AIFactory

농산물 가격 변동률 예측 AI 경진대회

서현석 손보영 신진명 이명덕 이태영 이정민 조윤진 조현준 정인서

핀테크 디지털 금융 사이언티스트 양성과정
(2022.07.21~2022.09.30)

TABLE OF CONTENTS

01 공모전 소개

한국농수산식품유통공사
Korea Agro-Fisheries & Food Trade Corporation

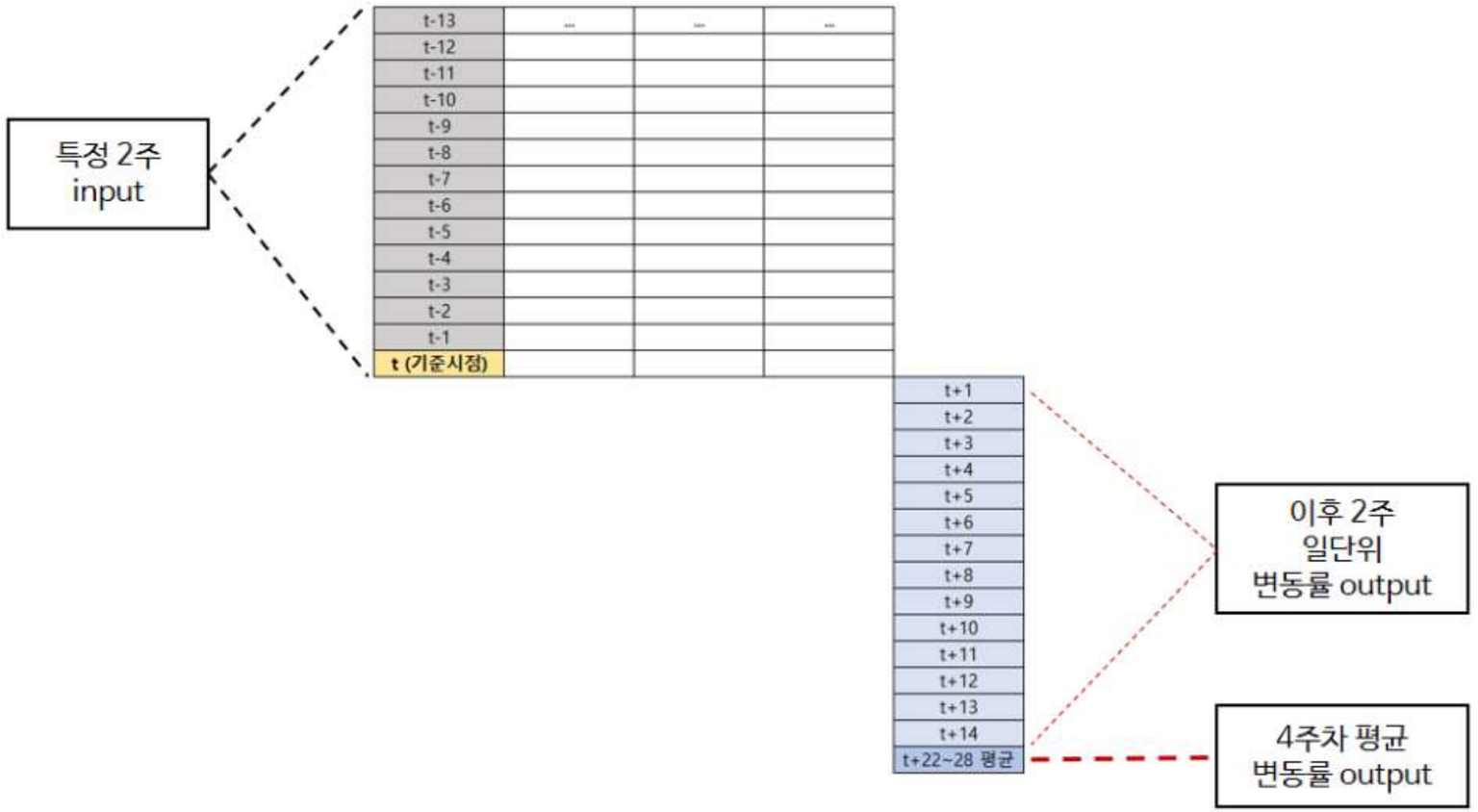
주최/주관 : 한국농수산식품유통공사
운영 : 인공지능팩토리

본 대회를 통해 입상한 우수 모형은 추후 농넷의 가격예측 모델 고도화에 활용될 예정입니다.
https://aifactory.space/competition/detail/2091

01

공모전 소개

변동률 예측



예측해야 할 변동률

각 set, 품목별

1) 입력 데이터 기간 이후 14일 동안 매일의, 기준시점 대비 변동률 14개값

2) 입력 데이터 기간 이후 22~28일 (+4주차) 기간의 기준시점 대비 변동률의 평균 1개값

이상의 15개

37개 품목

54,057

가격 데이터

Raw Data

도매 시장 데이터

품목별 거래시장, 도매가격,
등급, 조사단위

도매 경락 데이터

픔목별 경매 거래량,
거래 대금,단가

소매 시장 데이터

품목별 거래시장, 농산물 부류명,
소매가격,등급, 조사단위

수입 및 수출

전체 품목별
수출입 가격, 중량, 무역수지

주산지 날씨

품목당 주산지 0,1,2 의
온도, 습도, 강수량

Missing data

- 수출입 데이터

  2015.12 날짜 의 Null 값 :
  2013/2014/2016.01 데이터의 흐름 분석 후
  Null 값은 수출입이 없었던 날로 판단하여 0으로 대치
  하나의 데이터 프레임으로 만들기 위해
  월별 데이터를 일별 데이터 형식으로 변형

- 날씨 데이터

  습도, 강수량, 온도 모든 columns에서의 null값은 4년치
  데이터 평균으로 채우기

02

Data EDA

수입 및 수출

전체 품목 별
수출입 가격, 중량,
무역수지

주산지 날씨

품목당 주산지 0,1,2 의
온도, 습도, 강수량

Missing data

도매/소매 데이터

거래가 없는 날 : 월, 화요일은 0으로 채워주기

조사단위 첫 단위로 통일

도매, 소매 거래 가격 데이터를 일자별 가격 최대/최소/평균
단위가격으로 파생 변수 생성

등급이 '상품', 'S' 만 해당하는 품목 데이터만 사용

	조사단위	도매 최대가격	도매 최소가격	도매 평균가격	거래시장	등급명	...
2013.01.01	kg	5161.0	32000.0	4020.0	경동	상품	
2013.02.01	kg	43000.0	2200.0	3305.0	양동	S	
2013.03.01	1 개	3500.0	3225.0	3440.0	B-유통	S	
...	포기	35000	19000.0	2890.0	칠성	상품	
2016.03.01	접	4225.0	25000	3400.0	c-유통	S	

02

Data EDA

도매 시장 데이터

품목 별 거래시장,
도매가격,
등급, 조사단위

소매 시장 데이터

품목 별 거래시장 ,
소매가격 ,농산물 부류명,
등급, 조사단위

Missing data

품목 데이터

해당일자 가격 null : 0으로 채우기
일자 거래량 합계로 해당일자 전체 거래물량 변수 추가

평균가격의 단가가 평균보다 낮으면 하위, 평균보다 높으면 상위로 구분

품목의 데이터를 가져와 '해당일자 전체거래물량', '하위가격 평균가', '상위가격 평균가', '하위가격 거래물량', '상위가격 거래물량' 의 파생변수를 생성

