

# AI를 활용한 법인 별 농산물 이상 경락 가격 알림 플랫폼

정보컴퓨터공학부 201724533 이성호  
정보컴퓨터공학부 202055520 김수현  
핀테크융합전공 201845938 최은진  
IT 응용공학과 202045834 진서현



## 0. 요약

농산물 도매시장 내 도매시장법인의 불법 거래로 농가 소득 저하와 농산물 생산자의 정보 불균형문제가 발생한다. 이는 도매법인의 독과점 구조원인으로 작용하기 때문에 생산자의 도매법인 선택과정에서 경쟁구도가 필요하다. 따라서 농산물 생산자에게 합리적인 법인 선택을 위한 충분한 법인 별 경락 정보를 제공하고, 불법 거래 의심 법인을 피하기 위해 이상 경락 가격 모델을 제시한다. 또한 평균 연령이 높은 농산물 생산자를 고려해 이상 경락 가격 발생 시 알림 기능을 추가해 활발한 정보 이용을 촉진한다.

충분한 정보 이용으로 농산물 생산자는 더 합리적인 가격의 법인을 선택하고, 도매법인의 경락 가격 경쟁수준을 높일 수 있다. 기존 모델이 불법 거래 법인 정보를 제공하지 못한다는 점에서 제시하는 새로운 모델이 좋은 해결책이 될 것이라 기대한다. 추가적으로 도매시장 관리의 효율성 증대, 소비자 가격 안정, 대중들의 이용으로 도매법인의 독과점 경계를 기대할 수 있다.

농림축산식품부에서 제공하는 ‘전국 도매시장 일별 정산 경락가격 상세정보(법인사용코드포함)’을 사용해, 일자 별 평균가로 LSTM 가격 예측 모델을 만들었다. 웹에는 법인 별 당일 가격 정보 및 익일 예측 가격 정보를 제시하고, 법인 별 가격 그래프를 구현했다. 카카오 봇을 이용하여 알림 설정 후 이상가격 발생 시 메일로 알림 기능을 구현하였다.

도매시장 내 도매법인의 불법 거래를 방지하고, 도매 법인 간의 충분한 경쟁으로 합리적인 경락가격을 기대하여 AI를 활용한 법인 별 농산물 이상 경락 가격 알림 플랫폼 모델을 제시한다.

Github : <https://github.com/eunjin917/PnuFintechCompetition>

웹서버 주소 : <https://github.com/eunjin917/PnuFintechCompetition>

카카오채널 : [https://pf.kakao.com/\\_Njlxixj](https://pf.kakao.com/_Njlxixj)

Colab - 데이터 생성 : <https://colab.research.google.com/drive/1QJzhqAn5gp7tRR4VnYTsvBz-GdiBWbHo?usp=sharing>

Colab - 모델 성능 평가 : <https://colab.research.google.com/drive/1c7qjx0ul46XYaKFdl3d-ib5WVX-R6jb3?usp=sharing>

Colab - 최종 모델 : <https://colab.research.google.com/drive/1kKNYzxLzJrabxae0VvALWlz6TAbx6vP6?usp=sharing>

# 목차

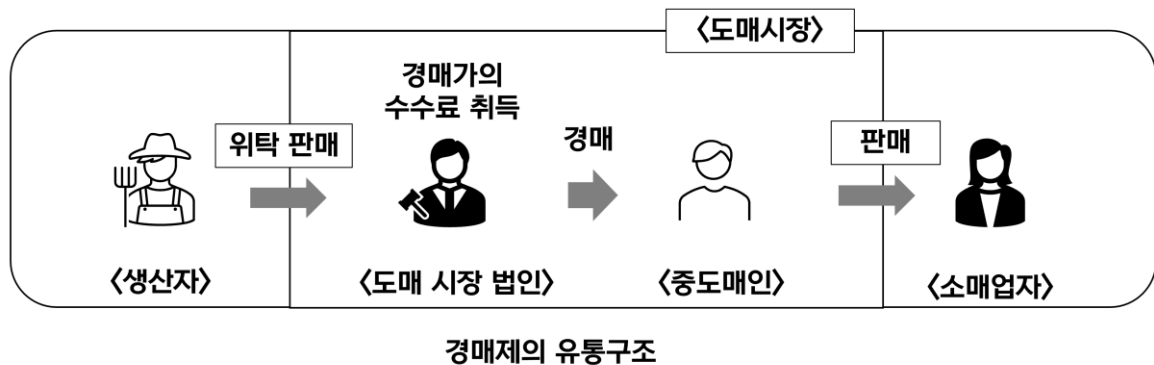
- 0 요약
- 1 모델 고안 배경
  - (1) 농산물 도매시장의 불법 거래 상황
  - (2) 불법 거래 피해
  - (3) 불법 거래 원인
- 2 모델 아이디어
  - (1) 모델 구성
  - (2) 제안 모델
    - 1> 공공데이터를 활용한 가격 예측모델과 가격 판단 모델
    - 2> 공공데이터를 활용한 법인 가격 비교 웹 구현
    - 3> 이상 가격 발생시 알람 구현
- 3 필요성
  - (1) 기존 해결방법의 한계점
  - (2) 제시 모델 해결방법의 장점
- 4 기대 효과
  - (1) 모델 기대 효과
  - (2) 추가 기대 효과
- 5 실제 구현
  - (0) 개발환경
  - (1) 인공지능 모델 설계
  - (2) 웹 관련 설명
  - (3) 카카오 봇을 통한 메일 알람 구현
- 6 제약상황과 발전 가능성
  - (1) 알람 구현 측면에서
  - (2) 추가 기대 효과
- 7 마무리하며
- 8 출처 확인

## 1.모델 고안 배경

## 1-(1) 농산물 도매시장의 불법 거래 상황

### [도매시장의 경매제]

<sup>i</sup>우리나라 도매시장에서 농산물 거래는 전체 물량의 80% 이상이 경매거래로 이루어진다. (농민신문 2021.03.03) 경매거래는 도매법인이 다수의 중도매인 중 가장 높은 가격을 적어낸 중도매인과 위탁거래를 하는 제도이다. 하지만 현재 도매법인의 불법거래로 경매제는 공정성을 잃고 생산자와 소비자의 가격부담이 커지고 있다.



### [도매법인의 불법 수수료 취득]

도매법인은 위탁거래 시 경매 낙찰가의 4~7% 수수료로 이득을 취한다. 도매법인은 중도매인과 로비활동으로 경매 전에 미리 가격을 정하고 불법 거래를 통해 수수료를 취하고 있다. <sup>ii</sup>서울시 조사 결과에 따르면 2019 년~2022 년동안 시장도매인 58 개가 중도매인 144 개와 농산물 불법 거래한 것으로 나타났으며, 불법 거래 규모는 637 억에 달했다. (농민신문 2022.07.25) <sup>iii</sup>실제로 2019 년동안 다수의 중도매인이 아닌 1 대 1 경매건수가 24 만 3378 건으로 전체 거래의 3.8%이다. 또한 충분히 입찰 가격을 기다리지 않고 3 초 내로 낙찰한 경매는 383 만건으로 전체의 60%에 육박한다. (한국 농정, 2020,1213)

## 1-(2) 도매법인의 불법 거래 문제

### [농가 소득 저하로 이어지는 불법 거래]

이런 불법 거래의 가장 큰 문제는 법인 낙찰가격의 신뢰성이 떨어지는 상황에서 농산물 생산자에게 공정한 실소득을 보장해주지 못한다는 것이다. 같은 품목 같은 등급이라도 법인마다 큰 경락 가 차이가 있다. <sup>iv</sup>실제 2022 년 8 월 10 일 평창의 출하자 최 모 씨는 풋고추 10 kg을 A 청과에 12 박스 B 청과에 14 박스를 경매를 통해 판매한 결과, A 청과는 박스당 24,000 원, B 청과는 고작 2000 원으로 12 배의 경락가격의 차이가 있었다. (월간원예, 2020.12.02)

## [농산물 생산자의 정보 불균형]

도매법인의 불법 거래에서 생산자는 정확한 경락가격 안내를 받지 못해 정보 불균형을 겪는다. <sup>v</sup>광주 서부 도매시장에 8 월 13 일 실제 거래 물량은 10~20 톤에 달했지만 관리사무소에 보고된 양은 6~9 톤이었으며, 생산자에게 전달되는 경매 가격과 실제 경매 가격의 차이가 있었음을 확인했다. (KBS 2020.09.30) 또한 도매법인의 경락 기준을 신뢰할 수 없어 소득 예측에 어려움을 겪는다.

또한 생산자는 불법 거래 법인을 피할 수 있도록 충분한 정보를 제공받아야 한다. 그러나 불법 거래 법인에 대한 정보는 <sup>vi</sup>공개법 제 9 조 1 항 5 호에 의해 부분공개로 보호받고 있다. 생산자는 이로 인해 어떤 법인이 불법 거래를 하는지 알 수 없다.

## 1-(3) 불법 거래 원인 분석

### [도매시장 내의 도매법인의 독과점 구조]

이런 도매시장의 문제는 도매법인의 독과점 구조라는 데 있다. <sup>vii</sup>실제로 84 년 최초 지정된 이후 독과점 도매법인들은 한 번도 퇴출(재지정 취소)된 적이 없다. (월간월예. 2020.12.02) 이러한 구조는 도매법인이 독점 사업자가 되어 도매법인만의 의사대로 가격을 결정하고 시장 지배력을 갖게 된다. <sup>viii</sup>실제로 가락도매시장 도매시장법인 간 경쟁수준은 상당히 낮으며 독점시장이다. (농산물... 2021 년 논문 발췌) 따라서 합리적인 가격 결정을 위해서는 생산자가 도매법인을 선택하는 과정에서 경쟁요소가 필요하다. 그러나 현재 생산자가 법인 선택 과정에 정보 부족으로 어려움이 있어, 충분한 경쟁이 이루어지지 않는다.

따라서 생산자에게 적절한 법인 별 경락 가격 정보를 제공하고, 공정한 법인 선택을 할 수 있도록 불법 거래로 의심되는 법인을 알려주는 모델을 제시하여, 도매법인 간의 공정한 경쟁수준을 유지한다.

## 2.모델 아이디어

### 2-(1) 모델 구성

#### [모델 고안 해결 목표]

농산물 생산자에게 합리적인 법인 선택을 돕는 충분한 정보를 제시하고, 특히 불법 거래 의심 법인을 선택하지 않도록 도울 수 있어야 한다. 이를 통해 도매 법인 간의 경쟁구도를 만들고 합리적인 가격 선정이 되어야 한다.

## [모델의 주요 타깃]

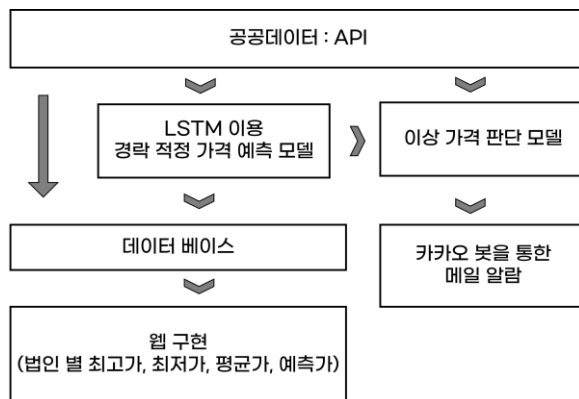
주요 타깃층은 정보 불균형을 겪는 농산물 생산자이며, <sup>ix</sup>특징은 평균 연령이 65.9 세로 디지털 기술 이용에 어려움을 겪을 것으로 예상된다. (지다넷 코리아.2022.10.19) 예상되는 추가 타깃층으로는 농산물 유통 관계자로 법인 별 가격 분석을 필요로 한다.

