# 농산물가격예측모델

2조. CSP

# 01. 프로젝트 개요

# 프로젝트 기획 배경 및 목표

‘농업 예측은 신(新)도 어렵다’

주요 농축산물 연간 소매가격 등락폭 단위:% ※ 전년대비 기준

| 구분 | 2017년 | 2018년 | 2019년 | 2020년 | 2021년 |
|---|---|---|---|---|---|
| 깐마늘(1kg) | -9.1 | -5.1 | -11.8 | -0.2 | 34.4 |
| 배추(1포기) | -6.5 | 5.9 | -12.8 | 49.2 | -24.8 |
| 사과(10개) | -4.1 | 32.0 | -13.6 | 21.1 | 1.2 |
| 삼겹살(1kg) | 6.1 | -7.6 | -4.8 | 15.1 | 10.4 |
| 쌀(20kg) | -5.8 | 29.8 | 8.1 | 2.2 | 13.2 |
| 양파(1kg) | 9.3 | -17.8 | -15 | 45.3 | 17.7 |


**문제인식**

날씨, 기온, 수요공급 예측 어려움에 따른
농산물 가격 폭등과 폭락으로
소비자와 생산자에 양측 피해

▽

**현황파악**

농식품부, aT 센터
기존 경험 기반 재배 위주 □ 데이터 확보 및 예측정교화 추진
최근 도매가격 제공서비스 확대

▽

**목표설정**

2016-2021 농산물 품종별 일별 거래가격을 기반으로,
**공급량에 영향을 줄 수 있는 주요 변수 탐색** 및
수급관리에 도움을 주는 **가격 예측모델 설계**

# 주요 진행과정

주요 진행과정

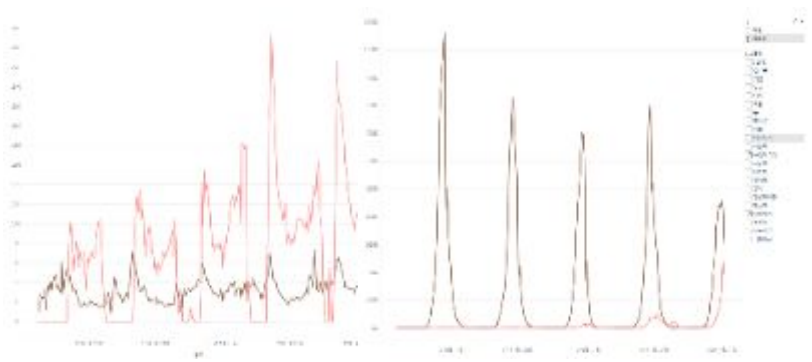
| Plan | Results |
| --- | --- |
| **1. 프로젝트 범위 정의 : 작물 선택(EDA)**<br>- 가격의 변화 추이 및 변동폭 기준 계절성과 상관관계가 높은 품목 우선<br>- 소비량이 많은 대중적 품목 선택 | **'양파' 선택**<br>고려사항 : 사회적 이슈, 대중성, 적은 품종수 |
| **2. 변수에 대한 가설 설정 및 1차 선정**<br>- 선택 품종에 대한 사전 스터디를 통한 가설 변수 설정.<br>- ex. 기온, 강수, 물류가격, 수확량, 수출입량 | **월별 거래가격(y)에 영향을 미치는 요인 탐색**<br>1) 전년도 재배면적<br>2) 10h당 생산량 : 기온, 강수, 일조량<br>3) 추가 공급량 : 수출입량 |
| **3. 데이터 수집 및 전처리**<br>- 품종별 일일 거래량 및 가격 : _농넷<br>- 독립변수용 외부데이터 : 기후정보(기상청), 농식품수출정보 등 | **데이터셋**<br>1) 월별/시도산지별/거래량/거래금액 : 거래가격(y)산출<br>2) 생산량 정보 : 연간 재배면적, 연간 생산성, 총 생산량<br>3) 기후 정보 : 월별 평균기온, 최고기온, 최저기온, 강수량, 일조량<br>4) 월별 양파 수입/수출 물량 |
| **4. 가격예측 모델을 위한 머신러닝 모델 탐색 및 1차 테스트** | **머신러닝**<br>1) 시계열기반 가격 예측 모델<br>: Prophet, ARIMA, SARIMA<br>2) 회귀분석<br>: Multivariate(Sklearn, Statsmodel)<br>Decision Tree, Random Forest 등 |
| **5. 모델 설계, 평가, 검증 반복을 통한 예측 정확도 개선** | **모델 검증 및 활용**<br>1) MSE 최소화를 위한 변수 및 모델 결정<br>2) 2022년 양파 도매가격 예측치 산출<br>3) 시사점 및 추후 보완사항 산출 |

# 작물 선택 : 양파



## 포도

[ 단위당 가격]



[거래량]

## 양파

[ 단위당 가격]

[거래량]

| | | |
|---|---|---|
| 장점 | 다양한 품종 - 세부주제 가능 | 적은 품종수 - 분석에 용이 |
| | 화제성/대중성 - 여름 최대 소비농산물 | 거래량 일관성 대비, 가격의 폭등/폭락 이슈 존재 |
| | 여름 집중 소비 - 거래량 최대 | 생산지역이 포괄적이고, 영세농가가 많아 정보유의성 有 |
| 단점 | 샤인머스캣 고비중 - 2019년 본격화 | 노지/하우스재배, 수확 후 1년 저장 등 환경변수 고려 필요 |
| | 품종별 가격 차이로 인한 거래비중에 따른 평균가격 왜곡 | |
| | 비닐하우스로 기온 등에 영향을 적게 받음 | |
| 기타 | 수입량 포함한 공급량 반영 필요 | |

# 도메인 탐색 및 변수 설정

1. EDA
   - i. 작물 선택
   - ii. 도메인 탐색
2. 변수 설정
3. 데이터셋
4. 모델링(1) – 시계열분석
   - i. Prophet
   - ii. ARIMA, SARIMA
5. 모델링(2) – 회귀분석
   - i. Multivariate Regression
   - ii. Decision Tree
   - iii. Random Forest
6. 최종 모델 결정
   - i. 변수 검증
   - ii. 모델검증
7. 예측
8. 인사이트 도출
9. 향후 개선 사항

농촌지도와 개발 vol.22.No.4 Journal of Agricultural Extension & Community Development, Vol.22 No.4[December 2015], 423-434
ISSN 1976-3107[print], 2384-3705[online] http://dx.doi.org/10.12653/jecd.2015.22.4.0423

## 양파 출하시기 도매가격 예측모형 연구*

남국현a** · 최영찬b
a 서울대학교 농업생명과학연구원(서울시 관악구 관악로 599)
b 서울대학교 농경제사회학부 지역정보전공(서울시 관악구 관악로 599)

## A Study on Onion Wholesale Price Forecasting Model

Kuk-Hyun Nam[a] · Young-Chan Choe[b]
[a] Research Institute of Agriculture and Life Science, Seoul National University, Korea
[b] Program in Regional Information, Department of Agricultural Economics and Rural Development, Seoul National University, Korea

**Abstract**

This paper predicts the onion's cultivation areas, yields per unit area, and wholesale prices during ship dates by using wholesale price data from the Korea Agro-Fisheries & Food Trade Corporation, the production data from the Statistics Korea, and the weather data from the Korea Meteorological Administration with an ARDL model. By analyzing the data of wholesale price, rural household income and rural total earnings, onion cultivation areas in 2015 are estimated to be 21,035, 17,774 and 20,557(ha). In addition, onion yields per unit area of South Jeolla Province, North Gyeongsang Province, South Gyeongsang Province, Jeju Island, and the whole country in 2015 are estimated to be 5,980, 6,493, 6,543, 6,614, 6,139 (kg/10a) respectively. By using onion production's predictive value found from onion's cultivation areas and yields per unit area in 2015, the onion's wholesale prices in June are estimated to be 780 won, 1,100 won, and 820 won for each model. Predicted monthly price after the onion's ship dates is analyzed to exceed 1,000 won after August.