	해당일자 전체거래물량	하위가격 평균가	상위가격 평균가	...
2013.01.01	11373.0	1161.744839291	3543.28383	
2013.01.02	20672.0	2819.36373	3577.23838	
2013.01.03	3838.0	3100.362181	3611.1938	
...	8663.0	3257.37924	3814.3642920	
2016.12.31	2305.0	2989.38920	3752.6754	

02 Data EDA

도매 경락 데이터

품목 별 경매 거래량,
거래 대금,단가

Feature Selection

Lasso Regression

```
from sklearn import *
def lasso_feature_selection(df):

    X_train = df.drop(['log_target'], axis=1)
    y_train = df["log_target"]

    regressor = linear_model.Lasso(alpha=0.01,
                                   positive=True,
                                   fit_intercept=False,
                                   max_iter=1000,
                                   tol=0.0001)
    regressor.fit(X_train, y_train)

    fs = pd.DataFrame(regressor.coef_, index=X_train.columns)
    fs.columns = ['weights']
    sc = list(fs[fs['weights'] > 0].index)



    df = df[sc]
    df['log_target'] = y_train
    #print(f'품목의 selected_feature: {sc}')
    return df
```

```
# train 데이터 라쏘회귀로 feature selection
for i in range(37):
  globals()[f'final_train_{i}'] = lasso_feature_selection(globals()[f'train_scaled_{i}'])
  print(globals()[f'final_train_{i}'].columns)
```

```
Index(['단가(원)', '습도(%)_1', '습도(%)_2', '일자별_somae가격_최대(원)', '무역수지(달러)',
       'log_target'],
      dtype='object')
Index(['단가(원)', '일자별_domae가격_최대(원)', '일자별_domae가격_평균(원)', '수입중량(kg)',
       '수입금액(달러)', 'log_target'],
      dtype='object')
Index(['단가(원)', '일자별_domae가격_최대(원)', '무역수지(달러)', 'log_target'], dtype='object')
Index(['단가(원)', '최저온도(℃)_0', 'log_target'], dtype='object')
Index(['단가(원)', '수입중량(kg)', 'log_target'], dtype='object')
Index(['단가(원)', '일자별_somae가격_최소(원)', '수출금액(달러)', 'log_target'], dtype='object')
Index(['단가(원)', '최대온도(℃)_0', '강수량(ml)_1', '최대온도(℃)_2', '수입금액(달러)',
       'log_target'],
      dtype='object')
Index(['단가(원)', '강수량(ml)_0', '최저온도(℃)_1', '강수량(ml)_1', 'log_target'], dtype='object')
Index(['단가(원)', '최저온도(℃)_0', '최대온도(℃)_1', '습도(%)_2', 'log_target'], dtype='object')
Index(['단가(원)', '최대온도(℃)_0', '수입금액(달러)', 'log_target'], dtype='object')
Index(['단가(원)', '최대온도(℃)_0', '최대온도(℃)_1', '습도(%)_2', 'log_target'], dtype='object')
Index(['단가(원)', '최대온도(℃)_0', '평균온도(℃)_0', '습도(%)_1', '수입중량(kg)', 'log_target'], dtype='ob
Index(['단가(원)', '수입금액(달러)', 'log_target'], dtype='object')
Index(['단가(원)', '무역수지(달러)', 'log_target'], dtype='object')
Index(['단가(원)', '습도(%)_0', 'log_target'], dtype='object')
Index(['단가(원)', '수입금액(달러)', 'log_target'], dtype='object')
Index(['단가(원)', '거래량', '최저온도(℃)_0', '습도(%)_2', '일자별_domae가격_최대(원)',
       '일자별_somae가격_평균(원)', 'log_target'],
      dtype='object')
Index(['단가(원)', 'log_target'], dtype='object')
Index(['단가(원)', '무역수지(달러)', 'log_target'], dtype='object')
Index(['단가(원)', '습도(%)_2', '일자별_domae가격_최대(원)', '무역수지(달러)', 'log_target'], dtype='object'
Index(['단가(원)', '일자별_domae가격_최대(원)', 'log_target'], dtype='object')
Index(['단가(원)', '습도(%)_0', '초기온도(℃)_2', '최저온도(℃)_2', '강수량(ml)_2',
       '일자별_domae가격_최소(원)', '일자별_somae가격_최소(원)', '수입중량(kg)', 'log_target'],
      dtype='object')
Index(['단가(원)', '습도(%)_0', '습도(%)_1', '습도(%)_2', '일자별_domae가격_최대(원)',
       '수입중량(kg)', '수입금액(달러)', 'log_target'],
      dtype='object')
Index(['단가(원)', '최저온도(℃)_0', '강수량(ml)_0', '초기온도(℃)_1', '최저온도(℃)_2', '습도(%)_2',
       '일자별_domae가격_최소(원)', 'log_target'],
      dtype='object')
Index(['단가(원)', '최저온도(℃)_0', '최저온도(℃)_1', 'log_target'], dtype='object')
Index(['단가(원)', '최저온도(℃)_0', '습도(%)_0', 'log_target'], dtype='object')
Index(['단가(원)', 'log_target'], dtype='object')
Index(['단가(원)', 'log_target'], dtype='object')
Index(['단가(원)', '초기온도(℃)_0', '습도(%)_0', '최대온도(℃)_1', '습도(%)_2',
       '일자별_domae가격_최소(원)', '수입중량(kg)', '수입금액(달러)', 'log_target'],
      dtype='object')
```

