

시퀀스 레이블러 및 금융거래전략 시스템 개발

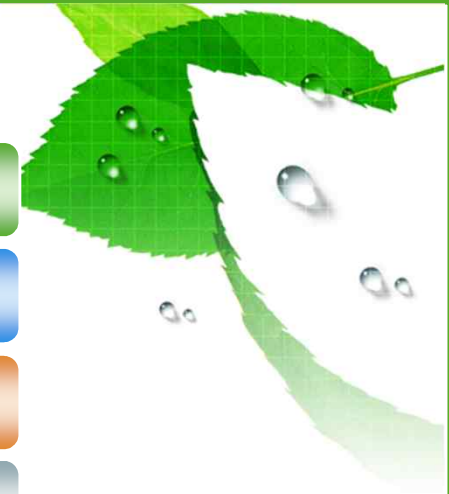
Development of Sequence Labeler and Financial Trading Strategy System



종합설계 2차 설계서

차 례

2



◆	종 합 설 계 개 요
◆	관 련 연 구 및 사 례
◆	시 스템 수 행 시 나 리 오
◆	시 스템 구 성 도
◆	시 스템 모 들 상 세 설 계
◆	개 발 환 경 및 개 발 방 법
◆	데 모 환 경 설 계
◆	업 무 분 담
◆	종 합 설 계 수 행 일 정
◆	필 요 기 술 및 참 고 문 헌

종합 설계 개요

❖ 연구 개발 배경

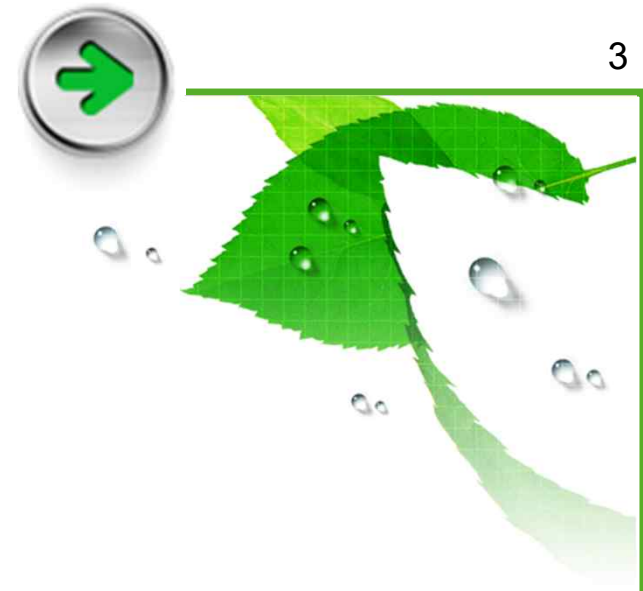
- 시스템 트레이딩의 수요 증가
- 레이블링 기술의 중요성 증가

❖ 연구 개발 목표

- 시계열 데이터 레이블링
- 거래 전략 개발
- 자동 거래 구현
- 다 종목 시뮬레이션 가능

❖ 연구 개발 효과

- 주식과 관련된 레이블링 데이터 제공
- 유동적인 전략 생성과 검증이 가능하다.



종합설계 개요

❖ Sequence Labeling

➤ 적용 분야

- ✓ DNA 암기 서열 분석
- ✓ 자연어 처리
- ✓ 음성 및 동작 인식

-> 즉 Sequence로 해석될 수 있는 어떠한 종류의 데이터 처리

➤ 시계열 데이터

- ✓ 하나 또는 여러 사건에 대해 시간의 흐름에 따라 일정한 간격으로 관찰하여 기록한 Sequence Data
- ✓ 예시) 주식 데이터, 날씨 데이터, 센서 데이터
- ✓ 장점
 - 시간이 지남에 따라 어떻게 달라지는지 명확함
 - 발생 추세가 명확함
 - 예측 모델링 및 예측에 가장 적합함



종합설계 개요

❖ Sequence Labeling

➤ Stock Charts에서의 Labeling

- ✓ 시계열 데이터의 각 지점에 identity를 표시
- ✓ 어떤 시간 t 에 발생한 데이터는 t 앞뒤의 데이터로 부터 의존적
- ✓ t 앞뒤의 데이터를 분해 및 해석하여 범주형 변수 Labeling
- ✓ Labeling한 데이터를 활용하여 다양한 거래 전략 생성
- ✓ 추세 분해, 주기 분해, 기술적 지표, 특성 단순화 등을 활용

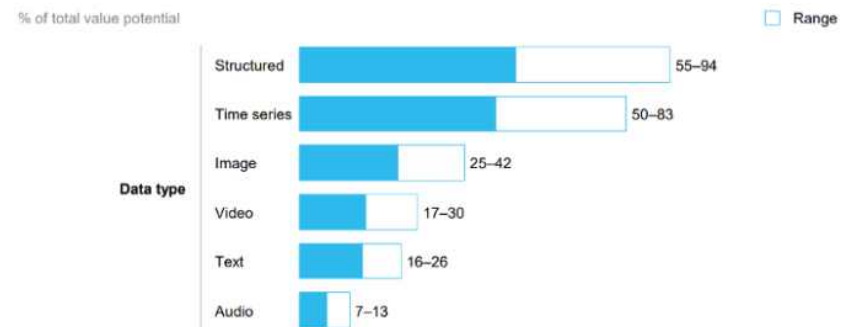
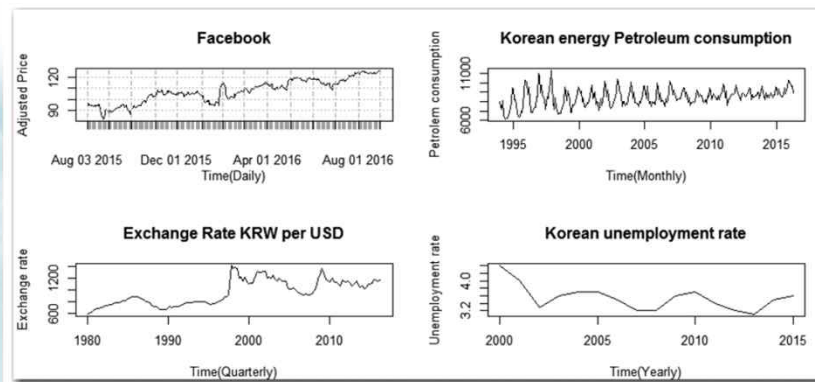


Figure 1. 데이터 유형에 따른 잠재적 인공지능 활용 가치 (Source: McKinsey Global Institute analysis)

관련 연구 및 사례

❖ 예스트레이더 - 예스랭귀지

- 프로그래머에서 기본 제공하는 지표나 전략을 사용하거나 사용자가 지표나 전략을 새로 생성할 수 있다.
- 시뮬레이션(가상매매), 성능평가 기능을 통해 지표와 전략을 투자 종목에 적용할 수 있다.
- 단점
 - ✓ 외부데이터를 추가하여 사용할 수 없다.
 - ✓ 시뮬레이션을 하나의 종목에 대해서만 적용할 수 있다.