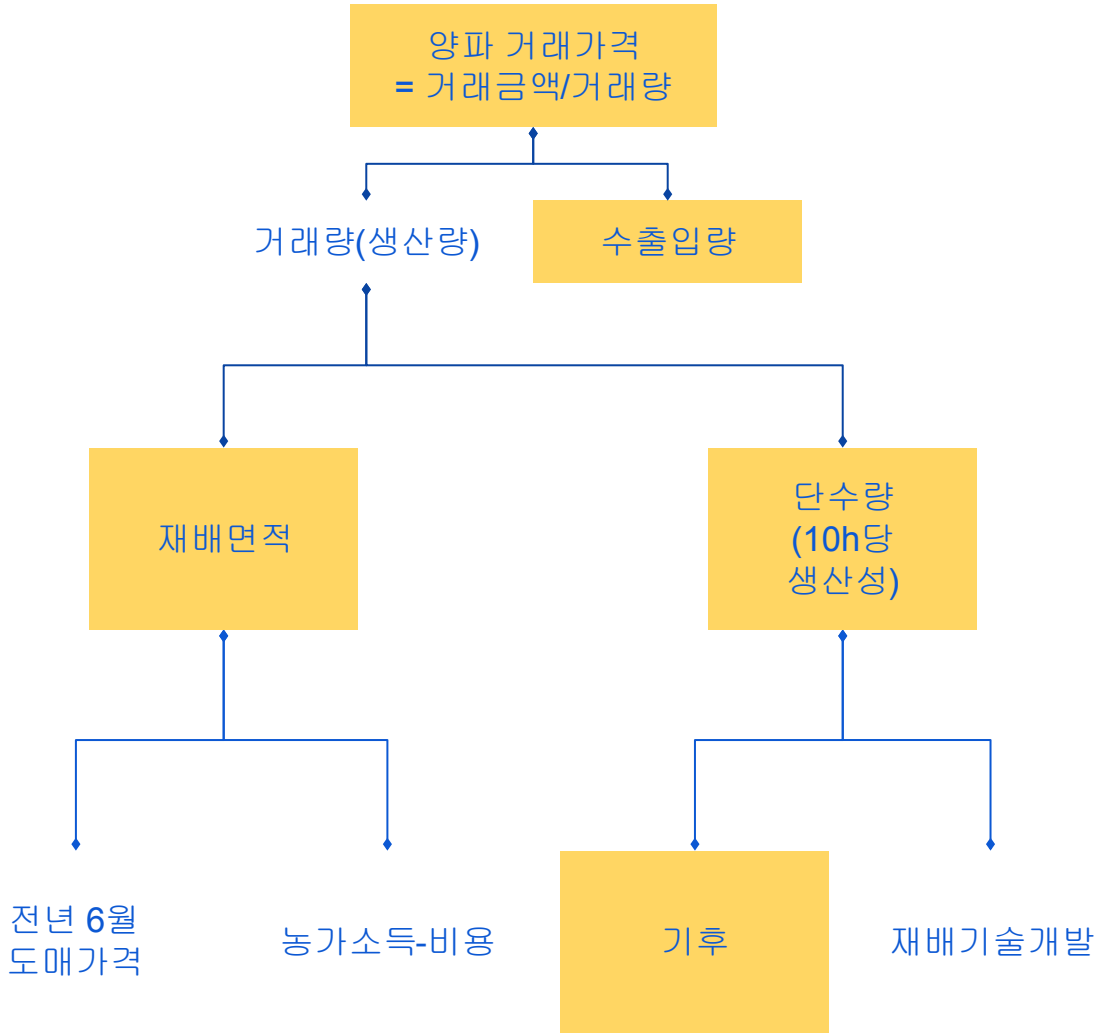
**Key words:** cultivation areas, yields, price, forecasting, onion

## [ Domain Rule ]

### 성공적인 양파 재배를 위한 노하우

| | | |
|---|---|---|
| 8월~10월 | 유묘기 | 발아적정온도 15~25도<br>강수빈도가 높아야 함. |
| 11월~1월 | 활착기 | 최저기온 영하 9도씨이면<br>동해발생 |
| 2월~3월 | 경엽 신장기 | 강수빈도가 높으면<br>습해 발생 |
| 4월~7월 | 구 비대기 | 수확기<br>평균 기온이 25도 이상이면<br>고온 장애 발생 |

## [ 1차 변수 설정 ]



# 02. 프로세싱

# 3. 데이터셋



| | 년월 | 품목 | 품종 | 시장 | 법인 | 광역산지 | 시군산지 | 도매가격(원/kg) | 거래량(톤) | 거래금액(백만원) |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2021. 10 | 양파 | 양파(일반) | 대구북부도매 | 대양청과 | 경북 | 영천 | 793.000000 | 40.00 | 31.72 |
| 1 | 2021. 10 | 양파 | 양파(일반) | 대구북부도매 | 효성청과 | 경북 | 의성 | 803.030303 | 31.68 | 25.44 |
| 2 | 2021. 10 | 양파 | 양파(일반) | 대구북부도매 | 대양청과 | 경남 | 합천 | 769.686727 | 24.26 | 18.67 |
| 3 | 2021. 10 | 양파 | 양파(일반) | 대구북부도매 | 대양청과 | 대구 | 대구중구 | 735.000000 | 15.20 | 11.17 |
| 4 | 2021. 10 | 양파 | 양파(일반) | 대구북부도매 | 효성청과 | 경북 | 영천 | 843.434343 | 7.92 | 6.68 |

2014년 1월 ~ 2021년 10월
월별 도매가격
출처: 농넷

| | 시점 | 시도별 | 양파:면적 (ha) | 10a당 생산량 (kg) | 생산량 (톤) |
|---|---|---|---|---|---|
| 0 | 2013 | 서울특별시 | 0 | 0 | 0 |
| 1 | 2013 | 부산광역시 | 23 | 8265 | 1901 |
| 2 | 2013 | 대구광역시 | 81 | 6909 | 5596 |
| 3 | 2013 | 인천광역시 | 25 | 2294 | 573 |
| 4 | 2013 | 광주광역시 | 58 | 7443 | 4317 |

2013년 ~ 2021년 연도별 재배면적, 생산량 (출처: 통계청)

| | 시군산지 | 기후관측소기준 |
|---|---|---|
| 0 | 무안 | 목포 |
| 1 | 함평 | 영광 |
| 2 | 창녕 | 밀양 |
| 3 | 신안 | 목포 |
| 4 | 합천 | 합천 |

시군산지별 기후관측소 (출처: 기상청)

| | inn | out | year | month |
|---|---|---|---|---|
| 0 | 24407.001 | 1.000 | 16 | 1 |
| 1 | 1417.701 | 26.206 | 16 | 10 |
| 2 | 2110.050 | 10.300 | 16 | 11 |
| 3 | 2619.802 | 4.100 | 16 | 12 |
| 4 | 6681.080 | 0.305 | 16 | 2 |

2016년 1월 ~ 2021년 8월 수입/수출량 (출처: 농넷)

| | weather | AT_1 | AT_2 | AT_3 | AT_4 | AT_5 | AT_6 | AT_7 | AT_8 | AT_9 | AT_10 | AT_11 | AT_12 | HT_1 | HT_2 | HT_3 | HT_4 | HT_5 | HT_6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 강릉 | -0.1 | 0.4 | 5.7 | 13.3 | 17.6 | 20.6 | 25.3 | 25.1 | 20.2 | 15.8 | 7.8 | -0.1 | 4.1 | 5.1 | 9.6 | 18.3 | 21.9 | 24.1 |
| 1 | 강릉 | -0.5 | 1.8 | 7.4 | 10.8 | 18.1 | 21.5 | 27.0 | 28.5 | 20.8 | 15.8 | 8.8 | 3.1 | 4.0 | 6.7 | 12.2 | 15.3 | 23.1 | 25.0 |
| 2 | 강릉 | 2.3 | 1.3 | 8.3 | 14.1 | 20.0 | 20.9 | 26.1 | 23.7 | 20.9 | 15.9 | 10.5 | 1.1 | 6.4 | 5.3 | 12.7 | 18.7 | 24.8 | 24.3 |
| 3 | 강릉 | 2.2 | 3.1 | 8.2 | 12.1 | 20.0 | 21.0 | 23.8 | 25.8 | 20.2 | 16.0 | 9.9 | 4.9 | 6.4 | 7.4 | 13.4 | 16.7 | 25.1 | 25.1 |
| 4 | 강릉 | 0.4 | 1.9 | 7.9 | 14.1 | 19.3 | 21.9 | 24.1 | 25.8 | 20.6 | 15.6 | 9.1 | 5.0 | 4.7 | 6.0 | 12.4 | 19.3 | 24.9 | 25.9 |

2012년 ~ 2021년
각 연도별 12개월의 기온, 강수, 일조량
출처: 기상청

# 3. 데이터셋



| | price | weather | kind | market | corp | wide | city | year | month | area | ratio | amount | inn | out | AT_1 | AT_2 | AT_3 | AT_ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 793.00 | 영천 | 4 | 5 | 23 | 2 | 26 | 21 | 10 | 2365 | 8098 | 191509 | 5213.902 | 7.88 | 1.1 | 3.3 | 8.5 | 12 |
| **1** | 843.43 | 영천 | 4 | 5 | 79 | 2 | 26 | 21 | 10 | 2365 | 8098 | 191509 | 5213.902 | 7.88 | 1.1 | 3.3 | 8.5 | 12 |
| **2** | 725.00 | 영천 | 0 | 31 | 20 | 2 | 26 | 21 | 10 | 2365 | 8098 | 191509 | 5213.902 | 7.88 | 1.1 | 3.3 | 8.5 | 12 |
| **3** | 804.59 | 영천 | 4 | 5 | 23 | 2 | 26 | 21 | 9 | 2365 | 8098 | 191509 | 3378.003 | 2.20 | 1.1 | 3.3 | 8.5 | 12 |
| **4** | 890.58 | 영천 | 4 | 5 | 79 | 2 | 26 | 21 | 9 | 2365 | 8098 | 191509 | 3378.003 | 2.20 | 1.1 | 3.3 | 8.5 | 12 |

| | |
|---|---|
| **X (73)** | kind(품종), market(시장), corp(법인), wide(광역산지), city(시군산지), year(연도), month(월) |
| | area(재배면적), ratio(10a당 생산량), amount(생산량) |
| | inn(수입량), out(수출량) |
| | AT_1~12(평균기온), HT_1~12(최저기온), RAIN_1~12(강수량), SUN_1~12(일조량) |
| **y** | price |