Lasso Regression : MAE Score - 0.47

Feature Selection

Ols Report

```
0 품목의 selected_columns : ['단가', '거래량', '습도_0', '초기온도_1', '최저온도_1', '평균온도_1', '일자별_somae가격_최대', '일자별_somae가격_최소', '해당일자_전체평균가격']
1 품목의 selected_columns : ['단가', '거래량', '습도_0', '최대온도_1', '평균온도_1', '습도_1', '최대온도_2', '최저온도_2', '평균온도_2', '습도_2', '일자별_somae가격_최대', '일자별_somae가격_평균', '일자별_somae가격_최소', '수출중량', '수출금액', '해당일자_
2 품목의 selected_columns : ['단가', '거래량', '최저온도_0', '습도_0', '최저온도_1', '최저온도_2', '습도_2', '일자별_domae가격_최소', '일자별_somae가격_평균', '일자별_somae가격_최소', '수출중량', '수출금액', '수입금액', '무역수지', '해당일자_전체평균가격']
3 품목의 selected_columns : ['단가', '거래량', '최저온도_1', '평균온도_1', '습도_2', '일자별_domae가격_최대', '일자별_domae가격_평균', '일자별_somae가격_평균', '일자별_somae가격_최소', '수출중량', '수출금액', '수입중량', '수입금액', '무역수지', '해당일자_전
4 품목의 selected_columns : ['단가', '거래량', '최저온도_0', '습도_2', '일자별_somae가격_최대', '일자별_somae가격_최소', '수출중량', '무역수지', '해당일자_전체평균가격']
5 품목의 selected_columns : ['단가', '거래량', '습도_0', '평균온도_1', '일자별_domae가격_평균', '일자별_domae가격_최소', '수출중량', '수출금액', '수입중량', '수입금액', '무역수지', '해당일자_전체평균가격']
6 품목의 selected_columns : ['단가', '거래량', '습도_1', '초기온도_2', '최대온도_2', '해당일자_전체평균가격']
7 품목의 selected_columns : ['단가', '거래량', '초기온도_0', '강수량_0', '초기온도_1', '최대온도_1', '일자별_somae가격_최대', '일자별_somae가격_평균', '해당일자_전체평균가격']
8 품목의 selected_columns : ['단가', '거래량', '최저온도_0', '평균온도_0', '습도_0', '최저온도_1', '습도_1', '습도_2', '일자별_domae가격_평균', '일자별_domae가격_최소', '일자별_somae가격_최대', '일자별_somae가격_최소', '해당일자_전체평균가격']
9 품목의 selected_columns : ['단가', '거래량', '최대온도_0', '최대온도_2', '최저온도_2', '평균온도_2', '일자별_domae가격_최대', '일자별_domae가격_평균', '일자별_domae가격_최소', '일자별_somae가격_평균', '일자별_somae가격_최소', '수출중량', '수입중량', '해당
10 품목의 selected_columns : ['단가', '거래량', '최저온도_0', '최대온도_1', '강수량_1', '강수량_2', '일자별_domae가격_최대', '일자별_domae가격_평균', '일자별_domae가격_최소', '일자별_somae가격_최대', '수입중량', '해당일자_전체평균가격']
11 품목의 selected_columns : ['단가', '거래량', '평균온도_0', '습도_0', '초기온도_1', '평균온도_1', '강수량_1', '일자별_somae가격_최소', '해당일자_전체평균가격']
12 품목의 selected_columns : ['단가', '거래량', '습도_0', '최대온도_2', '평균온도_2', '습도_2', '일자별_domae가격_최소', '일자별_somae가격_평균', '수출중량', '수입중량', '해당일자_전체평균가격']
13 품목의 selected_columns : ['단가', '초기온도_0', '습도_0', '평균온도_1', '평균온도_2', '일자별_domae가격_최대', '일자별_domae가격_평균', '일자별_domae가격_최소', '일자별_somae가격_최대', '일자별_somae가격_최소', '수출중량', '수출금액', '수입중량', '수입금
14 품목의 selected_columns : ['단가', '거래량', '습도_0', '평균온도_1', '최저온도_2', '강수량_2', '습도_2', '일자별_domae가격_최대', '일자별_domae가격_평균', '일자별_somae가격_최대', '일자별_somae가격_최소', '해당일자_전체평균가격']
15 품목의 selected_columns : ['단가', '거래량', '초기온도_0', '평균온도_0', '강수량_1', '습도_1', '평균온도_2', '일자별_domae가격_최대', '일자별_somae가격_최대', '일자별_somae가격_평균', '수출중량', '수출금액', '수입금액', '무역수지', '해당일자_전체평균가격
16 품목의 selected_columns : ['단가', '거래량', '평균온도_0', '습도_0', '초기온도_1', '최대온도_1', '평균온도_1', '습도_1', '초기온도_2', '최대온도_2', '최저온도_2', '평균온도_2', '해당일자_전체평균가격']
17 품목의 selected_columns : ['단가', '거래량', '최대온도_0', '최저온도_1', '최대온도_2', '일자별_somae가격_최대', '일자별_somae가격_평균', '수출중량', '무역수지', '해당일자_전체평균가격']
18 품목의 selected_columns : ['단가', '거래량', '최대온도_0', '평균온도_0', '습도_0', '초기온도_1', '초기온도_2', '강수량_2', '수출중량', '수출금액', '수입중량', '해당일자_전체평균가격']
19 품목의 selected_columns : ['단가', '거래량', '습도_0', '최저온도_1', '습도_1', '습도_2', '일자별_domae가격_평균', '일자별_domae가격_최소', '일자별_somae가격_평균', '수출금액', '수입중량', '수입금액', '무역수지', '해당일자_전체평균가격']
20 품목의 selected_columns : ['단가', '거래량', '최대온도_0', '최대온도_1', '일자별_somae가격_최대', '일자별_somae가격_최소', '수출중량', '수출금액', '수입중량', '수입금액', '무역수지', '해당일자_전체평균가격']
21 품목의 selected_columns : ['단가', '습도_0', '강수량_1', '강수량_2', '습도_2', '일자별_domae가격_평균', '일자별_domae가격_최소', '일자별_somae가격_최대', '일자별_somae가격_평균', '수출중량', '수출금액', '수입금액', '무역수지', '해당일자_전체평균가격']
22 품목의 selected_columns : ['단가', '거래량', '평균온도_0', '초기온도_1', '강수량_1', '습도_2', '일자별_domae가격_최대', '일자별_domae가격_최소', '일자별_somae가격_평균', '일자별_somae가격_최소', '수출중량', '수출금액', '수입중량', '수입금액', '해당일자_전
23 품목의 selected_columns : ['단가', '거래량', '평균온도_2', '강수량_2', '일자별_somae가격_평균', '해당일자_전체평균가격']
24 품목의 selected_columns : ['단가', '거래량', '최대온도_0', '최저온도_2', '일자별_somae가격_최대', '일자별_somae가격_평균', '일자별_somae가격_최소', '해당일자_전체평균가격']
25 품목의 selected_columns : ['단가', '거래량', '최대온도_0', '최저온도_0', '평균온도_0', '습도_0', '최대온도_1', '최저온도_1', '평균온도_1', '습도_2', '일자별_domae가격_최대', '일자별_somae가격_최대', '일자별_somae가격_평균', '수입중량', '무역수지', '해당일
26 품목의 selected_columns : ['단가', '거래량', '최대온도_0', '최저온도_1', '강수량_1', '최대온도_2', '습도_2', '일자별_domae가격_최대', '일자별_domae가격_평균', '일자별_domae가격_최소', '일자별_somae가격_최소', '해당일자_전체평균가격']
27 품목의 selected_columns : ['단가', '거래량', '일자별_somae가격_평균', '해당일자_전체평균가격']
28 품목의 selected_columns : ['단가', '거래량', '습도_0', '일자별_somae가격_최대', '수출중량', '수출금액', '수입금액', '해당일자_전체평균가격']
29 품목의 selected_columns : ['단가', '거래량', '최대온도_0', '초기온도_1', '일자별_domae가격_최소', '일자별_somae가격_평균', '수입중량', '해당일자_전체평균가격']
30 품목의 selected_columns : ['단가', '거래량', '최대온도_1', '평균온도_1', '습도_1', '최대온도_2', '강수량_2', '습도_2', '일자별_domae가격_최대', '일자별_domae가격_평균', '일자별_domae가격_최소', '일자별_somae가격_최대', '일자별_somae가격_평균', '일자별_so
31 품목의 selected_columns : ['단가', '거래량', '최대온도_0', '최대온도_1', '최저온도_1', '평균온도_1', '평균온도_2', '일자별_domae가격_평균', '일자별_domae가격_최소', '일자별_somae가격_평균', '해당일자_전체평균가격']
32 품목의 selected_columns : ['단가', '거래량', '일자별_domae가격_최대', '일자별_somae가격_최소', '수출중량', '수출금액', '수입중량', '수입금액', '무역수지', '해당일자_전체평균가격']
33 품목의 selected_columns : ['단가', '거래량', '습도_0', '습도_1', '일자별_domae가격_평균', '일자별_domae가격_최소', '일자별_somae가격_평균', '일자별_somae가격_최소', '수출중량', '수출금액', '수입중량', '수입금액', '무역수지', '해당일자_전체평균가격']
34 품목의 selected_columns : ['단가', '거래량', '초기온도_0', '일자별_somae가격_평균', '수출중량', '해당일자_전체평균가격']
35 품목의 selected_columns : ['단가', '거래량', '최저온도_0', '초기온도_1', '일자별_domae가격_최소', '일자별_somae가격_평균', '일자별_somae가격_최소', '수출중량', '수입중량', '해당일자_전체평균가격']
36 품목의 selected_columns : ['단가', '거래량', '최저온도_1', '일자별_domae가격_평균', '일자별_domae가격_최소', '일자별_somae가격_최대', '일자별_somae가격_평균', '수출중량', '해당일자_전체평균가격']
```

0.47 Lasso ⟶ 0.40 Ols Report

Lasso Regression 보다 Ols Report 가 성능이 좋다.

