FarmTech

소프트웨어융합학과 2018111395 박서영

소프트웨어융합학과 2018111575 김가을

농산물 가격 예측 AI 모델 개발 및 이를 활용한 소비 품목 제안 어플리케이션

CONTENTS

Contents 01
현재 진도 사항

Contents 02
향후 진행 계획

01 Contents 01

현재 진도 사항

데이터 출처 : 농업관측통계시스템(OASIS)
https://oasis.krei.re.kr/analyzer/marketTrends/wholesale.do(수요량:월별 / 가격)

데이터 살펴보기 :Features Descriptions

각 변수 :

1. 거래일자
2. 양파(일반) 1.0kg 그물망 / 특 등급 <가격>
3. 양파(일반) 1.0kg 그물망 / 상 등급 <가격>
4. 양파(일반) 1.0kg 그물망 / 중 등급 <가격>
5. 양파(일반) 1.0kg 그물망 / 하 등급 <가격>
6. 양파(전체) 반입량 - 일반 <종류1>
7. 만생양파 1.0kg 그물망 / 특 등급 <가격>
8. 만생양파 1.0kg 그물망 / 상 등급 <가격>
9. 만생양파 1.0kg 그물망 / 중 등급 <가격>
10. 만생양파 반입량 <종류2>
11. 저장양파 1.0 kg 그물망 /특 등급 <가격>
12. 저장양파 1.0 kg 그물망 /중 등급 <가격>
13. 저장양파 반입량 <종류3>
14. 조생양파 1.0kg 그물망 / 특 등급 <가격>
15. 조생양파 1.0kg 그물망 / 상 등급 <가격>
16. 조생양파 1.0kg 그물망 / 중 등급 <가격>
17. 조생양파 1.0kg 그물망 / 하 등급 <가격>
18. 조생양파 반입량 <종류4>
19. 기타 1.0kg 그물망 / 특 등급 <가격>
20. 기타 1.0kg 그물망 / 상 등급 <가격>
21. 기타 1.0kg 그물망 / 중 등급 <가격>
22. 기타 1.0kg 그물망 / 하 등급 <가격>
23. 기타 반입량 <종류5>

```
[88] ### 결측치 : 비어있는 데이터를 찾습니다.
     ### 여기서는 어떤 컬럼(변수, 특성, x)에 결측치가 많은지 봅니다.
     df.isnull().sum()
```

```
거래일자                    0
양파(일반) 1.0 kg 그물망 /특     3
양파(일반) 1.0 kg 그물망 /상    33
양파(일반) 1.0 kg 그물망 /중    50
양파(일반) 1.0 kg 그물망 /하    72
양파(전체) 반입량               1
만생양파 1.0 kg 그물망 /특      65
만생양파 1.0 kg 그물망 /상      72
만생양파 1.0 kg 그물망 /중      73
만생양파 반입량                 1
저장양파 1.0 kg 그물망 /특      57
저장양파 1.0 kg 그물망 /중      72
저장양파 반입량                 1
조생양파 1.0 kg 그물망 /특      67
조생양파 1.0 kg 그물망 /상      71
조생양파 1.0 kg 그물망 /중      72
조생양파 1.0 kg 그물망 /하      72
조생양파 반입량                 8
기타 1.0 kg 그물망 /특          3
기타 1.0 kg 그물망 /상         59
기타 1.0 kg 그물망 /중         62
기타 1.0 kg 그물망 /하         72
기타 반입량                     1
dtype: int64
```

```
[89] ### 만약 결측치가 있다면, 결측치를 처리해야 됩니다. 이것을 전처리라고 합니다.
     df.dropna(axis=1, thresh=30)
```