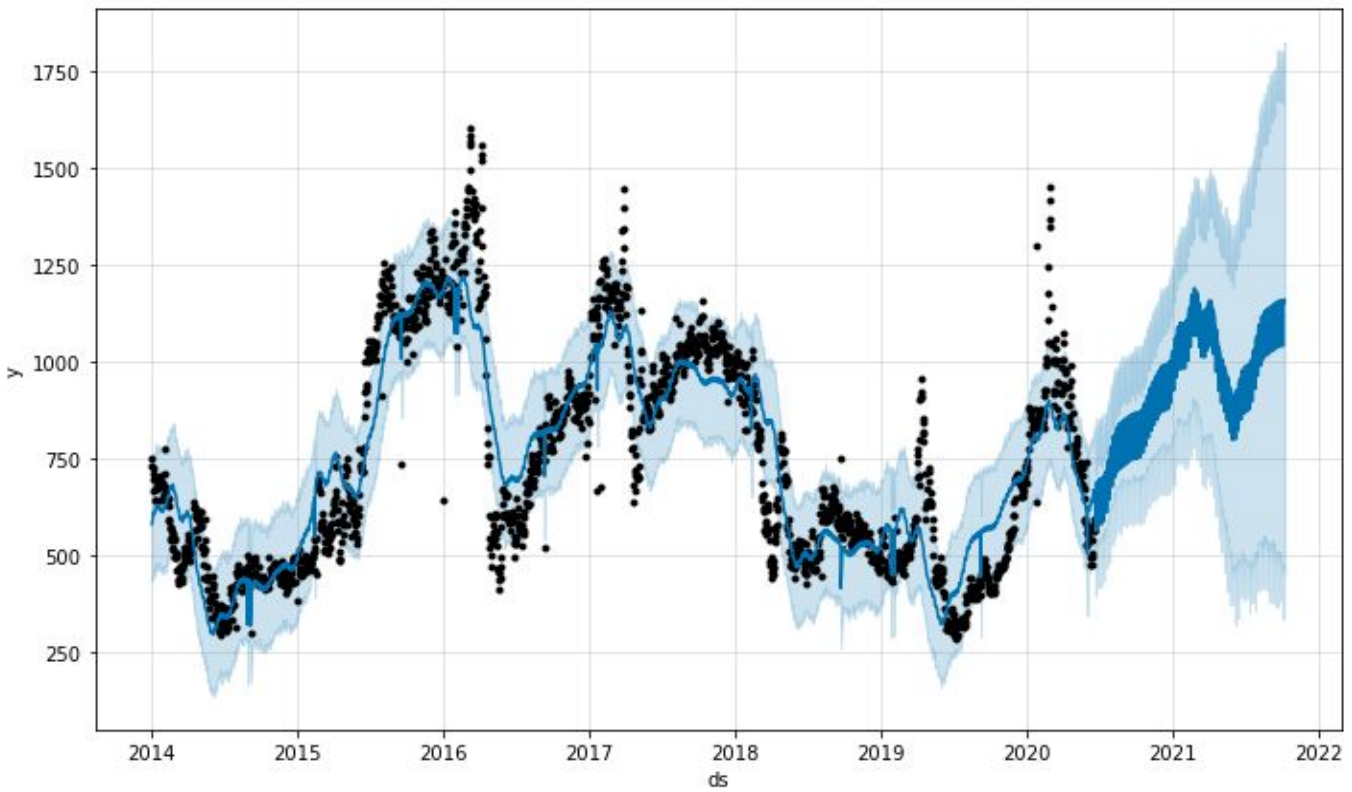
# 시계열분석-Prophet



페이스북에서 공개한 시계열 예측 라이브러리입니다.

prophet 알고리즘의 파라미터는 매우 직관적이기 때문에 시계열 데이터에 대한 지식이 부족하더라도 쉽게 사용할 수 있습니다.

우리가 하려는 작업에 대한 개괄을 파악하기 위해 일 평균 도매가격 데이터만 넣어서 모델을 만들어봤습니다.




| 사용 데이터 :2014년 1월 3일부터 2021년 10월 7일까지<br>일 평균 양파 도매가격 (원/kg) | |
|---|---|
| train data | 2014년 1월 3일 ~ 2020년 6월 19일 |
| test data | 2020년 6월 20일 ~ 2021년 10월 7일 |

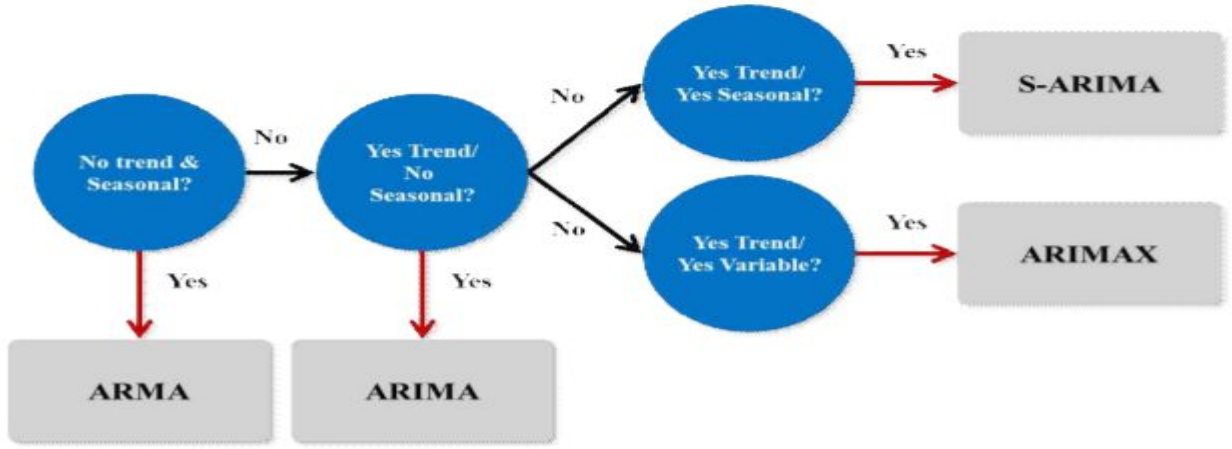
| 결과 | RMSE : 328.1 | NMAE:0.2886 |
|---|---|---|

# 시계열분석-ARIMA, SARIMA



## 시계열 분석 정의:

시계열 데이터는 시간에 따라 기록되는 데이터 로써 시간의 흐름에 영향을 받는다. 시계열 데이터는 4가지 성분으로 구성되어 있는데 경향성, 계절성, 주기는 일정한 패턴을 가지며 나머지 패턴이 없는 성분을 불규칙으로 정의한다.



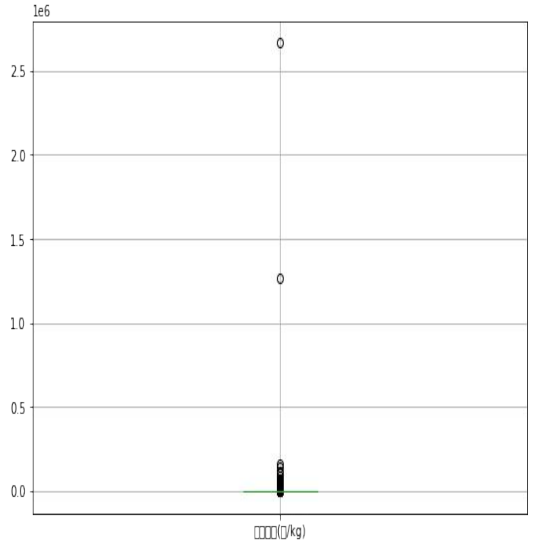

***1. ARIMA 모델 :*** ARMA(AR+MA)모델에 차분을 추가해준 모델을 의미한다. 여기서 차분을 해주는 이유는 바로 비정상성의 데이터를 정상화 시켜주기 위함이다.

***2. SARIMA 모델 :*** SARIMA 모델은 ARIMA 모델에 **계절적 성분이 S(Seasonal)를 추가된 모델**이다. 계절적 성분이기 때문에 주어진 데이터의 주기(m값)를 설정해줌으로써 원하는 SARIMA 모델을 적합시킬 수 있다.