품목별 유의수준 0.05 기준으로 Feature Selection

Data Scaling

Standard Scaler + log 변환

```
def scaling_df(df):
    scaler = StandardScaler()
    tmp = df.drop('해당일자_전체평균가격',axis=1).copy()
    scaler.fit(tmp)
    df_scaled = scaler.transform(tmp)
    df_scaled = pd.DataFrame(data=df_scaled, columns = tmp.columns)
    df_scaled['log_target'] = np.log1p(df['해당일자_전체평균가격']).values
    return df_scaled
```

	단가	도매시장코드	도매법인코드	해당일자_전체거래물량	wma05	하위가격 평균가	하위가격 거래물량	상위가격 평균가	상위가격 거래물량	습도_0	log_target
0	-0.797712	-1.945569	-1.945569	-0.551863	3.087280	-1.572964	-0.594756	-1.096476	-0.486393	0.0	8.798415
1	0.746332	1.382945	1.382945	-0.547318	3.087280	2.147779	-0.588939	8.623102	-0.483026	0.0	9.938528
2	-0.320691	0.115156	0.115156	-0.440105	3.087280	0.045214	-0.398922	0.730612	-0.440263	0.0	8.132373
3	1.335574	0.115156	0.115156	-0.494228	3.087280	0.257782	-0.556462	0.905587	-0.419049	0.0	8.787969
4	0.096076	1.382945	1.382945	-0.297153	3.087280	0.755579	-0.121166	1.259532	-0.400193	0.0	8.366389
...	...	...	...	...	...	...	...	...	...	...	...
1456	-0.797712	-1.945569	-1.945569	-0.551863	2.052657	-1.572964	-0.594756	-1.096476	-0.486393	0.0	8.592987
1457	-0.105904	0.115156	0.115156	-0.222372	2.013347	0.541306	-0.191453	1.320815	-0.229475	0.0	8.720925
1458	1.717955	0.115174	0.115174	-0.461589	2.115606	1.605676	-0.487144	1.195429	-0.413998	0.0	8.801625
1459	2.070148	1.193001	1.193000	-0.351070	2.289488	1.402698	-0.338814	1.332949	-0.336889	0.0	8.790799

02
Data EDA

파생변수 생성 과정

계절 변수 추가

```
for a in temp["month"]:
    if (a > 2) and (a <= 5):
        day_set.append(0)

# 0이면 봄 / 1이면 여름 / 2면 가을 / 3이면 겨울
    elif (a > 5) and (a <= 8):
        day_set.append(1)
    elif (a > 8) and (a <= 11):
        day_set.append(2)
    else:
        day_set.append(3)
temp["season"] = day_set
globals()[f'{name}_total_sep_{num}_{i}'] = temp
```

월 변수 추가

```
temp['month'] = temp['datadate'].dt.month
temp=temp.set_index('datadate')
```

거래 가격 최대/최소/평균 추가

```
 sep.rename(columns={"가격(원)": "가격"}, inplace=True)
        sep2 = sqldf(
            f"select datadate, max(가격) as '일자별_{text}가격_최대', avg(가격) as '일자별_{text}가격_평균', min(가격) as '일자별_{text}가격_최소', avg(단위가격) as '일자별_{text}평균단위가격' from sep group by datadate")
```

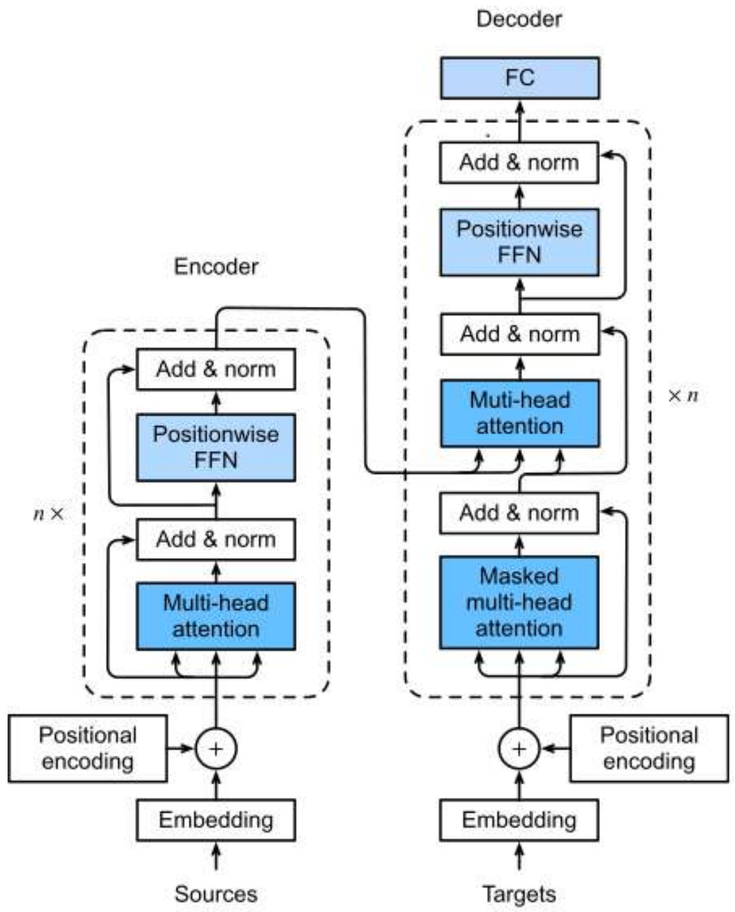
평균단가 변수추가

```
        pummok['high_low'] = pummok[['단가(원)','해당일자_전체평균가격(원)']].T.apply(lambda x: 1 if x['단가(원)'] >  =x['해당일자_전체평균가격(원)'] else 0)
        pummok_high=pummok[pummok['high_low']==1].groupby('datadate').sum()[['거래량','거래대금(원)']]
        pummok_low=pummok[pummok['high_low']==0].groupby('datadate').sum()[['거래량','거래대금(원)']]
        sep2=sep2.set_index('datadate')
        sep2['하위가격 평균가']=pummok_low['거래대금(원)']/pummok_low['거래량']
        sep2['하위가격 거래물량'] = pummok_low['거래량']
        sep2['상위가격 평균가']=pummok_high['거래대금(원)']/pummok_high['거래량']
        sep2['상위가격 거래물량'] = pummok_high['거래량']
```

03

MODELING Transformer

트랜스포머(Transformer)는 2017년 구글이 발표한 논문인 "Attention is all you need"에서 나온 모델로 기존의 seq2seq의 구조인 인코더-디코더를 따르면서도, 논문의 이름처럼 어텐션(Attention)만으로 구현한 모델이다. 이 모델은 RNN을 사용하지 않고, 인코더-디코더 구조를 설계하였음에도 번역 성능에서도 RNN보다 우수한 성능을 보여주었다.
Image classification 이나 Image detection, Image retrieval 등 NLP를 넘어 컴퓨터 비전분야까지 범위를 넓히고 있다.