	거래일자	양파(일반) 1.0 kg 그물망 /특	양파(일반) 1.0 kg 그물망 /상	양파(전체) 반입량	만
0	2015-01-01	NaN	NaN	13916582.0	
1	2015-02-01	900.0	NaN	11005449.0	
2	2015-03-01	1031.0	NaN	13611617.0	
3	2015-04-01	1109.0	NaN	22077825.0	
4	2015-05-01	1200.0	NaN	26358838.0	
...	...	...	...	...	
68	2020-09-01	2129.0	1725.0	14683073.0	
69	2020-10-01	2200.0	1800.0	13567706.0	
70	2020-11-01	2200.0	1700.0	14104641.0	
71	2020-12-01	2357.0	1300.0	12713812.0	
72	2021-01-01	NaN	NaN	NaN	

73 rows × 9 columns

01 Contents 01

현재 진도 사항

```
### data type을 확인합니다.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73 entries, 0 to 72
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   거래일자                  73 non-null     object
 1   양파(일반) 1.0 kg 그물망 /특  70 non-null     float64
 2   양파(일반) 1.0 kg 그물망 /상  40 non-null     float64
 3   양파(전체) 반입량            72 non-null     float64
 4   만생양파 반입량              72 non-null     float64
 5   저장양파 반입량              72 non-null     float64
 6   조생양파 반입량              65 non-null     float64
 7   기타 1.0 kg 그물망 /특      70 non-null     float64
 8   기타 반입량                72 non-null     float64
dtypes: float64(8), object(1)
memory usage: 5.3+ KB
```

```
#거래일자 데이터 현재 형식 : 0000-00-00 => 숫자형 변수로 바꾸기 위해 우선 "-" 제거
df['거래일자'] = df['거래일자'].str.replace("-", "").head()
```

```
#거래일자 데이터 현재 형식 : 00000000 => dtype : float32로 변환
df['거래일자'] = df['거래일자'].astype(float)
```

```
#거래일자 데이터 타입 바뀌었는지 확인
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73 entries, 0 to 72
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   거래일자                  5 non-null      float64
 1   양파(일반) 1.0 kg 그물망 /특  70 non-null     float64
 2   양파(일반) 1.0 kg 그물망 /상  40 non-null     float64
 3   양파(전체) 반입량            72 non-null     float64
 4   만생양파 반입량              72 non-null     float64
 5   저장양파 반입량              72 non-null     float64
 6   조생양파 반입량              65 non-null     float64
 7   기타 1.0 kg 그물망 /특      70 non-null     float64
 8   기타 반입량                72 non-null     float64
dtypes: float64(9)
memory usage: 5.3 KB
```

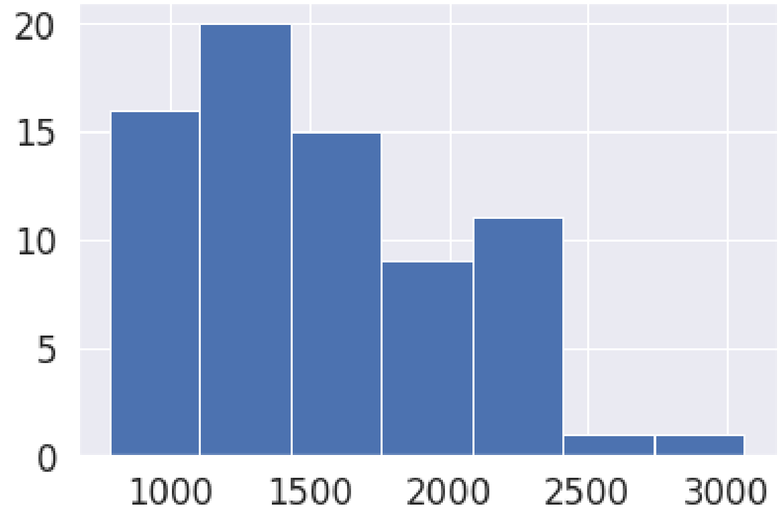
01 Contents 01

현재 진도 사항

타겟 변수로 지정한 양파 가격을 종류별 시각화하기 – 히스토그램

```
df['양파(일반) 1.0 kg 그물망 /특'].hist(bins=7)
```

