수산물 수입가격 예측

팀 명 : 만조

팀장 : 류 영 표 (youngpyoryu@dongguk.edu)
팀원 : 김 수 빈 (ksb.forest@gmail.com)
팀원 : 박 정 열 (pde159@naver.com)
팀원 : 양 재 영 (didwodud1025@naver.com)

프로젝트 소개

- 목표 : 수산물 수입가격 예측을 통한 최적의 가격 예측 모형 도출

➡ 가격 안정화가 되면 시장에 원활한 수급과 적정한 가격 유지가 됨.

학습데이터 | 검증데이터 | 평가 데이터

2015.12.28 ~ 2019.12.30 | 2020.01 ~ 2020.12 | 2021.01 ~ 2021.06

- 분석 방향

1. 수산물 가격을 머신러닝을 통해 가격을 예측이 가능할 것이다.
2. 다양한 수산물을 통해 시장 가격을 예측 할 수 있을 것이다.

- 평가 지표

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

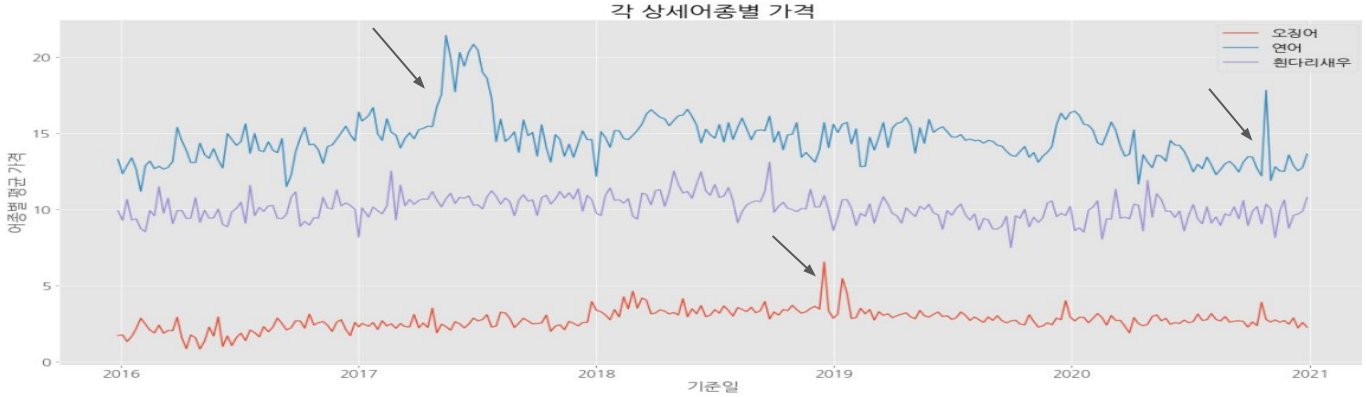
- 실험이나 관측에서 나타나는 오차(Error)를 제곱(Square)해서 평균(Mean)한 값의 제곱근(Root)을 의미합니다.

1.

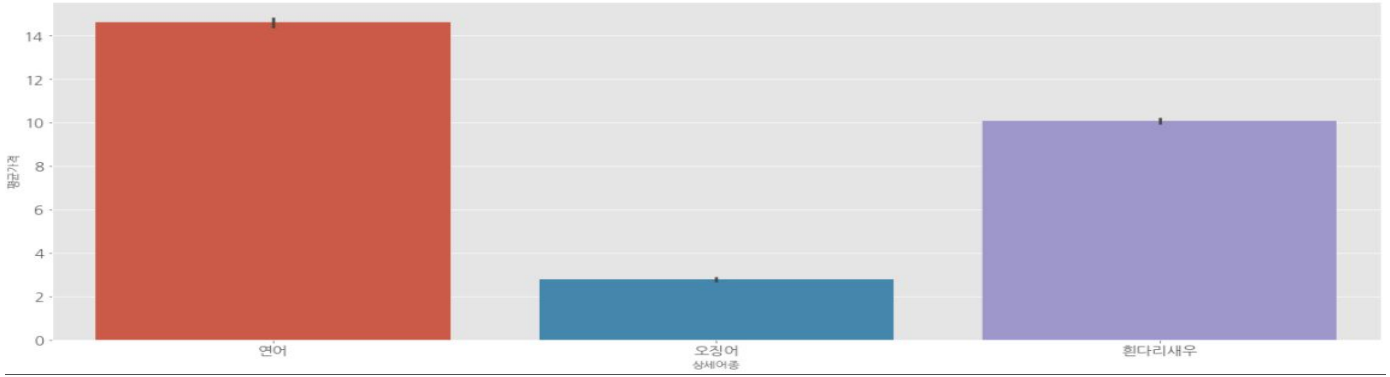
EDA (Exploratory Data Analysis, 탐색적 데이터 분석)