## [모델의 필수 기능]

- ① 디지털 이용에 어려움이 있을 생산자가 간단하게 이용할 수 있어야 한다.
- ② 생산자가 합리적인 선택을 도울 수 있을 만한 충분한 정보를 제공해야 한다.
- ③ 불법 거래 법인을 피할 수 있도록 이상가격 정보를 제공해야 한다.

## 2-(2) 제안모델

### [법인 별 농산물 이상 경락 가격 알림 플랫폼]



### 1> 공공데이터를 활용한 경락 적정 가격 예측 모델

농림축산식품의 공공데이터 ‘전국 도매시장 일별 정산 경락가격 상세정보(법인사용코드포함)’ 이용해 다음날의 경락가격을 예측하는 LSTM 모델을 만든다.

### 2> 법인 및 품목 별 가격 비교 웹

정보를 확인하는 웹사이트를 구현한다.

- ① 당일 실제가와 예측가 비교 그래프 및 정보 창
- ② 법인 별 경락 가격 비교 그래프

### ③ 전체 경락 가격 목록 표

#### 3> 이상 가격 발생 시 알람 구현

가격 예측 모델의 예측 가격과 당일 거래 가격이 다르면 도매법인의 이상 경락가격으로 가정한다.

이상가격 판단 기준은 '예측 값 절대 편차'를 이용하였다. 
$$\left[ \frac{\sum |실제값 - 예측값|}{데이터 개수} \right]$$

**1 년치 전체 법인 예측 값 절대 편차 > 당일 해당 법인의 예측 값 절대 편차**

**=> 이상가격으로 판단**

불법 거래로 인해 경락 가격 변동폭이 큰 거래를 탐지해야 하기 때문에, 다양한 대표 값 중에서 절대 편차를 이상 가격 판단 기준으로 선정하였다. 절대 편차는 데이터 중 크게 떨어진 값에 민감하다는 특징이 있어 적절하다.

선택한 법인에 대한 이상 가격 발생시 메일로 알람을 보내는 기능을 구현하여 상시 관찰이 가능하도록 한다.

## 3.필요성

### 기존 해결방법의 한계와 그에 대한 모델 제시해결방법의 장점

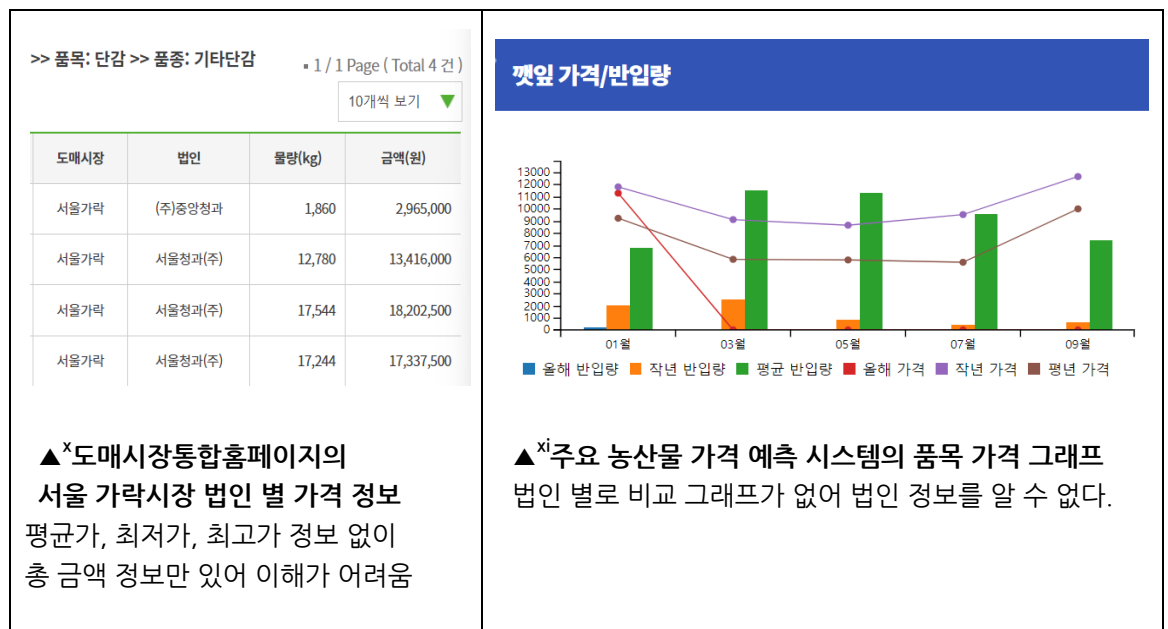
기준	기존 해결 방법	모델 제시 해결방법
농산물 생산자가 알아보기 쉽게 분석된 정보를 제공하는가?	부족하다.	그렇다.
(법인 별 품목 간의 경락가격 그래프, 당일 법인 별 경락 가격 변화 정보)	법인 별 경락 가격 보고서만 있거나, 법인 구분 없이 품목만 구분된 그래프만 있었다.  쉽게 이해할 수 있도록 분석된 사이트는 찾아볼 수 없었다.	법인 별 품목 간의 경락 가격 정보 및 그래프를 제공하고, 법인 별 경락 가격 그래프를 제공한다.
농산물 생산자가 불법 거래 법인을 피할 수 있도록 정보를 제공하는가?	그렇지 않다.	그렇다.
	따로 불법 거래가 의심되는 법인 정보를 제공하는 사이트를 찾을 수 없었다.	LSTM 모델을 사용하여 예측 가격을 제시한다.  예측 가격과 차이가 큰 법인을 불법 거래로 의심할 수 있도록 정보를 제공한다.

농산물 생산자가 지속적으로 정보를 제공받을 수 있도록 상시 알림 기능이 있는가?	그렇지 않다.	그렇다.
	이상 가격 정보를 알리를 제공하는 사이트는 찾을 수 없었다.	카카오 봇을 이용한 메일 알리를 구현한다.

### 3-(1) 기존 해결 방법의 한계점

#### ① 법인 선택을 위한 충분한 정보제공이 이루어지지 않는다.

선택을 위한 충분한 정보는 법인 별 경락 가격 그래프, 당일 법인 별 경락 가격 변화 정보이지만 기존 사이트는 두가지 정보 중 하나만 있거나 법인 구분 없이 품목별로만 가격 정보가 나와있다.



#### ② 불법 거래 확인을 위한 정보가 제공되지 않는다.

불법 거래 의심을 확인하는 사이트를 찾아볼 수 없었다.

#### ③ 농산물 생산자가 이상 가격 정보를 알림 받을 수 없다

이상 가격 정보 알리를 제공하는 사이트를 찾아볼 수 없었다.

### 3-(2) 제시 해결방법의 장점

단순 경락 가격 제시에 그치지 않고 상세한 법인 별 가격 정보와 이상 가격 정보를 제시한다.



법인별 당일 가격 분석 정보



당일 법인별 품목 가격 정보

- ① 법인 별로 당일 가격 분석 정보를 제공한다. 최고가, 최저가, 평균가, 예측가, 실제가 비교를 제시하여 현재 거래하는 법인에 대한 정보를 제공한다.
- ② 법인 별 경락가격 비교 그래프를 제공하여 사용자가 법인 가격 경향 정보를 쉽게 알 수 있도록 한다.
- ③ 불법 거래로 의심할 수 있는 이상 기준을 제시하여 합리적인 법인을 선택할 수 있게 한다.
- ④ 이상 가격 감지 시 알림 기능 덕분에 농산물 생산자의 관리가 수월하다.

## 4. 기대효과

### 4-(1) 모델 기대 효과

#### [예측가 제시를 통한 기대 효과 - 불법거래 방지]

다음날 예측 가격과, 예측 가격과 실제 거래가격 비교 정보를 제시한다. 사용자는 제공된 예측 가격을 기준으로 정상적인 가격 기준을 유추할 수 있어, 법인이 정상적인 가격 흐름에서 벗어나 경매를 하는지 알 수 있다.

이를 통해, 생산자는 불법 거래로 의심되는 법인을 피하고 정상 가격으로 경매하는 법인과 거래할 것이다.

#### [법인 별 상세 정보 제공을 통한 기대효과 - 법인 경쟁 구도 조성]

생산자는 기존에 거래하던 법인 외에도 다양한 법인의 거래 가격과 품목 별 출하 예측 가격을 알 수 있기 때문에 선택권이 보장될 것이다.

생산자는 더 합리적인 가격으로 낙찰하는 법인을 알 수 있고, 그 법인을 선택하게 될 것이다. 그 과정에서 법인은 도태되지 않기 위해 수수료 인하, 공정한 거래 등 경쟁을 통해 합리적인 가격 책정을 할 것이다.



### [이상가격 알림 기능 측면 - 농산물 생산자의 활발한 정보 사용]

평균 연령이 높은 농산물 생산자가 따로 웹사이트 이용 없이 정보를 받아 볼 수 있다. 이는 정보 불균형에 있던 생산자의 정보 이용이 더 수월 해진다. 이를 통해 더욱 효과적인 생산자의 정보 사용을 기대할 수 있고, 이를 통해 법인 간의 경쟁구도를 촉진할 수 있다.

앞선 과정을 통해 생산자는 정보 활용이 수월해지고 불법 거래로 의심되는 법인을 피할 것이고, 법인은 경쟁우위를 위해 합리적인 가격으로 경매할 것이다. 이를 통해 법인의 불법 거래가 방지되고 합리적인 가격 책정이 이루어질 것이다.

## 4-(2) 추가 기대효과

### [도매시장 관리의 효율성 증대]

상시 업데이트를 통해 도매시장 법인 별 거래 내역 분석을 받아 볼 수 있다. 또한 도매시장 법인에 대한 예측 가격 정보를 이용하여 불법 거래 법인을 유추할 수 있다. 이를 통해 상시적인 감독 및 관리의 효율성 증대를 기대할 수 있다.

### [유통비용율의 절감으로 소비자 가격 안정, 농가 실소득을 증가]

유통 과정에서 불법 거래를 지양하고 합리적인 가격 경매가 이루어질 것이다. 이는 먼저 유통비용율의 감소로 이어진다. <sup>xii</sup>현재 소비자 지불 가격 100기준으로 유통비용율은 47.5%, 그 중 도매단계는 10.8%, 직접비(도매시장 운송비, 도매시장 수수료, 하역비 등)는 16%이다. (전북중앙 2022.10.19) 경락 가격의 하락은 도매단계의 비용 축소와 간접비 축소를 기대할 수 있다.

<sup>xiii</sup>2022.9 월 양배추의 가격이 전일 대비 131% 급등한 다음날 46% 급락했다. 이처럼 현재는 불법 거래로 인한 농산물 가격 불안정성이 매우 크다. 앞선 기능과 과정을 통해 불법 거래 감소와 합리적인 경락가격 낙찰을 통해 농산물 가격 안정을 기대 할 수 있다.

이를 통해 소비자 가격 안정과, 농가 실소득을 높일 수 있다.

### [대중들의 관심 유도]

농산물 관계자가 아닌, 전문 지식이 없는 소비자나 일반 사용자들도 이해하기 쉽게 구현했다. 따라서 일반 대중들의 사이트 이용이 편리하고 이를 지속적인 관심으로 유도할 수 있다. 일반 대중들의 관심은 법인의 독과점을 경계할 수 있다.

## 5. 실제 구현

## 5-(0) 개발 환경

구현 관련 코드는 맨 앞장에 첨부했다.


모델 구현은 colab 에서 파이썬을 사용해 구현했다.

초반에는 Visual Studio Code 에서 웹 페이지의 프론트엔드는 html, css, Javascript 를 사용하였고, Django 로 백엔드를 구현했다. Django 를 사용하기 위해서 새로운 가상환경을 생성해 python, Django, tensorflow 등 필요한 패키지들을 설치하였다. 로컬 서버에서 웹을 구현했으나, 메일 연동을 위해 웹 서버가 필요했고 이를 위해 groomIDE 개발환경에서 웹 구현을 마무리하였다.

