

초보 농가를 위한 한우 낙찰가 예측 서비스

김대환 김찬수 나서영 박선경 이태형



□ 목 차



● 주제

- 주제 선정 이유

● 요구사항 정의서

● 전처리

- 컬럼 정리
- 결측치 / 이상치 / 중복값 처리

● 데이터 분포 확인

● 모델링

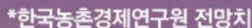
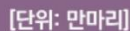
● 결과

- train score / test score
- rmse 오차

● 결론

● 출처



- # 한우 사육 농가 '이중고'에...
지역 내 하...
한우가격 폭락 여파...전북 농가소득, 전년보다 떨어져
... 343만원이던 ... 140만원으로 각각 65% ...
이처럼 한우 가격이 하락...
반면 배합사료 가...
이 가중되는...
... 소득이 4291만1000원으로, 전년보다 9.6% 하락한 것으로...
... 소득은 농업소득(3.8% ↑) ...
[편집자 칼럼] "예고된 한우가격 폭락 정부 책임 크다"
8월 전북도...
이 큰 폭으로 하락...
전국적인 농업소득 하락폭은 26.8%...
이같은 농업소득의 이례적...
원인으로 분석된다...
... 75% 수준에 그치...
...이라며 "정부의 사료 가...
...이라며 "농가의 어려움과 금리 인상으로 조합 경...



자료: 통계청



- 경상북도의회
 경북관광
 국가상징 알아보기

농업/축산  

검색어를 입력하세요

[Facebook](#)
[Twitter](#)
[Instagram](#)
[YouTube](#)

CHU

🏠 > 농업/축산 > 귀농귀촌종합지원센터 > 청년지원 > 도

- 귀농귀촌조례

• 시군

- 시군

· 시군

- 임시거주공간

도
 평창지역 초보 한우농가 대상, 맞춤형 종합 교육
한우 키우
인공수정교육

2023-07-04
 - 2023-09-04
☐ 기간
제목

장소

19-02

19-02

1

19-02

19-02

19-02

국내 최초 '기숙형 한우인 교육센터' 문 연다

요구사항 정의서



대분류	중분류	소분류	상세설명	우선순위	담당자
데이터 수집	데이터 수집	지역 데이터 추가 수집 (스마트한우경매사이트)	데이터 추가 수집 _ 전라남도(함평, 순천광양) _ 경상남도(합천, 의령, 창녕, 김해, 고성) _ 경상북도(영주, 고령성주)	1	전원
데이터 전처리	컬럼 정리	불필요한 컬럼 제거	번호, 출하주, 상태, 개체번호(중복값 확인 후 삭제)	1	전원
		비고 컬럼 처리, 새로운 컬럼 생성	결격사유 여부 컬럼 추가 _ 결격사유 있으면 1, 없으면 0 부여	1	전원
		범주형 컬럼 처리	범주형 데이터를 수치형으로 매핑 _ 성별: 수소-0, 암소-1 _ 종류: 큰소-0, 혈통우-1, 일반우-2	1	전원
	결측치 처리	Null 값 처리	낙찰가 0값 행 삭제	2	전원
	이상치 처리	컬럼 내 단위 통일	낙찰가 컬럼 중 단위가 다른 데이터 확인 _ 만 원 단위로 데이터 통일	2	전원
	중복값 처리	모든 요소 동일한 행 삭제	행 내의 모든 요소가 동일한 중복행을 삭제	2	전원



대분류	중분류	소분류	기능설명	우선순위	담당자
기능 설계	예측 타겟 설정	회귀모형 구현을 위한 타겟 설정	낙찰가 컬럼을 타겟으로 설정	3	전원
	예측 모델 구축	회귀 모형 구축	Ensemble Regression 모델 구축	3	이태형
			KNN Regression 모델 구축	3	김대환
			RandomForest Regression 모델 구축	3	박선경
			GradientBoosting Regression 모델 구축	3	나서영
			XGBoost Regression 모델 구축	3	김찬수



1. 종류

- 큰소 : 임신우(임신 중이거나 계획이 있는 암소)
- 혈통우 : 송아지(암, 수)
- 일반우 : 비육우(먹기위해 살찌우는 소)

2. 최저가

- 축협에서 지정한 시세 조정위원들이
소의 기본정보를 토대로 매긴 가격

3. 단위

- 최저가, 낙찰가 : 만 원
- 중량 : kg

축협에서 지정한 시세 조정위원들이 한우를 이리저리 살펴 보고 생년월일과 유전능력 등을 토대로 최저가격(기준가)을 매긴다.

