|  |
| --- |
|  |
| **Mock Coiner** |
|  |
| 지도교수 이길흥 교수님 |

****

14109315 김 영 환

14109367 장 종 진

목차

[1. 프로젝트 구성 명단 4](#_Toc25753646)

[1.1. 역할 4](#_Toc25753647)

[2. Mock Coiner 개요 및 목표 5](#_Toc25753648)

[3. 진행 방법 및 절차 7](#_Toc25753649)

[3.1. 진행방법 7](#_Toc25753650)

[3.2. 진행절차 7](#_Toc25753651)

[4. Project 설계 8](#_Toc25753652)

[4.1.1. 아키텍쳐 8](#_Toc25753653)

[4.1.2. 서버 구성 및 역할 8](#_Toc25753654)

[4.1.3. 클라우드 DB 구성 9](#_Toc25753655)

[4.1.4. DB 스키마 11](#_Toc25753656)

[4.1.5. 필요 환경 및 라이브러리 등 12](#_Toc25753657)

[5. Project 진행 내용 14](#_Toc25753658)

[5.1.1. 알파 모델 16](#_Toc25753659)

[5.1.2. 수학적 모델 (알고리즘 기본 설명) 16](#_Toc25753660)

[5.1.3. 수학적 모델 (분산투자) 18](#_Toc25753661)

[5.1.4. 가격예측 모델 19](#_Toc25753662)

[5.1.5. 평균회귀 예측모델 20](#_Toc25753663)

[5.1.6. 머신러닝 모델 종합 21](#_Toc25753664)

[5.1.7. 알파 모델의 데이터 처리 방식 22](#_Toc25753665)

[5.1.8. 포트폴리오 모델 23](#_Toc25753666)

[5.1.9. PyQt를 이용한 클라이언트 구성 23](#_Toc25753667)

[6. 모델 수행 결과 24](#_Toc25753668)

[6.1. 수학적 모델의 실행 결과 24](#_Toc25753669)

[6.3. 평균회귀 모델 학습결과 25](#_Toc25753670)

[6.4. 클라이언트 UI 26](#_Toc25753671)

[6.5. 정리 및 논의 30](#_Toc25753672)

[8. 설계요소 평가 31](#_Toc25753673)

[9. 추후 발전방향 32](#_Toc25753674)

[10. 종합 토의 33](#_Toc25753675)

[11. 일정 및 첨부 34](#_Toc25753676)

# 프로젝트 구성 명단

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 성명 | 소속 | 학번 | 연락처 | 이메일 |
| 김영환 | 컴퓨터공학과 | 14109315 | 01028017081 | [zkvp201@naver.com](mailto:zkvp201@naver.com) |
| 장종진 | 컴퓨터공학과 | 14109367 | 01055287018 | [triplejjj03@gmail.com](mailto:triplejjj03@gmail.com) |

## 역할

김영환 : 시스템 트레이딩 모델 개발 , DB 클라우드 서버 환경 설정 , GUI 메인 및 다이얼로그

장종진 : 머신러닝 모델 개발 , GUI 다이얼로그

# Mock Coiner 개요 및 목표

Mock Coiner는 모의 가상화폐 시스템 트레이딩 서비스로서 향후의 가격 변동성을 예측하여 자동 매수, 매도를 통해 수익률을 창출하고, 손실을 예방하는 서비스이다. 전자화폐는 최근에 주목 받기 시작한 분야로써, 이미 많이 존재하는 주식 관련 투자 서비스에 대비하여 관련 서비스가 많이 부족한 실정이며, 시스템 트레이딩은 컴퓨터 공학적 지식과 경제학적 지식을 기반으로 하므로 인력 또한 부족한 실정이다. 본 프로젝트는 이런 상황에서 투자 관련 지식이 부족한 사람들에게 최소한의 가격 예측 서비스를 제공하고, 안정적인 투자전략을 제공함으로써 초보 투자자들도 쉽게 시스템 트레이딩과 투자에 익숙해지는 것을 목표로 한다. 또한 합리적이고 논리적인 절차를 통해 수익률을 달성하는 수익률 모델을 만드는 것을 목표로 한다. 수익률과 안정성 두 가지 측면 중, 기본적으로 안정성을 목표로 하며, 최종 작품은 해당 프로젝트 중 조원이 판단하기에 가장 서비스하기 적절하다고 생각하는 하나의 모델을 제공하는 방향으로 한다. 또한 의사결정을 도와주는 하나의 도구로써 머신러닝을 접목해 금융 분야에서 머신러닝이 어디까지 활용 가능할지, 어떻게 접목 가능할지, 그 가능성을 테스트해보도록 한다.

* 시스템 트레이딩

컴퓨터 프로그램을 이용해 주식 등을 매매, 각종 기술적 지표를 토대로 수익 창출이 가능한 신호를 만들어 자동화하는 것

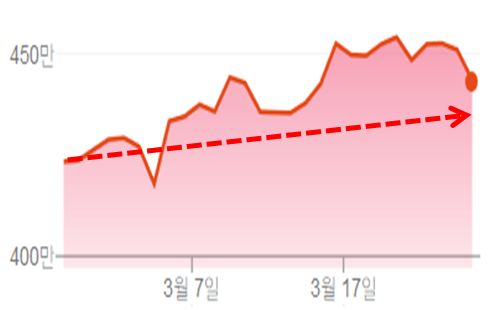
* 전자화폐 트레이딩 시스템

전자화폐의 가격을 예측하고, 가격 변화를 신호로 받아들여 학습시킨 후에 손해를 보지 않는 방향으로 매도, 매수를 해주는 자동화 시스템

캡스톤 디자인(2)에서는 기존에 개발 중이던 수익률 모델에 더하여 단일 코인이 아니라, 분산 투자상황에서의 여러 코인 기반으로의 모델로 확장을 해보도록 한다. 분산 투자는 위험도를 낮출 수 있는 가장 중요하고 보편적인 방법이며, 투자의 경우의 수를 늘려 백 테스팅 측면에서 여러 상황을 시뮬레이션하기에 적합하기 때문이다. 나아가 머신러닝을 활용한 서비스를 분산투자 상황에서도 한 가지 기획하여 그 가능성을 테스트해 보도록 한다. 적중률이나 효율성은 뒤로 하고 머신러닝이 서비스에 어떻게 접목가능하고, 활용 가능한지 사용 방법에 대한 논의와 그 결과를 눈에 확인하는 방법으로, 머신러닝을 활용하는 하나의 경험으로써 하나의 모델을 만들어 보도록 한다. 이후 PyQt5를 이용하여 GUI 환경 기반에서의 실행 파일로써 기본적인 서비스를 제공하고 만든 모델들을 테스트해볼 수 있는 디자인을 제작하도록 한다.

요약하면 다음과 같다.

1. 전자 화폐의 경우 변동성이 높고 가격 Risk가 매우 큼.
2. 향후 벌어질 가격 변동성을 예측, 매수와 매도 시점을 결정하여 손실을 예방하는 서비스
3. 나아가 수익성까지 보장하는 알고리즘의 개발
4. PyQt를 이용한 GUI 디자인



# 진행 방법 및 절차

## 진행방법

현재 2인으로 구성된 Mock Coiner 팀은 모델 별로 각자의 역할이 분담되어 있다. 역할 분담은 다음과 같다.

|  |  |
| --- | --- |
| **항목** | **담당** |
| 수익률 구성 모델 알고리즘 구현 및 클라우드 서버 관리 | 김영환 |
| 딥러닝 머신러닝 모델 구현 | 장종진 |
| PyQt5 GUI 개발 | 공통 |

각각의 역할분담이 확실하며, 산출물이 GUI인 특성으로 인해, 애자일 모델을 이용하여 점진적인 개발 후 GUI 개발에는 각자의 화면을 구성하는 다이얼로그 개발 형식으로 진행한다. 각자의 역할을 준수하고 하나의 버전을 만든 후에 다른 기능을 추가하면서 버전업을 하는 방식으로 진행한다.

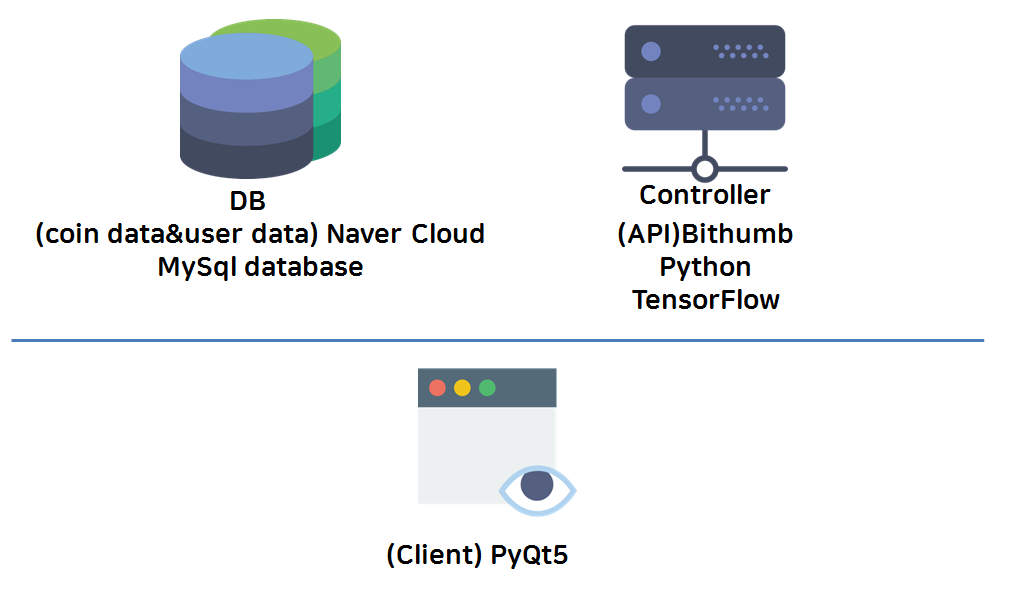
## 진행절차

3.1.에서 언급한 “버전”을 만드는 작업은 다음의 과정을 포함한다.