## 5-(1) 인공지능 모델 설계

- ① 공공데이터 API 를 활용해 도매법인별 가격 정보를 가져온다.

농림축산식품 공공데이터 포털에서 제공하는 오픈 API 를 활용하여 도매법인별 경락 가격 데이터를 가져온다.

**농림축산식품  
공공데이터 포털**

로그인 | 회원가입


데이터 검색   데이터 활용   데이터 신청   참여&소통   마이페이지

홈 > 데이터 검색 > 통합 검색

URL ☆

### 데이터 상세보기

관련 데이터 현황, 주요 기능을 확인할 수 있습니다.

  
유통소비

농림수산물교육문화정보원

#### 전국 도매시장 일별 정산 경락가격 상세정보(법인사용코드포함)

전국 35개 공영도매시장(일반법정 2개 포함)의 농수산물 정산전 정산 경매거래 가격 상세 정보로 경매 일별 도매시장법인별 경매 상세 정보 제공  
(법인 사용 품목, 품종, 산지 코드 포함)

수정일 | 2022.02.09   조회수 | 11499

제공기관	농림수산물교육문화정보원	분류체계	유통소비
------	--------------	------	------

제공기관: 농림축산식품부

데이터 : 전국 도매시장 일별 정산 경락가격 상세정보(법인사용코드포함)

([https://data.mafra.go.kr/opendata/data/indexOpenDataDetail.do?data\\_id=20180118000000000934](https://data.mafra.go.kr/opendata/data/indexOpenDataDetail.do?data_id=20180118000000000934))

## [서울가락 도매시장에서 한국 청과 법인의 풋고추(특) 검색 코드]

```
def date_range(start, end):
    start = datetime.strptime(start, "%Y%m%d")
    end = datetime.strptime(end, "%Y%m%d")
    dates = ((start + timedelta(days=i)).strftime("%Y%m%d") for i in range((end-start).days+1))
    return dates

marketname = "서울가락도매"
coname_c = "한국청과"
mclassname = "풋고추"
sclassname = "청양"
gradename = "11" # 등급 특 - 11, 상 - 12, 보통 13
grade = "특"

marketname_info = "6PBLMNG_WHSAL_MRKT_NM=" + marketname
coname_info_c = "6CPR_NM=" + coname_c
mclassname_info = "6PRDLST_NM=" + mclassname
sclassname_info = "6SPTIES_NM=" + sclassname
gradename_info = "6GRAD_CD=" + gradename

datas = [] # 가져온 데이터를 pandas 형태로 저장
datas = date_range("20230101", "20230119") # 탐색 범위(처음,끝)

for date in datas:
    url = 'http://211.237.50.150:7080/openapi/d60c7fa3f4501f62dbd2ca000625a2d12b764928e9a2cd513dfac4d628991c5/json/Grid_20180118000000000581_1/1/1000?'
    date_info = "DEUNG_DE="+date
    url = url + date_info + mclassname_info + sclassname_info + marketname_info + coname_info_c + gradename_info
    url_encoded_n = quote(url, safe=':/=26')
    response = requests.get(url_encoded_n)
    contents = response.text
    data = json.loads(contents)
    stat = data['Grid_20180118000000000581_1']['row']

    if stat == [] : #없으면 건너뛰기
        continue

    else : #해당 날짜에 데이터 있으면 추가
        datas.append(stat)

df = pd.DataFrame() # 가지고 온 데이터 pandas 형태로 변환
for i in range(len(datas)):
    datas[i] = pd.DataFrame(datas[i])
    df = pd.concat([df,datas[i]],ignore_index = True)
```

## ② 법인 별 품종의 최고가, 최저가, 평균가를 날짜마다 구한다.

### [법인 별 품종의 최고가, 최저가, 평균가 구하는 코드]

```
day_mean = [] # 날짜/ 평균값 pandas
x_shift_data = [] # 평균가/ shift된 입력 데이터 (input)
timestep = 10 # 최근 10영업일로 다음 값 예측
mean_values = []
date = []

for i in range(datasize):
    x_shift_data.append(data[i].groupby('DEUNG_DE', as_index=False)['PRI_AVE'].mean()) # 모든 법인의 날짜별 평균값 리스트
    x_shift_data[i]['PRI_AVE'] = round(x_shift_data[i]['PRI_AVE'],2)
    day_mean.append(x_shift_data[i][['DEUNG_DE','PRI_AVE']])
    for s in range(1, timestep+1):
        x_shift_data[i]['shift_'+str(s)] = x_shift_data[i]['PRI_AVE'].shift(s)

for i in range(datasize):
    x_shift_data[i] = x_shift_data[i].dropna().drop('PRI_AVE',axis=1).drop('DEUNG_DE',axis=1) #자신의 평균가 삭제 => 자신의 평균가를 예측하기 위해 자신의 평균가를 입력으로 넣지 않음, 날짜도 삭제
    mean_values.append(day_mean[i]['PRI_AVE'].tolist())
    date.append(day_mean[i]['DEUNG_DE'].tolist())

x_data = [] # 입력 데이터
y_data = [] # 타겟 데이터

for i in range(datasize):
    x_data.append(np.array(x_shift_data[i][x_shift_data[i].columns].values.tolist())) # 모든 컬럼 값들을 가진 행들을 리스트화
    y_data.append(np.array(mean_values[i][timestep:].reshape(-1,1)) #timestep만큼 삭제된 정보 이후의 데이터의 타겟값을 반영
    date[i] = date[i][timestep:]

[11] x_data

[array([[3894.98, 4169.22, 3598.96, ..., 4419.32, 3688.46, 3206.45],
       [4655.17, 3894.98, 4169.22, ..., 3708.36, 4419.32, 3688.46],
       [5353.97, 4655.17, 3894.98, ..., 3877.95, 3708.36, 4419.32],
       ...,
       [6266.37, 5313.31, 6052.83, ..., 5873.04, 5958. , 6485.52],
       [6489.91, 6266.37, 5313.31, ..., 5917.05, 5873.04, 5958. ],
       [7422.59, 6439.91, 6266.37, ..., 8568.75, 5917.05, 5873.04]])],
array([[2723.64, 2562.46, 2385.67, ..., 2729.23, 2687.29, 2348.27],
       [3835.38, 2723.64, 2562.46, ..., 2152.56, 2729.23, 2687.29],
       [2925. , 3835.38, 2723.64, ..., 2849.31, 2152.56, 2729.23],
       ...,
       [4829.92, 3973.92, 1333.33, ..., 4900. , 3343.41, 3183.95],
       [4877.84, 4829.92, 3973.92, ..., 3361.94, 4900. , 3343.41],
       [3979.92, 4877.84, 4829.92, ..., 3591.28, 3361.94, 4900. ]]])]
```

## ③ 학습 이용 데이터를 구조화한다.

### [ 학습에 이용한 데이터 ]

1. 2020~2022 년도 가락도매시장 [풋고추-청양-특]에 대한 서울청과, 동화청과, 한국청과, 중앙청과의 데이터 2191 개를 사용하였다.

2. 2020~2022 년도 가락도매시장 [새송이-새송이(일반)-특]에 대한 서울청과, 동화청과, 한국청과, 중앙청과의 데이터 1722 개를 사용하였다.)

[X 데이터] : 예측 날짜 이전의 10 개 거래 평균가의 리스트

[Y 데이터 (타겟값)] : 해당 날짜의 평균가

날짜	평균가
2022-12-01	1200
2022-12-02	1300
2022-12-03	1400
2022-12-04	1500
2022-12-05	1400
2022-12-06	1500
2022-12-07	1200
2022-12-08	1300
2022-12-09	1500
2022-12-10	1600
2022-12-11	1800

X데이터

Y데이터  
(타겟값)

#### ▲입력데이터 구조

[입력 데이터 구조 코드]

```
data = df[['DELNG_DE', 'DELNGBUNDLE_QY', 'PRICE']] #필요한 칼럼만 선택
data['DELNGBUNDLE_QY'] = data['DELNGBUNDLE_QY'].astype('float')
data['PRICE'] = round(data['PRICE']/data['DELNGBUNDLE_QY'], 2) # 1KG 기준으로 판매가를 조정해줌
data['DELNGBUNDLE_QY'] = "1KG"
data['DELNG_DE'] = pd.to_datetime(data['DELNG_DE'], format='%Y%m%d')

new_data = data
new_data['PRI_MAX'] = data.groupby('DELNG_DE', as_index=False)['PRICE'].transform('max')
new_data['PRI_MIN'] = data.groupby('DELNG_DE', as_index=False)['PRICE'].transform('min')
new_data['PRI_AVE'] = data.groupby('DELNG_DE', as_index=False)['PRICE'].transform('mean')
new_data = new_data.groupby('DELNG_DE', as_index=False).mean()
new_data['MRKT_NM'] = marketname
new_data['CPR_NM'] = cconame_c
new_data['PRDLST_NM'] = mclassname
new_data['SPCIES_NM'] = sclassname
new_data['GRAD'] = grade
new_data['weight'] = "1KG"
new_data = new_data[['DELNG_DE', 'MRKT_NM', 'CPR_NM', 'PRDLST_NM', 'SPCIES_NM', 'GRAD', 'weight', 'PRI_MAX', 'PRI_MIN', 'PRI_AVE']]

new_data.head()
```

	DELNG_DE	MRKT_NM	CPR_NM	PRDLST_NM	SPCIES_NM	GRAD	weight	PRI_MAX	PRI_MIN	PRI_AVE
0	2023-01-05	서울가락도매	한국청과	꽃고추	청양	특	1KG	8800.0	1550.0	6654.910714
1	2023-01-06	서울가락도매	한국청과	꽃고추	청양	특	1KG	8300.0	2100.0	6628.571429
2	2023-01-07	서울가락도매	한국청과	꽃고추	청양	특	1KG	7600.0	2100.0	6227.713140
3	2023-01-11	서울가락도매	한국청과	꽃고추	청양	특	1KG	7800.0	2200.0	5917.229730
4	2023-01-13	서울가락도매	한국청과	꽃고추	청양	특	1KG	8500.0	1875.0	5763.081395

#### ④ MinMaxScaler 를 이용해 데이터 전처리한다.

MinMaxScaler 는 다음과 같이 입력값을 반환해준다.

$$X\_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))$$

$$X\_scaled = X\_std * (max - min) + min$$

훈련시킨 데이터 값중 가장 큰 값을 1, 가장 작은 값을 0 이라 가정하고 scale 되는 데이터의 값을 보고 0 부터 1 사이의 수로 반환해준다. 이를 통해 입력값들의 차이가 크지 않게 만들어준다.

## [데이터 전처리 코드]

```
from sklearn.preprocessing import MinMaxScaler # 전처리
from pickle import dump

x_model_sc = []
y_model_sc = []
x_data_sc = []
y_data_sc = []
for i in range(datasize):
    x_model_sc.append(MinMaxScaler())
    x_model_sc[i].fit(x_data[i])
    y_model_sc.append(MinMaxScaler())
    y_model_sc[i].fit(y_data[i])
    x_data_sc.append(x_model_sc[i].transform(x_data[i]))
    y_data_sc.append(y_model_sc[i].transform(y_data[i]))
    dump(x_model_sc[i], open('/content/drive/MyDrive/fintech_project/scaler/x_sc{}.pkl'.format(i), 'wb'))
    dump(y_model_sc[i], open('/content/drive/MyDrive/fintech_project/scaler/y_sc{}.pkl'.format(i), 'wb'))
```

- ⑤ Train\_test\_split 을 통해 테스트 셋과 검증 셋을 나눴다.