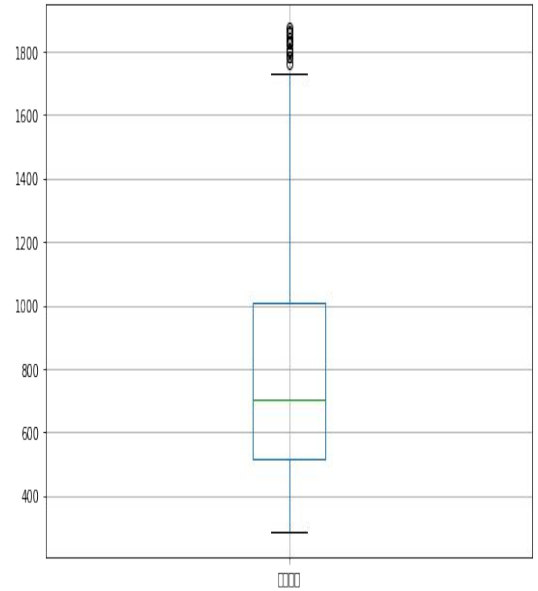
# 시계열분석-ARIMA, SARIMA



## [데이터 전처리 및 이상치 탐지]




이상치 처리




### 이상치 처리

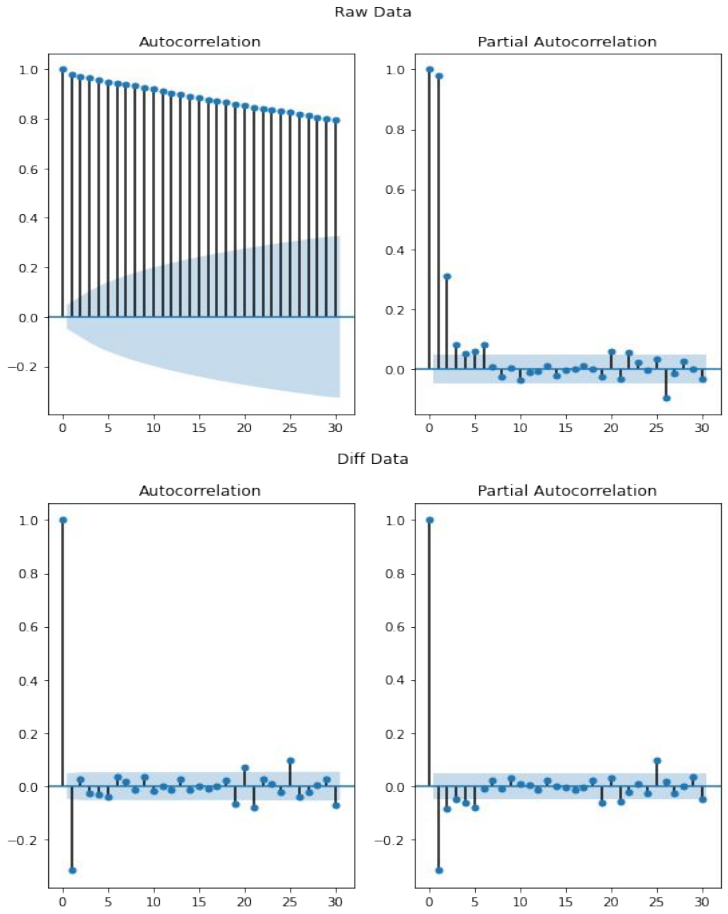
```
[5]
    quartile_1 = DF1['도매가격(원/kg)'].quantile(0.25)
    quartile_3 = DF1['도매가격(원/kg)'].quantile(0.75)
    IQR = quartile_3 - quartile_1
    search_df = DF1[(DF1['도매가격(원/kg)'] < (quartile_1 - 5 * IQR)) | (DF1['도매가격(원/kg)'] > (quartile_3 + 5 * IQR))]
    print(search_df)
```

```
              일자  품목      품종      시장  ...  시군산지    도매가격(원/kg)  거래량(톤)  거래금액(백만원)
787     2021-10-01  양파  양파(일반)    진주도매  ...    진주  1.600000e+04   0.001        0.02
1467    2021-09-18  양파      기타   천안도매  ...  서울중구  3.750000e+03   0.020        0.08
3376    2021-09-03  양파      기타   포항도매  ...    영천  1.000000e+04   0.090        0.90
4581    2021-08-24  양파    자주양파   포항도매  ...    영천  1.375000e+04   0.008        0.11
5306    2021-08-18  양파    저장양파  서울강서도매  ...    함양  2.664769e+06   0.005       13.32
...            ...  ..      ...     ...  ...   ...           ...     ...         ...
210997  2017-02-01  양파      기타   구리도매  ...    구리  3.389300e+03   0.050        0.17
222416  2016-04-02  양파     깐양파   안산도매  ...    서산  3.400000e+03   0.010        0.03
245249  2014-04-09  양파    자주양파  서울가락도매  ...    무안  4.041667e+03   0.070        0.29
245449  2014-04-03  양파    자주양파  서울가락도매  ...    무안  3.631579e+03   0.680        2.48
245487  2014-04-02  양파    자주양파  서울가락도매  ...    함양  3.892535e+03   0.580        2.24

[519 rows x 10 columns]
```

# 시계열분석-ARIMA, SARIMA

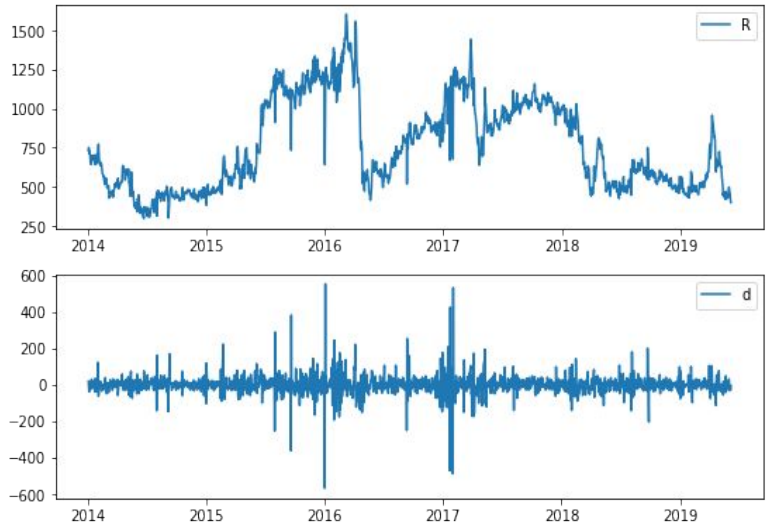


[비정상적 시계열 차분]




1차차분

차분 후 Data



# 시계열분석-ARIMA, SARIMA



[ARIMA 모델 설계 및 예측 시각화]

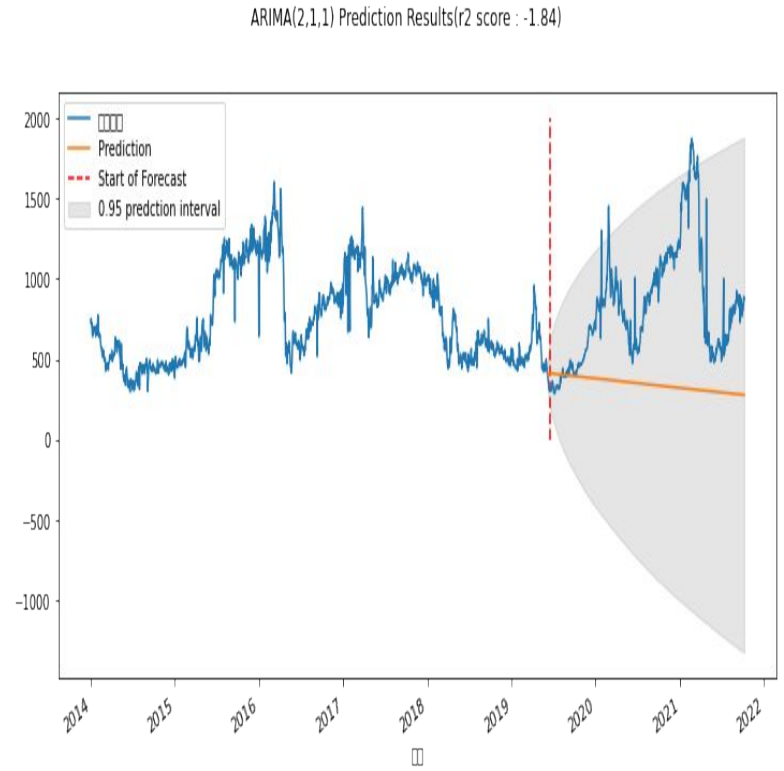
**ARIMA 모델 설계**

- 최적의 파라미터 찾기

```
import itertools
p = range(0,3)
d = range(1,2)
q = range(0,2)
pdq = list(itertools.product(p,d,q))

aic = []
params = []
for i in pdq:
    try:
        model = ARIMA(train_data['도매가격'].values, order=(i))
        model_fit = model.fit()
        print(f'ARIMA : {i} >> AIC : {round(model_fit.aic,2)}')
        aic.append(round(model_fit.aic,2))
        params.append((i))
    except:
        continue
```

```
ARIMA : (0, 1, 0) >> AIC : 18250.51
ARIMA : (0, 1, 1) >> AIC : 18059.21
ARIMA : (1, 1, 0) >> AIC : 18077.35
ARIMA : (1, 1, 1) >> AIC : 18060.82
ARIMA : (2, 1, 0) >> AIC : 18067.76
ARIMA : (2, 1, 1) >> AIC : 18056.15
```