03

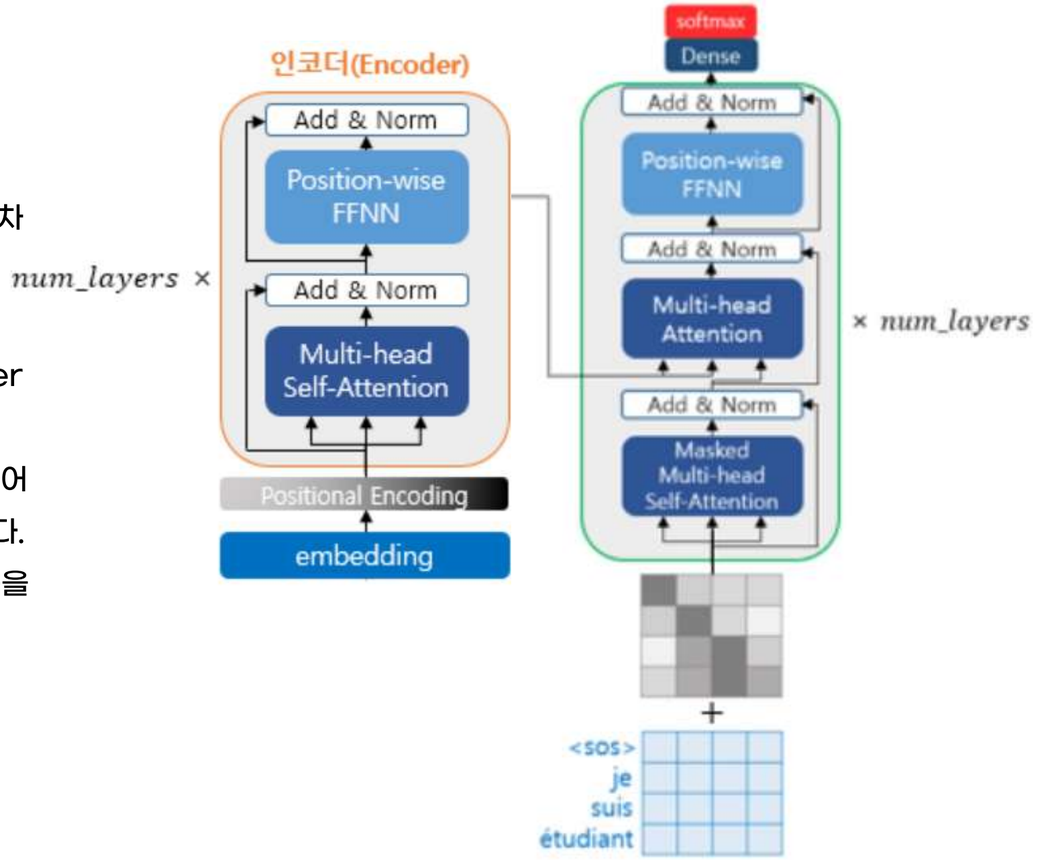
MODELING

Transformer

인코더와 디코더

인코더와 디코더 여러 개를 중첩한 구조를 갖고 있다는 점에서 이전 모델과의 차이점을 갖는다. 각각의 인코더와 디코더를 블록(Block)이라고 일컫는다

각 Encoder는 그것의 출력을 그 다음 Encoder에 보내고 마지막 Encoder는 입력 시퀀스의 표현(Representation)을 출력한다.
트랜스포머 구조는 인코더로부터 정보를 전달받아 디코더가 출력 결과를 만들어낸다. 디코더는 마치 기존의 seq2seq 구조처럼 입력을 받아 연산을 진행한다.
이는 RNN은 사용되지 않지만 여전히 인코더-디코더의 구조는 유지되고 있음을 보여준다,

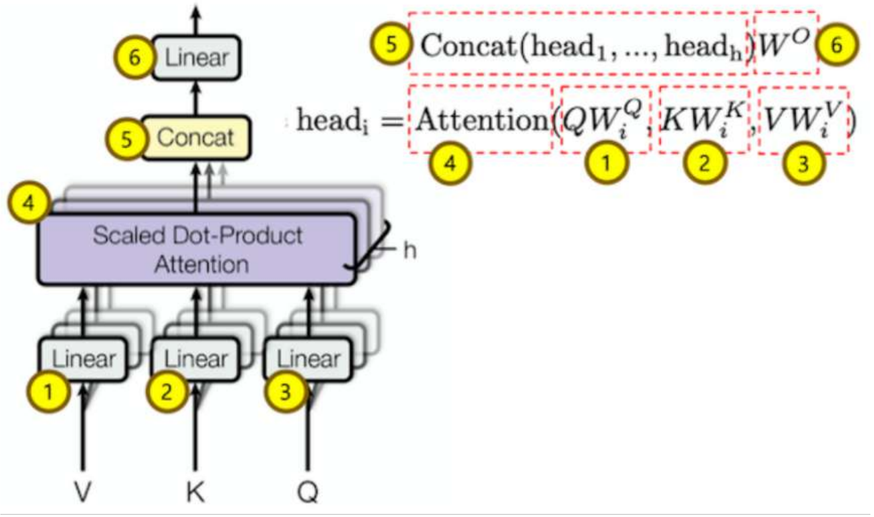


03

MODELING

Transformer

Multi-Head Attention



Scaled dot-product attention의 계산 수식

$$4\ soft\ \max\left(\frac{1\ QK^T}{\sqrt{d_k}\ 2}\right)V\ 5$$

멀티 헤드: 여러 개의 셀프 어텐션을 병렬로 수행

각 헤드(Single Self-Attention)의 계산을 서로 독립적으로 수행 가능하다.