## [테스트 셋과 검증 셋 나누는 코드]

```
from sklearn.model_selection import train_test_split

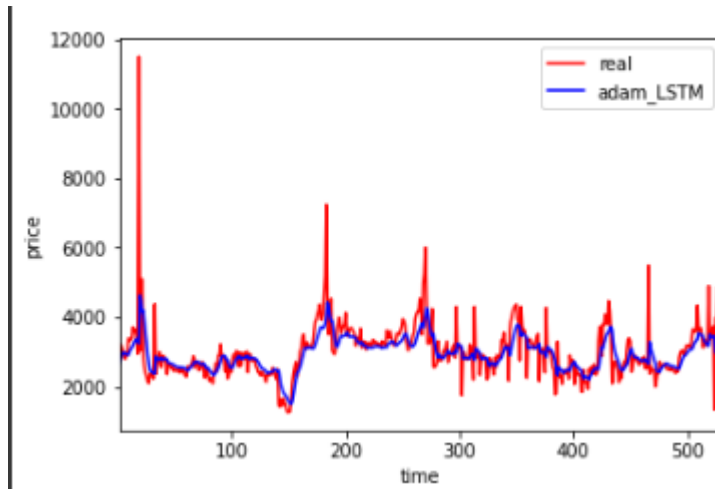
x_train = []
x_test = []
y_train = []
y_test = []

for i in range(datasize):
    a,b,c,d = train_test_split(x_data_sc[i], y_data_sc[i], test_size=0.2)
    x_train.append(a)
    x_test.append(b)
    y_train.append(c)
    y_test.append(d)
```

## [모델 성능 테스트 과정]

	첫번째모델	두번째모델	세번째 모델	네번째모델
모델	simpleRNN	LSTM	LSTM	LSTM
입력층	1	2	4	4
유닛수	-	각 층마다 8	512/256/128/64/32	512/256/128/64/32
드롭아웃	-	0.3	0.1	0.1
Optimizer	adam	rmsprop	adam	rmsprop
손실함수	mean_squared_error	mean_squared_error	mean_squared_error	mean_squared_error
Early Stopping 기준	3번이상 성능이 나아지 지 않으면 조기 종료	20번이상	20번 이상	20번 이상
epochs	1000	1000	1000	1000
배치 크기	64	64	64	64
r2score	0.0948	0.4335	0.5471	0.2870
MAE	308.9381	254.4358	208.0414	309.2661

세번째 모델이 가장 좋은 r2score 와 가장 적은 MAE 이 나옴으로 가격 측정 시 세번째 모델을 선택한다.



▲ 세번째 코드의 실제가격과 예측 가격의 그래프

풋고추- 청양 - 특 모델과, 새송이 - 새송이(일반)- 특 모델을 구현했다.

[첫번째 모델 코드]

```
from tensorflow import keras
from keras.models import Sequential
from keras.layers import SimpleRNN
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping

model = Sequential([
    SimpleRNN(1, input_shape=[timestep, 1])
])

print(model.summary())

Model: "sequential"
Layer (type)                Output Shape              Param #
-----
simple_rnn (SimpleRNN)        (None, 1)                 3
-----
Total params: 3
Trainable params: 3
Non-trainable params: 0
None

model.compile(optimizer = "adam", loss = 'mean_squared_error')
checkpoint_cb = keras.callbacks.ModelCheckpoint('/content/drive/MyDrive/project_model/best-simplernn-model_adam.h5')
early_stopping_cb = keras.callbacks.EarlyStopping(monitor='loss', patience=3, restore_best_weights=True)
history = model.fit(x_train_sc, y_train_sc, epochs=1000, batch_size=64,
                    validation_data=(x_test_sc, y_test_sc), callbacks=[checkpoint_cb, early_stopping_cb])
```

[두번째 모델 코드]

```
model1 = keras.Sequential()
model1.add(keras.layers.LSTM(8, dropout=0.3, return_sequences = True, input_shape = (x_train.shape[1], 1)))
model1.add(keras.layers.LSTM(8, dropout=0.3))
model1.add(keras.layers.Dense(1))
model1.summary()

Model: "sequential_1"
Layer (type)                Output Shape              Param #
-----
lstm (LSTM)                  (None, 10, 8)            320
lstm_1 (LSTM)                (None, 8)                 544
dense (Dense)                (None, 1)                 9
-----
Total params: 873
Trainable params: 873
Non-trainable params: 0

model1.compile(optimizer = "rmsprop", loss = 'mean_squared_error')
checkpoint_cb = keras.callbacks.ModelCheckpoint('/content/drive/MyDrive/project_model/simple-LSTM-model1_rmsprop.h5')
early_stopping_cb = keras.callbacks.EarlyStopping(monitor='loss', patience=20, restore_best_weights=True)
history1 = model1.fit(x_train_sc, y_train_sc, epochs=1000, batch_size=64,
                     validation_data=(x_test_sc, y_test_sc), callbacks=[checkpoint_cb, early_stopping_cb])
```

## [세번째 모델 코드]

```
model2 = keras.Sequential()
model2.add(keras.layers.LSTM(512, dropout=0.1, return_sequences = True, input_shape = (x_train.shape[1], 1)))
model2.add(keras.layers.LSTM(256, dropout=0.1, return_sequences = True))
model2.add(keras.layers.LSTM(128, dropout=0.1, return_sequences = True))
model2.add(keras.layers.LSTM(64, dropout=0.1, return_sequences = True))
model2.add(keras.layers.LSTM(32, dropout=0.1))
model2.add(keras.layers.Dense(1))
model2.summary()

Model: "sequential_3"

```

Layer (type)	Output Shape	Param #
lstm_7 (LSTM)	(None, 10, 512)	1052672
lstm_8 (LSTM)	(None, 10, 256)	787456
lstm_9 (LSTM)	(None, 10, 128)	197120
lstm_10 (LSTM)	(None, 10, 64)	49408
lstm_11 (LSTM)	(None, 32)	12416
dense_2 (Dense)	(None, 1)	33

```

Total params: 2,099,105
Trainable params: 2,099,105
Non-trainable params: 0

model2.compile(optimizer = "adam", loss = 'mean_squared_error')
checkpoint_cb = keras.callbacks.ModelCheckpoint('/content/drive/MyDrive/project_model/LSTM-model2_adam.h5')
early_stopping_cb = keras.callbacks.EarlyStopping(patience=20, restore_best_weights=True)
history2 = model2.fit(x_train_sc, y_train_sc, epochs=1000, batch_size=64,
                    validation_data=(x_test_sc, y_test_sc), callbacks=[checkpoint_cb,early_stopping_cb])
```

## [네번째 모델 코드]

```
model3 = keras.Sequential()
model3.add(keras.layers.LSTM(512, dropout=0.1, return_sequences = True, input_shape = (x_train.shape[1], 1)))
model3.add(keras.layers.LSTM(256, dropout=0.1, return_sequences = True))
model3.add(keras.layers.LSTM(128, dropout=0.1, return_sequences = True))
model3.add(keras.layers.LSTM(64, dropout=0.1, return_sequences = True))
model3.add(keras.layers.LSTM(32, dropout=0.1))
model3.add(keras.layers.Dense(1))
model3.summary()

Model: "sequential_4"

```

Layer (type)	Output Shape	Param #
lstm_12 (LSTM)	(None, 10, 512)	1052672
lstm_13 (LSTM)	(None, 10, 256)	787456
lstm_14 (LSTM)	(None, 10, 128)	197120
lstm_15 (LSTM)	(None, 10, 64)	49408
lstm_16 (LSTM)	(None, 32)	12416
dense_3 (Dense)	(None, 1)	33

```

Total params: 2,099,105
Trainable params: 2,099,105
Non-trainable params: 0

model3.compile(optimizer = "rmsprop", loss = 'mean_squared_error')
checkpoint_cb = keras.callbacks.ModelCheckpoint('/content/drive/MyDrive/project_model/LSTM-model2_rmsprop.h5')
early_stopping_cb = keras.callbacks.EarlyStopping(monitor='loss',patience=20, restore_best_weights=True)
history3 = model3.fit(x_train_sc, y_train_sc, epochs=1000, batch_size=64,
                    validation_data=(x_test_sc, y_test_sc), callbacks=[checkpoint_cb,early_stopping_cb])
```

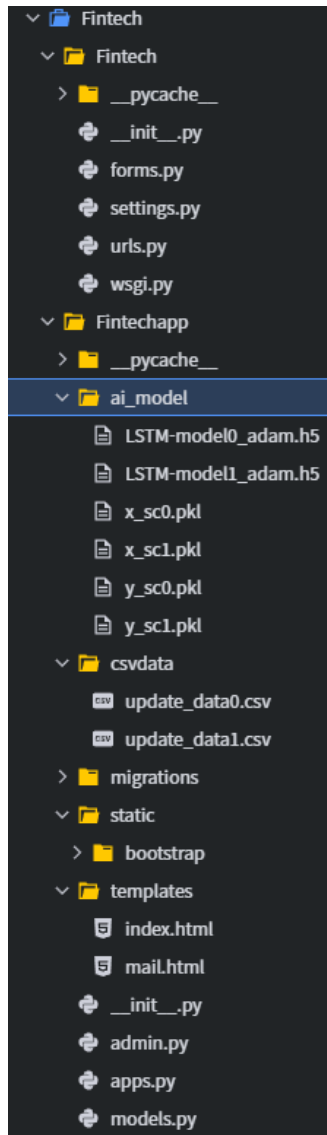
## 5-(2) 웹 구상

### 1. 파일 설명

핵심 파일들만 나열하였으며, 전체 파일은 GitHub 를 통해 확인할 수 있다.

(<https://github.com/eunjin917/PnuFintechCompetition>)

웹 주소 : (<https://pnu-fintech.run.goorm.app/>)



A file explorer view showing the project structure. The 'ai\_model' folder is selected, showing files like LSTM-model0\_adam.h5, LSTM-model1\_adam.h5, x\_sc0.pkl, x\_sc1.pkl, y\_sc0.pkl, and y\_sc1.pkl. Other folders like csvdata, migrations, static, bootstrap, and templates are also visible.

templates/index.html	홈페이지
templates/mail.html	Mail 발송 시 HTML
csvdata/update_data0.csv	농협가격(공), 서울청과, 동화청과, 한국청과, 중앙청과의 2년치 풋고추 데이터
csvdata/update_data1.csv	농협가격(공), 서울청과, 동화청과, 한국청과, 중앙청과의 2년치 새송이 데이터
ai_model/LSTM_model0_adam.h5	Colab에서 학습시킨 풋고추 가격 예측 모델
ai_model/LSTM_model1_adam.h5	Colab에서 학습시킨 새송이 가격 예측 모델
ai_model/x_sc0.pkl	Pickle로 저장시킨 풋고추 입력 데이터에 대한 MinMaxScaler scaler
ai_model/x_sc1.pkl	Pickle로 저장시킨 새송이 입력 데이터에 대한 MinMaxScaler scaler
ai_model/y_sc0.pkl	Pickle로 저장시킨 풋고추 출력 데이터에 대한 MinMaxScaler scaler
ai_model/y_sc1.pkl	Pickle로 저장시킨 새송이 출력 데이터에 대한 MinMaxScaler scaler
Urls.py	url 요청시 views.py에 있는 함수와 연결
models.py	데이터베이스 정의
views.py	url에서 넘어온 request 처리 함수 설정, models.py에서 정의한 데이터베이스 이용



## 2. 함수 별 설명

### (1) urls.py

- `path('company_good/c<int:company_pk>/g<int:good_pk>', ...)`
- 사용자가 [법인-품목] 선택 후 검색 버튼을 누르면 해당 url로 이동한다.
- [https://pnu-fintech.run.goorm.app/company\\_good/c0/g0](https://pnu-fintech.run.goorm.app/company_good/c0/g0)
- 선택한 법인은 'c+(법인번호)', 선택한 품목은 'g+(품목번호)'가 되어, 사용자에게 주소 뒤에 /company\_good/c법인번호/g품목번호 형식으로 들어온 url이 보여진다.