시각화




| | R2 | Corr | RMSE | MAPE |
|---|---|---|---|---|
| 0 | -183.874 | -0.463 | 612.485 | 48925.055 |

| 결과 | RMSE : 612.49 | MAPE : 48925.05 |
|---|---|---|

# 시계열분석-ARIMA, SARIMA



## [SARIMA 모델 설계 및 예측 시각화]

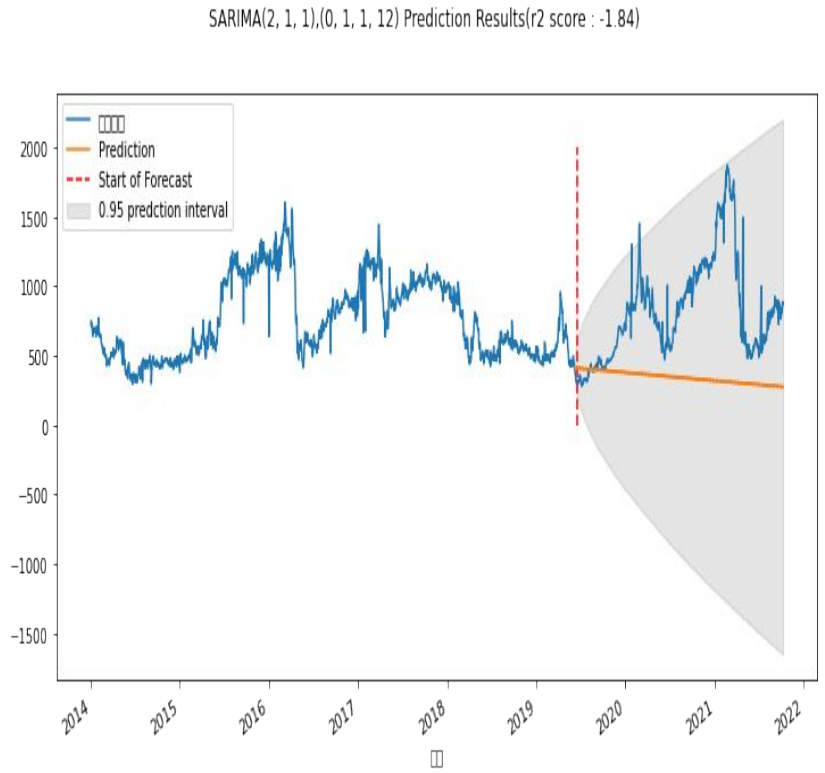
### SARIMA 모형

- SARIMA 모형의 최적화 된 하이퍼 파라미터 설정

```
[ ] %%time

    import itertools
    p = range(0,3)
    d = range(1,2)
    q = range(0,2)
    pdq = list(itertools.product(p,d,q))
    seasonal_pdq = [(x[0],x[1],x[2],12) for x in list(itertools.product(p,d,q))]

    aic = []
    params = []
    for i in pdq:
        for j in seasonal_pdq:
            try:
                model = SARIMAX(train_data['도매가격'].values,order = (i), seasonal_order=(j))
                model_fit = model.fit()
                print(f'SARIMA : {i}{j} >> AIC : {round(model_fit.aic,2)}')
                aic.append(round(model_fit.aic,2))
                params.append((i,j))
            except:
                continue
```

```
SARIMA : (0, 1, 0)(0, 1, 0, 12) >> AIC : 19307.06
SARIMA : (0, 1, 0)(0, 1, 1, 12) >> AIC : 18183.61
SARIMA : (0, 1, 0)(1, 1, 0, 12) >> AIC : 18842.01
SARIMA : (0, 1, 0)(1, 1, 1, 12) >> AIC : 18185.35
SARIMA : (0, 1, 0)(2, 1, 0, 12) >> AIC : 18664.12
```

시각화

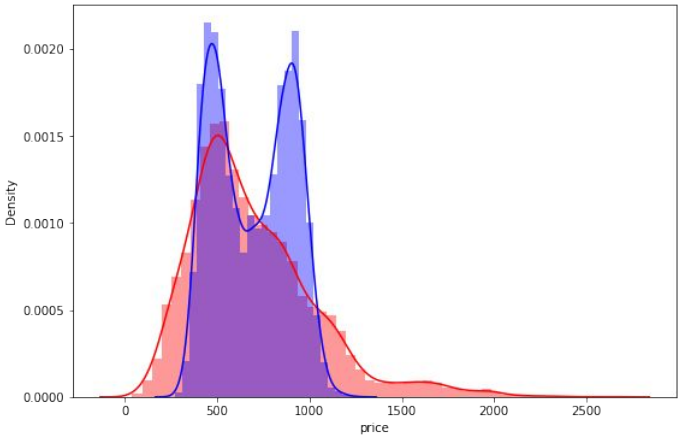



| | R2 | Corr | RMSE | MAPE |
|---|---|---|---|---|
| 0 | -184.19 | -0.462 | 612.827 | 48962.413 |

| 결과 | RMSE : 612.82 | MAPE : 48962.41 |
|---|---|---|

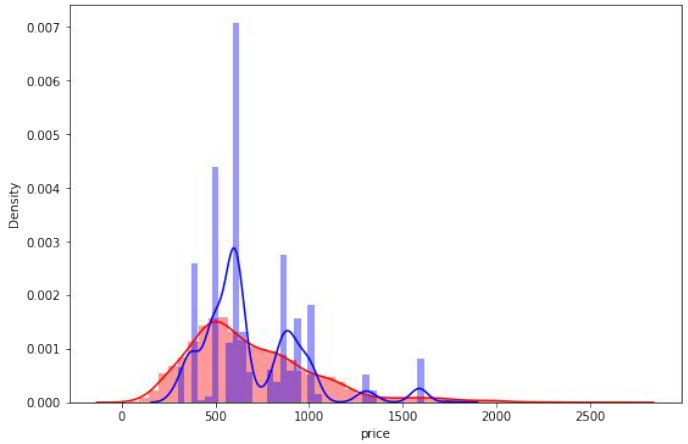
# 회귀분석





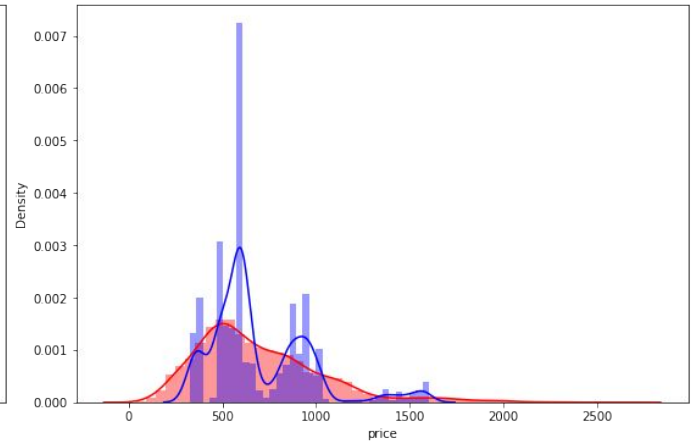

```
from sklearn.linear_model import LinearRegression

RA = LinearRegression()
RA.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```




```
from sklearn.tree import DecisionTreeRegressor

DTR = DecisionTreeRegressor(max_depth = 5,
                            criterion ='mse',
                            random_state = 2045)

DTR.fit(X_train, y_train)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=5,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=2045, splitter='best')
```



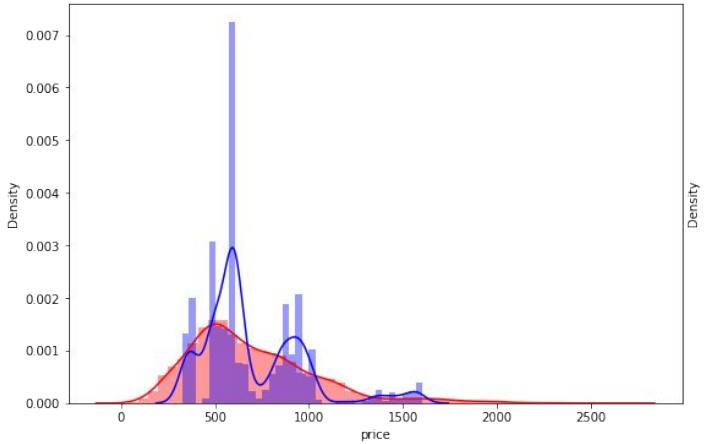

```
RFR1 = RandomForestRegressor(n_estimators = 2000,
                             max_depth = 5,
                             criterion ='mse',
                             n_jobs = -1,
                             random_state = 2045)