EDA

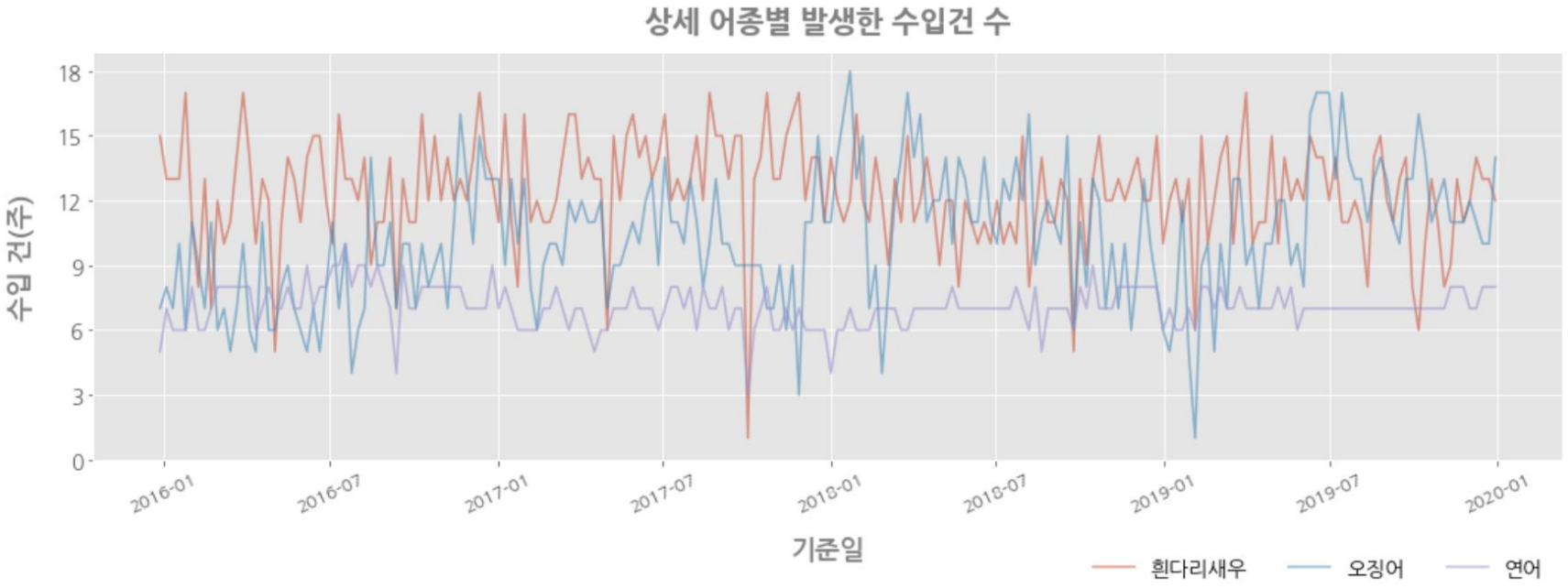
다른 기간에 이상치 발견



상세어종별 평균



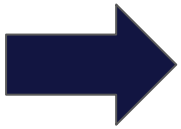
EDA


상세 어종별 발생한 수입건 수
수입 건(주)
18
15
12
9
6
3
0
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
기준일
흰다리새우
오징어
연어

EDA

```
# 날씨 변수 중 이상치 변환
train['기준일'] = train['기준일'].apply(lambda x:
pd.Timestamp('2017-01-02') if x.strftime('%Y-%m-%d')=='2017-01-01' else
x)

train['기준일'] = train['기준일'].apply(lambda x:
pd.Timestamp('2017-01-09') if x.strftime('%Y-%m-%d')=='2017-01-06' else
x)
```

기준일	평균단가($)
2016-12-26	8.787198
2017-01-01	7.032494
2017-01-06	8.511175
2017-01-16	8.981492



기준일	평균단가($)
2016-12-26	8.787198
2017-01-02	7.032494
2017-01-09	8.511175
2017-01-16	8.981492

학습 및 훈련 데이터 중 “월요일”이 아닌 날짜

“2017-01-01”