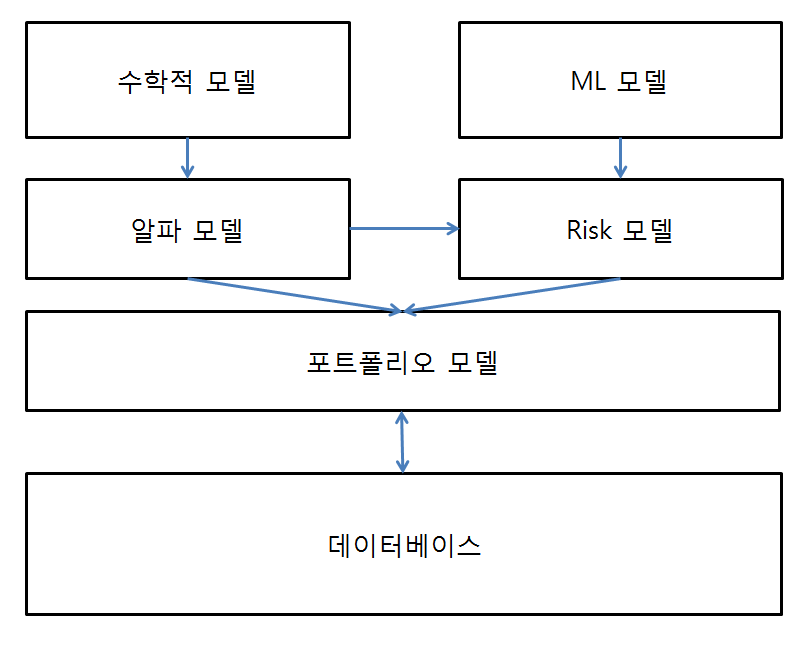
하나의 사이클을 통해 하나의 버전을 만들고 나서, 다시 한 사이클을 통해 버전업을 하는 방식이다.

# Project 설계

### 아키텍쳐



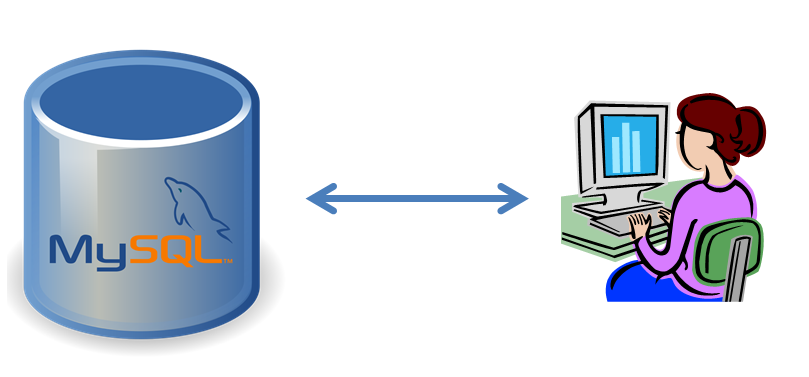
기존 MVC 디자인 패턴에서 변경된, 클라이언트-서버 패턴을 이용하여 시스템을 구성하였다. 해당 아키텍처는 위와 같은 그림으로 구성되며, Server는 클라우드 DB 서버를 사용한다. API는 Client가 요청에 따라 호출되게 된다. (pybithumb을 이용한 bithumb API 호출 방식) 클라이언트는 PyQt5로 제작된 GUI 기반 exe 파일을 바탕으로 통신을 하게 된다.



그 외에 알고리즘을 개발하는 일련의 논리적 아키텍처는 위와 같다. 자세한 설명은 뒤에서 다루도록 한다.

### 서버 구성 및 역할

서버는 회원들의 정보를 저장하고, 코인 명세를 저장해줄 수 있는 DB 서버를 운영한다. 개별적으로 제공되는 GUI 기반의 exe 파일을 통해서 해당 서버와 통신할 수 있으며, 서버는 유저들의 거래 결과 만을 저장하는 역할을 담당하게 된다. 서버가 담당하는 부담은 크지 않으며, 대부분 로컬에서 진행되기 때문에 개인 사용자 컴퓨터의 성능이 요구되는 경향이 있다. 머신러닝같이 많은 자원을 요구하는 기능들이 포함되어 있기 때문에, 서버에서 모든 일을 동작하기에는 비용이나 기능적 한계가 있음으로, 더욱 개인화된 로컬환경의 중요도를 높였다.

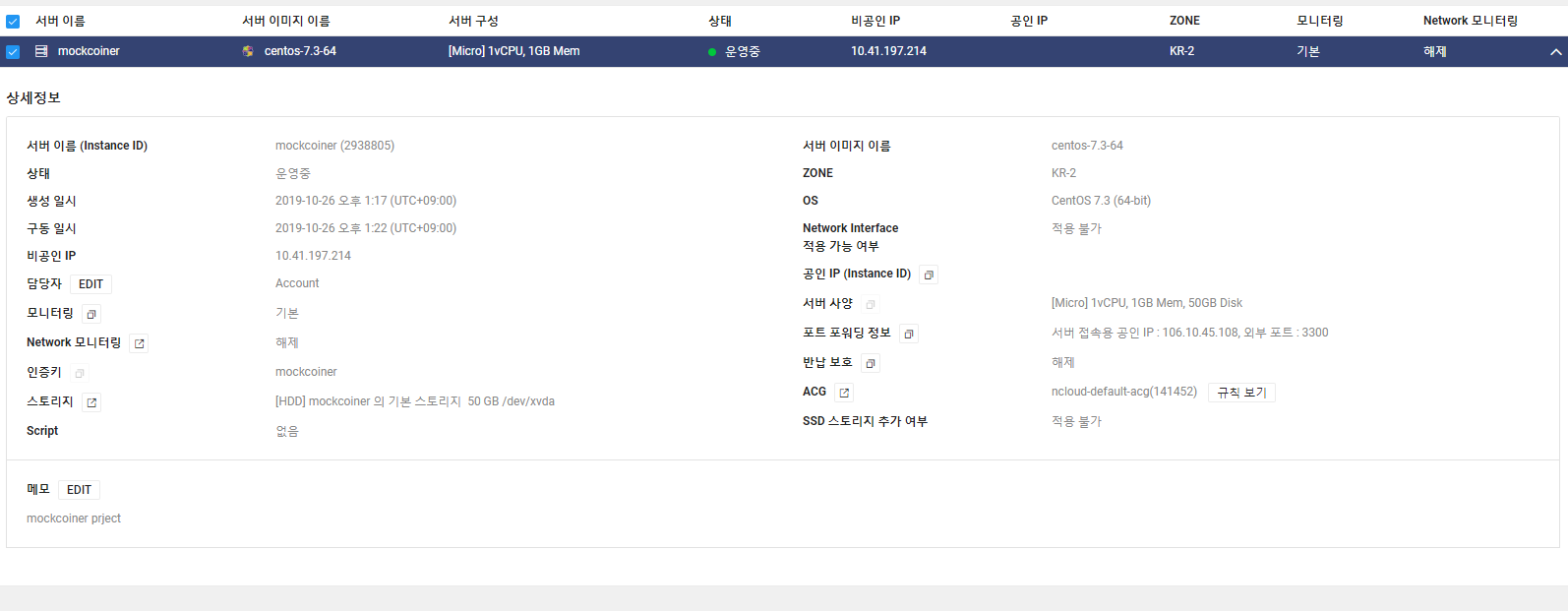


### 클라우드 DB 구성

서버를 24시간 돌리기에는 환경적인 제약이 있기 때문에, 클라우드 서버를 이용하여 DB 서버를 구성하였다. 서버는 네이버 클라우드 플랫폼을 이용한다.