RFR1.fit(X_train, y_train)                    ## 2분 23초
```

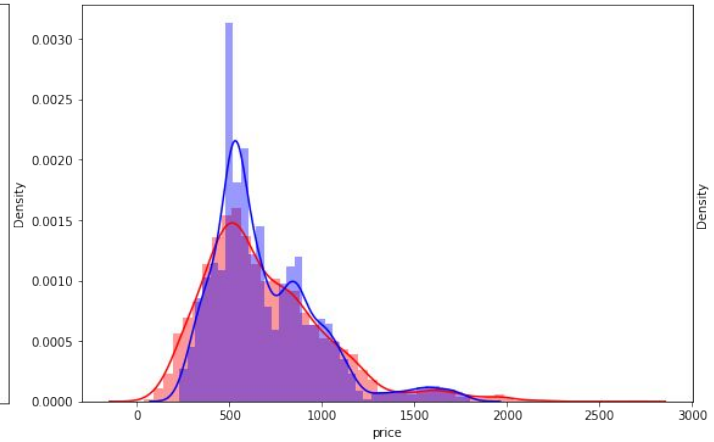
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=5, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=2000, n_jobs=-1, oob_score=False,
                      random_state=2045, verbose=0, warm_start=False)
```

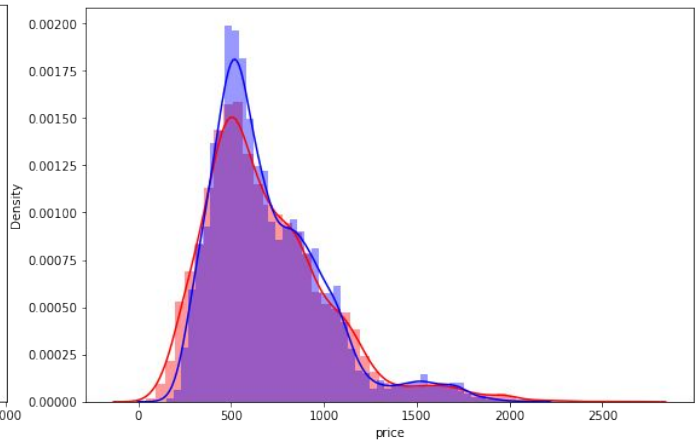
| Linear Regression | Decision Tree | Random Forest |
|---|---|---|
| 234.5 | 204.9 | 200.5 |

# 회귀분석 – Random Forest Regression





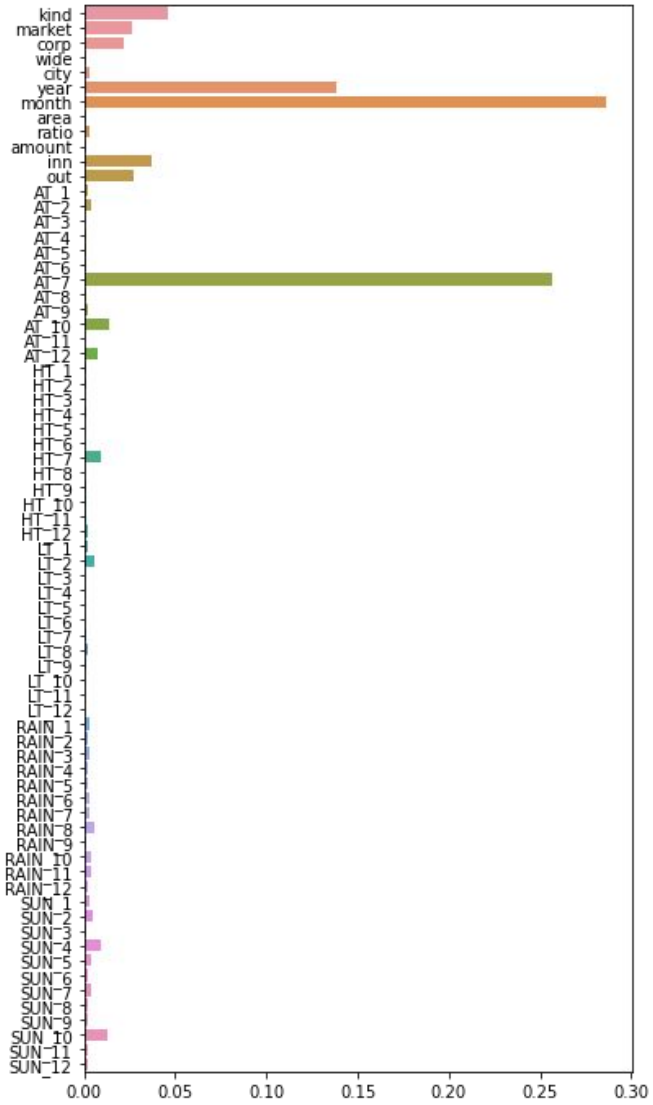




| max_depth = 5 | max_depth = 10 | max_depth = 40 |
|---|---|---|
| 200.5 | 175.5 | 168.2 |

# 최종모델결정 – 변수검증(1)






feature_importance_
가장 높은 'month' 제거


| max_depth = 5 | 200.5 | 215.7 |
|---|---|---|

# 최종모델결정 – 변수검증(2)



| Weight | Feature |
|---|---|
| 75241.6854 ± 2228.8625 | month |
| 52439.3199 ± 3319.0828 | AT_7 |
| 20361.3155 ± 1661.4260 | year |
| 13359.1068 ± 1806.2028 | kind |
| 6609.7944 ± 453.9644 | inn |
| 6390.7759 ± 1030.3814 | market |
| 3214.0278 ± 541.7261 | corp |
| 2758.4503 ± 459.6011 | AT_10 |
| 1416.5728 ± 276.5899 | HT_7 |
| 1359.1029 ± 227.3214 | out |
| 1330.7835 ± 283.3742 | SUN_10 |
| 769.1233 ± 361.6012 | SUN_4 |
| 721.1322 ± 312.6767 | HT_10 |
| 611.6846 ± 212.8931 | SUN_5 |
| 396.5768 ± 249.2848 | LT_2 |
| 328.2557 ± 296.1101 | ratio |
| 327.5650 ± 195.3209 | SUN_2 |
| 322.1483 ± 200.0869 | AT_1 |
| 284.7783 ± 161.8355 | AT_12 |
| 236.0782 ± 186.3299 | RAIN_10 |
| 188.8742 ± 134.0049 | RAIN_3 |
| 174.0529 ± 134.9659 | SUN_11 |
| 126.3135 ± 116.8949 | SUN_1 |
| 121.2153 ± 291.2661 | RAIN_6 |
| 40.5079 ± 163.7455 | city |
| 25.7649 ± 222.8844 | RAIN_7 |
| 7.6689 ± 237.6848 | SUN_7 |
| -46.6684 ± 143.1277 | RAIN_8 |
| -105.0298 ± 126.7190 | wide |
| -133.4306 ± 57.9009 | area |
| -201.5272 ± 90.2259 | amount |

**Permutation Importance** 특정 feature의 데이터를 shuffle 했을 때 모델의 예측 성능이 얼마나 안 좋아졌는지 보여준다.

scoring : neg_mean_squared_error

Weight가 음수라는 건 feature의 데이터를 shuffle 했을 때 모델의 성능이 개선되었다는 뜻입니다. Weight의 편차 범위 전체가 음수인 feature를 주로 제거하는 방식으로 5차에 걸쳐 제거했습니다.

73개 였던 feature의 수가 1차 23개, 2차 11개, 3차 4개, 4차 3개, 5차 2개 제거하여 최종 31개로 줄었습니다.

각 차수마다 교차검증을 했으나 RMSE의 개선이 뚜렷하게 나타나지는 않았습니다.

31개 feature가 남은 모델로 Test를 해본 결과 RMSE는 170이 나왔습니다.

| 결과 | CV RMSE : 203 | test RMSE : 170.4 |
|---|---|---|

# 최종모델결정 – 변수검증(3)



| 변수 구분 | 변수 |
|---|---|
| 생산량 관련 | 재배면적<br>생산성<br>기후 |
| 거래량 관련 | 지역별<br>월별<br>거래량 및 금액<br>수출입물량 |
| y | 거래가격 |

## 총 23791 rows x 62 columns

| | price | weather | area | ratio | AT_1 | AT_2 | AT_3 |
|---|---|---|---|---|---|---|---|
| 0 | 793.00 | 영천 | 2365 | 8098 | 3.3 | 4.2 | 8.2 |
| 1 | 843.43 | 영천 | 2365 | 8098 | 3.3 | 4.2 | 8.2 |
| 2 | 725.00 | 영천 | 2365 | 8098 | 3.3 | 4.2 | 8.2 |
| 3 | 804.59 | 영천 | 2365 | 8098 | 3.3 | 4.2 | 8.2 |
| 4 | 890.58 | 영천 | 2365 | 8098 | 3.3 | 4.2 | 8.2 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 23786 | 819.59 | 여수 | 9521 | 5981 | 2.5 | 4.3 | 9.0 |
| 23787 | 872.83 | 서귀포 | 1566 | 6458 | 6.2 | 7.4 | 11.1 |
| 23788 | 731.67 | 서귀포 | 1566 | 6458 | 6.2 | 7.4 | 11.1 |
| 23789 | 774.76 | 서귀포 | 1566 | 6458 | 6.2 | 7.4 | 11.1 |
| 23790 | 793.19 | 서귀포 | 1566 | 6458 | 6.2 | 7.4 | 11.1 |

23791 rows × 74 columns

### Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor

RFR = RandomForestRegressor(n_estimators = 2000,
                            max_depth = 5,
                            criterion ='mse',
                            n_jobs = -1,
                            random_state = 2045)