Multi-head Attention 계산

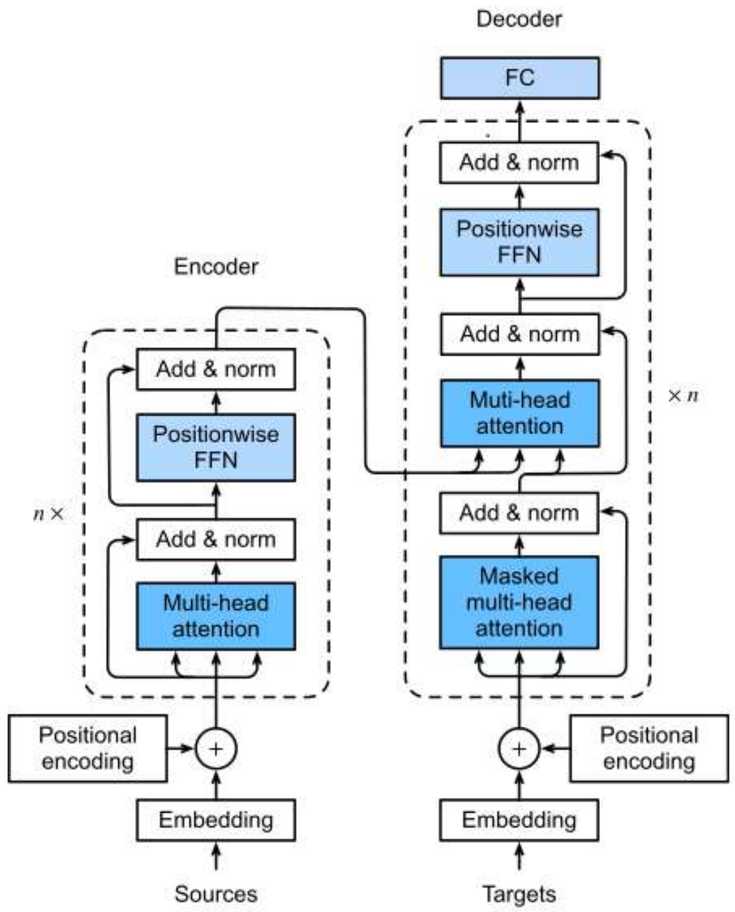
1. Linear 연산을 취하고 이를 각각의 Head로 나눈다.
4. 각각의 Head에서 Scaled dot-product attention을 수행한다.
5. 각 head에서 일어나는 연산을 Concatenation한다.
6. 마지막 linear layer

03

MODELING

Transformer

```
def build_model(input_shape, head_size, num_heads, ff_dim, num_trans
former_blocks, mlp_units, dropout=0, mlp_dropout=0):
    inputs = keras.Input(shape=input_shape)
    x = inputs
    for _ in range(num_transformer_blocks):
        x = transformer_encoder(x, head_size, num_heads, ff_dim, dropout)
    x = layers.GlobalAveragePooling1D(data_format="channels_first")(x)
    for dim in mlp_units:
        x = layers.Dense(dim, activation="relu")(x)
        x = layers.Dropout(mlp_dropout)(x)
    outputs = layers.Dense(28)(x) # 4주 예측
    return keras.Model(inputs, outputs)
```



모델 개선 총 18회

제출일	모델이름	공개점수	성공여부
22.09.28 22:54:34	니가가라하와이 _220928_225346	0.408276	성공
22.09.28 22:53:48	니가가라하와이 _220928_225339	-	실패
22.09.28 17:37:19	니가가라하와이 _220928_162611	0.4734248	성공
22.09.28 16:26:09	니가가라하와이 _220928_162605	0.4774207	성공
22.09.28 12:59:16	니가가라하와이 _220928_125909	0.3851827	성공
22.09.28 11:23:13	니가가라하와이 _220928_112257	0.4448069	성공
22.09.28 11:16:56	니가가라하와이 _220928_111602	-	실패
22.09.28 09:58:36	니가가라하와이 _220928_095831	0.3868508	성공
22.09.27 19:34:47	니가가라하와이 _220927_191538	0.383511	성공
22.09.27 19:15:35	니가가라하와이 _220927_191520	-	실패
22.09.27 15:03:07	니가가라하와이 _220927_150257	0.3856325	성공
22.09.24 01:54:49	니가가라하와이 _220924_015443	1.6602851	성공
22.09.23 10:20:10	니가가라하와이 _220923_102002	0.8221895	성공
22.09.12 14:33:03	니가가라하와이 _220912_133255	0.5709514	성공
22.09.05 19:35:26	니가가라하와이 _220905_193520	0.5229937	성공

03

02

01

제출일	모델이름	공개점수	성공여부
22.09.30 10:49:42	ATT_220930_104923	0.3518988	성공
22.09.29 19:06:16	ATT_220929_150221	0.377318	성공
22.09.29 15:02:16	ATT_220929_124429	0.3882336	성공
22.09.29 11:36:53	ATT_220929_113647	0.3965971	성공
22.09.27 21:52:35	ATT_220927_212222	-	실패
22.09.27 21:22:21	ATT_220927_212206	-	실패
22.09.26 21:11:11	ATT_220926_211057	0.3736803	성공
22.09.22 17:48:05	ATT_220922_174715	0.7173788	성공

04

시도 01- MAE Score: 0.57

주산지 날씨 데이터의 습도: 00이 많아서 drop
모든 데이터의 0 값은 4년치 평균으로 대치
Baseline 코드 사용

시도 02- MAE Score : 0.82

습도 데이터 0으로 대치 후 추가
타겟- log scale
변수- standard scale 적용

모델 개선 총 18회

제출일	모델이름	공개점수	성공여부
22.09.28 22:54:34	니가가라하와이 _220928_225346	0.408276	성공
22.09.28 22:53:48	니가가라하와이 _220928_225339	-	실패
22.09.28 17:37:19	니가가라하와이 _220928_162611	0.4734248	성공
22.09.28 16:26:09	니가가라하와이 _220928_162605	0.4774207	성공
22.09.28 12:59:16	니가가라하와이 _220928_125909	0.3851827	성공
22.09.28 11:23:13	니가가라하와이 _220928_112257	0.4448069	성공
22.09.28 11:16:56	니가가라하와이 _220928_111602	-	실패
22.09.28 09:58:36	니가가라하와이 _220928_095831	0.3868508	성공
22.09.27 19:34:47	니가가라하와이 _220927_191538	0.383511	성공
22.09.27 19:15:35	니가가라하와이 _220927_191520	-	실패
22.09.27 15:03:07	니가가라하와이 _220927_150257	0.3856325	성공
22.09.24 01:54:49	니가가라하와이 _220924_015443	1.6602851	성공
22.09.23 10:20:10	니가가라하와이 _220923_102002	0.8221895	성공
22.09.12 14:33:03	니가가라하와이 _220912_133255	0.5709514	성공
22.09.05 19:35:26	니가가라하와이 _220905_193520	0.5229937	성공

03

02

01

제출일	모델이름	공개점수	성공여부
22.09.30 10:49:42	ATT_220930_104923	0.3518988	성공
22.09.29 19:06:16	ATT_220929_150221	0.377318	성공
22.09.29 15:02:16	ATT_220929_124429	0.3882336	성공
22.09.29 11:36:53	ATT_220929_113647	0.3965971	성공
22.09.27 21:52:35	ATT_220927_212222	-	실패
22.09.27 21:22:21	ATT_220927_212206	-	실패
22.09.26 21:11:11	ATT_220926_211057	0.3736803	성공
22.09.22 17:48:05	ATT_220922_174715	0.7173788	성공

04

시도 03- MAE Score: 0.47

거래 내역이 없는 화요일 가격 데이터 0으로 대치
파생 변수 추가: 계절, 월
거래 가격 최대/최소/평균
평균단가 변수추가

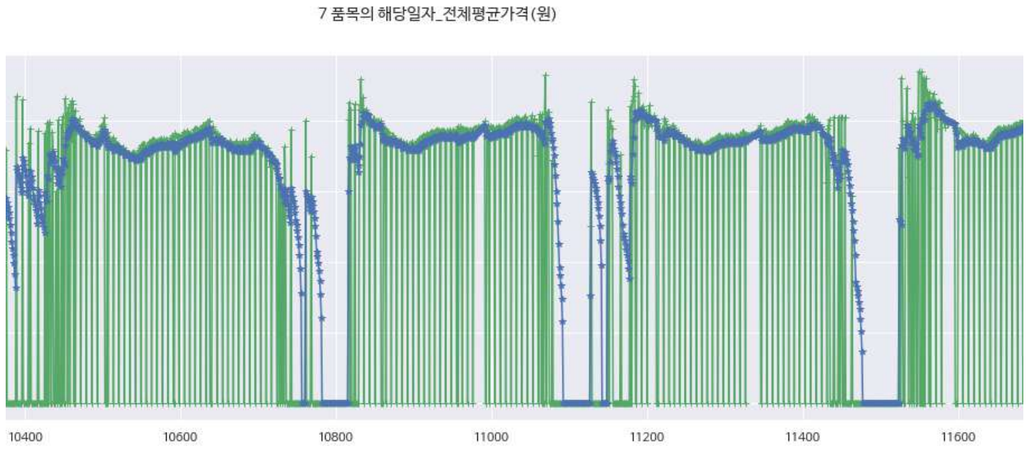
타겟- log scale
변수- standard scale 적용
Lasso regression feature selection적용