관련 연구 및 사례

❖ 예스트레이더

➤ 예스트레이더와 예스랭귀지 화면



관련 연구 및 사례

❖ 예스트레이더 - 예스스팟

- 자바스크립트 기반 시스템 트레이딩 프로그램으로 예스랭귀지보다 더 자유롭게 개발자의 의도대로 흐름을 정할 수 있다.
- 단점
 - ✓ 시뮬레이션은 불가능하다.
 - ✓ 편의성이 떨어진다.
 - ✓ 자바스크립트 활용능력이 필요하다.



관련 연구 및 사례

❖ 예스트레이더 - 예스스팟

YesSpot Studio - 하이투자증권 - 메인프로그램을 찾을 수 없습니다.

The screenshot shows the YesSpot Studio IDE with the following components:

- Menu Bar:** 파일(F), 편집(E), 보기(V), 도움말(H)
- Toolbar:** Standard development tools like file operations, undo, redo, and search.
- Left Panel:**
 - test1:** Project tree showing a 'Main' script object.
 - 스크립트 객체:** List of script objects with columns '이름' (Name) and '자료형' (Data Type).
 - 사용자정의 모듈:** List of user-defined modules with columns '이름' (Name) and '설명' (Description).
- Center Panel:** C# script editor for 'Main' class.


```

25 | 20); // 최대진입횟수
26 |
27 |
28 | // 해당 차트는 최대 5개까지만 가능하므로 아래와 같이 경우를 나눠서 코딩해 주어야 합니다.
29 |
30 | if ( JongmokNum <= 5)
31 | {
32 |     for (var i = 0; i < JongmokNum ; i++)
33 |     {
34 |         JongmokCode[i] = Main.GetItemCodeInInterest(관심종목명, i);
35 |         ChartSet[i] = new ReqChartItem(JongmokCode[i], 5, CHART_PERIOD_MINUTE, 5000, CHART_
36 |         SystemSet[i] = new SystemInfo(시도명, YL_TYPE_NORMAL, null, TradeSet, null);
37 |         Main.ReqChartEx(ChartSet[i], SystemSet[i]);
38 |     }
39 | }
40 |
41 | else
42 | {
43 |     for (var i = 0; i < 5 ; i++){
44 |         JongmokCode[i] = Main.GetItemCodeInInterest(관심종목명, i);
45 |         ChartSet[i] = new ReqChartItem(JongmokCode[i], 5, CHART_PERIOD_MINUTE, 5000, CH
46 |         SystemSet[i] = new SystemInfo(시도명, YL_TYPE_NORMAL, null, TradeSet, null);
47 |         Main.ReqChartEx(ChartSet[i], SystemSet[i]);
48 |     }
49 | }
50 |
51 | }
52 |
53 |
54 | // 해당 차트에서 발생한 신호에 의해 주문을 넣는 로직
55 |
56 |
57 | function Main_OnRiseSignal(ChartEx, Signal)
58 | {
59 |     //buy신호 발생
60 |     if (Signal.signalKind == 1)
61 |     {
62 |         // 매수주문
63 |         Account1.OrderBuy(Signal.code, Signal.count, Signal.price, 0);
64 |     }

```
- Right Panel:**
 - 속성 (Properties):** Shows properties for 'Method' (GetUserValue, 반환값: 문자열) and '매개변수' (sName: 문자열).
 - 화면데이터 (Screen Data):** Table with columns '이름' (Name) and '인덱스' (Index).
 - 객체 정보:** Section for '화면데이터' and '사용자정의 모듈 관리자'.
 - 사용자정의 모듈 관리자:** Table with columns '이름' (Name) and '설명' (Description).
- Bottom Panel:**
 - 오류보고 (Error Log):** Shows an error for 'test1' with message '검증이 완...'.
 - 디버깅:** Includes '오류보고', '찾기결과1', '찾기결과2', and '실행결과' tabs.

관련 연구 및 사례




❖ 뉴지스탁 – 젠포트

- 프로그램에서 제공하는 전략을 조합하여 알고리즘 트레이딩을 할 수 있는 프로그램이다.
- 코딩을 하지 않고 Drag&Drop 방식으로 투자 전략을 생성할 수 있다.
- 투자 종목을 다양한 기준으로 선정하여 시뮬레이션이 가능하다.
- 단점
 - ✓ 사용자가 새로운 지표나 전략을 자유롭게 생성할 수 없다.
 - ✓ 시뮬레이션 사용이 유료이다.(일봉, 틱 백테스트 각각 일 1회 무료)
 - ✓ 시뮬레이션은 일봉에 대해서만 지원한다.
 - ✓ 외부데이터 사용이 불가능하다.
 - ✓ 2007년 이전 데이터로는 백테스팅이 불가능하다.

관련 연구 및 사례

❖ 뉴지스탁 – 젠포트



젠포트란? 젠스토어 젠프로 젠트레이더 아카데미

일봉 백테스팅 포트 만들기 젠포트가 제공하는 다양한 조건으로 나만의 투자전략을 만들어 보세요. 젠프로 > 일봉 백테스팅 포트 만들기

일반 NH

Step 1. 기본 설정
기본 조건들을 설정하세요.

Step 2. 매매대상 설정
어떠할 대상들을 설정하세요.

Step 3. 매매조건 설정
매수/매도조건을 설정하세요.

Step 4. 포트 최적화
최적의 조건을 찾아보세요.

Step 5. 포트 완성!
백테스팅 포트가 완성되었습니다.

포트 제목 생물 포트 만들기 **포트 설명** (선택사항)

tonykim_20210106_142317 포트물리요에 대한 설명을 100자 이내로 남겨주세요.

기본 조건

운용자금 ⑦	운용기간 ⑦	잔여 과거데이터 백테스팅 이용권 ⑦ 1 회	수수료율 ⑦	슬리피지 ⑦
5,000 만원	2020/01/06 부터 2021/01/05 까지	전제 무료기준일 상한기준 30% 변동일 이전 이후	0.100 %	0.000 %

자산 배분 조건

자산 배분 사용 ⑦	투자 성향 ⑦	자산 배분 비중 ⑦	리밸런싱 주기 ⑦
<input checked="" type="radio"/> 미사용 <input type="radio"/> 사용	중립형 성장선택	ETF: 30 % 주식: 60 %	3개월

*자산배분 중립의 수평과 가격은 '전일종가 +5%' 기준으로 산정됩니다.