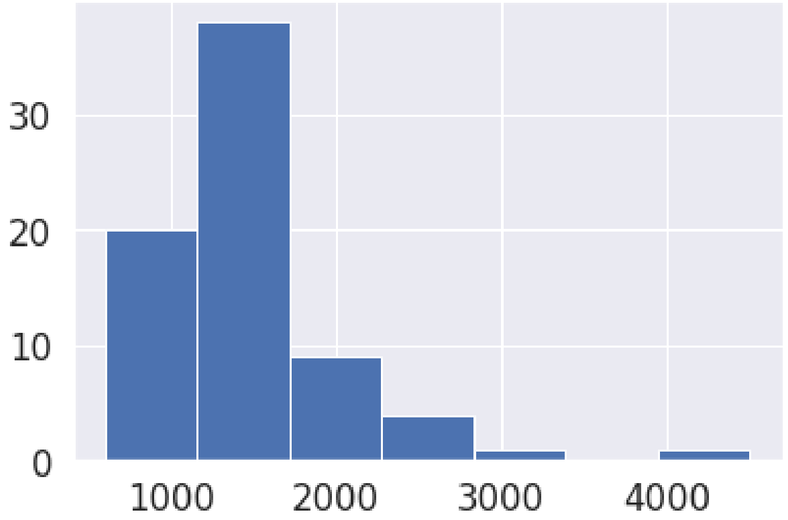
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff39075c6d0>
```



1000원과 2000원대 사이 가격이 일반적이고,
2500원과 3000원대 가격은 거의 형성되지 않음

```
df['기타 1.0 kg 그물망 /특'].hist(bins=7)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff3906fab50>
```



1000원과 2000원대 사이 가격이 일반적이고,
가장 비싼 가격이 4000원대임

01 Contents 01

현재 진도 사항

데이터 전처리

- Feature들의 scale 차이를 없애기 위한 표준화 진행

▾ 3-1 데이터 전처리

- 먼저 Feature 들의 scale 차이를 없애기 위해 수치형 Feature에 대해서 표준화를 진행해야 합니다.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()  # 평균 0, 표준편차 1
scale_columns = ['p_onion1', 's_onion1', 'p_onion2', 's_onion2']
df[scale_columns] = scaler.fit_transform(df[scale_columns])
```

```
df.head()
```

```
df[scale_columns].head()
```

	p_onion1	s_onion1	p_onion2	s_onion2
0	0.000000	0.382018	-1.342480	-0.696513
1	-1.199877	-0.228088	-1.247918	-0.816844
2	-0.940136	0.318104	-0.961082	-0.559957
3	-0.785481	2.092425	-0.940594	-0.311757
4	-0.605051	2.989625	-0.363769	-0.324822

sklearn라이브러리에서 train_test_split 이용

-train data와 test data 나누기

```
from sklearn.model_selection import train_test_split

# split dataset into training & test
X = df[numerical_columns]
y = df['p_onion1']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