이를 기준으로 가장 높은 가격을 스마트폰에 입력한 구매자가 낙찰받는다.

1회 유찰 시 기준가보다 10만원씩 낮게 책정돼 추가 경매에 부쳐진다.

2차례 이상 유찰돼 낙찰자가 나오지 않으면 거래 없이 빈손 돌아가는 농장주도 적지 않다고 축협 직원들은 귀띔한다.





1. 컬럼 정리

- 번호, 출하주, 상태, 개체번호(중복값 확인 후 삭제)
- 결격사유 여부 컬럼 추가(상처 여부, 이모색 여부)
- 범주형 데이터를 수치형으로 매핑

1. 이상치 처리

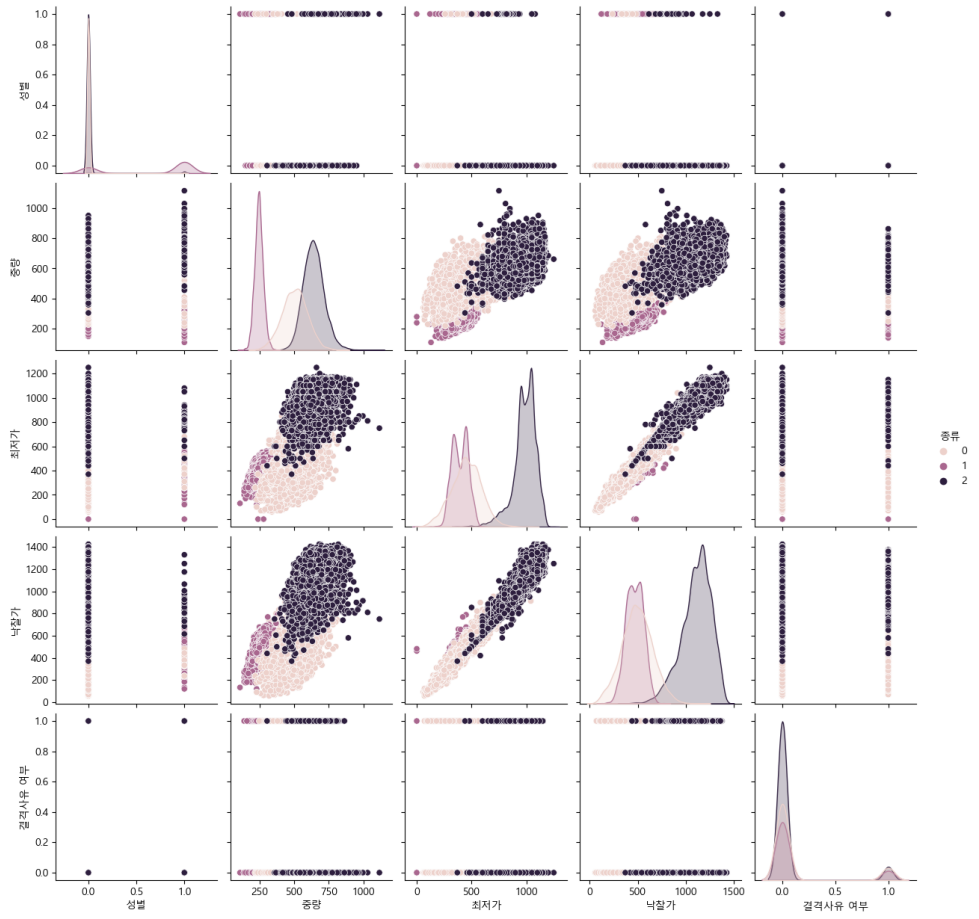
- 낙찰가 컬럼 만 원 단위로 통일
- 낙찰가 0값 행 삭제

1. 중복값 처리

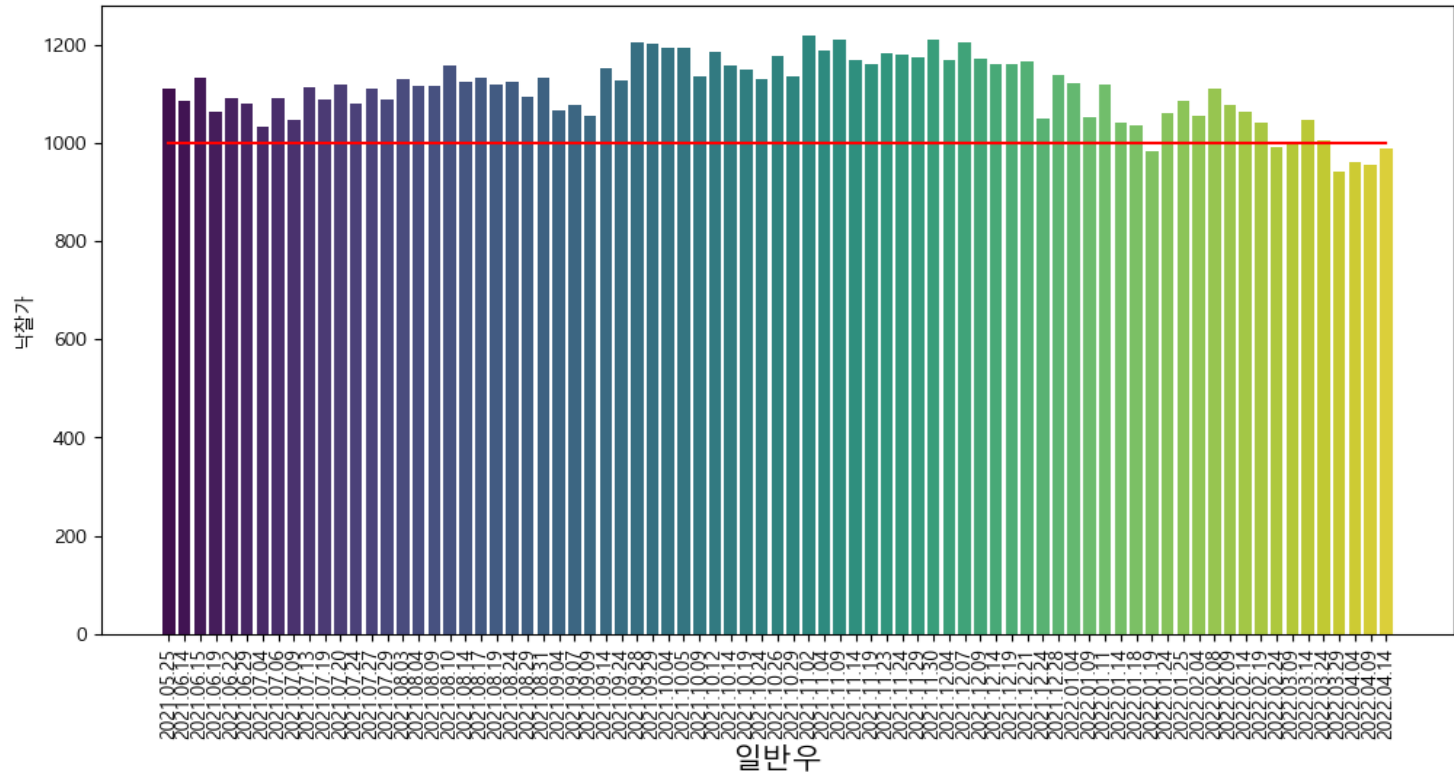
	성별	중량	최저가	낙찰가	결격사유 여부	종류	성별_n	종류_n
0	암	580	360	363	1	큰소	0	0
1	암	460	320	353	0	큰소	0	0
2	암	340	400	471	1	큰소	0	0
3	암	380	400	432	1	큰소	0	0
4	암	550	650	766	0	큰소	0	0
...
17443	암	635	970	11750	0	일반우	0	2
17444	암	620	940	10950	0	일반우	0	2
17445	암	614	1100	13500	0	일반우	0	2
17446	암	569	900	9450	0	일반우	0	2
17447	암	543	760	7600	0	일반우	0	2

	중량	최저가	낙찰가	결격사유 여부
count	17448.000000	17448.000000	17448.000000	17448.000000
mean	499.346114	669.672513	5122.761692	0.112391
std	173.233963	297.998572	5350.663271	0.315856
min	110.000000	0.000000	57.000000	0.000000
25%	310.000000	410.000000	460.000000	0.000000
50%	545.000000	560.000000	636.000000	0.000000
75%	634.000000	980.000000	10950.000000	0.000000
max	1114.000000	1250.000000	14250.000000	1.000000

Pairplot



Barplot - 시간별 낙찰가



- 전반적으로 한우 가격이 하락하는 것을 볼 수 있다.