매수조건 일봉매수 분할매수 등락매수

매수 가격기준 ⑦

전일종가 +0.0 %

종목별 매수조건

종목당 매수비율 ⑦	종목당 최대 매수금액 ⑦
10 %	제한없음 만원

매수 종목수 조건

최대 보유종목 수 ⑦	1일 최대 매매종목 수 ⑦
-------------	----------------

매도조건

퇴출(stop) 조건 ⑦

목표가	손절가
매수가 + 10.0 %	매수가 - 10.0 %

보유일 관련 조건

종목 최소 보유일 ⑦	종목 최대 보유일 ⑦
0 일	5 일

보유일 만기 매도 가격기준 ⑦

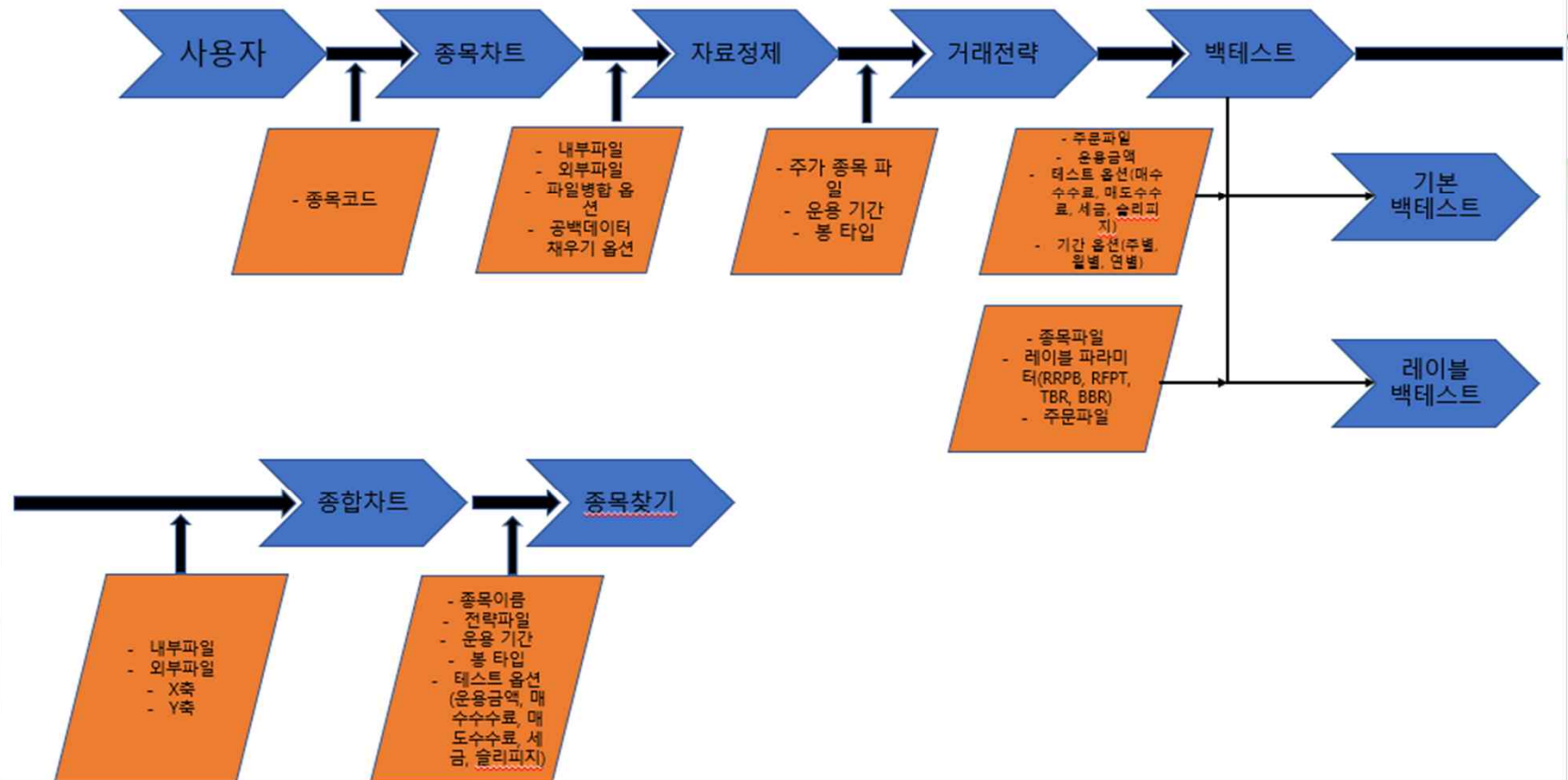
관련 연구 및 사례

❖ 차별성



	에스 트레이더	에스 스팟	젠포트	목표 시스템
시뮬레이션	- 한 가지 전략에 대해 다양한 종목 시뮬레이션이 불가능	- 불가능	- 2007년 과거 데이터로 시뮬레이션이 불가능 - 시뮬레이션 사용이 유료	- 한 가지 전략으로 다 종목 시뮬레이션이 가능하다. - 과거 모든 데이터 시뮬레이션 가능
사용성	에스랭귀지 학습 필요(난이도: 중)	자바스크립트 학습 필요(난이도: 상)	사용자 수식 작성 불필요하므로 쉬움(난이도: 하)	파이썬 언어를 기반으로 한 조건문 학습 필요
사용자 정의 함수	가능	가능	불가능	가능
외부 데이터	불가능	주가 데이터	불가능	시계열 데이터 및 레이블 데이터

시스템 수행 시나리오



시스템 수행 시나리오 - 레이블

	open	high	low	close	volume	change	top	bottom	top_zone	bottom_zone	label_target	code	dType
Date													
2020-03-19	75500	75800	65800	66000	11140367	-0.056088	0	1	0	1	close	000660	D
2020-03-20	71800	76200	70100	74800	8750101	0.084056	1	0	1	0	close	000660	D
2020-03-23	69700	71600	66000	66400	8192363	-0.072193	0	1	0	1	close	000660	D
2020-03-24	73000	78800	72100	78700	7247345	0.134006	0	0	0	0	close	000660	D
2020-03-25	82000	84700	79600	84500	8787217	0.073696	1	0	1	0	close	000660	D