* **네이버 클라우드 플랫폼의 장점**

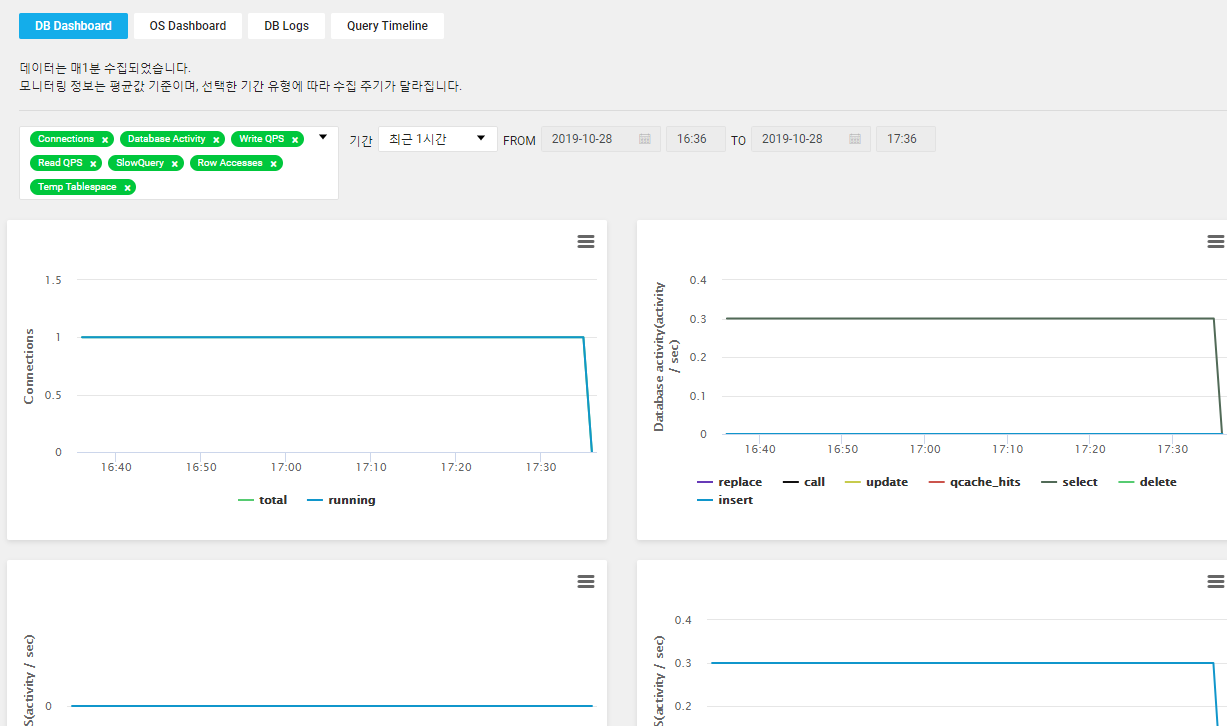
1. 피드백이 빠르다. (문의 시 네이버 측에서 개인 사용자에게 전화 연락 )
2. 한국어로 되어있어 초보자도 누구나 쉽게 이용 가능하다.
3. 기본 크레딧 10만 크레딧이 제공되어 학교 프로젝트에 용이하다.
4. 설명이 잘 되어 있고, UI가 깔끔하다.
5. MySql Workbench와의 연동이 가능하다.
6. IP가 아닌 도메인 주소로 접근함으로 보안에 용이하다.

<데이터 베이스에 접근할 수 있는 네이버 클라우드 서버>



<클라우드 DB 서버>

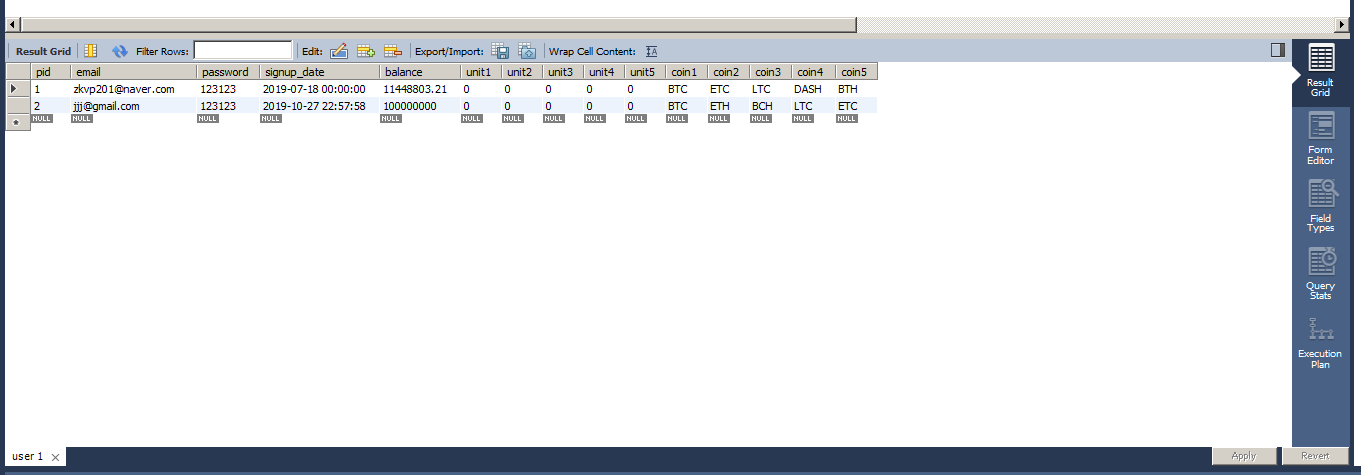
클라우드 DB 서버 환경 (Public 도메인은 보안상 가림.) 네이버 클라우드 DB 서버의 경우 도메인 주소로만 접근이 가능한데, Private 도메인의 경우 네이버 클라우드 내부 서버를 통해서만 접근이 가능하기 때문에, 내부 접근용 클라우드 서버도 하나 만들어야 된다.



<고퀄리티 UI 제공으로 관리에도 용이함을 알 수 있다>

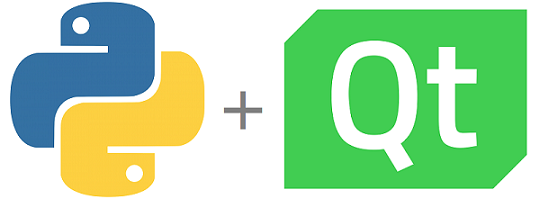
### DB 스키마

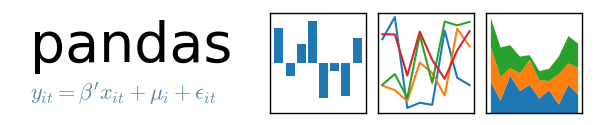
Database의 경우에는 Mysql을 활용하도록 한다. 복잡한 기능을 수행한다기 보단 데이터를 저장하고 갱신하는 가벼운 역할만 하기 때문에 구조가 간단하게 구성된다.

(이외에도 필요 시 스키마를 추가할 예정.) 총 다섯 개의 코인으로 분산투자를 실시할 것이기 때문에 5개의 코인 이름과 그 코인의 보유량을 저장한다. 그 외의 가입일자, 이름, 비밀번호 등 회원을 구분하기 위한 가장 기본적인 요소만을 데이터베이스에 저장한다.

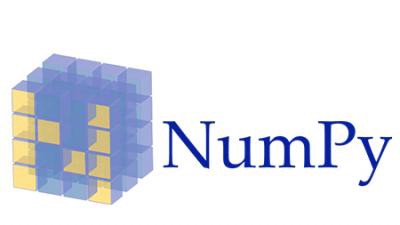
<Workbench 로 관리가 가능한 네이버 클라우드 MySql>

### 필요 환경 및 라이브러리 등







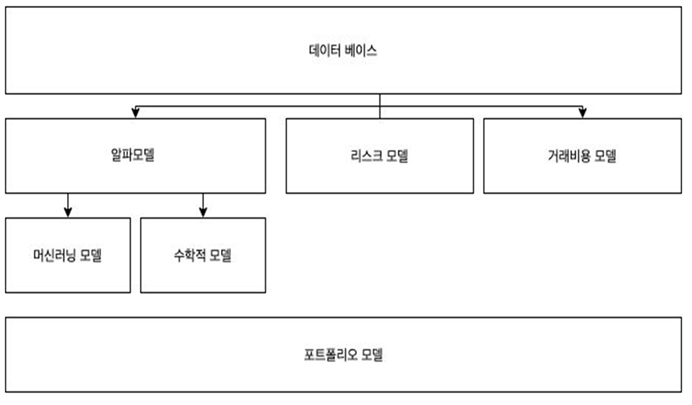


GUI 표현에는 PyQt5를, 데이터 수집을 위한 파서로는 pybithumb 및 bithumb를, 데이터 정제를 위해선 Pandas와 NumPy의 라이브러리들을 사용 하였으며, 클라우드 DB로는 네이버 클라우드 플랫폼을 사용하였다. 머신러닝에는 TensorFlow를 이용하였다.

# Project 진행 내용

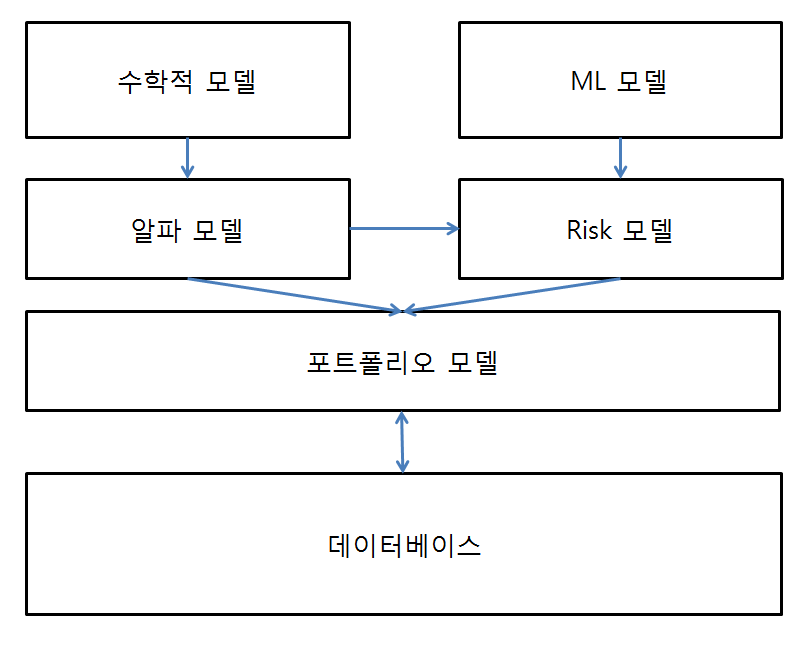
해당 장에서는 MockCoiner Project의 전체적인 구조 및 구현 방법 및 방향성에 및 진행 절차에 따른 과정에 대하여 설명하도록 한다.