□ 모델링 - KNN Regression



```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.utils import *
from sklearn.metrics import *
for i in range(3,21,2):
    train_x, test_x, train_y, test_y = train_test_split(cow_DF[['중량', '최저가', '종류_N']], cow_DF[['낙찰가']], random_state=0)
    cow_li = KNeighborsRegressor(n_neighbors=i)
    cow_li.fit(train_x, train_y)
    train_score = cow_li.score(train_x, train_y)
    test_score = cow_li.score(test_x, test_y)
    print(f'[k={i}] train score : {train_score}, test score : {test_score}')
    preds_y = cow_li.predict(test_x)
    mse = mean_squared_error(test_y, preds_y)
    rmse = np.sqrt(mse)
    print(f'MSE : {mse}, RMSE : {rmse}')
    print(f'R2 : {r2_score(test_y, preds_y)}')
```

KNN Regression을 활용한 ML 모델 구축

- 주요 분석 방법 : 회귀 분석
- 상세 분석 방법 : KNeighborsRegressor
- 피쳐 : 중량, 최저가, 종류
- 타겟 : 낙찰가
- 평가지표 : r2, rmse

```
[k=3] train score : 0.9791, test score : 0.9633, train - test : 0.0157
MSE : 4221.6910, RMSE : 64.9745
R2 : 0.9633407167691251
[k=5] train score : 0.9772, test score : 0.9665, train - test : 0.0107
MSE : 3861.5289, RMSE : 62.1412
R2 : 0.9664682042069871
[k=7] train score : 0.9760, test score : 0.9685, train - test : 0.0075
MSE : 3628.4229, RMSE : 60.2364
R2 : 0.9684923923613974
[k=9] train score : 0.9752, test score : 0.9694, train - test : 0.0059
MSE : 3528.8660, RMSE : 59.4043
R2 : 0.9693569006599976
[k=11] train score : 0.9746, test score : 0.9699, train - test : 0.0047
MSE : 3470.0317, RMSE : 58.9070
R2 : 0.969867791681857
[k=13] train score : 0.9743, test score : 0.9701, train - test : 0.0042
MSE : 3444.1061, RMSE : 58.6865
R2 : 0.9700929176559352
[k=15] train score : 0.9740, test score : 0.9703, train - test : 0.0037
MSE : 3424.4608, RMSE : 58.5189
R2 : 0.9702635090131824
[k=17] train score : 0.9738, test score : 0.9705, train - test : 0.0033
MSE : 3394.4171, RMSE : 58.2616
R2 : 0.9703243330243230
[k=19] train score : 0.9737, test score : 0.9707, train - test : 0.0030
MSE : 3376.9437, RMSE : 58.1115
R2 : 0.9706761262355044
```

□ 모델링 - Random Forest Regression



```
params={'RandomForestRegressor':{'max_depth':[5,10,15,25],  
                                  'n_estimators':[10,50,100,300],  
                                  'min_samples_leaf': [12,18,24],  
                                  'min_samples_split' : [6,12,18]}}
```

```
def ml_GridSearchCV(grid_Models,params_,n_splits_,train_X,train_Y):  
    ...  
    GridSearchCV 이용해서 하이퍼파라미터 찾고 객체 반환  
    n_splits_ :KFold의 n_splits  
    ...  
    grid_Dict={}  
    kf=KFold(n_splits=n_splits_,shuffle=True,random_state=0)  
    for model in grid_Models:  
        model_name = model.__class__.__name__  
        if model_name in params_.keys():  
            param_grid_ = params_[model_name]  
            gridCV = GridSearchCV(model, param_grid=param_grid_,cv=kf)  
            gridCV.fit(train_X, train_Y)  
            grid_Dict[gridCV.best_estimator_]=[gridCV.best_params_,gridCV.best_score_]  
    return grid_Dict
```