주가 데이터

<INPUT>
- 레이블 기준 및 대상
- 사용자 파라미터

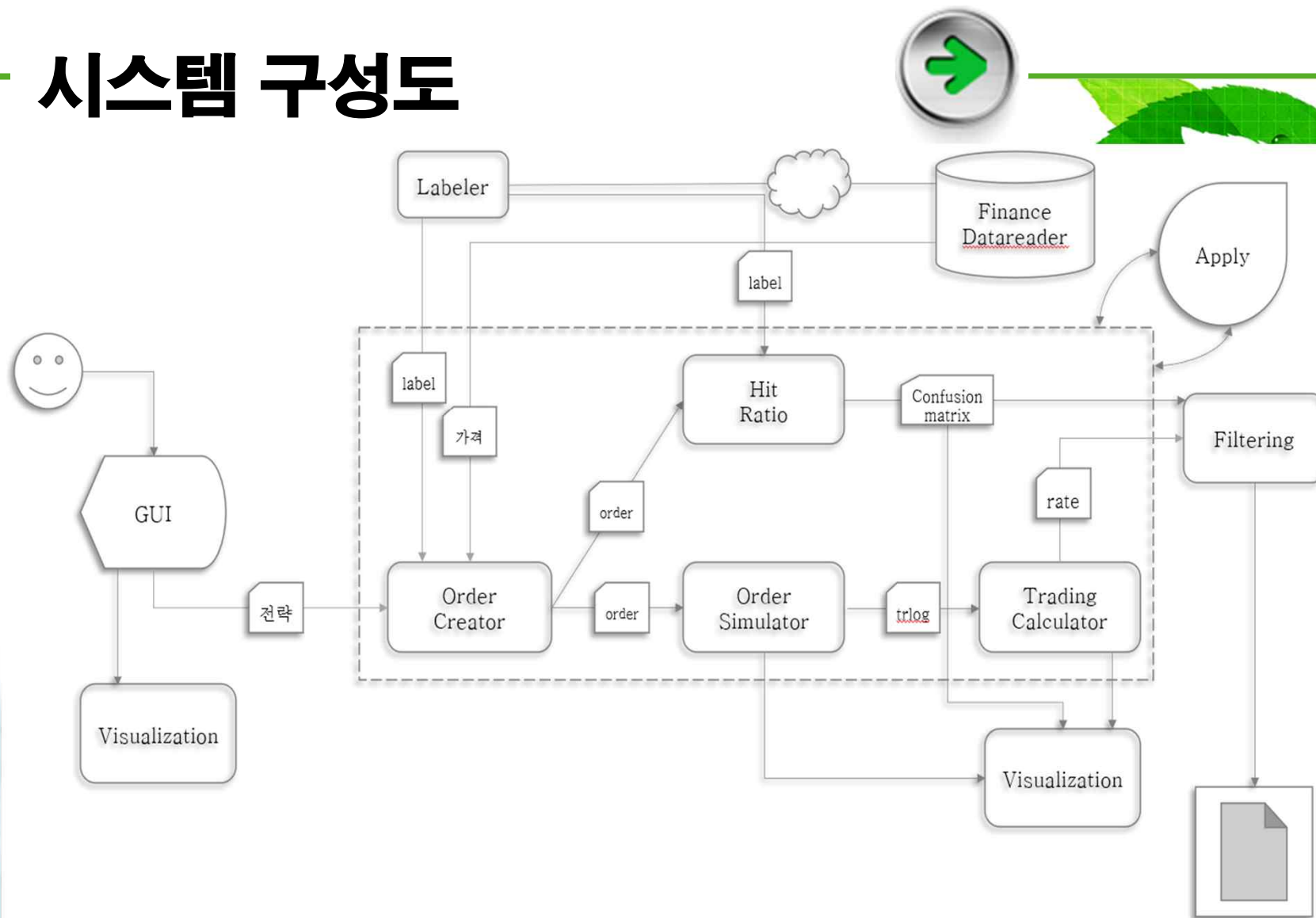
<예시>
일봉 차트에서 종가 기준 n% 이상
차이 나는 지점들을
top, bottom 컬럼으로 레이블

레이블링

<OUTPUT>
- 레이블 컬럼이 추가
된 데이터

레이블링 데이터를
이용한 거래 전략
생성

시스템 구성도



시스템 모듈 상세 설계

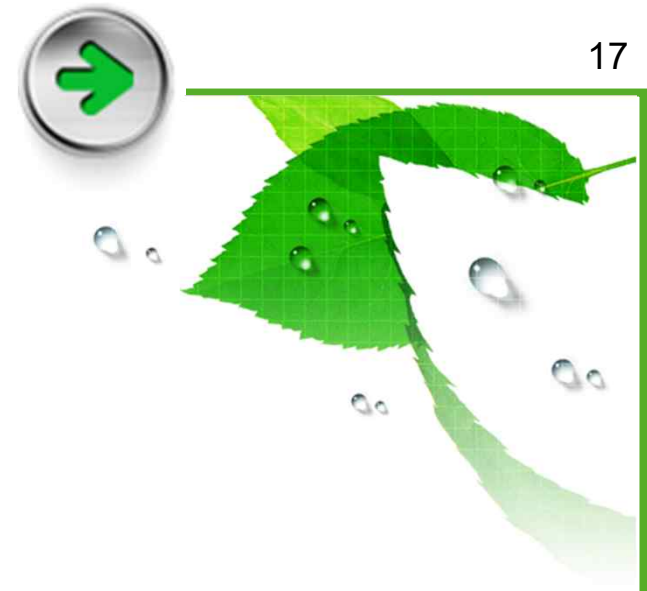
0. 모듈 목록

- gatherer: 주가 가격데이터 수집 및 저장하는 모듈
- indicator: 기술적지표 값을 생성하여 데이터프레임 컬럼에 추가하는 모듈
- labeler: 주가 차트의 지점들을 특정 범주 및 그룹으로 분류하는 모듈
- order_creator: 전략파일을 입력 받아 전략에 맞는 주문파일을 생성하는 모듈
- simulator: 주문파일을 이용하여 거래 시뮬레이션하는 모듈
- calculator: 거래내역을 입력 받아서 주(월, 연)간 수익률을 계산하는 모듈
- hit_tester: 저점/고점 레이블링 데이터와 일치성의 정도를 비교하는 모듈
- GUI: 파일 입/출력, 차트 시각화 등을 수행하는 모듈

시스템 모듈 상세 설계

1. gatherer

- 주가 가격데이터 수집 및 저장하는 모듈
- Class: Gatherer()
- 사용 라이브러리: FinanceDataReader, Pandas
- 멤버변수(public)
 - df_krx: 한국거래소에 상장된 종목정보 데이터프레임



시스템 모듈 상세 설계

2. indicator

- 기술적지표 값을 생성하여 데이터프레임 컬럼에 추가하는 모듈
- Class: 없음, 함수로만 모듈 구성
- 사용 라이브러리: ta-lib
- 지원하는 지표:
 - 볼린저밴드
 - RSI
 - MACD
 - MA(이동평균)
 - EMA(지수이동평균)
 - CMO(상드 모멘텀 오실레이터)
 - 스톡캐스틱
 - 스톡캐스틱 패스트
 - ATR
 - Super Trend
 - Clustering



시스템 모듈 상세 설계

3. Labeler

- 주가 차트의 지점들을 특정 범주 혹은 그룹으로 분류하는 모듈
- 레이블링 데이터는 전략 개발 과정 및 전략 검증 과정에서 사용할 수 있다.
- Class: 없음, 함수로만 모듈 구성
- 사용 라이브러리: ta-lib
- 지원하는 레이블러:

✓ candles

- 캔들 종류(Candle Type)
- 캔들 모양(Candle Shape)

✓ cross

- 이동평균 크로스(MA Cross)
- 이중지수이동평균 크로스 (DEMA Cross)
- 거래량가중이동평균 크로스(VWMA Cross)
- 이동평균수렴확산지수 크로스(MACD Cross)
- 패스트-스토캐스틱 크로스(Fast-Stochastic Cross)
- 슬로우-스토캐스틱 크로스(Slow-Stochastic Cross)

✓ Classification

- 적삼병(Three white soldiers)
- 흑삼병(Three black crows)
- 볼린저 밴드(Bollinger Bands)
- 가격 변화 비율(Rate Of Change)
- 탐바툼: top_bottom



시스템 모듈 상세 설계

4. order_creator

- 전략파일을 입력 받아 전략에 맞는 주문파일을 생성하는 모듈
- Class: OrderCreator
- 사용 라이브러리: FinanceDataReader, Pandas
- 멤버변수(public)
 - mode
 - True: network, False: local
 - Type: boolean
 - order_requests
 - Request(Class)로 생성된 객체들을 listtype으로 모아둠
 - type: list of Request(instance)
- 멤버변수(private)
 - trade_list
 - 전략식을 만족하여 주문에 필요한 정보들의 리스트를 담은 리스트
 - type: list