```
X_train.shape, y_train.shape
```

```
((58, 2), (58,))
```

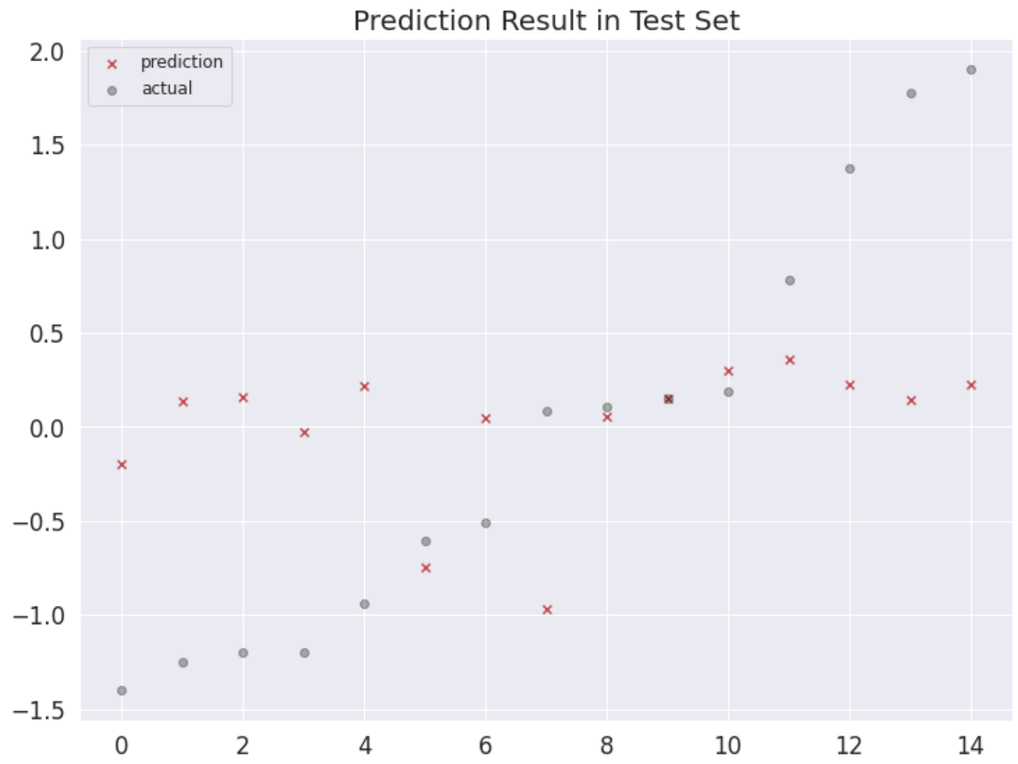
```
X_test.shape, y_test.shape
```

```
((15, 2), (15,))
```

01 Contents 01

현재 진도 사항

```
plt.figure(figsize=(12, 9))
plt.scatter(df.index, df['prediction'], marker='x', color='r')
plt.scatter(df.index, df['actual'], alpha=0.3, marker='o', color='black')
plt.title("Prediction Result in Test Set", fontsize=20)
plt.legend(['prediction', 'actual'], fontsize=12)
plt.show()
```



test set 예측 결과 예측과 실제 값이 차이나는 부분이 많았다.

이 부분들을 수정 보완해나가는 방향으로 진행할 것이다.

수집 가능한 데이터의 양이 부족해서

이 점을 k-fold를 이용해 데이터 보완을 할 예정이다.

01 Contents 01

현재 진도 사항

모델 성능 평가 (R squre 와 RMSE)를 실시했는데

그 결과 R square(결정계수)는 0.05이고,

RMSE(표준편차)는 1.04이다.

크게 나쁘지도 좋지도 않은 성능이라

모델 보완이 필요하다.

```
### R square
### 결정계수 : 0 부터 1까지의 값을 갖고, 1에 가까울수록 설명력이 높음을 의미한다.
### 선형 회귀 모델이 데이터에 대해 얼마나 잘 설명해주는지에 대한 값
print(model.score(X_train, y_train))  # training set
print(model.score(X_test, y_test))  # test set
```

```
0.10551890402990416
0.050801703155534894
```

```
### RMSE
### 제곱근을 취한 평균 제곱근 오차(Root mean squared error, RMSE) = 표준편차
### 특정 수치에 대한 예측의 정확도를 표현할 때,
### Accuracy로 판단하기에는 정확도를 올바르게 표기할 수 없어, RMSE 수치로 정확도 판단을 하곤 한다.
### 일반적으로 해당 수치가 낮을수록 정확도가 높다고 판단한다.
from sklearn.metrics import mean_squared_error
from math import sqrt

# training set
pred_train = lr.predict(X_train)
print(sqrt(mean_squared_error(y_train, pred_train)))

# test set
print(sqrt(mean_squared_error(y_test, pred_test)))
```

```
0.9272098849898941
1.0435614499262107
```

```
from sklearn import linear_model

# fit regression model in training set
lr = linear_model.LinearRegression()
model = lr.fit(X_train, y_train)

# predict in test set
pred_test = lr.predict(X_test)
```

```
### 해설) Test set에서 해당 예측 모델의 R square가 0.05이고, RMSE가 1.04입니다.
```

01 Contents 01

현재 진도 사항

모델 성능 평가 (R squre 와 RMSE)를 실시했는데

그 결과 R square(결정계수)는 0.05이고,

RMSE(표준편차)는 1.04이다.

크게 나쁘지도 좋지도 않은 성능이라

모델 보완이 필요하다.