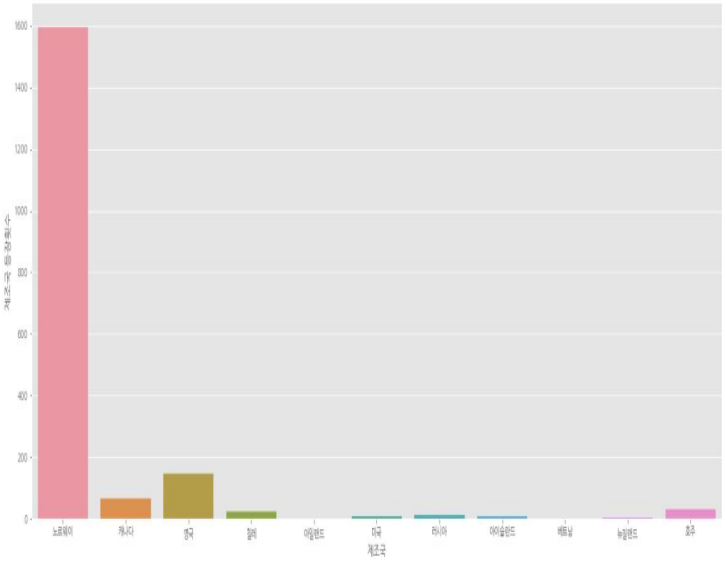
“2017-01-06”

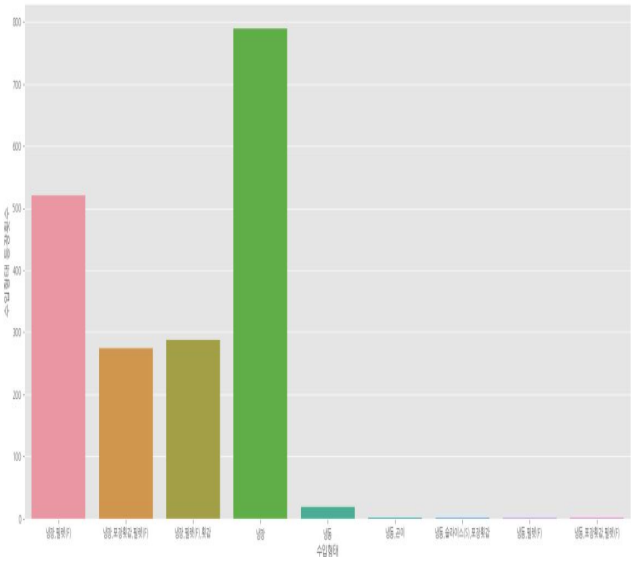
주별(“기준일”) 평균이 크게 차이가 나지 않는다고 판단했기 때문에 전후를 확인한 뒤, 월요일 날짜로 바꿈

EDA_연어

연어

- 연어는 노르웨이의 비율이 압도적으로 높고 냉장 형태로 많이 수입이 된다.