시스템 모듈 상세 설계

5. simulator

- 주문파일을 이용하여 거래 시뮬레이션하는 모듈
- 주식종목, ETF, 가상화폐(미구현)에 대해서 시뮬레이션이 가능하다.
- Class: Simulator
- 사용 라이브러리: FinanceDataReader, Pandas
- 멤버변수(public)
 - _trd_log
 - 거래기록을 저장하는 DataFrame
 - _stu_log
 - 상태를 저장하는 DataFrame
 - _cash
 - 자본금을 저장한다.
 - _order_sheet
 - 주문내역이 들어있는 데이터프레임
 - DataFrame화한 주문서(파일)를 저장한다.
 - _buying_fee
 - 매도 수수료
 - _selling_fee
 - 매수 수수료
 - _national_tax
 - 국가 세금
 - _weight
 - 매도 수수료 + 매수 수수료 + 국가 세금
 - _slippage
 - slippage 값을 저장한다.
 - _connect_network
 - 네트워크 연결 여부를 저장한다.
 - stock_name
 - 종목이름을 저장한다.



시스템 모듈 상세 설계

6. calculator

- 거래내역을 입력 받아서 주(월, 연)간 수익률을 계산한다.
- Class: Calculator
- 사용 라이브러리: FinanceDataReader, Pandas
- 멤버변수(public)
 - _wast_log
 - 주간 수익률을 기록하는 DataFrame
 - _mast_log
 - 월간 수익률을 기록하는 DataFrame
 - _yast_log
 - 연간 수익률을 기록하는 DataFrame
 - _dates
 - 거래 시작 날짜와 마지막 날짜 사이에 모든 날짜를 저장하는 리스트
 - _weeks
 - 금요일에 해당하는 날짜를 저장하는 리스트
 - _months
 - 월에 마지막 날짜를 저장하는 리스트
 - _years
 - 연에 마지막 날짜를 저장하는 리스트
 - stock_name
 - 주가 종목이름을 저장
 - _trd_log
 - 거래 기록이 있는 데이터프레임
 - DataFrame화한 거래기록(파일)을 저장한다.
 - atlog
 - 손익률계산 결과를 담은 딕셔너리
 - {손익률 계산 날짜, 자산, 기간별 손익률, 누적 손익률}
 - type: dictionary



시스템 모듈 상세 설계

7. hit_tester

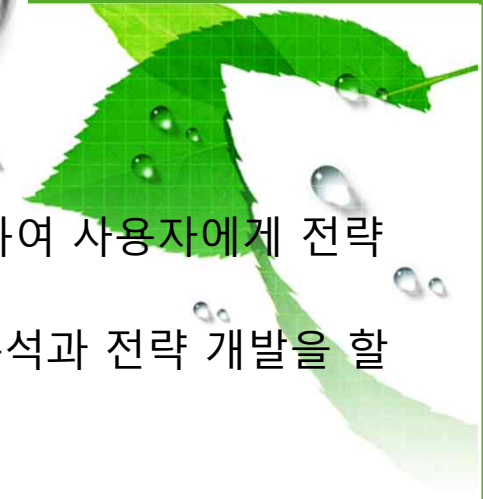
- 가격레이블과 주문이 얼마나 일치하는지 비교한다.
- 사용 라이브러리: numpy



시스템 모듈 상세 설계

8. GUI

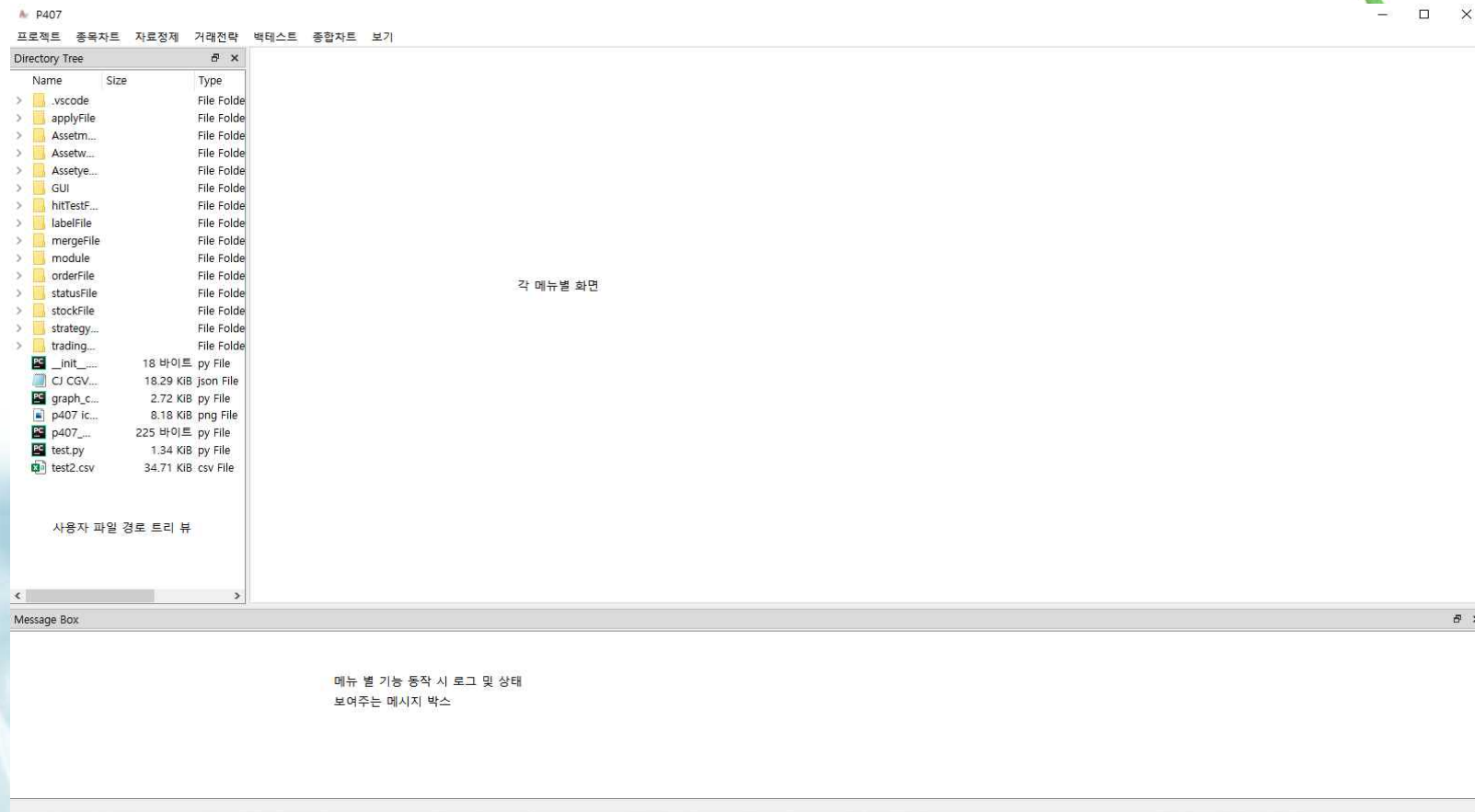
- 앞서 소개한 7개의 내부 모듈을 종합하여 GUI로 시각화 하여 사용자에게 전략 개발 서비스를 제공한다.
- 주식 차트를 시각화 하여 사용자가 보다 편리한 기술적 분석과 전략 개발을 할 수 있도록 돕는다.