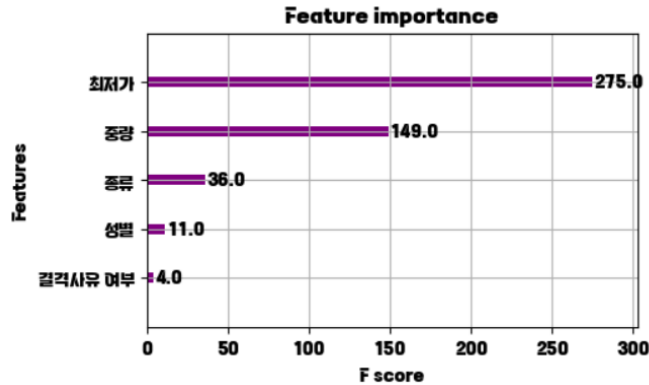
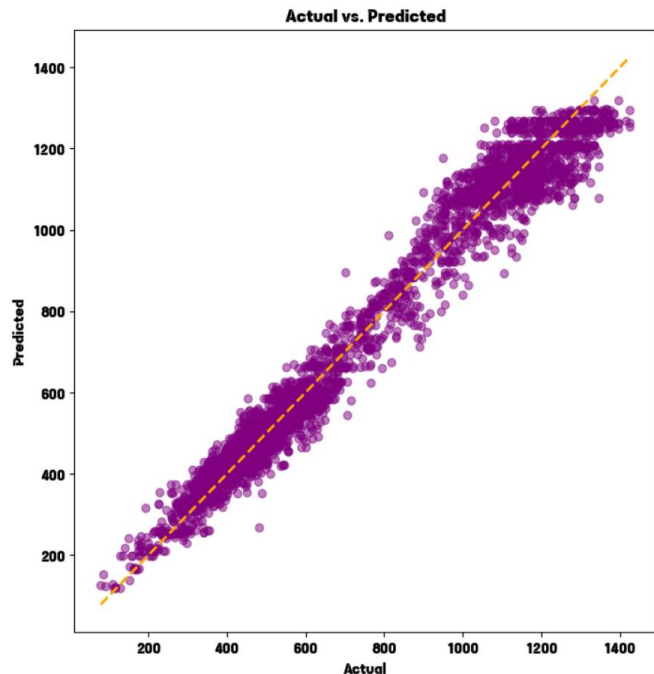
- 주요 분석 방법 : 회귀 분석
- 상세 분석 방법 :
 - Random Forest Regression
- 피쳐 : 중량, 최저가, 종류
- 타겟 : 낙찰가
- 교차 검증 :
 - GridSearchCV, KFold
- 평가지표 :
 - r2, rmse

train_score : 0.9741, test_score : 0.9715

성능평가 => r2 : 0.972, RMSE : 56.8

```
= [RandomForestRegressor(max_depth=5, min_samples_leaf=18, min_samples_split=12, n_estimators=10)]
```

모델링 - XGBoost Regression



XGBoost Regression 을 활용한 ML 모델 구축

- 주요 분석 방법 : 트리 기반 앙상블 학습
- 상세 분석 방법 : XGBoost Regression
- 피쳐 : 중량, 최저가, 종류, 성별, 결격사유
- 타겟 : 낙찰가
- 평가지표 : r2, rmse

```
[test score : 0.9713717299966214]
[train score : 0.9727002597324552]
[val score : 0.9701157237596847]
[RMSE : 57.781954195511204, R2 : 0.9713717299966214]
```

▣ 모델링 - Ensemble Regression



```
from autogluon.tabular import TabularPredictor
# 정규화는 AutoGluon 내부에서 자동으로 수행됩니다.

predictor3 = TabularPredictor(label='낙찰가')
|
predictor3.fit(train_data=train_data[['중량', '최저가', '낙찰가', '종류']], presets= 'medium_quality')
```

```
performance: -55.28517066854829
model: WeightedEnsemble_L2
model_type: WeightedEnsembleModel
hyperparameters: use_orig_features: False
                  max_base_models: 25
                  max_base_models_per_type: 5
                  save_bag_folds: True
inference_latency: 0.05186009407043457
training_time: 3.1904146671295166
```

Autogluon 패키지를 활용한 ML 모델 구축

- 주요 분석 방법 : 앙상블
- 상세 분석 방법 : WeightedEnsemble_L2
- 피쳐 : 중량, 최저가, 종류
- 타겟 : 낙찰가
- 평가지표 : r2, rmse

train r2 : 97.52%, test r2 : 97.25%

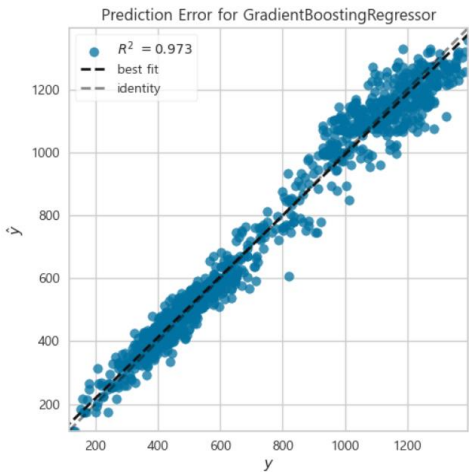
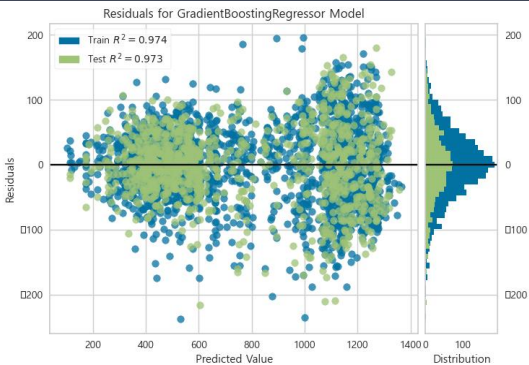
train rmse : 54, test rmse : 57

```
{'GBM': {}, 'CAT': {}, 'XGB': {}, 'RF': {}, 'XT': {}, 'KNN': {}, 'LR': {}, 'NN_TORCH': {}, 'FASTAI': {}}
```