```
### R square
### 결정계수 : 0 부터 1까지의 값을 갖고, 1에 가까울수록 설명력이 높음을 의미한다.
### 선형 회귀 모델이 데이터에 대해 얼마나 잘 설명해주는지에 대한 값
print(model.score(X_train, y_train))  # training set
print(model.score(X_test, y_test))  # test set
```

```
0.10551890402990416
0.050801703155534894
```

```
### RMSE
### 제곱근을 취한 평균 제곱근 오차(Root mean squared error, RMSE) = 표준편차
### 특정 수치에 대한 예측의 정확도를 표현할 때,
### Accuracy로 판단하기에는 정확도를 올바르게 표기할 수 없어, RMSE 수치로 정확도 판단을 하곤 한다.
### 일반적으로 해당 수치가 낮을수록 정확도가 높다고 판단한다.
from sklearn.metrics import mean_squared_error
from math import sqrt

# training set
pred_train = lr.predict(X_train)
print(sqrt(mean_squared_error(y_train, pred_train)))

# test set
print(sqrt(mean_squared_error(y_test, pred_test)))
```

```
0.9272098849898941
1.0435614499262107
```

```
from sklearn import linear_model

# fit regression model in training set
lr = linear_model.LinearRegression()
model = lr.fit(X_train, y_train)

# predict in test set
pred_test = lr.predict(X_test)
```

```
### 해설) Test set에서 해당 예측 모델의 R square가 0.05이고, RMSE가 1.04입니다.
```

01 Contents 01

현재 진도 사항

배추도 양파와 마찬가지로 전처리+선형회귀 예측 모델을 만들었다.

```
df.head()
```

	거래일자	봄배추 반입량	여름배추 반입량	김장(가을)배추 반입량	월동배추 반입량	가격
0	2021-01-11	-0.504223	-0.546712	-0.512769	0.063917	4219.75
1	2021-02-11	-0.498271	-0.550317	-0.540178	-0.010509	7082.75
2	2021-03-11	-0.488285	-0.549409	-0.535521	0.366874	4518.50
3	2021-04-11	0.345640	-0.547703	-0.589095	-0.315150	3505.50
4	2021-05-11	2.683002	-0.536649	-0.587308	-0.774616	1760.25

학습 이후 [최고기온,평균기온,최저기온,강수량]

데이터를 추가해 학습시켜보았는데 크게 달라진 점이 없어

데이터를 잘게 나누어 양을 늘려 학습시켜볼 계획이다.

```
df_last.head()
```

	거래일자	봄배추 반입량	여름배추 반입량	김장(가을)배추 반입량	월동배추 반입량	최고기온	평균기온	최저기온	강수량	가격
0	2011-01-01	-0.504223	-0.546712	-0.512769	0.063917	-2.474648	-1.852270	-1.388074	-1.071434	4219.75
1	2011-02-01	-0.498271	-0.550317	-0.540178	-0.010509	-0.845615	-1.192151	-1.189318	-0.492478	7082.75
2	2011-03-01	-0.488285	-0.549409	-0.535521	0.366874	-0.682712	-0.970949	-0.822382	-0.805992	4518.50
3	2011-04-01	0.345640	-0.547703	-0.589095	-0.315150	-0.180427	-0.236515	-0.373904	0.339379	3505.50
4	2011-05-01	2.683002	-0.536649	-0.587308	-0.774616	0.410098	0.490371	0.365064	-0.091180	1760.25

02

Contents 02

향후 진행 계획

4대 품목에 대한 [반입량, 기상] 데이터에 대해 전반적으로 학습시킨 상태이며
정확도 향상을 위해 정제할 예정

앱에 적용하기 위한 추천 알고리즘 개발

감사합니다