시스템 모듈 상세 설계

8. GUI

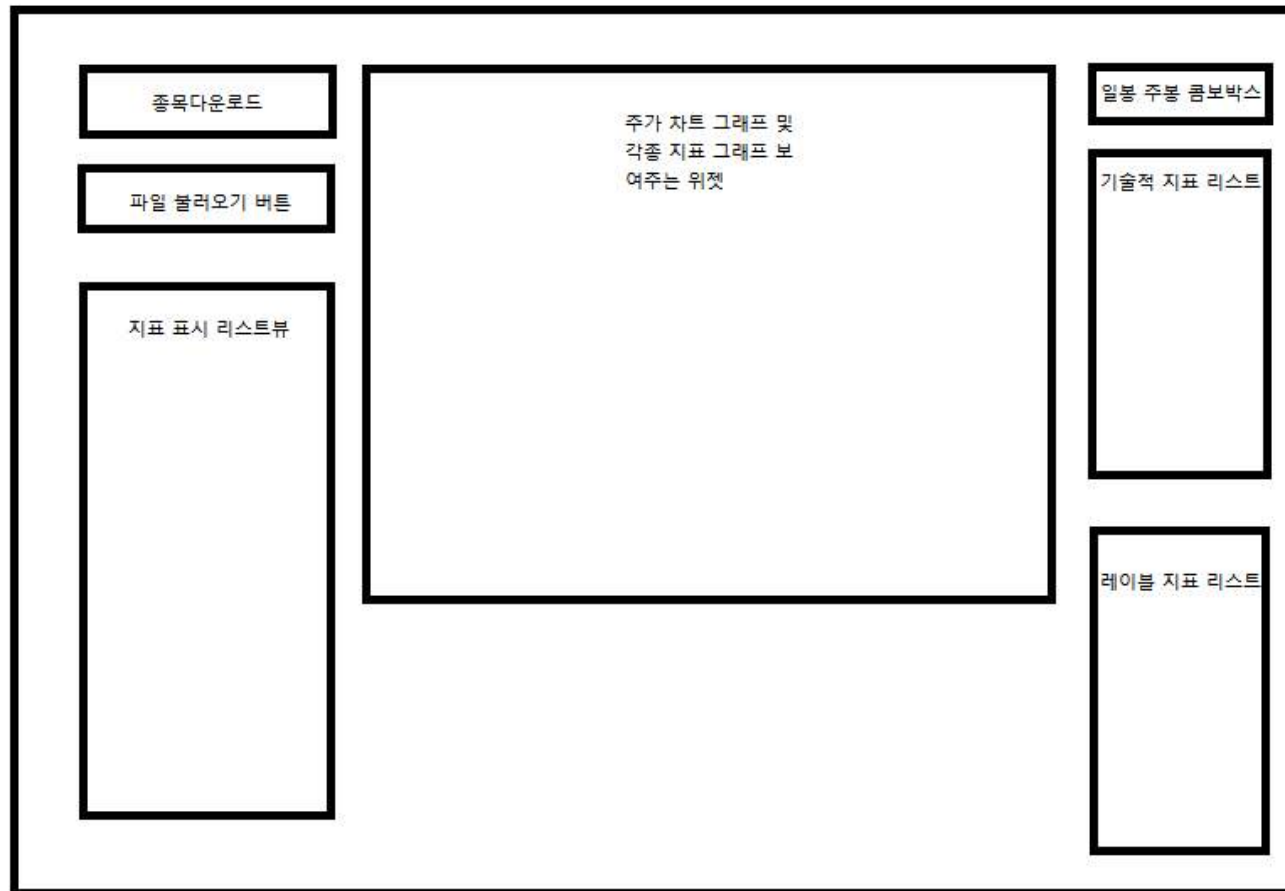
- GUI 화면 설계
 - 전체화면 및 메뉴



시스템 모듈 상세 설계

8. GUI

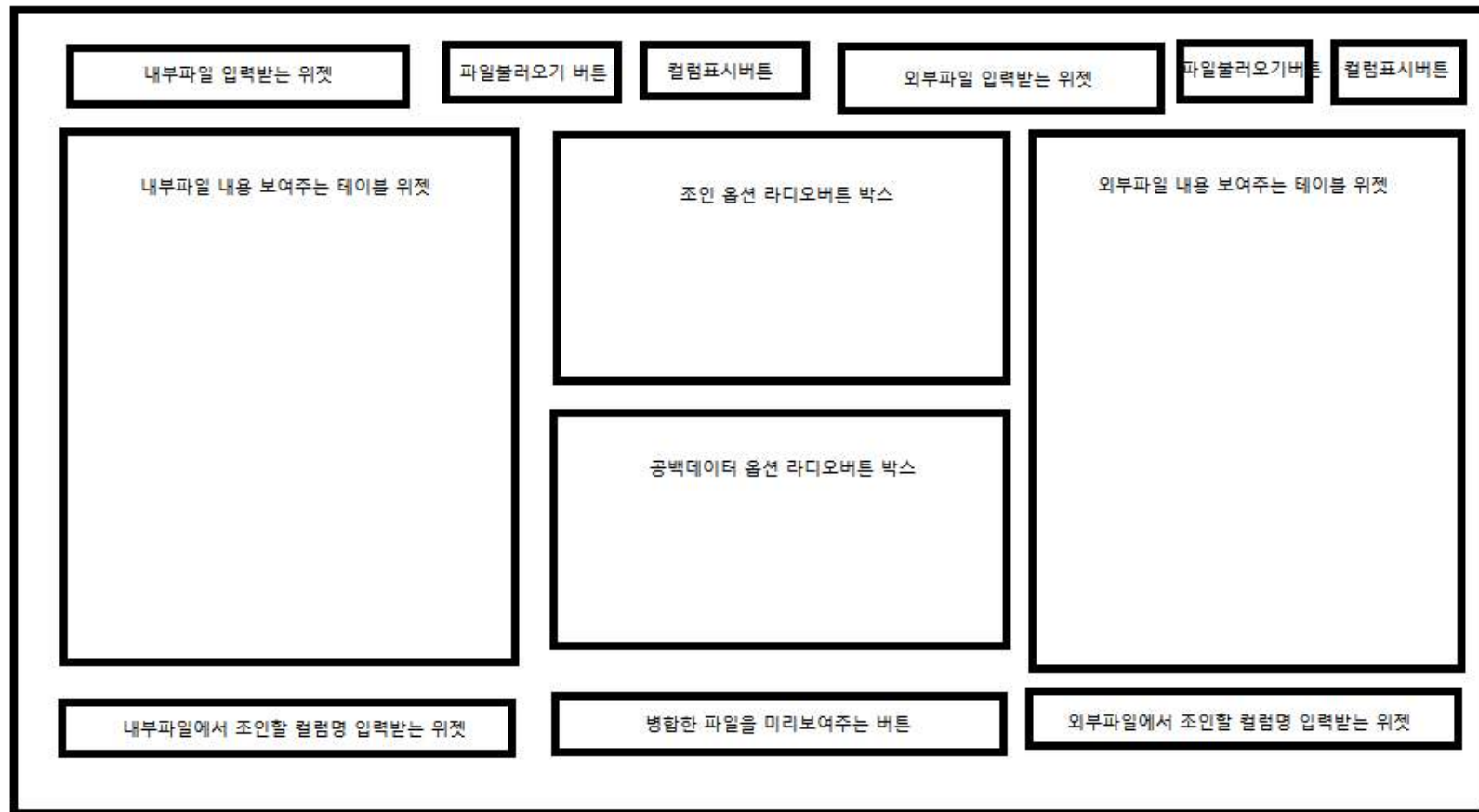
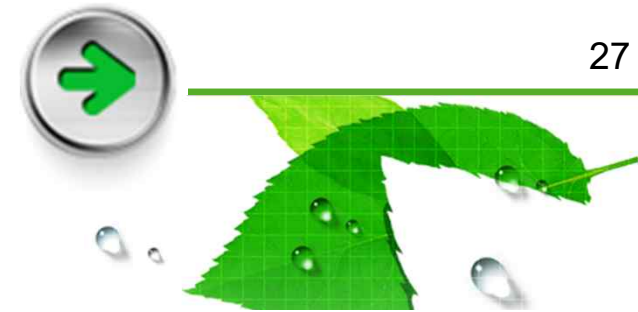
- 종목차트 메뉴