모델링 - Gradient Boosting Regression



Parameters	
alpha	0.9
ccp_alpha	0.0
criterion	friedman_mse
init	None
learning_rate	0.1
loss	squared_error
max_depth	3
max_features	None
max_leaf_nodes	None
min_impurity_decrease	0.0
min_samples_leaf	1
min_samples_split	2
min_weight_fraction_leaf	0.0
n_estimators	100
n_iter_no_change	None
random_state	0
subsample	1.0
tol	0.0001
validation_fraction	0.1
verbose	0
warm_start	False



Pycaret 패키지를 활용한 ML 모델 구축

- 주요 분석 방법: 회귀 분석
- 상세 분석 방법: Gradient Boosting Regression
- 피쳐: 중량, 최저가, 종류
- 타겟: 낙찰가
- 평가지표: R^2 , RMSE

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Gradient Boosting Regressor	41.9723	3013.9577	54.8995	0.9743	0.0798	0.0614
	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Gradient Boosting Regressor	44.3209	3334.1213	57.7419	0.9714	0.0827	0.0644



Autogluon 패키지를 활용한 ML 모델 구축

- 주요 분석 방법 : 앙상블
- 상세 분석 방법 : weightedEnsemble_L2
- 피쳐 : 중량, 최저가, 종류
- 타겟 : 낙찰가
- 평가지표 : r2, rmse

train r2 : 97.52%, test r2 : 97.25%

train rmse : 54, test rmse : 57

중량 : 600kg, 최저가 : 800만원, 종류 : 일반우

모델 적용



858(\pm 50) 만원



97.52%의 모델 적합도(R^2)를 가진 모델을 구현,

1,106만 원 일반우의 경우

가격의 약 5%인 55만 원 내외의 오차 내에서

낙찰가를 예측하는 모델을 개발함

초보 농가에서 해당 서비스를 활용하여

적절한 판매가를 미리 예측하여 활용할 수 있도록

서비스를 무료로 배포하고 적극 홍보할 예정

▣ 데이터 출처

© 캐글 <https://www.kaggle.com/datasets/jskim1738/smart-korean-beef-auction>

© 한우경매 시스템 <http://www.xn--289a13w02jixo.kr/dev/web/index.html>



▣ 기사 출처

- © 뉴제주일보 (<http://www.jeuilbo.net/news/articleView.html?idxno=208816>)
- © 5개국어 글로벌 경제신문' 아주경제 (<https://www.ajunews.com/view/20230608162702348>)
- © 팜인사이트 (<https://www.farminsight.net/news/articleView.html?idxno=10101>)
- © BizWatch (<http://news.bizwatch.co.kr/article/consumer/2022/12/15/0026>)
- © 팜인사이트 (<https://www.farminsight.net/news/articleView.html?idxno=6781>)
- © 중부메일 (<http://www.jbnews.com/news/articleView.html?idxno=1406021>)
- © 예천연합뉴스 (<https://www.y-cnews.com/news/articleView.html?idxno=13360>)
- © 식약일보 (<http://www.kfdn.co.kr/42808>)
- © 팜인사이트 (<https://www.farminsight.net/news/articleView.html?idxno=10057>)
- © 농축유통신문 (<https://www.amnews.co.kr/news/articleView.html?idxno=54088>)
- © 경향신문 (<https://m.khan.co.kr/economy/economy-general/article/202303061425001>)
- © 한국농정 (<http://www.ikpnews.net/news/articleView.html?idxno=49698>)
- © 한국농어촌방송 (<http://www.newskr.kr/news/articleView.html?idxno=84774>)
- © 농축유통신문 (<https://www.amnews.co.kr/news/articleView.html?idxno=54157>)