[urls.py 구현코드]

```
views.py  models.py  urls.py x
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4   https://docs.djangoproject.com/en/2.2/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import:  from my_app import views
8     2. Add a URL to urlpatterns:  path('', views.home, name='home')
9 Class-based views
10    1. Add an import:  from other_app.views import Home
11    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18
19 import Fintechapp.views
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', Fintechapp.views.index, name="index"),
24     path('company_good/<int:company_pk>/<int:good_pk>', Fintechapp.views.company_good, name="company_good"),
25     path('setting_data', Fintechapp.views.setting_data, name="setting_data"),
26     path('update', Fintechapp.views.update, name="update"),
27     path('alarm_setting', Fintechapp.views.alarm_setting, name="alarm_setting"),
28     path('alarm_bell', Fintechapp.views.alarm_bell, name="alarm_bell"),
29 ]
```

### <메인 페이지>

- `path('', Fintechapp.views.index, name="index"),`
- <https://pnu-fintech.run.goorm.app>
- 사용자가 주소만 치면 보이는 화면이다. 기본으로 [중앙청과-청양고추]를 보여준다.

<code>path('admin', admin.site.urls)</code>	/admin url 실행 시 admin.site.urls 실행
<code>path('', Fintechapp.views.index, name="index")</code>	"(초기 화면) url 실행 시 Fintechapp.views.index 함수 실행
<code>path('company_good/&lt;int:company_pk&gt;/&lt;int:good_pk&gt;', Fintechapp.views.company_good, name="company_good")</code>	/company_good/<int:company_pk>/<int:good_pk>/company_good url 실행 시 Fintechapp.views.company_good 함수 실행
<code>path('setting_data', Fintechapp.views.setting_data, name="setting_data")</code>	/setting_data url 실행 시 Fintechapp.views.setting_data 함수 실행
<code>path('update', Fintechapp.views.update,</code>	/update url 실행 시

name="update")	Fintechapp.views.update 함수 실행
path('alarm_setting', Fintechapp.views.alarm_setting, name="alarm_setting")	/alarm_setting url 실행 시 Fintechapp.views.alarm_setting 함수 실행
path('alarm_bell', Fintechapp.views.alarm_bell, name="alarm_bell")	/alarm_bell url 실행 시 Fintechapp.views.alarm_bell 함수 실행

## (2)models.py

Price	공공데이터(농산물도매법인별경락가격조회) API로 가져온 데이터를 이용해, 거래날짜, 거래시장, 법인, 품목, 품종, 품질, 무게, 최고가, 최저가, 평균가, 예측가를 저장하는 데이터베이스 정의
Predict	예측 모델에서 실행한 결과인 내일 예측가를 저장하는 데이터베이스 정의
KakaoUser	카카오봇 사용자의 이메일, (법인-품목) 알림설정 여부 정보를 저장하는 데이터베이스 정의

### [models.py 코드]

```

1 from django.db import models
2
3 class Price(models.Model):
4     DELNG_DE = models.CharField(max_length=30)
5     MRKT_NM = models.CharField(max_length=30)
6     CPR_NM = models.CharField(max_length=30)
7     PRDLST_NM = models.CharField(max_length=30)
8     SPCIES_NM = models.CharField(max_length=30)
9     GRAD = models.CharField(max_length=30)
10    weight = models.CharField(max_length=30)
11    PRI_MAX = models.DecimalField(max_digits=10, decimal_places=2)
12    PRI_MIN = models.DecimalField(max_digits=10, decimal_places=2)
13    PRI_AVE = models.DecimalField(max_digits=10, decimal_places=2)
14    PRI_PRED = models.DecimalField(max_digits=10, decimal_places=2)
15
16 class Predict(models.Model):
17     price0 = models.DecimalField(max_digits=10, decimal_places=2) # 청양고추
18     price1 = models.DecimalField(max_digits=10, decimal_places=2) # 새송이
19
20 class KakaoUser(models.Model):
21
22     user_email = models.EmailField(unique=True)
23
24     alarm_0_0 = models.BooleanField(default=False) # 중앙청과 - 청양고추
25     alarm_1_0 = models.BooleanField(default=False) # 서울청과 - 청양고추
26     alarm_2_0 = models.BooleanField(default=False) # 동화청과 - 청양고추
27     alarm_3_0 = models.BooleanField(default=False) # 농협가락(공) - 청양고추
28     alarm_4_0 = models.BooleanField(default=False) # 한국청과 - 청양고추
29
30     alarm_0_1 = models.BooleanField(default=False) # 중앙청과 - 새송이
31     alarm_1_1 = models.BooleanField(default=False) # 서울청과 - 새송이
32     alarm_2_1 = models.BooleanField(default=False) # 동화청과 - 새송이
33     alarm_3_1 = models.BooleanField(default=False) # 농협가락(공) - 새송이
34     alarm_4_1 = models.BooleanField(default=False) # 한국청과 - 새송이
35
36
37     def __str__(self):
38         return self.user_email
39
40

```

## (3) views.py

### ① index(request) 함수

처음 보이는 메인 페이지로, 화면 구성에 필요한 법인, 품목 정보, 그래프 차트에 필요한 8일 데이터, 표 테이블에 넣을 데이터를 데이터베이스에서 가져온다. 기본값으로 [중앙청과(0)-풋고추(0)]로 설정하였다.

### [index 함수 구현 코드]

```

## 메인 페이지 (중앙청과, 청양고추가 대표로)
def index(request):
    # 해당 법인에 대한 정보 가져오기
    price_info = Price.objects.filter(CPR_NM=company_list[0],
                                      PRDLST_NM=good_list[0]["PRDLST_NM"], SPCIES_NM=good_list[0]["SPCIES_NM"], GRAD=good_list[0]["GRAD"])

    # 5개 법인에 대한 전체 정보 가져오기
    company_info = [] # [[]]
    day8_info = []
    for company_no in range(5):
        prices = Price.objects.filter(CPR_NM=company_list[company_no], PRDLST_NM=good_list[0]["PRDLST_NM"],
                                      SPCIES_NM=good_list[0]["SPCIES_NM"], GRAD=good_list[0]["GRAD"])

        count = prices.count()
        for i in range(7, -1, -1): # 7일전 ~ 0일전 총 8일 데이터
            day8_info.append(prices[count-1-i])
        company_info.append(day8_info)
        day8_info = []

    next_predict = Predict.objects.last().price0

    return render(
        request,
        'index.html',
        {
            'price_info': price_info, # 해당 법인만 있는 정보
            'company_info': company_info, # 법인 전체가 있는 정보
            'this_info': price_info.last(), # 마지막으로 업데이트된 날짜의 정보
            'next_predict': Predict.objects.last().price0, # 그 다음에 업데이트될 날짜를 예측
            'error': Predict.objects.last().price0 - price_info.last().PRI_PRED, # 나중에 빠지!!!!!!!!!!!!!!!!!!!!
        }
    )

```

② company\_good(request, company\_pk, good\_pk) 함수

사용자가 드롭다운에서 선택한 company\_pk 와 good\_pk 에 해당하는 법인 정보, 품목 정보, 그래프 차트에 필요한 8 일 데이터, 표 테이블에 넣을 데이터를 가져온다.

[company\_good 함수 구현 코드(1)]

```

## 법인, 품목별 세부 페이지
def company_good(request, company_pk, good_pk): #/company_good/c0(중앙청과)/g0(청양고추)

    # 해당 (법인-품목)에 대한 정보 가져오기
    price_info = Price.objects.filter(CPR_NM=company_list[company_pk],
                                      PRDLST_NM=good_list[good_pk]["PRDLST_NM"], SPCIES_NM=good_list[good_pk]["SPCIES_NM"], GRAD=good_list[good_pk]["GRAD"])

    # 8일 간 법인별 정보 가져오기
    company_info = [] # [[0법인8일], ..., [4법인8일]]
    day8_info = []
    for company_no in range(5): # 법인별
        prices = Price.objects.filter(CPR_NM=company_list[company_no], PRDLST_NM=good_list[good_pk]["PRDLST_NM"], SPCIES_NM=good_list[good_pk]["SPCIES_NM"], GRAD=good_list[good_pk]["GRAD"])

        count = prices.count()
        for i in range(7, -1, -1): # 7일전 ~ 0일전 총 8개
            day8_info.append(prices[count-1-i])
        company_info.append(day8_info)
        day8_info = []

    # 품목(풋고추, 새송이)에 따라 next_predict에 다음날 예측값 가져오기
    if good_pk == 0:
        next_predict = Predict.objects.last().price0
    elif good_pk == 1:
        next_predict = Predict.objects.last().price1

```

## [company\_good 함수 구현 코드 (2)]

```
# 5개 법인 전체 정보 가져오기
all_info = Price.objects.filter(PRLST_NM=good_list[good_pk]["PRLST_NM"], SPCIES_NM=good_list[good_pk]["SPCIES_NM"], GRAD=good_list[good_pk]["GRAD"])
y = [item.PRI_AVE for item in all_info]
pred = [item.PRI_PRED for item in all_info]
mean_error = round(mean_absolute_error(y, pred), 2) # 이상치 판별 기준값 설정

# 해당 법인에 대한 가장 최신의 정보 가져오기
price_info = Price.objects.filter(CPR_NM=company_list[company_pk],
                                  PRLST_NM=good_list[good_pk]["PRLST_NM"], SPCIES_NM=good_list[good_pk]["SPCIES_NM"], GRAD=good_list[good_pk]["GRAD"])

this_info = price_info.last()
this_error = abs(this_info.PRI_AVE - this_info.PRI_PRED) # (법인-품목)의 예측값과 실제값 편차

# 이상값 시 (실제값-예측값 편차가 기준 이상)
is_error = False
if this_error > mean_error:
    is_error = True

print(is_error)

return render(
    request,
    'index.html',
    {
        'price_info': price_info, # 해당 법인만 있는 정보
        'company_info': company_info, # 법인 전체가 있는 정보
        'this_info': this_info, # 마지막으로 업데이트된 날짜의 정보
        'next_predict': next_predict, # 그 다음에 업데이트될 날짜를 예측
        'mean_error': mean_error, # 전체 법인의 전체 기간에 대한 오차 평균
        'this_error': this_error, # 선택된 법인1개의 오늘에 대한 오차 평균
        'is_error': is_error, # 이상인지 여부
    }
)
```

## [함수 설명]

- 원하는 [법인-품목]을 선택하고 검색 버튼을 누르면 만들어진 url([https://pnu-fintech.run.goorm.app/company\\_good/c0/g0](https://pnu-fintech.run.goorm.app/company_good/c0/g0))에 대해, 0 을 company\_pk(법인) 로, 0 을 good\_pk(품목)로 받아온다.
- 선택한 [법인-품목]에 대한 정보를 Price 모델을 통해 가져와 price\_info 에 저장한다.
- 전체 법인에 대한 정보를 Price 모델을 통해 가져온다. 이 때, 최근 8 일에 대한 정보만 가져온다. company\_info 에 2 중리스트로 [ [0 법인의 1 일, 2 일, ...7 일], ..., [4 법인의 1 일, 2 일, ...7 일] ] 형식으로 저장한다.
- 선택한 품목에 대한 다음날 예측값을 Predict 모델을 통해 가져온다.
- 전체 법인, 전체 기간에 대한 선택한 품목 정보를 Price 모델을 통해 가져와 all\_info 에 저장한다. 가져온다. y 에는 실제평균값들, pred 에는 전날에 예측한 해당 날짜의 값들이다. 이 때, y 와 pred 의 절대값평균을 mean\_error 에 저장한다.
- 선택한 법인, 선택한 품목에 대한 정보를 Price 모델을 통해 가져와 this\_info 로 저장한다. 이 때, this\_info 의 실제평균값과 예측값 차이를 this\_error 에 저장한다.
- This\_error 가 mean\_error 보다 크면 이상값이므로 is\_error 에 True 를 대입한다.