시스템 모듈 상세 설계

8. GUI

- 자료정제 메뉴



시스템 모듈 상세 설계

8. GUI

- 단순전략 메뉴 - 로컬파일모드

로컬파일모드 라디오버튼 네트워크모드 라디오버튼 적용종목 입력받는 위젯 지표표시 버튼 전략 운용기간 입력받는 위젯

사용할 지표 보여주는 리스트뷰 지표 선언식 텍스트뷰

거래 전략 편집기 텍스트뷰

전략 조건식 검증 버튼 주문 생성 버튼

시스템 모듈 상세 설계

8. GUI

- 단순전략 메뉴 - 네트워크모드

종목파일 입력받는 위젯

전략 운용할 기간 입력받는 위젯

로컬파일모드 라디오버튼 네트워크모드 라디오버튼

일봉, 주봉 라디오버튼 박스

사용할 지표 보여주는 리스트 위젯

지표 선언식 텍스트뷰

거래 전략 편집기 텍스트뷰

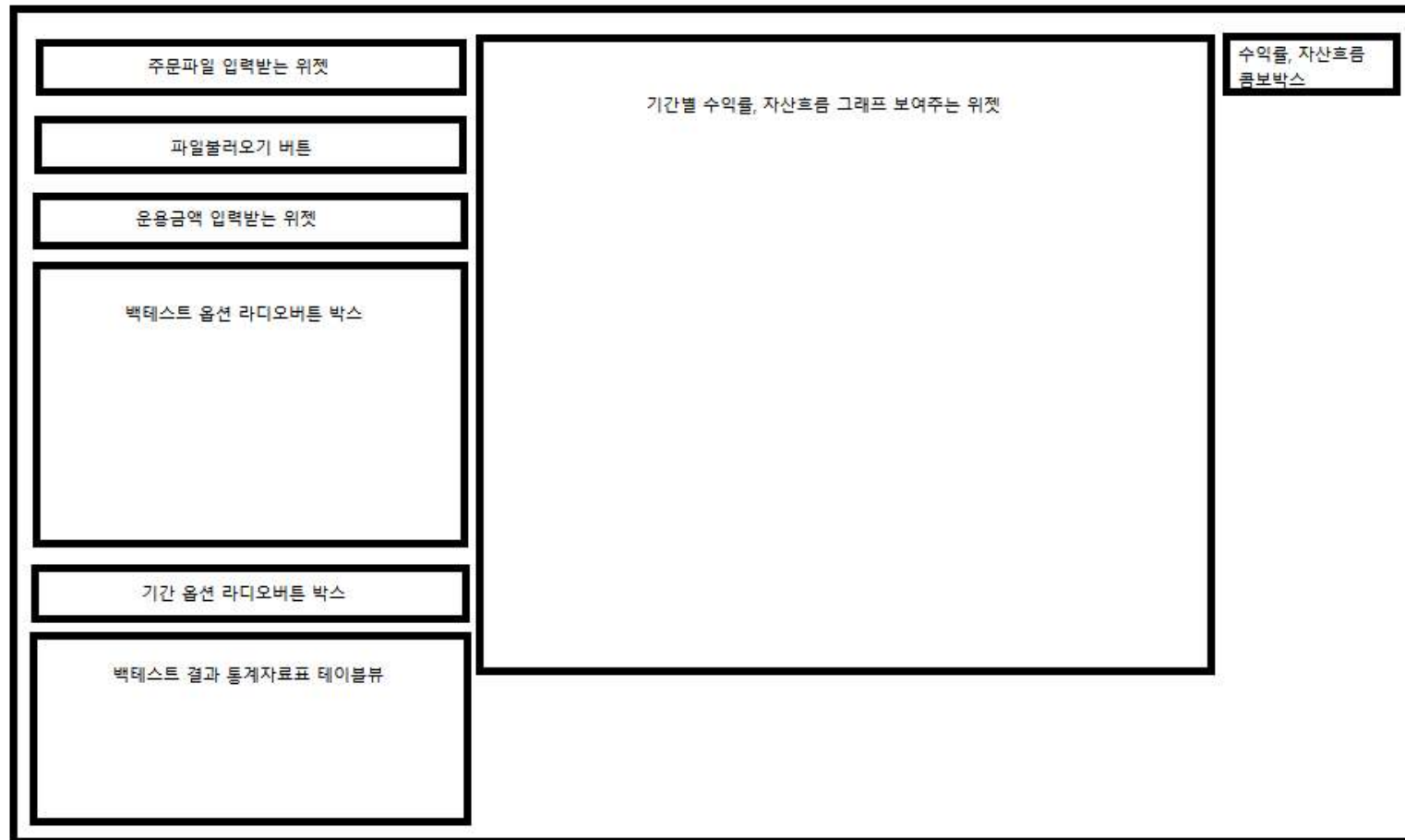
전략 조건식 검증 버튼

주문 생성 버튼

시스템 모듈 상세 설계

8. GUI

- 백테스트 메뉴 – 기본 백테스트



The GUI layout is as follows:

- Left Sidebar (Input/Action Area):**
 - 주문파일 입력받는 위젯 (Widget for receiving order file)
 - 파일불러오기 버튼 (File load button)
 - 운용금액 입력받는 위젯 (Widget for receiving operation amount)
 - 백테스트 옵션 라디오버튼 박스 (Backtest option radio button box)
 - 기간 옵션 라디오버튼 박스 (Period option radio button box)
 - 백테스트 결과 통계자료표 테이블뷰 (Table view for backtest results and statistics)
- Central Area:**
 - 기간별 수익률, 자산흐름 그래프 보여주는 위젯 (Widget for displaying period-wise return and asset flow graph)
- Right Sidebar:**
 - 수익률, 자산흐름 콤보박스 (Return and asset flow combobox)

시스템 모듈 상세 설계

8. GUI

- 백테스트 메뉴 – 레이블 백테스트

종목파일 입력받는 위젯

파일불러오기 버튼

레이블 파라미터 입력받는 위젯 박스

레이블 데이터 파일 생성 버튼

주문 파일 입력받는 위젯

파일 불러오기 버튼

레이블 파일 입력받는 위젯

레이블 백테스트 실행 버튼

레이블된 그래프 보여주는 위젯

시스템 모듈 상세 설계

8. GUI

- 종합차트 메뉴

내부파일 설정 그룹박스

내부파일 입력받는 위젯

컬럼표시 버튼

내부파일에 존재하는 컬럼 보여주는 리스트뷰

병합할 기준컬럼 입력받는 위젯

그래프 합성 라디오버튼

그래프 병렬 라디오버튼

외부파일 설정 그룹박스

외부파일 입력받는 위젯

컬럼표시 버튼

외부파일에 존재하는 컬럼 보여주는 리스트뷰

병합할 기준컬럼 입력받는 위젯

그래프 추가 버튼

그래프 초기화 버튼

합성 및 병렬 배치된 그래프 보여주는 위젯

개발 환경



33



구분	상세내용
S/W 개발 환경	OS Windows 10
	개발 환경(IDE) VS code, Spyder
	개발 언어 Python
	프레임 워크 PySide2
	주요 라이브러리 matplotlib, plotly, PySide2, pandas, finance-datareader, ta-lib, Scikit-learn
프로젝트 관리 환경	형상관리 Git, GitHub
	의사소통 관리 Notion

개발 환경 (2)

❖ 졸업작품 GitHub 주소

➤ [https://github.com/\[redacted\]/\[redacted\].git](https://github.com/[redacted]/[redacted].git)

❖ 팀원 별 GitHub ID

- 팀장: [redacted]
 - ✓ ID: [redacted]
- 팀원: [redacted]
 - ✓ ID: [redacted]
- 팀원: [redacted]
 - ✓ ID: [redacted]



개발 방법

1. Application

- 개발언어는 Python을 이용
- PySide2를 이용한 GUI 구현

2. Server 및 DB

- 라이브러리를 이용하여 데이터 수집 및 처리
- 각 모듈 별 파일 산출물 사용자 로컬 파일 디렉토리 저장



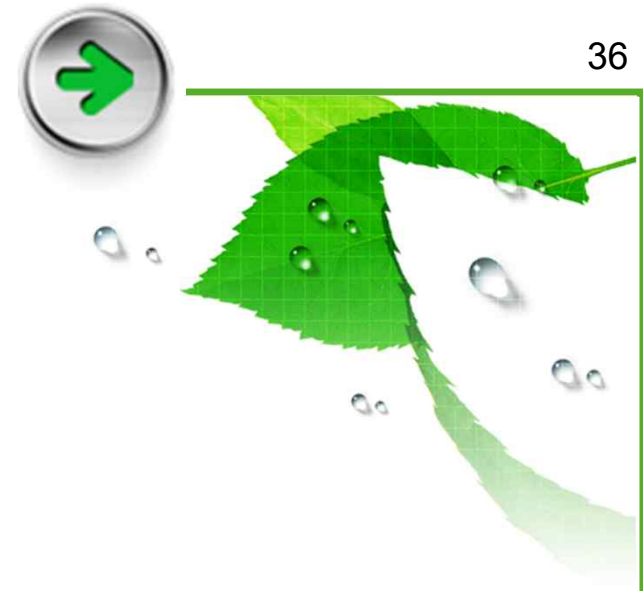
데모 환경 설계

❖ 데모 환경

- Windows 10
- Python 3.7

❖ 데모 방법

- GUI 프로그램으로 데모



업무 분담



37

자료수집	<ul style="list-style-type: none"> ❖ 레이블 알고리즘 ❖ 트레이딩 전략 	<ul style="list-style-type: none"> ❖ 시각화 라이브러리 ❖ HTS 	<ul style="list-style-type: none"> ❖ 기술적 지표 ❖ 증권사 API
설 계	<ul style="list-style-type: none"> ❖ 시계열 데이터 분석 및 레이블링 모듈 	<ul style="list-style-type: none"> ❖ 프로그램 GUI 모듈 	<ul style="list-style-type: none"> ❖ 전략 개발 모듈
구 현	<ul style="list-style-type: none"> ❖ Data Labeling ❖ 종목 필터링 	<ul style="list-style-type: none"> ❖ GUI ❖ Chart Visualization 	<ul style="list-style-type: none"> ❖ 전략 개발 및 검증 ❖ 거래 자동화
테스트	<ul style="list-style-type: none"> ❖ 모듈 별 테스트 및 디버깅 ❖ 모듈 통합 테스트 및 디버깅 		

종합설계 수행일정

항목	추진사항	12월	1월	2월	3월	4월	5월	6월	7월	8월	9월	10월
계획 수립	- 전체 계획 수립	■										
요구사항 정의 및 분석	- 일고리증 트레이딩 자료 조사	■	■									
	- 시스템 트레이딩 톨 자료 조사	■	■									
시스템 설계	- stock data label 일고리증 설계		■	■								
	- 전략 syntax 설계		■	■								
	- 주문모듈 설계		■	■								
	- simulation 설계		■	■								
	- 시각화 설계		■	■								
프로토타입 구현	- stock data label 구현			■	■							
	- stock data gathering 구현			■	■							
	- 주문모듈 구현			■	■							
	- simulation 구현			■	■							
	- visualization 구현			■	■							
프로토타입 테스트	- 프로토타입의 결과 확인 및 오류 해결				■							
프로그램 구현	- GUI				■	■	■					
	- 시스템 통합					■	■	■				
프로그램 테스트	- 시스템 시험 및 검증						■	■	■			
최종마무리 작업	- 보고서 작성								■	■		
	- 시스템 문서화								■	■		
	- 발표 준비								■	■		
논문 작성	- 논문 작성								■	■	■	
	- 산업 기술대전 참가								■	■	■	

[표 3] 개발 일정 및 추진 사항



필요기술 및 참고 문헌

❖ Ta-lib

➤ <http://mrjbq7.github.io/ta-lib/funcs.html>

❖ YesStrock - Yeslanguage

➤ <https://www.yesstock.com/>

❖ 키움증권 - API

➤ <https://www3.kiwoom.com/nkw.templateFrameSet.do?m=m1408000000>

❖ 뉴지스탁 - 젠포트

➤ <https://genport.newsysstock.com/Main.aspx>

❖ PySide2

➤ <https://pypi.org/project/PySide2/>