품목 별 가격 데이터

	품목0	품목1	품목2	품목3	품목4	품목5
0	3871.125000	1362.117613	2909.783785	3400.075583	3947.809169	9253.947514
1	971.092350	1900.397600	1512.204300	3305.641400	1075.473100	2652.604500
2	7994.371000	300.648500	2095.661000	1671.990500	4160.522500	389.472170
3	0.061326	2.833791	0.611477	-0.461002	0.148949	120.073890
4	7460.459000	8240.599000	7303.744000	5576.103500	10153.708000	8830.293000
5	10157.980000	9099.087000	4516.143600	3144.575000	6883.584000	22029.730000
6	11813.398000	12134.480000	2589.355700	4765.356400	5451.572300	16602.125000
7	13942.736000	3971.536000	6466.801000	5845.348600	5291.869000	53768.760000
8	3173.517800	1818.167100	1274.578400	3546.257800	759.361400	1753.453200
9	20746.436000	248.761600	1766.043700	1718.775900	2455.926300	213.967270
10	0.832743	1.740247	0.393107	-0.526571	-0.543422	67.868030
11	9001.332000	6755.576700	6881.194300	5761.327000	5048.908000	6884.483400
12	9402.553000	8674.906000	4640.172400	3756.346200	4395.777000	21689.520000
13	8765.638000	10562.799000	2251.796900	5381.314500	4944.975600	14422.710000
14	9457.263000	3589.704800	5010.111300	6080.101600	5973.201000	44947.477000
15	1932.159900	1425.153600	1030.310200	4020.479200	881.636300	1775.592300
16	13494.104500	213.968400	1729.905400	2164.244000	2532.817900	241.343340
17	0.557717	1.372246	0.194808	-0.593657	-0.490347	69.150024
18	8187.442400	6119.344700	5504.093800	5664.027300	6695.584500	6612.657700
19	9441.464000	7392.825700	3484.614700	3587.743200	5891.921000	22581.336000
20	8369.595000	10338.410000	1953.910500	5335.020000	6630.158000	16798.494000
21	10014.023000	3706.690400	4995.904300	5840.667000	8143.781200	51232.684000

문제상황 : 가격이 0.xx 인 값이 있다

화요일 : 거래 발생 안하는 날이다
거래 발생 없는 날을 0으로 채워주면 잡음이 정보에 비해 큰 영향을
미치기 때문에 가격 변동 흐름을 왜곡시킨다
가격 변동 흐름을 개선하기 위해 smoothing 해야할 필요성을 느꼈다.



WMA

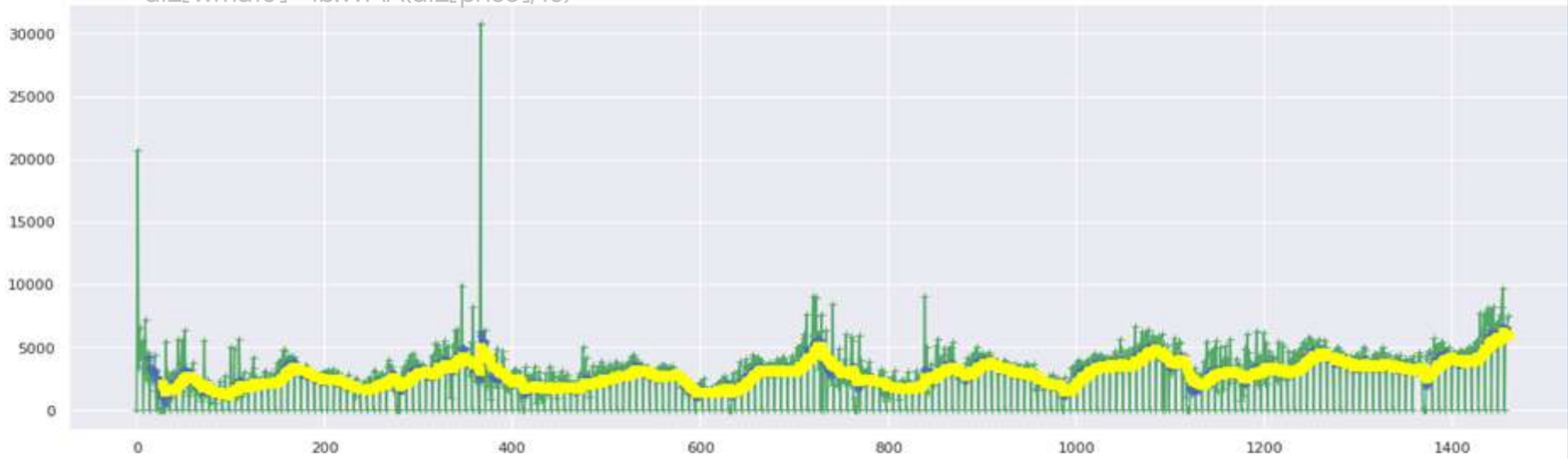
Weighted Moving Average
가중치 이동 평균선

가격 변동 흐름을 개선하기 위해 smoothing 필요성을 느낌

품목 가격 데이터를 WMA 로 대치

1주, 2주 ,4주 단위 수행 후, 2주차 (WMA =14) 점수가 가장 높게 나왔다.

```
for i in range(37):
  df2 = df[df['품목']==i]
  df2['해당일자_전체평균가격(원)'].replace(0,np.nan, inplace = True)
  df2.dropna(axis=0, inplace=True)
  df2['price'] = df['해당일자_전체평균가격(원)']
  df2['wma7'] = tb.WMA(df2['price'], 7)
  df2['wma14'] = tb.WMA(df2['price'], 14)
  df2['wma13'] = tb.WMA(df2['price'], 13)
  df2['wma12'] = tb.WMA(df2['price'], 12)
  df2['wma28'] = tb.WMA(df2['price'], 28)
  df2['wma10'] = tb.WMA(df2['price'], 10)
```