RFR.fit(Xc_train, yc_train)          # 2분 6초
```

```
mse = mean_squared_error(yc_test, RFR.predict(Xc_test))
np.sqrt(mse)
```

```
279.69116221815693
```

## 총 23791 rows x 9 columns

| | price | market | corp | wide | city | year | month |
|---|---|---|---|---|---|---|---|
| 0 | 793.00 | 5 | 23 | 2 | 26 | 21 | 10 |
| 1 | 843.43 | 5 | 79 | 2 | 26 | 21 | 10 |
| 2 | 725.00 | 31 | 20 | 2 | 26 | 21 | 10 |
| 3 | 804.59 | 5 | 23 | 2 | 26 | 21 | 9 |
| 4 | 890.58 | 5 | 79 | 2 | 26 | 21 | 9 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 23786 | 819.59 | 5 | 23 | 6 | 22 | 17 | 5 |
| 23787 | 872.83 | 11 | 1 | 8 | 18 | 17 | 5 |
| 23788 | 731.67 | 10 | 30 | 8 | 18 | 17 | 5 |
| 23789 | 774.76 | 10 | 75 | 8 | 18 | 17 | 5 |
| 23790 | 793.19 | 5 | 23 | 8 | 18 | 17 | 5 |

### Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor

RFR = RandomForestRegressor(n_estimators = 2000,
                            max_depth = 5,
                            criterion ='mse',
                            n_jobs = -1,
                            random_state = 2045)

RFR.fit(X_train, y_train)
```

```
mse = mean_squared_error(y_test, RFR.predict(X_test))
np.sqrt(mse)
```

```
197.31304251915336
```

| | | |
|---|---|---|
| 생산량 관련 | RMSE : 279.69 | NMAE: 0.298 |
| 거래량 관련 | RMSE : 197.31 | NMAE: 0.202 |

# 최종모델결정 – 변수검증(4)



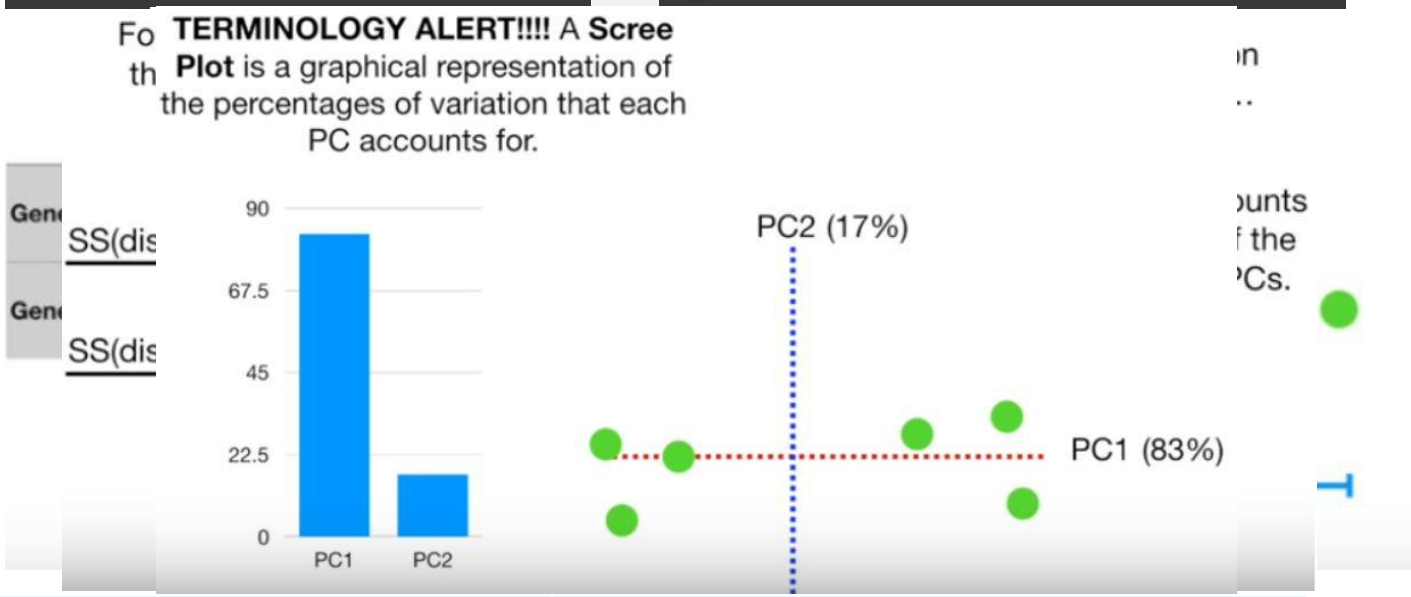
## PCA(주성분 분석)

고차원의 데이터를 저차원의 데이터로 환원시키는 기법을 말한다. 이 때 서로 연관 가능성이 있는 고차원 공간의 표본들을 선형 연관성이 없는 저차원 공간(**주성분**)의 표본으로 변환하기 위해 직교 변환을 사용한다. 데이터를 한개의 축으로 사상시켰을 때 그 분산이 가장 커지는 축을 첫 번째 주성분, 두 번째로 커지는 축을 두 번째 주성분으로 놓이도록 새로운 좌표계로 데이터를 선형 변환한다. 이와 같이 표본의 차이를 가장 잘 나타내는 성분들로 분해함으로써 데이터 분석에 여러가지 이점을 제공한다. 이 변환은 첫째 주성분이 가장 큰 분산을 가지고, 이후의 주성분들은 이전의 주성분들과 직교한다는 제약 아래에 가장 큰 분산을 갖고 있다는 식으로 정의되어있다. 중요한 성분들은 공분산 행렬의 고유 벡터이기 때문에 직교하게 된다.

총 23791 rows x 25 columns → 총 23791 rows x 21 columns

```
[ ] from sklearn.decomposition import PCA

    pca = PCA(n_components = 25)
    pca.fit(X_scaled)

    X_pca = pca.transform(X_scaled)
    print('원본 데이터 형태:', str(X_scaled.shape))
    print('축소된 데이터 형태:', str(X_pca.shape))

    원본 데이터 형태: (23791, 72)
    축소된 데이터 형태: (23791, 25)
```

```
pd.Series(np.cumsum(pca.explained_variance_ratio_))

0     0.247014
1     0.413781
2     0.530751
3     0.626018
4     0.701418
5     0.733304
6     0.764299
7     0.789316
8     0.809635
9     0.827132
10    0.843680
11    0.858180
12    0.871571
13    0.884380
14    0.896035
15    0.905882
16    0.914932
17    0.923347
18    0.931373
19    0.938953
20    0.945505
21    0.951364
22    0.956671
23    0.961293
24    0.965483
dtype: float64
```




| | | |
|---|---|---|
| PCA(RF) | MSE: 254.1 | NMAE: 0.2725 |
| PCA(DTR) | MSE: 278.5 | NMAE: 0.2905 |

# 최종모델결정



| Model | | Variates | RMSE | NMAE |
|---|---|---|---|---|
| 시계열 | Prophet | 도매가격(원/Kg) | 328 | 0.2886 |
| | ARIMA | 도매가격(원/Kg) | 612 | - |
| | SARIMA | 도매가격(원/Kg) | 613 | - |
| 회귀분석 | Multivariate Regression | 전체 | 234.5 | 0.254 |
| | Decision Tree | 전체 | 204.9 | 0.213 |
| | Random Forest | 전체 | 200.5 | 0.208 |
| | | 도메인룰 참고 | 197.31 | 0.202 |
| | | Feature Importance | 215.7 | 0.218 |
| | | Permutation Importance | 170.4 | 0.163 |
| | | PCA(주성분분석) | 254.1 | 0.273 |

# 03. 기대효과

# 22년 월별 양파 가격 예측



예측값 산출 시연

```
print('2021년 10월 ~ 2022년 8월 중 언제의 양파 가격을 출력하겠습니까?₩nex) 2021년 10월 -> 21/10₩n    2022년  8월 -> 22/8')
y,m=map(int, input().split('/'))

print('₩n양파 가격을 예측할 도시를 따옴표 없이 입력하세요₩n')
print('도시 목록:', ', '.join(city_list))
city=input()

city_num=LE_city_mapping[city]

X_input=DF_input[(DF_input['city']==city_num) & (DF_input['month']==m) & (DF_input['year']==y)]

print('{0}의 20{1}년 {2}월 월 평균 양파 도매 가격 예측 값은 {3}원/kg 입니다.'.format(city,y,m,load_RF.predict(X_input)))
```

```
2021년 10월 ~ 2022년 8월 중 언제의 양파 가격을 출력하겠습니까?
ex) 2021년 10월 -> 21/10
    2022년  8월 -> 22/8
22/5

양파 가격을 예측할 도시를 따옴표 없이 입력하세요

도시 목록: 거창, 경산, 고령, 고창, 고흥, 구리, 구미, 김천, 나주, 논산, 무안, 문경, 밀양, 부안, 서귀포, 서산, 안동, 여수, 연
논산
논산의 2022년 5월 월 평균 양파 도매 가격 예측 값은 [601.60166538]원/kg 입니다.
```

# 인사이트 도출



**결과 분석**

- 양파 도매가격 분석시, 시계열 분석 MSE 328~613원/kg 반면, 회귀분석 MSE 170~254원/kg으로 y : 가격, x : 지역, 거래량/금액, 기후를 변수로 설정한 분석이 좀더 유의미한 것으로 판단됨.
- 총 30,000개의 데이터row를 통해 가장 MSE가 낮은 모델은 Random Forest 이며, 이 중 Permutation Importance 를 통한 변수 설정이 가장 좋은 성과를 나타냄.
- 모델 검증을 통해 22년 월별/지역별 양파 도매가격 예측치를 출력할 수 있으며, MSE 170원/kg으로 편차가 작은 편은 아니나, 월별 추세를 통해 폭락/폭등의 흐름여부를 미리 판단해볼 수 있을 것으로 기대됨.