MockCoiner Project는 프로그램의 전체적인 구조를 설계하고 모델별 역할 군을 정의하였으며 초기 프로그램의 설계도는 다음과 같다.



<변경 전 프로그램 구조도>

기존 방식은 데이터베이스를 기반으로 여러 모델을 구성하고 그 모델을 총괄하는 포트폴리오 모델의 방식으로 구성하려고 하였으나, 진행 상황과 필요성에 의해 조금씩 역할이 변동되어 보다 간단한 모습의 구조로 변경되었다.



<변경 후 프로그램 구조도>

변경된 모델은 수익률을 계산하는 알파 모델의 역할이 수학적 모델만을 이용하는 것으로 변경되었다. 이후, ML모델의 경우 가격 예측 및 투자 비율 백테스팅 등을 통해 리스크를 검토하고 유저들에게 의사결정 과정에서 도움을 줄 수 있는 리스크 모델의 일부 역할로써 편입되었다. 전체적인 통신과정은 데이터 베이스와 포트폴리오 모델이 정보를 저장,가져오기를 반복하면서 데이터를 갱신하게 된다.

### 알파 모델

실제적인 투자 전략 및 가격 예측 서비스 등 수익률과 안전성을 확보하기 위한 기능을 하는 모델이다. 해당 프로젝트에서는 투자 전략을 활용하여 매수, 매도 기준을 판단하는 수학적 모델을 활용하여 알파 모델을 구성하였다. 투자자에게 일종의 가이드라인을 제시해주는 로보 어드바이저의 성격을 지녔다고 할 수 있다.

### 수학적 모델 (알고리즘 기본 설명)

변동성 돌파 전략을 기반으로 한 전자화폐 투자 알고리즘 개발을 목적으로 한다. 변동성 돌파 알고리즘은 기존에 주식 투자에서도 높은 수익률을 이뤄낸 래리 윌리엄스의 유명한 투자 전략이지만, 변동성이 큰 전자 화폐에서 더욱더 좋은 결과를 보일 것으로 예상하였기 때문에 현 프로젝트의 base 전략으로 사용하기로 하였다. 이 수학적 모델의 가장 중심이 되는 아이디어는 추세 추종 전략이다. 즉, 가격이 오르는 경향이라면 미래에도 계속 오를 것이라는 가정하에서 세워지는 전략이다. 일반적인 변동성 돌파 알고리즘을 간략화하면 다음과 같다.

**목표 가격 = 당일 시가 + 전날 변동 폭 \* 돌파계수(0.5)**

여기서 0.5라는 값은 경험을 통해 얻은, 가장 변동성 돌파를 잘 실현할 수 있다고 생각되는 평균적인 돌파 계수이다.

알고리즘의 실행 과정은 다음과 같다. 당일 가격에 변동성을 반영하였을 때, 가격이 목표를 달성하면 그 코인은 현재 추세를 따르는 상태라고 판단, 향후 더 오를 것으로 예측하게 되는 것이다. 따라서 목표가격에 매수하고, 매도는 **당일 종가**에 매도를 시켜 **하루 동안** **매수와 매도를 모두 처리**한다. 이러한 **단기적인 거래 패턴을 모아 장기적인 추세를 형성하여 수익률을 달성**하는 알고리즘이다.

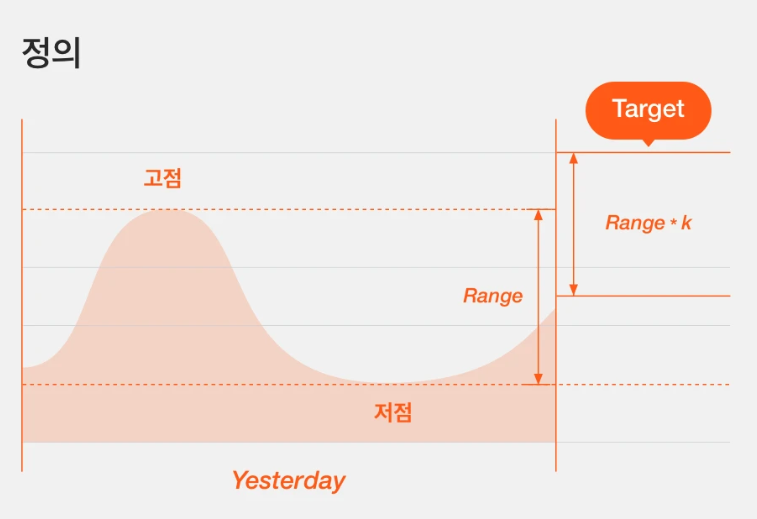
해당 프로젝트는 변동성 돌파 전략을 기반으로 기존에 널리 알려져 있던 투자 전략 등을 결합하여 프로그래밍 환경에서 알고리즘화하여 자동으로 이뤄지게 하여 사람이 지니는 시간, 공간적 한계 (ex. 24시간 모니터링) 등을 뛰어넘어 수익률을 실현한다. 여기서 MockCoiner의 수익률(수학적) 알고리즘의 개발 방향은 수익률보다는 **안전성을 우선**시하여 **보수적인 투자**를 시행하는 모델을 목표로 하며, 나아가 수익률까지 실현하는 것을 목표로 한다.

현재까지 개발된 수익률 모델은 기존의 변동성 돌파 전략만을 기반으로 하였을 때 보다 손실률이 적고, 그래프의 모양이 안정적으로 변하면서 해당 전략이 사용할 수 있는 전략임을 확인하였다. 그 방법에 대하여 세부적으로 보자면, 평균이동선 모델을 결합하여 위험 신호를 필터링하고, 가격의 추세 기간에 따른 투자 비율을 유동성 있게 조절하여 손실을 최소화하는 방향으로 알고리즘을 구성하였다. 평균이동선이란, 특정기간의 가격을 평균하여 나타낸 추세선이다.

이 외에 변동성 돌파 계수를 고정 계수가 아닌 데이터를 통하여 실시간으로 계산되는 지표로 대체하여 overfitting을 방지하고, 다양한 상황에 적응할 수 있는 계수로 활용하였다. 해당 지표는 장기 Noise 비율이라는 개념으로, 전자화폐 가격의 긴 기간 동안 고점-저점 대비 시가-종가의 비를 계산하여 해당 코인이 추세를 거스를 확률이 어느 정도 되는지를 측정하는 지표이다. 그 결과 시장이 불안정한 상황에서, 약간의 수익률 손해를 감수하면, 손실예상액은 많은 감소 효과를 보임을 확인하였다. 또한 노이즈 비율이라는 개념을 접목해 불필요한 신호를 걸러낸 결과 더 나은 수익률과 손실방지율을 보임으로써, Noise 비율이라는 개념이 실제로 사용 가능한 결과임을 확인하는 데 성공했다.

하지만 단일 종목의 경우 단기 Noise 비율을 적용해야 하는데, 이 단기 Noise 비율 모델을 적용하는 것에는 상황적인 한계와 값을 계산하는 것의 문제가 있기 때문에 실제 투자에는 아직 더 많은 연구가 필요하다고 판단했다. 하지만, 장기적인 Noise 비율 지표의 경우는 분산투자 환경에서 접목 시에 더 좋은 결과를 보일 것임을 기대할 수 있었다.

이처럼 다양한 전략과 모델을 결합한 결과 기존에 일반적인 변동성 돌파 전략보다 더 안정적이며 적응력 있는 모델로 개선했음을 볼 수 있다.



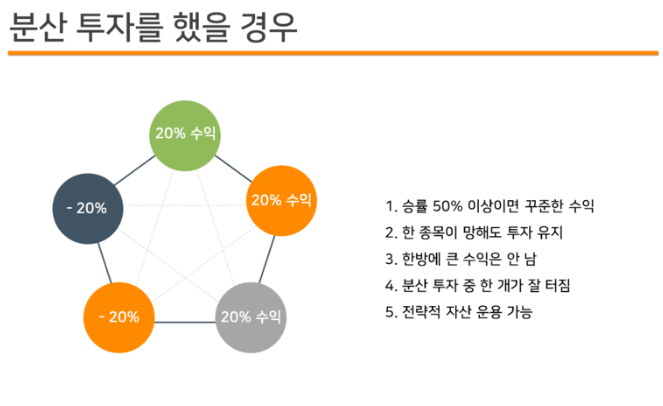
<변동성 돌파 전략>

### 수학적 모델 (분산투자)

분산 투자 모델은 이미 위에서 개발된 단일 코인들의 집합을 이용하여 분산 투자를 시행하는 것이다. MockCoiner 프로젝트는 안전성을 우선으로 하는 프로젝트이기 때문에, 분산 투자를 시행하는 것은 아주 당연한 절차이다. 분산 투자를 할 경우 MDD 즉, 최대 손실 가능성을 획기적으로 낮출 수가 있다. 어느 한 날의 코인 다섯 개의 수익 현황이 다음과 같다고 하자.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| BTC +10% | LTC +10% | DASH -50% | ETC +30% | ETH -5% | Total -5% |

다음과 같은 상황이 발생했을 경우, DASH 코인에 단일 투자를 한 투자자는 손실률이 50%라고 기록될 것이다. 하지만 분산투자를 했을 경우에는 해당 코인에서 손해를 보더라도 다른 코인에서 이득을 보게 되면 손해를 메꿀 수 있다는 것이다. 물론 ETC 코인에다 투자를 한 사람보다 분산 투자를 한 사람이 수익률은 낮게 책정되겠지만, 실제 거래에서 어떤 코인이 수익을 낼지는 미지수이기 때문에, 분산 투자를 하여 위험성을 낮추는 것이 안전성을 목표로 하는 MockCoiner와는 아주 잘 맞아떨어지는 전략이 되는 것이다. MockCoiner는 여기에 더하여, 단순히 5개의 코인이 있다면 5개의 코인에 모두 분산 투자 하는 것이 아니라, 필터링 조건을 적용하여 3개의 코인만을 선정하여 투자하는 알고리즘을 구현하였다. 이 경우, 분산투자 효과에 의한 위험도는 5개 종목에 분산하는 경우보다 높아질 가능성은 있지만, 약간의 위험성을 감수하고 그 이상의 수익률을 보장해주는 알고리즘으로써 안전성뿐만 아니라 수익률도 확보하는 모델임을 확인하였다.