## ③ setting\_data(request) 함수

update\_data.csv 를 읽어서 Price 데이터베이스에 저장시키고, LSTM model 을 가져와 update\_data.csv 에서 마지막 저장일을 기준으로 다음날 예측 값을 구한다. 구한 예측 값은 Predict 데이터베이스에 저장한다.

## [setting\_date 구현 코드 (1)]

```

def setting_data(request): # AI모델에서 학습된 데이터 가져오기 + DB에 저장
    # DB에 Price 저장
    for i in range(datasize):
        with open("./Fintechapp/csvdata/update_data{}.csv".format(i), 'r', encoding="UTF-8") as f:
            dr = csv.DictReader(f)
            s = pd.DataFrame(dr)
            ss = []
            for j in range(len(s)):
                st = (s["DELNG_DE"][j], s["MRKT_NM"][j], s["CPR_NM"][j], s["PRDLST_NM"][j], s["SPCIES_NM"][j],
                    s["GRAD"][j], s["weight"][j], s["PRI_MAX"][j], s["PRI_MIN"][j], s["PRI_AVE"][j], s["PRI_PRED"][j])
                ss.append(st)
                Price.objects.create(DELNG_DE=ss[j][0], MRKT_NM=ss[j][1], CPR_NM=ss[j][2], PRDLST_NM=ss[j][3],
                                    SPCIES_NM=ss[j][4], GRAD=ss[j][5], weight=ss[j][6], PRI_MAX=ss[j][7], PRI_MIN=ss[j][8],
                                    PRI_AVE=ss[j][9], PRI_PRED=ss[j][10])

    # DB에 Predict 저장
    data = []
    x_shift_data = [] # 평균가가 shift된 입력 데이터 (input)
    x_input_data = []
    model = []
    x_sc = []
    y_sc = []
    good = []
    company = []
    ori_data = []
    for i in range(datasize):
        ori_data.append(pd.read_csv('./Fintechapp/csvdata/update_data{}.csv'.format(i)))
        model.append(tf.keras.models.load_model('./Fintechapp/ai_model/LSTM-model{}_adam.h5'.format(i)))
        x_sc.append(load(open('./Fintechapp/ai_model/x_sc{}.pkl'.format(i), 'rb')))
        y_sc.append(load(open('./Fintechapp/ai_model/y_sc{}.pkl'.format(i), 'rb')))
        data.append(ori_data[i].sort_values(by=['DELNG_DE'])[['DELNG_DE', 'PRI_AVE']])

    for i in range(datasize):
        x_shift_data.append(ori_data[i].groupby('DELNG_DE', as_index=False)['PRI_AVE'].mean()) # 날짜별 평균값 리스트
        x_shift_data[i]['PRI_AVE'] = round(x_shift_data[i]['PRI_AVE'], 2)

```

#### [setting\_data 구현 코드 (2)]

```

    for i in range(datasize):
        temp = x_shift_data[i][-10:]['PRI_AVE'].tolist()
        temp.reverse()
        x_input_data.append(np.array(temp).reshape(1, -1))

    # 다음날 예측값 가져오기 + DB에 저장
    next_predict = [] # 내일 예측가(법인전체) : 청양고추, 새송이
    for i in range(datasize):
        x_data_sc = x_sc[i].transform(x_input_data[i])
        next_predict.append(y_sc[i].inverse_transform(model[i].predict(x_data_sc))[0][0])

    Predict.objects.create(price0=next_predict[0].item(), price1=next_predict[1].item())

    return render(request, 'index.html')

```

#### ④ update(request) 함수

백그라운드에서 실행되는 비동기 스케줄러를 선언하고, 스케줄러에 job 과 alarm\_bell 함수를 추가시킨다. job 은 1 분마다, alarm\_bell 은 매일 13 시에 실행된다. job 함수를 통해 api 로 새로운 데이터를 데이터베이스에 저장하고 그 데이터를 이용하여 다음날 예측값을 업데이트 시킨다.

## [update 함수 구현 코드]

```

189 ### 스케줄러 등록
190 def update(request):
191     sched = BackgroundScheduler(timezone="Asia/Seoul")
192
193     # 데이터 업데이트 스케줄러
194     sched.add_job(job, 'cron', second="0", id="job")
195
196     # 알람 보내기 스케줄러
197     sched.add_job(alarm_bell, 'cron', second="0", id="alarm_bell") # 매일 13시마다 실행. test는 매분마다
198     # sched.add_job(alarm_bell, 'cron', hour="13", id="alarm_bell")
199
200     sched.start()
201
202     return render(request, 'finish.html')
203

```

## ⑤ job () 함수

백그라운드에서 실행시키는 함수다. API 를 통해 오늘날짜를 호출 인자로 데이터를 요청한다. 이때 오늘 거래 데이터가 있으면 데이터를 수집한다

## [job 함수 구현 코드 (1)]

```

def job():
    print("job 실행!")
    updateCheck = 0
    ori_data = []
    ori_data = []

    # 법인-품목 별로 오늘 데이터 있는지 API 확인
    for i in range(datasize):
        ori_data.append(pd.read_csv('./Fintechapp/csvdata/update_data{}.csv'.format(i)))
        update_data.append(ori_data[i])
        for j in range(len(company_list)):
            marketname = market_list[j] # 도매시장명
            coname_c = company_list[j] # 법인명
            mclassname = good_list[i]['PRODST_NM'] # 품목
            sclassname = good_list[i]['SPECIES_NM'] # 품목
            gradename = good_list[i]['GRAD_NO'] # 등급 번호 (특 - 11, 상 - 12, 보통 13)
            grade = "특"
            marketname_info = "%APBLNG_WHISAL_MKTY_NM%" + marketname
            coname_info_c = "%CPR_NM%" + coname_c
            mclassname_info = "%APROLST_NM%" + mclassname
            sclassname_info = "%SPECIES_NM%" + sclassname
            gradename_info = "%GRAD_CD%" + gradename

            df_data = []
            today = date.today().strftime("%Y%m%d")
            url = "http://211.237.50.150:7800/openapi/d60c7fa3f4501f62dbd2ca000625a2d12b764928e9a2cd513dfac4d628991c5/json/GrId_20180118000000000501_1/1/1000?"
            date_info = "%DELNG_DE%" + today
            url = url + date_info + mclassname_info + sclassname_info + marketname_info + coname_info_c + gradename_info
            url_encoded_n = quote(url, safe=':/=&')
            response = requests.get(url_encoded_n)
            contents = response.text
            data = json.loads(contents)
            stat = data['GrId_20180118000000000501_1']['row']
            if stat == 0:
                print("no data") # 오늘 데이터가 아직 없는 경우
                continue
            else:
                df_data.append(stat) # 오늘 데이터가 있는 경우

```

데이터 베이스를 조회 후 방금 수집 데이터가 이미 데이터 베이스에 저장됐다면, 저장하지 않는다. 해당 데이터가 없다면 데이터 베이스에 저장한다.

## [job 함수 구현 코드 (2)]

```

# DB에 저장된 법인-품목이 있으면, # DB에 Predict 저장
if updateCheck:
    data = []
    x_shift_data = [] # 평균가가 shift된 입력 데이터 (input)
    x_input_data = []
    model = []
    x_sc = []
    y_sc = []
    good = []
    company = []
    for i in range(datasize):
        ori_data.append(pd.read_csv('./Fintechapp/csvdata/update_data{}.csv'.format(i)))
        model.append(tf.keras.models.load_model('./Fintechapp/ai_model/LSTM-model{}_adam.h5'.format(i)))
        x_sc.append(load(open('./Fintechapp/ai_model/x_sc{}.pkl'.format(i), 'rb')))
        y_sc.append(load(open('./Fintechapp/ai_model/y_sc{}.pkl'.format(i), 'rb')))
        data.append(ori_data[i].sort_values(by=['DELNG_DE', 'PRI_AVE']))

    for i in range(datasize):
        x_shift_data.append(ori_data[i].groupby('DELNG_DE', as_index=False)['PRI_AVE'].mean()) # 날짜별 평균값 리스트
        x_shift_data[i]['PRI_AVE'] = round(x_shift_data[i]['PRI_AVE'], 2)

    for i in range(datasize):
        temp = x_shift_data[i][-10:]['PRI_AVE'].tolist()
        temp.reverse()
        x_input_data.append(np.array(temp).reshape(1, -1))

    # 다음날 예측값 가져오기 + DB에 저장
    next_predict = [] # 내일 예측값(법인전체) : 청양고추, 새송이
    for i in range(datasize):
        x_data_sc = x_sc[i].transform(x_input_data[i])
        next_predict.append(y_sc[i].inverse_transform(model[i].predict(x_data_sc))[0][0])

    Predict.objects.create(price0=next_predict[0].item(), price1=next_predict[1].item())

```

## 5-(2)-2 웹으로 예측가 및 법인 별 가격 비교 확인하기

관리자가 웹을 설정하는 부분	사용자가 웹에서 사용하는 기능
<ul style="list-style-type: none"> <li>● 최초 한 번 데이터베이스를 생성한다.</li> <li>● 서버를 새로 실행할 때마다 DB 를 업데이트하고 알림을 보낸다.</li> </ul>	<ul style="list-style-type: none"> <li>● (법인-품목)별로 페이지 확인</li> <li>● 메인 홈페이지</li> <li>● 선택한 (법인-품목)에 대해 오늘 실제 가격과 예측 가격을 막대 그래프로 확인</li> <li>● 선택한 (법인-품목)에 대해 오늘 거래 정보와 이상 여부 확인</li> <li>● 선택한 (법인-품목)에 대해 다음 날 예측 가격 확인</li> <li>● 법인 별 최근 8 일의 가격 정보 선그래프 확인</li> <li>● 법인 별 오늘 가격 데이터 막대그래프로 확인</li> <li>● 선택한 (법인-품목)에 대해 전체 거래 목록 확인</li> </ul>

## 5-(2)-3 실제 동작확인

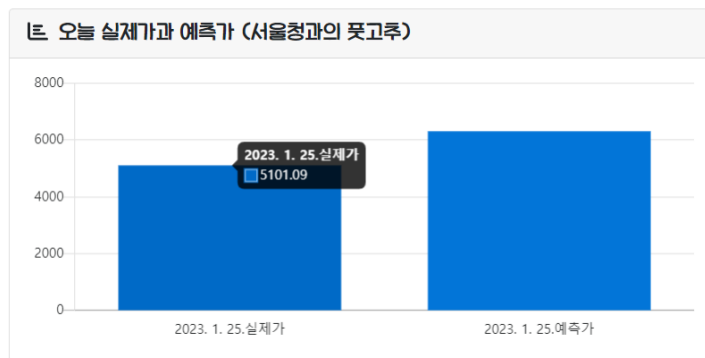
1. 메인 페이지 접속한다. ([중앙청과/풋고추-청양-특]이 기본으로 선택되어 있음)



2. 원하는 [법인-품목] 선택 후 검색 버튼을 누르면, 그에 해당하는 거래 데이터 및 그래프를 확인할 수 있다.



(1) 선택한 [법인-품목]에 대해 오늘 실제 가격과 예측 가를 막대그래프 확인할 수 있다.