**모델 한계**

- 시계열 → 회귀분석 → 변수검증의 3단계를 통해 점차적으로 MSE를 줄여나가는 모델 탐색
- 주어진 데이터셋 이외에, '도메인룰' 탐색을 통한 추가적인 변수 삽입(수출입량, 기후 등)
- 변수설정을 위한 다양한 각도에서의 검증으로 최저 MSE 산출

**향후 개선**

- 논문 참고시, 기후/지역별 도메인차이를 더 세밀하게 반영하기 어려워, 기후 변수의 영향력이 대체적으로 낮은 편이며, 도메인룰과 다소 차이가 있음
- 공급량에 지대한 영향을 미치는 재배면적/생산성에 대한 데이터가 광역권으로 정리되어있어 시도산지별 데이터셋에 모델 설계시 영향력이 제한적임.

# 향후 개선사항



## 생산량(공급량)에 집중한 데이터셋을 통한 수확기 3개월 전(3월경) 공급량 예측으로 6월 도매가격 예측

**y = 6월 도매가격으로 한정**
- 양파 수확기는 5월 하순 - 6월 하순으로, 매년 6월 최대 거래량 기록
- 재고량이 떨어지는 4월부터 최대출하량 6월까지 가격이 떨어지는 패턴을 일관적으로 보임

**기후 데이터 중 상관관계가 높은 월로 선택(Pearson)**
- 평균기온 : 3,7,8월
- 최고기온 : 8월
- 최저기온 : 7,8월
- 강수량 : 6,12월
- 일조량 : 3,11월

| | year | region | cityl | area_size | productivity | AT_3 | AT_7 | AT_8 | HT_8 | LT_7 | LT_8 | RAIN_6 | RAIN_12 | SUN_3 | SUN_11 | prices |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2014 | 대구 | 달성 | 120 | 6089 | 9.7 | 28.7 | 29.0 | 34.4 | 25.1 | 25.0 | 28.2 | 5.9 | 219.4 | 220.3 | 379 |
| 1 | 2014 | 경북 | 경산 | 2602 | 7193 | 9.7 | 28.7 | 29.0 | 34.4 | 25.1 | 25.0 | 28.2 | 5.9 | 219.4 | 220.3 | 396 |
| 2 | 2014 | 광주 | 광주서구 | 83 | 6628 | 8.6 | 27.1 | 28.4 | 33.4 | 24.5 | 24.7 | 72.0 | 18.2 | 213.4 | 173.4 | 282 |
| 3 | 2014 | 전남 | 나주 | 12080 | 6319 | 8.6 | 27.1 | 28.4 | 33.4 | 24.5 | 24.7 | 72.0 | 18.2 | 213.4 | 173.4 | 311 |
| 4 | 2014 | 경기 | 구리 | 217 | 4080 | 7.9 | 25.5 | 27.7 | 31.1 | 23.4 | 24.8 | 98.1 | 24.7 | 214.7 | 188.0 | 356 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 276 | 2021 | 경남 | 창녕 | 4023 | 8423 | 10.1 | 23.1 | 27.8 | 32.5 | 20.2 | 24.2 | 68.9 | 5.1 | 217.5 | 184.0 | 501 |
| 277 | 2021 | 경남 | 의령 | 4023 | 8423 | 10.1 | 23.3 | 27.7 | 32.5 | 20.5 | 24.2 | 73.0 | 9.2 | 209.4 | 187.9 | 518 |
| 278 | 2021 | 경남 | 창원의창 | 4023 | 8423 | 11.9 | 23.6 | 28.7 | 33.0 | 21.1 | 25.6 | 120.3 | 9.5 | 214.2 | 167.8 | 480 |
| 279 | 2021 | 경남 | 함양 | 4023 | 8423 | 9.5 | 22.5 | 26.7 | 31.4 | 19.6 | 23.3 | 86.8 | 5.1 | 208.2 | 167.4 | 615 |
| 280 | 2021 | 제주 | 제주 | 906 | 8265 | 12.5 | 24.2 | 29.4 | 32.8 | 21.8 | 26.8 | 122.4 | 29.5 | 177.0 | 130.3 | 516 |

| 결과 | MSE : 140 | NAME:0.2886 |
|---|---|---|

**[ 이슈사항 ]**

Data row 280개로 한정
2014년~2021년 6월의 소수 지역

**[ 보완사항 ]**

(1) 재배면적을 現)광역시 → 後)시도산지도 세분화
(2) 양파 시도산지와 기상관측소의 매칭 정교화
(3) 지역별 기후에 따른 재배양상을 반영
(4) 데이터셋 기간 범위 확대
⇒ 최소 3,000개 확보하여 머신러닝을 통한 모델 정교화

**[ 기대효과 ]**

3월경, 전년도 재배면적 및 생산성 자료 입수를 통해
거래량 최대인 수확기 6월의 가격을 예측하여
월별 공급량 및 수입/수출량 조정으로
가격 폭등/폭락의 위험을 최소화할 수 있음.

# 04. 개발후기 및 느낀점

# 구성원 역할 및 개발후기

## 2조 CSP

### 김승연

- 데이터 수집
- 데이터 전처리
- 회귀 분석

주제 선정부터 가격 예측까지 많은 시행착오가 있었고 여전히 모델에 결함이 있지만 할 수 있는 것 내에서 완성을 했다는 점에서 뿌듯하다.
전처리 과정에서 코드 작성에 어려움이 있어 오랜 시간 소요되었다. 이 부분은 여러 프로젝트를 진행하며 실전적 감각을 익혀 나가야 할듯하다.
주제 선정 이전에는 양파의 가격에 그리 큰 관심이 없었지만 우리의 예측이 맞는지 앞으로 주의깊게 살펴보게 될 것 같다.

### 박지현

- 데이터 수집
- 데이터 분석
- 산출물,리포트

- 빅데이터 및 머신러닝 등 분석 모델뿐만 아니라 목적에 맞는 데이터셋 구성과 모델과의 fit을 미리 판단할 수 있어야 한다는 점을 많이 느꼈다. 그런 의미에서 이번 프로젝트는 익숙치않은 농업분야라 데이터셋 모음에 제한이 있었지만, 여러 모델을 검토해봄으로서 유의미한 추가 과제를 경험해볼 수 있었던 것 같다. 개인적으로 농업분야는 큰 부가가치가 있을 수 있다고 생각하는데, 아직 데이터관련된 부분이 다른 분야대비 체계적이지 않은 것 같아서 아쉽다.

### 신찬우

- 데이터 수집
- 데이터 전처리
- 시계열 분석,PCA

단순히 농산물 거래량과 기후관련 지표로만 가격을 예측할 수 있을 것이라 생각하면서 시작하였지만 생산량과 수입량 등 여러 다양한 변수들이 많은 영향을 준다는 것을 알게 되어 많은 어려움을 겪으면서 작업을 시행했다. 기온 또한 단순히 일일 기온으로는 의미있는 인사이트는 도출하지 못하고 재배기간 동안의 특정 기후가 중요하다는 사실 까지 인지 하게되어 데이터를 여러 방면으로 전처리를 해보았다.
전처리부터 다양한 모델링들을 직접 해보면서 각 모델들의 특징에 대해 많은 내용을 알 수 있는 프로젝트였던 것 같다.

### 임성국

- 시계열 분석
- 회귀 분석
- 예측 프로그램

프로젝트를 시작할 때는 Regression 모델을 통해 미래의 어떤 값을 예측한다는 게 논리적 결함이 많을 것 같아 scope을 매우 한정적으로 정해야 할 것으로 여겼다. 프로젝트를 통해 배운 점은 논리적 결함이 있어보여도 일단 모델을 만들어보는 게 예측값 모델링을 하는 방식이라는 것이다. 그 모델의 성능은 미래가 되어 모델이 실제로 얼마나 예측을 잘했는지를 봐야 비로소 알 수 있다.
그러므로 지금 단계에서 할 수 있는 일은 어떤 최선의 모델을 만드는 것이 아니라 여러 모델을 만들어보고 검증을 통해서 성능이 낮을 것으로 예상되는 모델을 제외하는 과정이지 않을까 생각해봤다. 프로젝트는 마무리되지만 각 모델에 대한 평가는 새로 시작될 것이다.