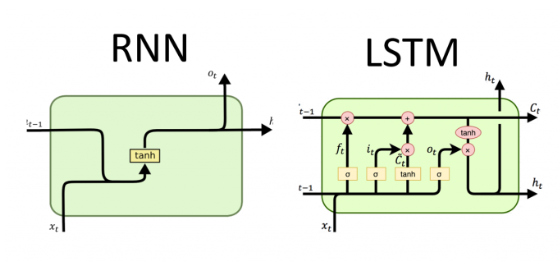
< 왜 분산 투자를 해야 할까? >

### 가격예측 모델

가격예측 모델은 딥러닝 모델 중 하나인 LSTM 모델을 사용하여 전자화폐 데이터를 학습시켜 다음날의 종가를 예측하는 모델이다. 시계열 데이터와 같이 연속적인 데이터는 RNN(Recurrent Neural Network, 순환신경망)이 효과적인데, RNN의 경우 장기 의존성(long-term dependencies) 문제가 발생하므로 이를 해결하기 위해 LSTM(Long Short Term Memory, 장단기 기억 메모리) 모델을 활용했다. 장기의존성이란 정보의 차이(gap)가 벌어질수록 이전 정보를 기억하지 못하는 것을 의미한다.

이를 해결하기 위해서 장기기억과 단기기억을 나누는 LSTMs(Long Short Term Memory networks)가 필요하다. 데이터를 얻는 방식은 초기에는 정적으로 로컬 환경에 존재하는 데이터를 받아와 학습을 시키는 방향에서 라이브러리를 사용하여 매번 새로운 코인 데이터를 받아오고 학습하여 다음 날의 종가를 출력해주는 동적 로딩 방식으로 발전시켰다.

또한, 초기에는 데이터 세트의 양이나 큰 변동을 겪지 못한 데이터 세트의 구성으로 테스트 시 제대로 학습되지 못한 결과를 보였으나, 데이터 세트를 적절히 재구성한 후에는 학습 과정 자체는 잘 이루어지는 모델임을 확인할 수 있었다.



<RNN과 LSTM의 도식적인 구조>

### 평균회귀 예측모델

평균 회귀 예측모델은 평균 회귀 방법을 사용하여 가격을 예측하는 모델이다.

여기서 의미하는 포트폴리오란 여러 가지 자산의 모음을 말한다. 포트폴리오에는 5가지 가상 화폐를 현금과 함께 설정했다. 리밸런싱 방법은 전날 7일 평균과 전날 종가 데이터를 사용했는데, (전날 종가 데이터 7일 평균 / 전날 종가)를 조정 비율로 설정했다. 조정비율이 1보다 낮으면 시장이 과열되었다고 판단하여 매도하고, 조정 비율이 1보다 높으면 시장이 침체라고 판단하여 그 반대로 종목을 매수하는 방법을 사용했다. 또한, 조정이 너무 과다하게 이뤄지면 어느 한 비율이 0이 되어 더는 리밸런싱 할 수 없는 문제가 생기는데 이를 해결 하기 위해 약간의 변수를 추가하여 해결했다. 이러한 리밸런싱 방법을 바탕으로 포트폴리오의 비율을 수정해 나갔으며 이를 리밸런싱 로그에 현금, 현재 자산, 현재 자산에서 전날 자산을 뺀 수익, 그리고 수익이 났을 경우 1, 아닐 경우 0으로 분류한 열과 함께 기록하여 데이터 세트를 만들었다.

이후에 머신러닝으로 학습을 시키고 예측을 하는 과정은 일반적인 로지스틱 회귀 분류의 방법으로써 널리 알려진 절차를 따라 학습을 진행했다. 간단하게, 수익이 난 경우 1 , 나지 않은 경우 0 로 기록한 열을 Y 함수로 두고, X 함수는 그 외의 현금 및 자산들을 넣어 해당 자산의 분배 비율이 수익이 발생하는 경우와 어떤 관계가 있는지 분류해 내는 과정을 의미한다. 그리고 그러한 분류의 과정에서 가설 함수가 얼마나 높은 확률로 수익률을 구분해 내는지를 Accuracy 개념으로 구분해 내어 해당 비율의 신뢰도를 구할 수가 있다. 이는 이미 당뇨병 환자의 몸 수치 등을 활용하여 당뇨라면 1, 당뇨가 아니라면 0의 방식으로 구분해 내는 등의 의학계에서도 흔히 활용되고 있는 구분법이며, 이를 응용하면 금융 데이터의 특정 코인 비율이 수익률과 어떤 관계를 맺고 얼마만큼의 신뢰도를 보일 수 있는지의 데이터로 해석이 가능할 것으로 판단하였다. 물론, 기본적으로 전자 화폐 및 주식 시장은 랜덤 워크이므로 평균회귀 성을 따르는 코인이 있을 수도 있고, 그렇지 않은 코인이 있을 수도 있음에 따라 모델의 성능 자체는 훌륭하게 나오지 않을 수 있다. 실제로 logistic classification을 수행해 본 결과 적중률 50~60% 정도로 겨우 절반을 넘은 실정이지만, 50%가 넘었다는 것은 기본적으로 수익이 날 확률이 더 높다는 것을 의미하기 때문에 아주 의미가 없는 데이터라고 판단할 수 있었다. 또한 이러한 아이디어를 가지고 금융 시장에 머신러닝 기술을 도입해 보는 그 과정 자체에 의미를 두고자 하였기 때문에 결과적인 수치를 넘어 경험적인 측면에 의의가 있다고 볼 수 있다. 실제로 금융 시장에서도 이러한 문제 때문에 아직 머신러닝의 적극적인 도입을 주저하고 있는데, 이러한 모델들을 더욱더 발전시킨다면 실제 산업에도 활용할 수 있다고 생각한다.

### 머신러닝 모델 종합

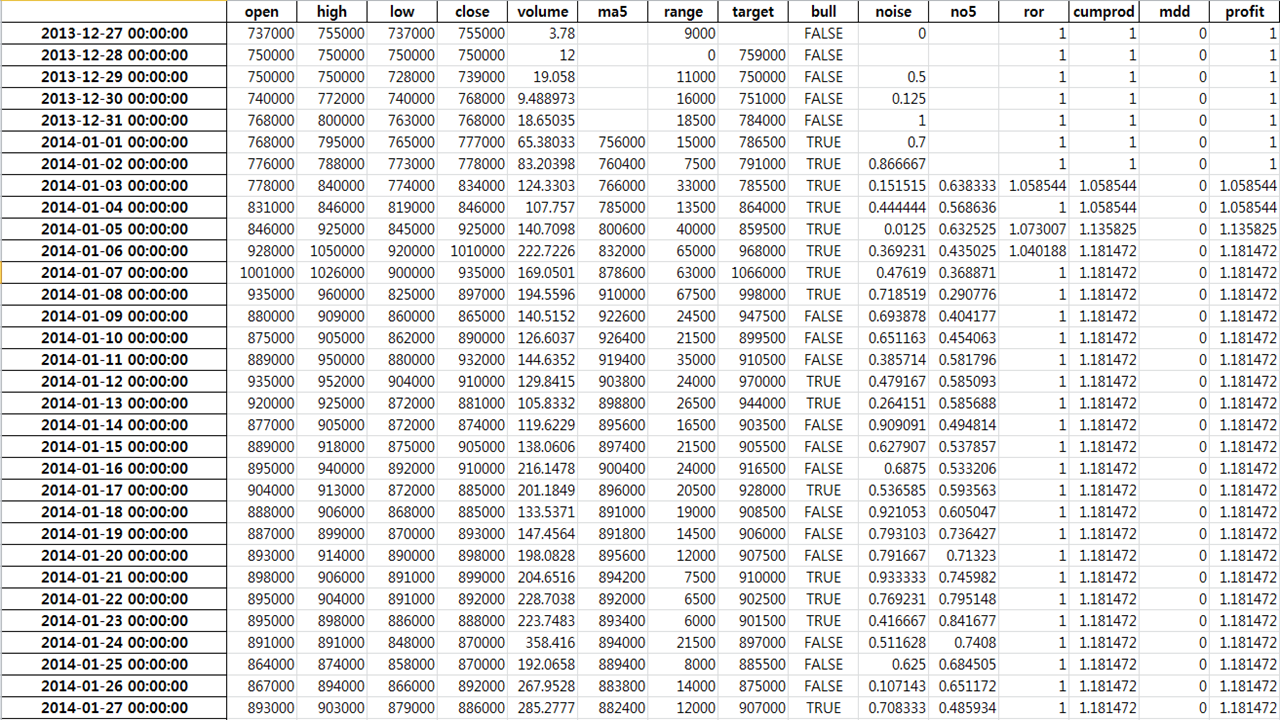
머신러닝 모델은 사실, 진행하면서 기본적으로 전자 화폐는 미래를 알 수 없는 랜덤워크 경향성이 강한 데이터기 때문에, 머신러닝으로 크게 의미 있는 데이터를 얻기가 어려울 것으로 예측하였다. 따라서, 결과를 가지고 피드백을 할 수 없기 때문에 정확한 이론적 지식 및 조원 간의 토의가 중요하게 여겨졌던 분야이다. 머신러닝 관련 스터디를 통해 머신러닝을 어떤 식으로 우리 프로젝트의 접목할 수 있고 접근할 수 있으며 구현할 수 있는지 그 방향성과 방법론을 논의하는 것만으로도 의미가 있다고 생각하였기 때문에 프로젝트에서도 사실 서비스 측면 보다는, 자기 계발적인 측면이 강한 파트라고 볼 수 있다. 또한, 데이터 세트를 구성하고, 처리하는 그 과정 자체는 수학적 모델을 구성하는 방법과도 일맥상통 하였기 때문에 프로젝트의 방향성과도 크게 엇나가지 않았다고 볼 수 있다.

