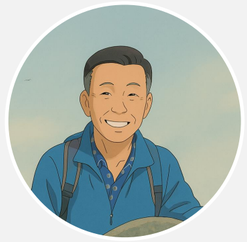
지친 팀원 채찍질 담당

와, 5주 동안 예측 페이지 하나 만든다고 제 영혼을 갈아 넣었습니다. 파이낸스 데이터로더에서 데이터 불러올 때부터 DB를 차곡차곡 정리하고, 그걸 또 모델에 넣어서 학습시키고... 예측 그래프가 화면에 딱! 하고 뜰 때까지 매 순간이 도전이었죠. 솔직히 중간중간 '아, 이걸 왜 한다고 했을까' 싶은 순간도 있었지만... (소주 한 잔이 참 간절했습니다 ㅎ), 결국 팀원들과 으쌰으쌰해서 여기까지 온 것 같습니다. 특히 제가 맡은 예측 파트가 잘 돌아가서 표시되는 걸 보니, 그동안의 고생이 눈 녹듯 사라지네요. 힘들었지만 이 번 프로젝트 덕분에 정말 값진 경험했습니다!

조지형 팀원

홍일점 팀 비쥬얼 담당

비록 chat GPT를 많이 이용하긴 했지만 배우지 않은 것들을 스스로 찾아내는 능력을 얻었고 예상치 못한 에러가 발생하여 내가 원하는대로 프론트가 나오지 않아 멘붕이고 다시 처음 부터 작업하는 위기가 있었지만 다양한 시도를 통해 에러를 해결하는 위기 대처를 경험했습니다. 다른 팀원들이 구현한 FinanceDataReader, Transformer, Kobert, Cuda, LSTM, Keras, TensorFlow 등 많은 라이브러리를 스스로 배우고 경험해봤습니다. 많은 분들이 같은 목적을 갖고 진행하였기에 혼자 진행했다면 사용해보지 못했을 기능들도 다양하게 경험한 좋은 기회였습니다.



조준행 팀원

백코치 담당

AI 관련하여 처음 접했는데 신기하고 즐거웠어요
실습을 통해 감을 잡을 수 있고 초년 사회생활 기분이었습니다
변화하는 세상을 여러분들과 함께 느낄 수 있어 더욱 행복한 시간을 보냈습니다
모두 건강하고 잘 되길 바랍니다

- 라이브 기능 시연

> python manage.py runserver

감사합니다

아트로포스 – AI를 활용한 차세대 주식 예측 서비스