WMA=14

모델 개선 총 18회

제출일	모델이름	공개점수	성공여부
22.09.28 22:54:34	니가가라하와이 _220928_225346	0.408276	성공
22.09.28 22:53:48	니가가라하와이 _220928_225339	-	실패
22.09.28 17:37:19	니가가라하와이 _220928_162611	0.4734248	성공
22.09.28 16:26:09	니가가라하와이 _220928_162605	0.4774207	성공
22.09.28 12:59:16	니가가라하와이 _220928_125909	0.3851827	성공
22.09.28 11:23:13	니가가라하와이 _220928_112257	0.4448069	성공
22.09.28 11:16:56	니가가라하와이 _220928_111602	-	실패
22.09.28 09:58:36	니가가라하와이 _220928_095831	0.3868508	성공
22.09.27 19:34:47	니가가라하와이 _220927_191538	0.383511	성공
22.09.27 19:15:35	니가가라하와이 _220927_191520	-	실패
22.09.27 15:03:07	니가가라하와이 _220927_150257	0.3856325	성공
22.09.24 01:54:49	니가가라하와이 _220924_015443	1.6602851	성공
22.09.23 10:20:10	니가가라하와이 _220923_102002	0.8221895	성공
22.09.12 14:33:03	니가가라하와이 _220912_133255	0.5709514	성공
22.09.05 19:35:26	니가가라하와이 _220905_193520	0.5229937	성공

03

02

01

제출일	모델이름	공개점수	성공여부
22.09.30 10:49:42	ATT_220930_104923	0.3518988	성공
22.09.29 19:06:16	ATT_220929_150221	0.377318	성공
22.09.29 15:02:16	ATT_220929_124429	0.3882336	성공
22.09.29 11:36:53	ATT_220929_113647	0.3965971	성공
22.09.27 21:52:35	ATT_220927_212222	-	실패
22.09.27 21:22:21	ATT_220927_212206	-	실패
22.09.26 21:11:11	ATT_220926_211057	0.3736803	성공
22.09.22 17:48:05	ATT_220922_174715	0.7173788	성공

04

시도 04- MAE Score : 0.37

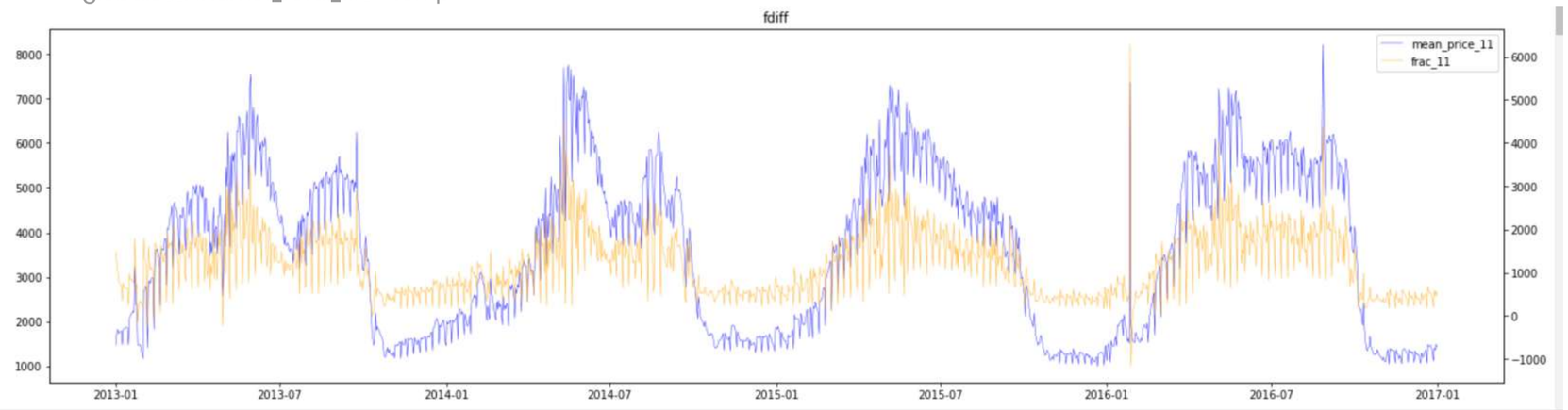
타겟- log scale
변수- standard scale
거래 내역이 없는 화요일 가격 데이터: WMA(14) 로 대치
Ols Report 로 피쳐 셀렉션

파생변수 추가 :
거래 가격 최대/최소/평균
평균단가 변수추가

소수점 차분 Fractional Differencing

차분 :시계열 데이터에서 나타나는 비정상성 특징을 정상성 특징으로 전환.
일반적인 정수 차원의 차분은 시계열 데이터의 정상성을 확보하지만 메모리를 지워버리는 특징이 있다.
시계열 데이터의 정상성을 어느정도 확보하고 메모리를 기억할 수 있는 실수 차원의 차분을 활용하였다.

```
a = fdiff(temp['해당일자_전체평균가격(원)'], n=0.35)
        diff = pd.DataFrame(a, index=temp.index, columns=["fracdiff"])
        temp['fracdiff'] = diff['fracdiff']
        temp.replace([np.inf, -np.inf], np.nan, inplace=True)
        temp.fillna(0,inplace=True)
        globals()[f'{name}_total_{i}'] = temp
```



N=0.35

04 결론

제출일	모델이름	공개점수	성공여부
22.09.28 22:54:34	니가가라하와이 _220928_225346	0.408276	성공
22.09.28 22:53:48	니가가라하와이 _220928_225339	-	실패
22.09.28 17:37:19	니가가라하와이 _220928_162611	0.4734248	성공
22.09.28 16:26:09	니가가라하와이 _220928_162605	0.4774207	성공
22.09.28 12:59:16	니가가라하와이 _220928_125909	0.3851827	성공
22.09.28 11:23:13	니가가라하와이 _220928_112257	0.4448069	성공
22.09.28 11:16:56	니가가라하와이 _220928_111602	-	실패
22.09.28 09:58:36	니가가라하와이 _220928_095831	0.3868508	성공
22.09.27 19:34:47	니가가라하와이 _220927_191538	0.383511	성공
22.09.27 19:15:35	니가가라하와이 _220927_191520	-	실패
22.09.27 15:03:07	니가가라하와이 _220927_150257	0.3856325	성공
22.09.24 01:54:49	니가가라하와이 _220924_015443	1.6602851	성공
22.09.23 10:20:10	니가가라하와이 _220923_102002	0.8221895	성공
22.09.12 14:33:03	니가가라하와이 _220912_133255	0.5709514	성공
22.09.05 19:35:26	니가가라하와이 _220905_193520	0.5229937	성공

03

02

01

제출일	모델이름	공개점수	성공여부
22.09.30 10:49:42	ATT_220930_104923	0.3518988	성공
22.09.29 19:06:16	ATT_220929_150221	0.377318	성공
22.09.29 15:02:16	ATT_220929_124429	0.3882336	성공
22.09.29 11:36:53	ATT_220929_113647	0.3965971	성공
22.09.27 21:52:35	ATT_220927_212222	-	실패
22.09.27 21:22:21	ATT_220927_212206	-	실패
22.09.26 21:11:11	ATT_220926_211057	0.3736803	성공
22.09.22 17:48:05	ATT_220922_174715	0.7173788	성공

05

04

최종모델 05- MAE Score : 0.35

타겟- log scale
변수- Standard scale
거래 내역이 없는 화요일 가격 데이터: WMA(28) 로 대치
Ols Report 로 피쳐 셀렉션
소수점 차분 적용 (n=0.35)
파생변수 추가 :
거래 가격 최대/최소/평균
평균단가 변수추가

05 참고문헌

https://neptune.ai/blog/comprehensive-guide-to-transformers
https://wikidocs.net/162098 -------attention 관련 자료
https://wikidocs.net/31379

Vaswani A. Shazeer N. Parmar N. Uszkoreit J. Jones L. Go
mez A. N. ... & Polosukhin I. (2017). Attention is all you need. A
dvances in neural information processing systems 30.

Thank you