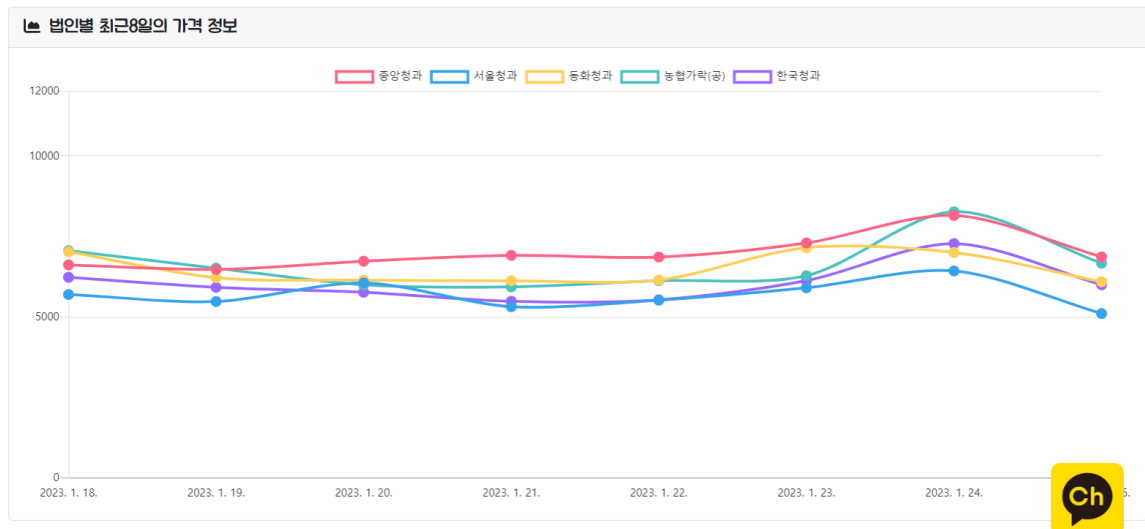


(2) 선택한 [법인-품목]에 대해 오늘 예측가와 실제 거래 평균가, 최고가, 최저가를 확인할 수 있고, 예측가와 실제가의 편차 값을 확인할 수 있다. 만약 편차 값이 이상치를 넘었을 경우, 이상 감지 경고문이 보인다.

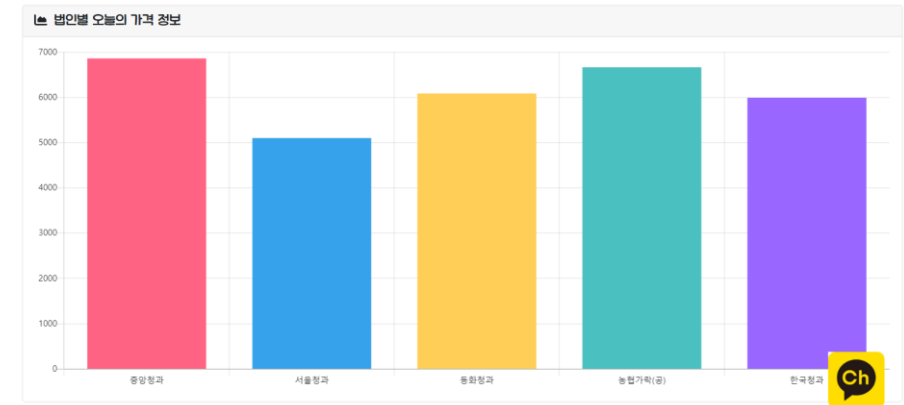


(3) 선택한 품목에 대해 모든 법인의 최근 8 일 가격 정보와 경향을 선그래프 확인할 수 있다.





(4) 선택한 품목에 대해 모든 법인의 오늘 실제 가격을 막대그래프로 확인할 수 있다.



(5) 선택한 [법인-품목]에 대해 전체 거래 목록을 확인할 수 있다.

중앙청과의 풋고추 거래 목록 전체 확인

10 entries per page

Search...

날짜	시장	법인	품목	품질	무게	최고가	최소가	평균가	예측가
2021-01-15	서울가락도매	중앙청과	풋고추	특	1KG	5400.00	3300.00	4854.55	3797.00
2021-01-16	서울가락도매	중앙청과	풋고추	특	1KG	6350.00	3750.00	5623.56	4080.33
2021-01-18	서울가락도매	중앙청과	풋고추	특	1KG	8500.00	5950.00	7810.75	4486.39
2021-01-20	서울가락도매	중앙청과	풋고추	특	1KG	8000.00	5000.00	6667.66	5177.82
2021-01-21	서울가락도매	중앙청과	풋고추	특	1KG	7500.00	2250.00	6503.65	5582.29
2021-01-22	서울가락도매	중앙청과	풋고추	특	1KG	8600.00	5150.00	7779.03	5697.98
2021-01-23	서울가락도매	중앙청과	풋고추	특	1KG	9550.00	3900.00	7847.35	5989.68
2021-01-25	서울가락도매	중앙청과	풋고추	특	1KG	9500.00	3000.00	8544.29	6256.22
2021-01-27	서울가락도매	중앙청과	풋고추	특	1KG	8350.00	3700.00	7020.83	6251.12
2021-01-28	서울가락도매	중앙청과	풋고추	특	1KG	9300.00	6100.00	8241.95	6191.12

Showing 1 to 10 of 484 entries

1 2 3 4 5 6 7 ... 49 >

## 5-(3) 카카오 봇을 통한 메일 알림 기능

- ① 사용자는 카카오채널 친구추가로 간편하게 알람 설정이 가능하다.
- ② AI 모델이 제시하는 적정가와 실제 가격의 비교 후 이상 가격이 발생했을 때 사용자에게 자동으로 메일 알람을 준다.

### 1. 카카오 봇으로 알람 설정하기

#### 1-1 함수 구현

##### (1) View.py 의 alarm\_setting()

```
@csrf_exempt
def alarm_setting(request):
    if request.method == "POST":
        # [봇시스템 -> 웹서버] 받은 json을 decode 후, 파이썬 객체로 변환
        answer = ((request.body).decode('utf-8'))
        return_json_str = json.loads(answer)

        i = return_json_str['action']['clientExtra']['company']          # user가 선택한 법인
        j = return_json_str['action']['clientExtra']['class']          # user가 선택한 품목
        user_email = return_json_str['action']['detailParams']['email']['value']  # user의 email

        # 이메일 유효성 검사
        email = re.compile('[a-zA-Z0-9+-._.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\.$')
        result = email.match(user_email)

        text = ""
        if result is None:
            text = "올바른 이메일이 아닙니다. 처음으로 돌아갑니다."
        else:
            # 해당 이메일을 가진 user에서 알람받을 (법인, 품목) 체크
            text = "알림 설정 완료!"
            try:
                # 기존 회원
                thisuser = KakaoUser.objects.get(user_email = user_email)
                fn_checkbox(thisuser, int(i), int(j))
            except:
                # 신규 회원
                thisuser = KakaoUser.objects.create(user_email = user_email)
                fn_checkbox(thisuser, int(i), int(j))

        # [웹서버 -> 봇서버] 보낼 json
        context = {
            "version": "2.0",
            "template": {
                "outputs": [
                    {
                        "simpleText": {
                            "text": text
                        }
                    }
                ]
            }
        }
```

- ① 카카오 봇 시스템 에서 -> 웹 서버로 POST 요청을 보낸다.
- ② 웹 서버는 json 을 받는다. 선택한 법인, 선택한 품목, 이메일만 뽑아낸다.
- ③ 웹 서버는 이메일 유효성을 검사한다. 알람 설정을 한 적이 있어서 KakaoUser 모델에 이미 등록된 회원이라면 get 메소드로 해당 object 에 적용시키고, KakaoUser 모델에 등록되지 않았던 회원이라면 create 메소드로 object 를 생성 후 적용시킨다. 이 때, fn\_checkbox 함수를 이용하여 적용된다.
- ④ 웹서버에서 -> 카카오 봇 시스템으로 json 을 보내어 response 응답한다.

##### (2) Model.py 의 KakaoUser

```
class KakaoUser(models.Model):

    user_email = models.EmailField(unique=True)

    alarm_0_0 = models.BooleanField(default=False) # 중앙청과 - 청양고추
    alarm_1_0 = models.BooleanField(default=False) # 서울청과 - 청양고추
    alarm_2_0 = models.BooleanField(default=False) # 동화청과 - 청양고추
    alarm_3_0 = models.BooleanField(default=False) # 농협가락(공) - 청양고추
    alarm_4_0 = models.BooleanField(default=False) # 한국청과 - 청양고추

    alarm_0_1 = models.BooleanField(default=False) # 중앙청과 - 새송이
    alarm_1_1 = models.BooleanField(default=False) # 서울청과 - 새송이
    alarm_2_1 = models.BooleanField(default=False) # 동화청과 - 새송이
    alarm_3_1 = models.BooleanField(default=False) # 농협가락(공) - 새송이
    alarm_4_1 = models.BooleanField(default=False) # 한국청과 - 새송이
```

## ① object

☐ KAKAO USER

☐ tngus4789@naver.com

☐ cgoya01@naver.com

2 kakao users

## ② 필드 구성

User email:

☒ Alarm 0 0

☐ Alarm 1 0

☒ Alarm 2 0

☐ Alarm 3 0

☐ Alarm 4 0

☒ Alarm 0 1

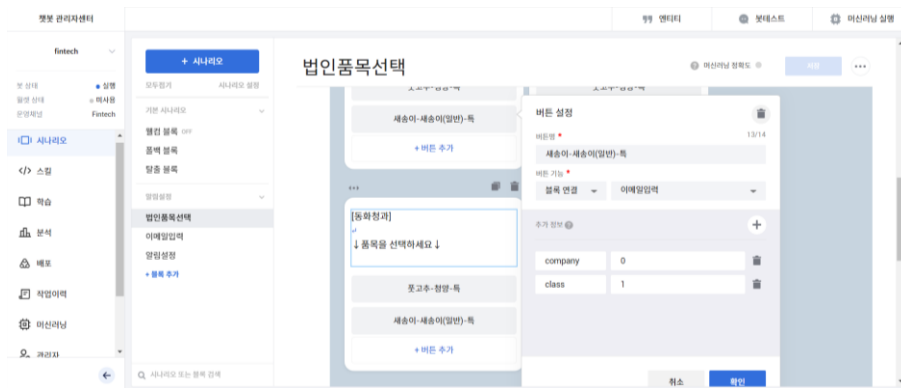
☐ Alarm 1 1

☐ Alarm 2 1

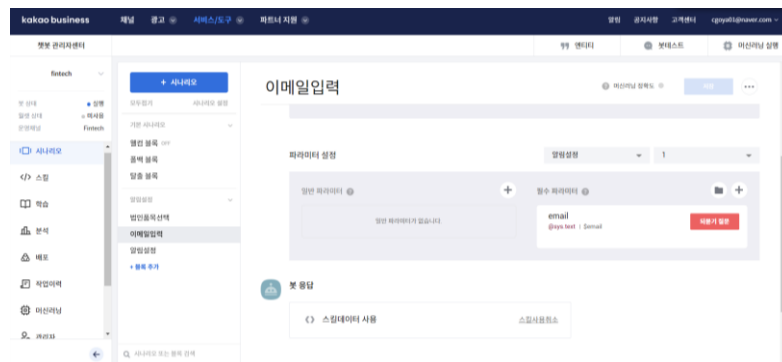
☐ Alarm 3 1

☐ Alarm 4 1

## (3) 카카오 i 오픈빌더



→ 웹서버에 json 전송 시, 선택한 company와 class에 대한 정보를 함께 전송한다.