딥러닝으로 대표 되는 가격 예측 기능과 머신러닝으로 대표되는 평균회귀 예측 모델은 현재 실전에서 사용하기에는 더 다듬어야 할 부분이 많지만, 이후의 부분은 전문가들의 영역으로써 학부생 수준으로는 여기까지가 한계라고 판단하였다.

머신러닝 관련 프로젝트를 진행하면서 느낀 점은, 머신러닝을 돌리는 과정 자체는 결국, 이미 구현되어 있는 라이브러리를 사용하는 것에 지나지 않는다는 점이다. 그것보다 중요한 것은, 데이터 세트를 어떻게 나누고, 구성하고 처리하는지, 어떤 아웃풋을 목적으로 할건지 등의 따라 신뢰도 및 학습률이 기하급수적으로 변한다는 것이다.

### 알파 모델의 데이터 처리 방식

Pandas의 DataFrame 구조를 이용한다. 해당 프로젝트에서 다뤄야 하는 데이터는 시계열 데이터로써 매일 지정된 시간에 주식의 시가, 종가 등을 받아 온다. 데이터의 양은 많지만, 항상 같은 Form의 데이터를 가지고 오고 유사한 동작을 하여 데이터를 가공하기 때문에 CSV나 Excel과 같은 형식으로 구성되어 행간 연산이나 열간 연산에 유리해야 한다. Pandas의 DataFrame은 이러한 조건을 충분히 만족시키며, 강력한 데이터 속성(행,열) 간의 연산 기능을 제공한다.



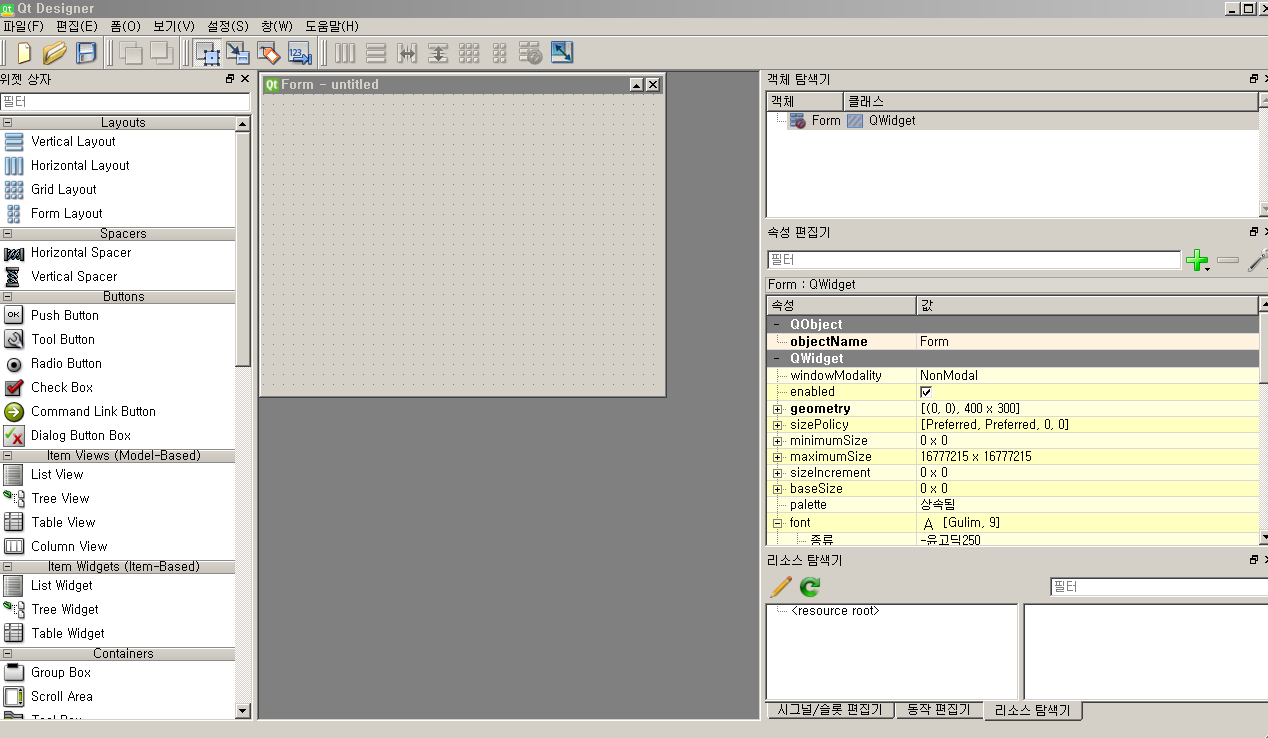
다음은 수익률 모델에서 백 테스팅에 사용하는 데이터 저장의 일부이다. 데이터의 행간 및 열간 연산을 수없이 반복하여 결괏값을 누적시키는 방법으로 수익률을 실현하고 있다. 시계열 데이터의 경우, 하나의 행에 있는 요소만으로 데이터가 의미를 찾기보다는 여러 값을 모아놓고 누적된 값의 분석이나, 데이터의 경향성을 찾아내는 등 각 데이터 간의 추가적인 관계를 탐지하는 것이 중요하다. 그런 점에 있어서 DataFrame 구조는 시각적으로 확인하기가 용이하며, 행, 열간 연산이 용이하기 때문에 다음과 같은 DataFrame 구조가 가장 적합한 것으로 판단이 되었다. 또한 웹 서비스로 제공할 때의 경우를 상정하면, DataFrame 형식은 쉽게 Json으로 변환이 가능하기 때문에, 웹 서버와의 통신에서 input과 output의 형식을 맞추기에도 용이한 측면이 있다. 이처럼 DataFrame 구조의 확장성은 프로젝트를 확장 시키는 경우에도 용이할 것으로 예상 되었다.

### 포트폴리오 모델

실제 거래 시에 알파 모델과 머신러닝 모델 등을 이용하여 단일 코인에서 다중 코인으로의 확장성을 꾀한다. 그와 관련된 정보를 DB와 연동하여 저장하거나 불러오는 역할을 한다. 분산투자의 경우, 종목마다 수익률이나 거래 비용 등을 변화를 감지하여 통합하는 과정이 필요한데, 이러한 일련의 과정을 체계적으로 관리하기 위해 포트폴리오 모델이라는 개념을 사용한다. 실제적인 거래를 담당하고, 유저의 정보를 갱신시키기 때문에 일종의 Trader(거래자) 역할을 한다고 볼 수 있다. 포트폴리오 모델을 구성하는 요소 중 하나인 리스크 모델의 경우 분산 투자 환경에서 각 종목 투자 별 노이즈에 따라서 효과적인 종목을 선별해내는 역할로써 필터링 기능을 담당하고 있다. 그 외에 ML을 이용하여 구성한 모델들로 여러 변수를 테스트해 봄으로써 유저들의 의사결정 과정을 돕는다. 해당 모델들의 구분은 개발의 용이함과 역할 군을 명확히 하기 위한 하나의 논리적인 구조이며, 실제 구현 시에는 통합 과정에 의해 통합되어 있거나, 구현상 뚜렷한 경계가 존재하지 않을 수 있다.

### PyQt를 이용한 클라이언트 구성

만들어진 투자 모델 및 가격 예측 모델들을 표현할 수 있는 수단으로 PyQt5를 활용한 exe 실행 파일 환경을 선택하였다. 기존에 목표로 하였던 웹 기반 오픈 환경이 아닌, 로컬에서 제공하는 방식을 채택함으로써, 모델의 업데이트 및 수정 배포 등에 대한 불리함을 안고는 있지만, 컴퓨터 환경에서 사용자의 성능에 따라 웹 서버에서 제공하는 방식보다 더 원활하게 Tensorflow 같은 경우 잘 동작할 수 있고, 보안 유지에도 용이하다. 또한 실행 프로그램 형식으로 간단하게 제공됨으로 사용하기가 쉬운 경향이 있다. PyQt는 다양한 Tool, 스레드 기능과 같은 강력한 프로세싱 및 라이브러리들과의 호환성을 제공하기 때문에 구현에 용이하며, 퀄리티 높은 GUI 기반 프로젝트를 진행할 수 있다는 장점이 있다. 또한 PyQt 뿐 아니라 Qt라는 이름 아래 다양한 언어를 지원하기도 하며, 해당 프로젝트의 경우 CSS를 이용하여 퀄리티 높은 GUI 기반의 스타일 시트 디자인을 꾸밀 수 있었다.