→ 웹서버에 json 전송 시, 입력한 email를 함께 전송

## 1-2. 실제 동작 확인

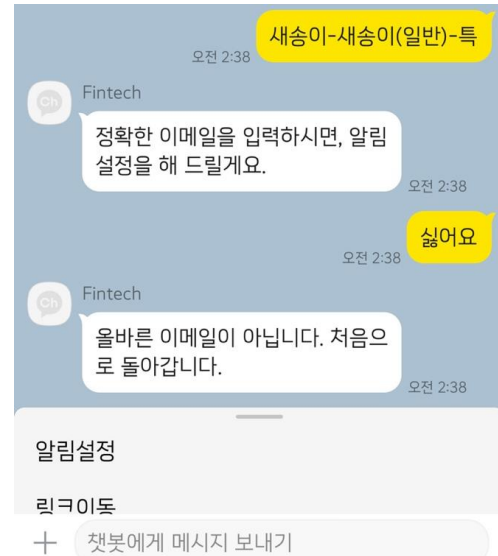
1. 홈페이지 오른쪽 하단에 '카카오채널' 아이콘 클릭해 Fintech 봇을 친구추가 한다.



2. '알림설정'을 누른 뒤 원하는 [법인-품목]을 선택하고 알림 받을 메일을 입력한다.



알림 설정 성공 시 화면



알림 설정 실패 시

## 2. 이상 발생 시 메일 보내기

### 2-1 함수 설명

#### (1) Url.py

서버를 실행시키고 최초 한 번 <https://pnu-fintech.run.goorm.app/update> 한다.

```
path('update', Fintechapp.views.update, name="update"),
```

#### (2) View.py - update()

매일 13 시마다 alarm\_bell()을 호출한다. (현재는 test 용으로 매분마다 호출)

```

### 스케줄러 등록
def update(request):
    sched = BackgroundScheduler(timezone="Asia/Seoul")

    # 데이터 업데이트 스케줄러
    sched.add_job(job, 'cron', second="0", id="job")

    # 알람 보내기 스케줄러
    sched.add_job(alarm_bell, 'cron', second="0", id="alarm_bell") # 매일 13시마다 실행. test는 매분마다
    # sched.add_job(alarm_bell, 'cron', hour="13", id="alarm_bell")

    sched.start()

    return render(request, 'finish.html')

```

### (3) View.py - alarm\_bell()

```

### 이상 발생 시 알림 메일 발송 기능
def alarm_bell():
    print("alarm 실행!")

    for i, good in enumerate(good_list): # 각 품목 별로
        # 5개 법인 전체 정보 가져오기
        all_info = Price.objects.filter(PDLST_NM=good["PDLST_NM"], SPCIES_NM=good["SPCIES_NM"], GRAD=good["GRAD"])

        y = [item.PRI_AVE for item in all_info] # 실제 평균값 리스트
        pred = [item.PRI_PRED for item in all_info] # 예측값 리스트
        mean_error = round(mean_absolute_error(y, pred), 2) # 전체 법인의 전체 기간

        # 해당 법인에 대한 가장 최신의 정보 가져오기
        for j, company in enumerate(company_list):
            price_info = Price.objects.filter(CPR_NM=company, PDLST_NM=good["PDLST_NM"], SPCIES_NM=good["SPCIES_NM"], GRAD=good["GRAD"])
            this_info = price_info.last()
            this_error = abs(this_info.PRI_AVE - this_info.PRI_PRED) # (법인-품목)의 예측값과 실제값 편차

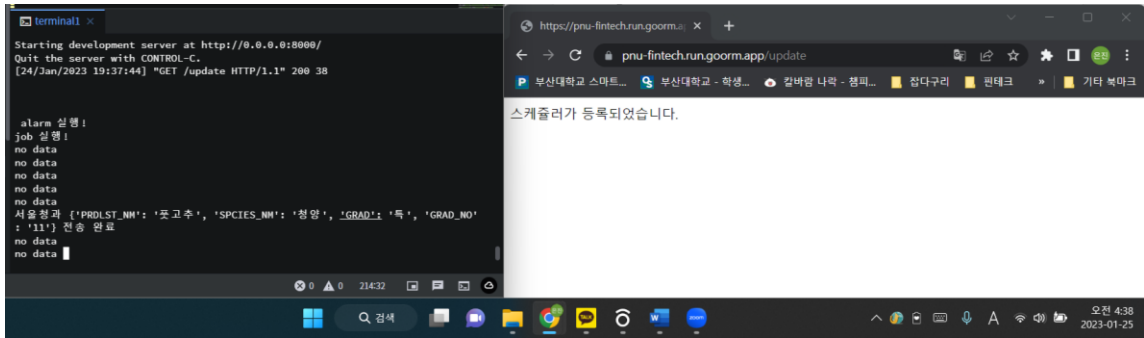
        # 이상값 시 (실제값-예측값 편차가 기준 이상일 때)
        if this_error > mean_error :
            # 알림설정된 사용자가 있으면 메일 전송
            users = fn_search(i, j)
            if users is not None:
                subject = this_info.DELNG_DE + " " + this_info.CPR_NM + " " + this_info.PDLST_NM + "-" + this_info.SPCIES_NM + " " + this_info.GRAD + " 이상 발생"
                message = render_to_string('mail.html', {
                    'this_info': this_info, # 마지막으로 업데이트된 날짜의 정보
                    'company_no': i,
                    'class_no': j,
                    'this_error': this_error,
                    'mean_error': mean_error,
                })
                email = EmailMessage(subject, message, to=[user.user_email for user in users])
                email.content_subtype = "html"
                email.send()
            print(company, good, "전송 완료")

```

- ① Price 모델에서 전체 기간, 전체 법인에 대한 품목별 정보를 가져온다. y 에는 실제 평균값을 pred 에는 다음날 예측값을 가져온다.
- ② Price 모델에서 가장 최신 날짜의 법인 별-품목 별 가격 정보를 가져온다.
- ③ [ “당일 실제 평균 값” 과 “당일 예상 값”의 편차 ] > [mean\_absolute(y, pred)(1 년치 전체 법인 편차)의 평균] 일 때 이상가격으로 판별한다.
- ④ 알림 설정한 사용자들을 fn\_search()을 이용하여 KakaoUser 모델에서 찾는다.
- ⑤ 해당 사용자들에게 알람 메일을 전송한다.

## 2-2. 실제 동작 확인

1. 사이트에 접속 후 최초 한 번 <https://pnu-fintech.run.goorm.app/update> 를 입력하여 스케줄러 등록한다. 1 분마다 실행된다.



## 2. 특정 [법인-품목]에 대해 알림을 설정한 사용자에게 이상치 발생 시 메일이 발송된다.

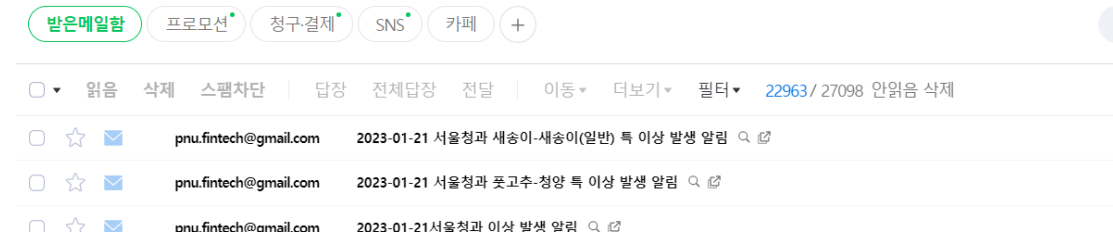


그림 1 이상 발생 시 메일 알림 발송



그림 2 해당 메일 내용

## 6. 제약상황과 발전 가능성

### 6-(1) 알람 구현 측면에서

농산물 생산자의 특징을 고려하여 간편한 카카오톡 채널을 통한 알람 톡으로 구상하였다. 하지만 카카오톡 채널의 알람 톡 기능은 사업자 등록이 되어 있는 사용자에게 한해 카카오톡 공식딜러사를 통한 계약으로만 사용이 가능했다.

따라서 카카오톡채널 대신 메일로 알림을 전송해주는 방식을 선택했다.

이후 사업화 과정에서 사용빈도가 적은 메일보다, 카카오톡채널이나 문자메시지 기능을 사용한다면 사용자가 더 활발하게 정보를 활용하고 편리하게 사용할 것으로 기대된다.

### 6-(2) 모델 구현 측면에서

현재 모델 학습시에 사용하는 입력 데이터는 평균가격을 사용하고 있다. 농산물 가격 선정과 관련이 있을 것으로 유추되는 기온이나 재해, 해외 수급 문제등 다양한 요인 변수를 사용하고 싶었으나 명확한 인과관계를 밝혀 내지 못해 어려움이 있었다.

예측할 날짜와 비교적 근처의 값들을 가중치를 두고 넣고 모델을 학습 시 오히려 성능이 떨어지는 모습을 보여 다양한 입력데이터 사용이 어려웠다.

이후 추가적인 연구를 통해 기온이나 재해, 해외 수급 문제등의 요인과 농산물 경매 가격의 명확한 인과관계를 통해 다양한 입력 값을 사용한다면 학습 모델의 성능을 개선할 수 있을 것으로 기대된다.

## 7. 마무리하며

디지털 사각지대에 놓여있는 농산물 생산자의 정보 불균형을 해결하고, 도매법인 독과점 문제를 해결하길 바란다. 새로운 모델인 법인 별 이상가격 알람 기능이 도매법인의 불법 거래를 근절하는 해결책 되길 바라며 모델을 제시한다.



## 8. 출처 확인

<sup>i</sup> 농민신문 [도매시장 경가/수의 매매 활성화의 맹점] 2021.03.03

<https://www.nongmin.com/334318>

<sup>ii</sup> 농민신문[경매제 vs 시장도매인제…법원 “강서시장, 영업장소 구분해야”] 2020.07.24

<https://www.nongmin.com/359764>

<sup>iii</sup> 한국 농정 [시장도매인제는 시기상조? 오히려 늦었다] 2020.12.13

<http://www.ikpnews.net/news/articleView.html?idxno=42733>

<sup>iv</sup> 원간 원예 [경매제는 땅 짚고 헤엄치기?! 거래 제도의 다양성 확보 지금이 ‘골든타임’]

2020.12.02.<http://www.hortitimes.com/news/articleView.html?idxno=26382>

<sup>v</sup> KBS NEWS [농산물 도매법인 유통 추적!…가격의 반은 ‘농민의 눈물’] 2020.09.30

<https://news.kbs.co.kr/news/view.do?ncd=5015829>

<sup>vi</sup> 서울정보소통광장 [농안법 위반 도매시장법인 행정처분 알림] - 공개구분

공공기관이 보유 · 관리하는 정보는 공개 대상이 된다. 다만, 다음 각 호의 어느 하나에 해당하는 정보는 공개하지 아니할 수 있다.

<https://opengov.seoul.go.kr/sanction/27235242>

<sup>vii</sup> 원간 원예 [경매제는 땅 짚고 헤엄치기?! 거래 제도의 다양성 확보 지금이 ‘골든타임’] 2020.12.02.

<http://www.hortitimes.com/news/articleView.html?idxno=26382>

<sup>viii</sup> [농산물도매시장의 시장구조와 효율성 간의 관계분석] 한국산학기술학회논문지 2020

<sup>ix</sup> 지다넷 코리아 [고령화된 농업, 디지털 전환방법은?] 2021.10.19

<https://zdnet.co.kr/view/?no=20221019172500>

<sup>x</sup> 도매시장 홈페이지 <https://at.agromarket.kr/index.do>

<sup>xi</sup> 주요 농산물 가격 예측 시스템

<https://www.gyeongnam.go.kr/bigdatafarm/index.es?sid=a1#close>

<sup>xii</sup> 전북중앙[농산물 유통비용 개선 수년째 제자리 걸음] 2022.10.19

<http://www.jjn.co.kr/news/articleView.html?idxno=922307>

<sup>xiii</sup> 원간 원예 [경매제는 땅 짚고 헤엄치기?! 거래 제도의 다양성 확보 지금이 ‘골든타임’] 2020.12.02.

<http://www.hortitimes.com/news/articleView.html?idxno=26382>