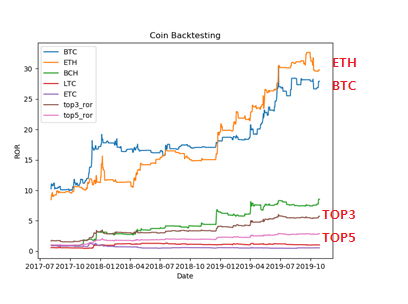
<pyqt의 UI구성 tool 인 qtDesigner >

# 모델 수행 결과

해당 이미지들은 구현된 각 기능별로 수학적 모델의 실행결과, 가격예측 모델의 실행 결과, 서비스 구현 상황 등 을 개별적으로 보여준다.

## 수학적 모델의 실행 결과

* 다중 코인



Top3 MDD: 18.494446223898393 Top5 MDD: 18.57620656239654

Top3의 MDD와 Top5의 MDD가 비슷한 수준임에도 필터링을 적용한 top3의 수익률이 더 높은 수준을 보임음을 그래프를 통해서 확인할 수 있다. 또한, 다른 개별 코인들의 지표보다 분산 투자한 코인들의 결과 지표가 일반적으로 더 굴곡(변동폭)이 적음을 눈으로 확인할 수 있다.(MDD(Max Draw Down) : 거래 중 발생한 최대 손실율)

추가로, BTC나 ETH의 경우는 얼핏 높은 수익률을 보이는 듯하지만, 시작점이 다르다는 것을 유의하자. 해당 그래프의 모양을 보이는 이유는 분산투자가 시행되기 이전부터 거래가 시작되어 수익률이 누적 돼 있는 상태이다. 즉, 거래 횟수가 많다. 따라서, 분산 투자의 그래프와 시작점을 같은 곳으로 놓으면 큰 차이가 나지 않는다. 또한 그래프의 모양이 매우 불안정하기 때문에 손해를 볼 가능성도 높다고 해석할 수 있다. 예를 들어 저 그래프 중 하락세에 있는 지점에 내가 구매를 하고 오르기 전에 심리적인 압박에 의해 팔 경우, 큰 손해를 볼 수 있는 구간들이 존재한다. 단일코인 투자의 경우, 운이 좋아 내가 오를 확률이 높은 코인에 투자했다고 하더라도, 그만큼 손해 볼 확률 또한 높기 때문에 MockCoiner가 추구하는 안전성 우선과는 거리가 먼 투자 방법이다.

## 6.3. 평균회귀 모델 학습결과

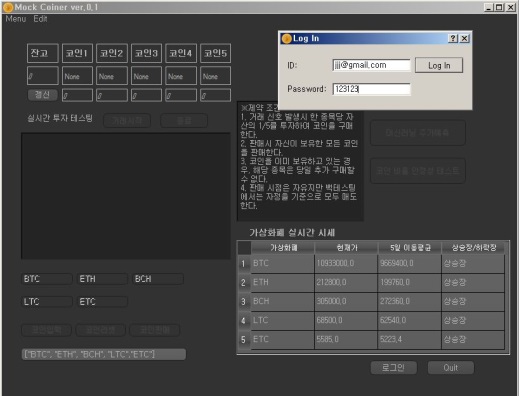
## 클라이언트 UI



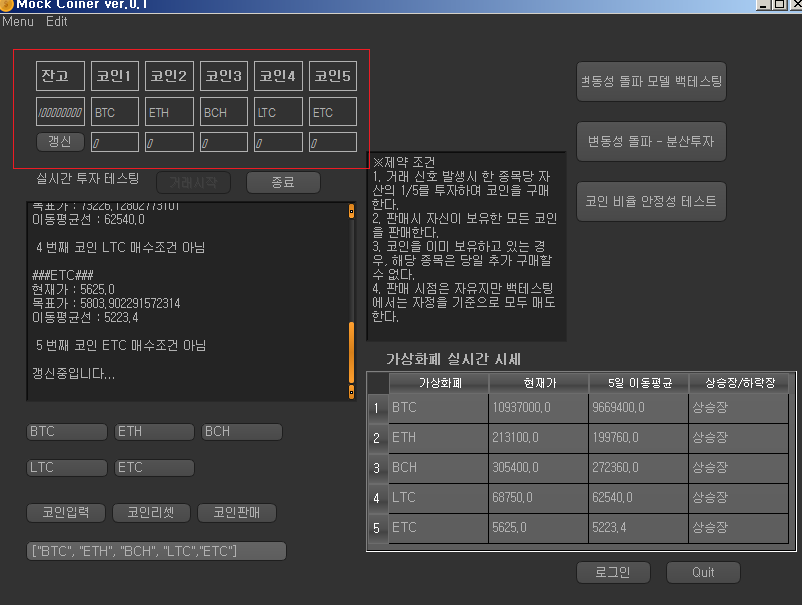
가상화폐 실시간 시세 부분은 실시간으로 계속 변화하는 창이며, 실시간 투자 테스팅 창의 경우도 거래 시작 버튼을 누를 시에 스레드로써 작업이 끝날 때마다 자동으로 갱신된다. 그 외에는 버튼이 누르면 동작하는 이벤트 처리 형식으로 구현이 되어 있다.



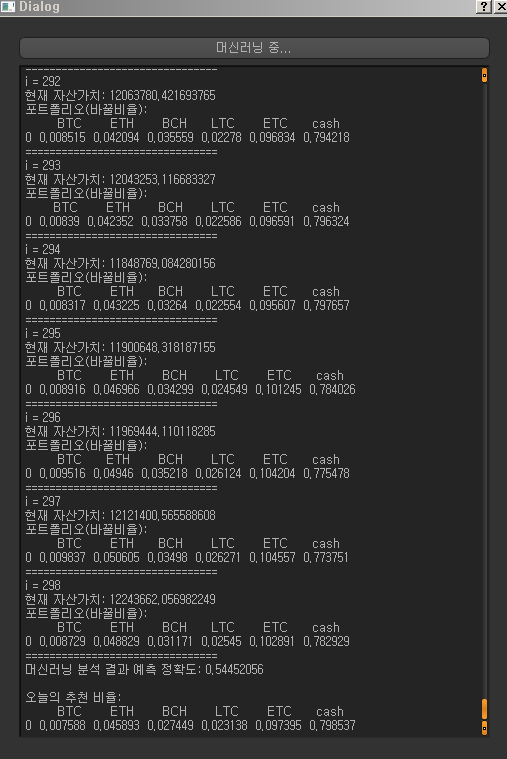
상단 Status Bar의Edit 부분은 아무것도 구현이 되어 있지 않으며, Menu 버튼을 누르면 회원 가입 및 Exit 버튼이 있다. Exit 버튼은 Ctrl+Q 단축키로 실행 가능하며, 메인화면의 Quit 버튼과 같은 기능을 한다. (종료)



<**좌** : 회원가입 창 **우**: 로그인 전 모든 버튼이 비활성화 되어있는 모습>



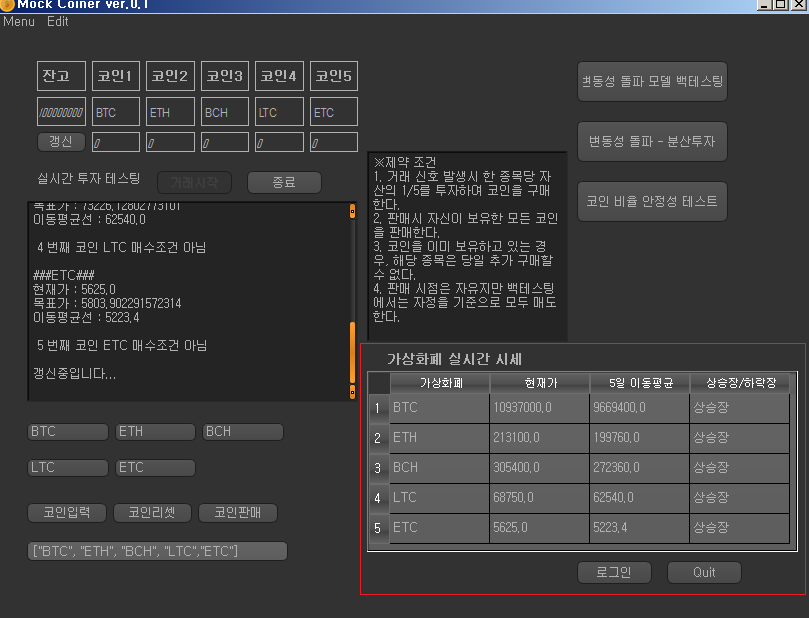
<갱신 버튼은 해당 부분을 갱신 시킨다>



<머신러닝 다이얼로그의 실행모습>



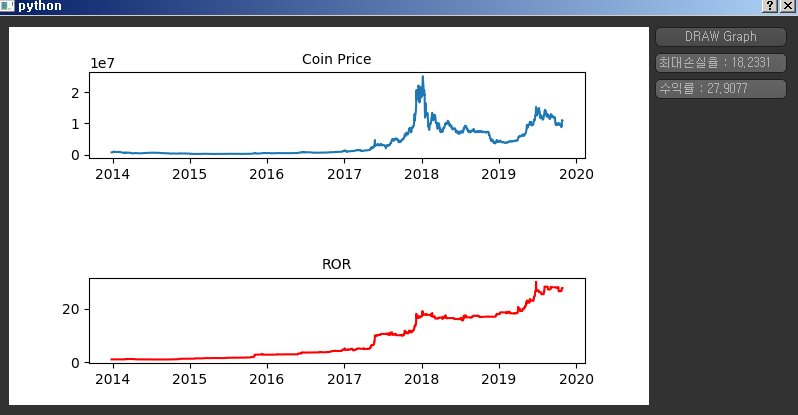
<해당 부분이 스레드로 동작 하고 있다.>



<가상화폐 실시간 시세가 스레드로 동작하고 있다.>



<회원가입 창>



<실행시 화면>

## 정리 및 논의

MockCoiner Project의 목표는 안전성을 보장해주면서 수익률을 확보하는 알고리즘을 개발하여 유저들에게 제공하는 것이다. 그 과정으로 MockCoiner에서 제작한 모델을 이용하여 원하는 코인을 선택하여 Back testing 해볼 수 있다. 이 외에도 머신러닝을 이용하여 의사결정에 도움을 주는 부가적인 서비스도 제공하는 것을 목적으로 한다. 기본적으로 시뮬레이션을 가정한 가상 트레이딩 시스템이기 때문에, 실제 거래소와의 연동이 아닌, 가상의 화폐를 가지고 가상의 투자를 제공한다. 제공 방식은 GUI를 기반으로 PyQt5로 제작한 exe 파일 형식이다. 구현되었지만 여러 이유에 따라 포함되지 않은 코드들은 python 파일의 형식으로 제공된다. 해당 책자에서는 모델의 발전 과정 및 여러 테스트 결과를 기재하였으나, 제공되는 exe 파일에서는 서비스 측면 및 효율성을 고려하여 가장 최종 단계의 모델들을 제공하기로 협의하였다.

LSTM 딥러닝 모델의 경우 딥러닝이라는 특성상 exe 파일에 포함하여 프로젝트를 진행해본 결과 심각한 사양의 저하 등이 발생하여, Python 코드 및 파일로 대체하기로 하였다.

거래 수수료 모델의 경우 거래소마다 수수료가 다르고, 가상 시뮬레이션의 경우 수수료가 크게 중요하지 않다고 판단되어 시스템의 간결함을 위하여 exe 파일에 포함하지 않기로 하였다.

# 설계요소 평가

|  |  |
| --- | --- |
| 성능 | 기본적으로 클라이언트의 성능에 좌우하는 경향이 있다. 딥러닝이나 머신러닝의 경우 서버에 맡기기에는 비용이나 공간적 제약이 따르며, 따라서 로컬로 구현하기에 이르렀는데, 딥러닝의 경우 EXE 파일로 제공하였을 때 심각한 성능 저하가 발생하여 파이썬 파일로 따로 제공되고 있다. 그 외에는 준수한 성능을 보인다고 볼 수 있다. |
| 안전성 | 아직 네이버 클라우드 서비스 플랫폼의 경우에는 무료 서비스를 이용하고 있기 때문에 다수의 유저 접속에 대한 안정성이 담보되기 어려움. 제한된 메모리와 트래픽 환경으로 더 효율적으로 구성하려면 실제 서비스를 진행하면서 조정이 필요할 것으로 보인다.  다만, 예측모델과 수학적 모델의 경우 안정적으로 결과가 출력되는바, 코드나 기능 자체의 안정적인 측면은 보장이 되는 것으로 예상된다. |
| 보안성 | 네이버 클라우드의 경우 미리 유저에 대한 접근 권한을 두었음. 특정 기능이 담긴 컴포넌트를 이용하기 위해서는 유저 로그인을 해야 하는데, 이 과정에서는 네이버 클라우드를 이용한 클라우드 서버를 이용하여, 서버에 접근해야 하므로, 보안성이 강화된다. |
| 구현 기간 | 개발 팀원의 변동으로 인하여 기존의 웹 서비스에서 GUI 기반의 로컬 서비스로 변화 되었기 때문에, 기존의 방향성과 맞지 않는 부분이 존재함. 해당 서비스를 기반으로 계속 발전 시켜 나간다면 웹 서비스 구현 혹은 어플리케이션 측면으로 개선 시킬 여지가 있다.  수학 및 머신러닝 모델의 구현 기간(3~10)보다 GUI 구현 기간이 짧았다.(9~10) |
| 구현 단가 | 웹 서비스와 같은 기준으로 구현 단가를 측정할 수는 없지만, 들어가는 유지비 같은 측면의 경우, 네이버 클라우드 플랫폼의 클라우드 DB Mysql을 사용하는 요금만 제외하면 유지비가 거의 들지 않는다. |

# 추후 발전방향

# 종합 토의

* 딥러닝을 이용해서 독자적인 수익률 모델을 구현하려 하였으나, 신경망 학습이 제대로 쌓이지 않아서 실제 가격과의 갭이 발생하기도 하고, 수학적 모델과 결합하여 거래 신호를 발생시키기에는 무리가 따르기 때문에, 딥러닝을 이용한 가격예측 모델은 가격 예측을 도와주는 하나의 지표로써 활용하기에 적합할 뿐, 수익률을 창출하는 데에 직접적으로 사용하기에는 무리가 따를 것으로 보인다.
* 네이버 클라우드 플랫폼의 경우 데이터 워크로드가 심하지 않아도 유지하는 데만 비용이 드는 측면이 있어서 AWS에 비해 장기간 사용하기에는 학생들의 입장에선 부적합 할 수 있다.
* UI 구성에 있어서 유저의 편의성이나 통일성 등의 부족함이 있는 것 같다.
* 머신러닝을 금융 시장 및 시스템 트레이딩 전자화폐 등에 사용하는 데에는 예상대로 적중률이 크게 높지 않기 때문에 현재 실제 필드에서 적극적으로 활용하기에는 어려울 것이라는 결론에 다다랐다. 하지만, 머신러닝을 이용하여 금융 관련 기능을 만들어보는 것 그 자체에 의의를 두기로 하였으며, 오히려 적중률이 낮기 때문에 앞으로 더 개선시켜 나갈 여지가 많은 것 같다.
* 백테스팅 상황과 실제 거래가 일어나는 상황에서의 환경 차이 때문에, 입출력을 통일하고 실시간 서비스에 맞게 코드를 지속적인 피드백이 필요할 것으로 보인다.
* 프로그램이 폐쇄적인 경향이 있다. 예를 들어 유저가 투자하고 싶은 금액만큼 투자하지 못하고 꼭 고정된 값으로만 투자를 해야 된다거나 하는 점 같이 서비스 적인 측면에서 폐쇄적인 경향이 있는데, 이는 구현상의 어려운 점은 아니며 단순히 MockCoiner 팀원들의 자체 판단에 의하여 고정 시키는 것이 좋을 것 이라는 생각에 결정된 사안이기 때문에 피드백을 받아 이러한 서비스 적인 측면을 개선해 나가야 할 여지가 있다. MockCoiner의 중심 과제는 특정 알고리즘을 만들고 모델을 구성하는 그 자체에 있었기 때문에, 서비스 적인 측면의 고려는 프로젝트 말미에 피드백을 거치면서 부족했음을 깨달았다.

# 일정 및 첨부

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 단계 | 업무 | 상세 내용 | | | 예정 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 진행 |  |  |  |  |  |  |  |  |  |  |
| 1. 프로젝트 계획 및 설계 | | | | | 완료 |  |  |  |  |  |  |  |  |  |  |
| 정의 및 설계 | 프로젝트 정의 | 프로젝트 정의 | | |  |  |  |  |  |  |  |  |  |  |  |
| 프로젝트 기간 및 일정 | | |  |  |  |  |  |  |  |  |  |  |  |
| 개발 준비 | GitHub,AWS,TensorFlow 설치 및 가입 | | |  |  |  |  |  |  |  |  |  |  |  |
| 2. GUI | | | | |  |  |  |  |  |  |  |  |  |  |  |
| GUI | 설계 및 구현 | UI/UX 설계 및 구현 | | |  |  |  |  |  |  |  |  |  |  |  |
| 프론트 엔드 작업 | | |  |  |  |  |  |  |  |  |  |  |  |
| 3. 프로그램 및 알고리즘 개발 | | | | |  |  |  |  |  |  |  |  |  |  |  |
| 머신 러닝 | 구현 | Tensorflow 기본 활용 | 샘플데이터학습 | 분석 모델 구현 |  |  |  |  |  |  |  |  |  |  |  |
| 시스템 트레이딩 | 설계 및 구현 | 시스템 트레이딩 알고리즘 설계 | 변동성 돌파 전략 수립 |  |  |  |  |  |  |  |  |  |  |  |
| 4. 분석 모델 웹 서비스 간 접목 및 백엔드 | | | | |  |  |  |  |  |  |  |  |  |  |  |
| 접목 및 통합 | 설계 및 구현 | DB 설계 및 구축 | | |  |  |  |  |  |  |  |  |  |  |  |
| 구현 | 학습 모델 동기화 | | |  |  |  |  |  |  |  |  |  |  |  |
| 5. 피드백 | | | | |  |  |  |  |  |  |  |  |  |  |  |
| 확인 | 백테스팅 | 알고리즘 백테스팅 및 개선 | | |  |  |  |  |  |  |  |  |  |  |  |
| 분석 | 계획서 대조 | 취약점 분석 및 기능 개선 | | |  |  |  |  |  |  |  |  |  |  |  |
| 완료 | 최종 정리 | 보고서 작성 | | |  |  |  |  |  |  |  |  |  |  |  |

* Github : https://github.com/zkvp201/mockcoiner.git
* 이외 계획서, 중간보고, 주차보고서 등.