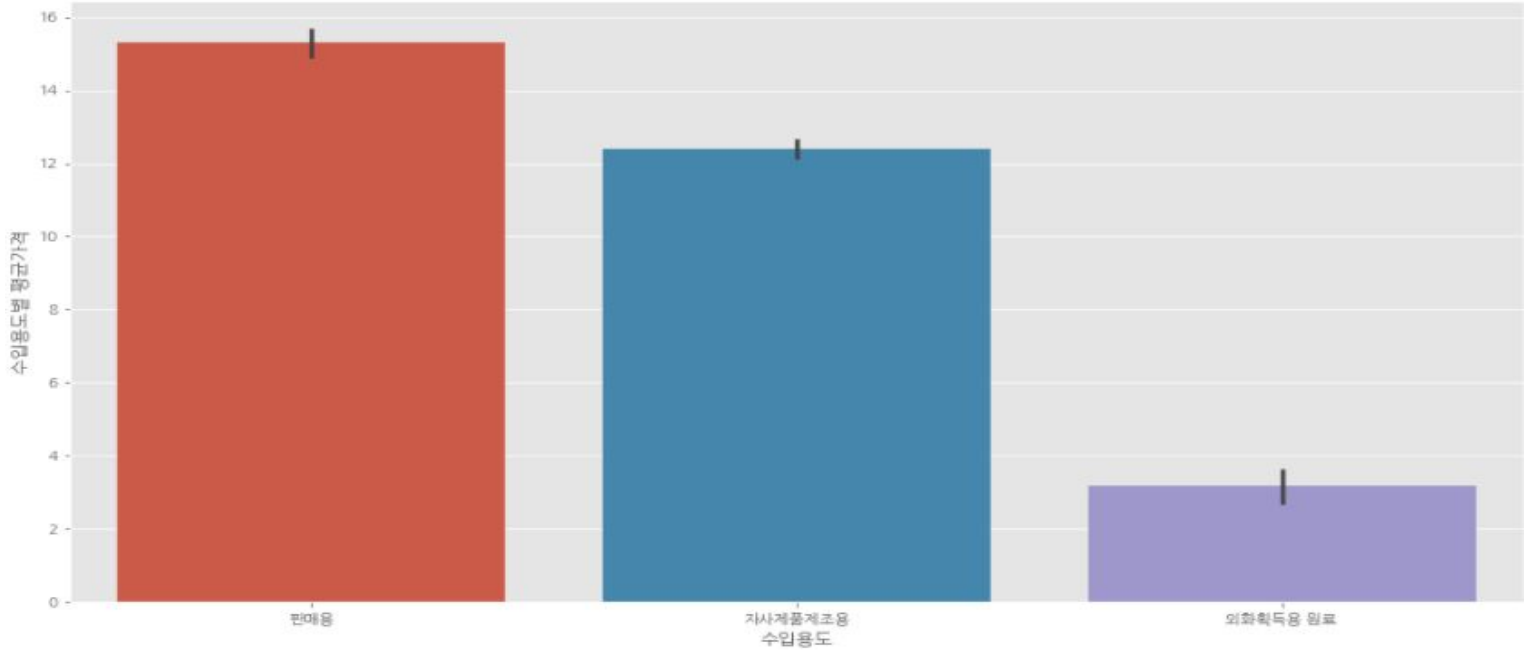




EDA_연어

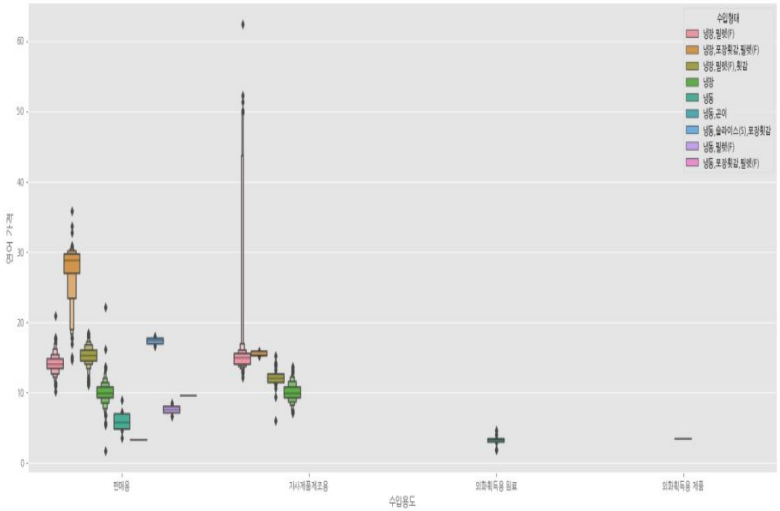
연어

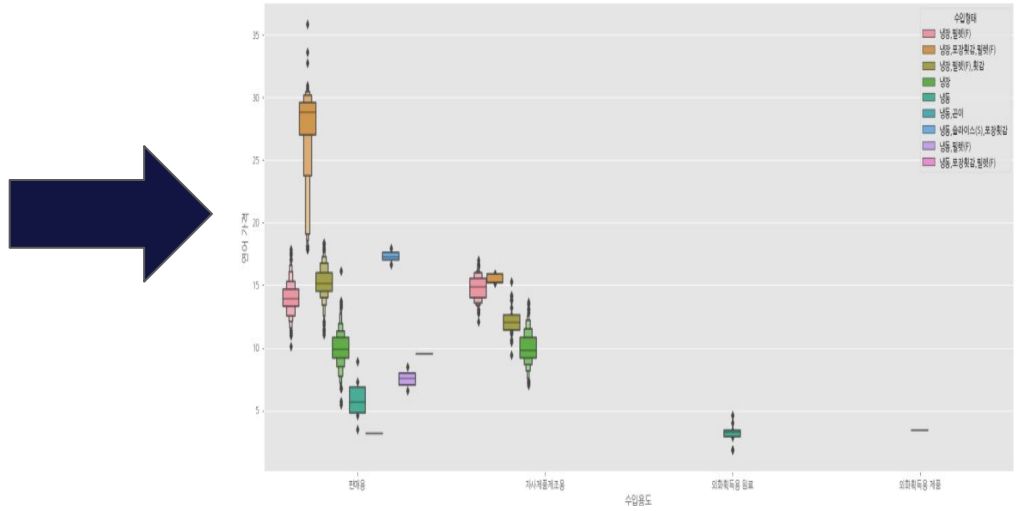
- 수입용도는 판매용 > 자사제품 제조용 > 외화획득용 원료 순으로 평균 가격이 높다.



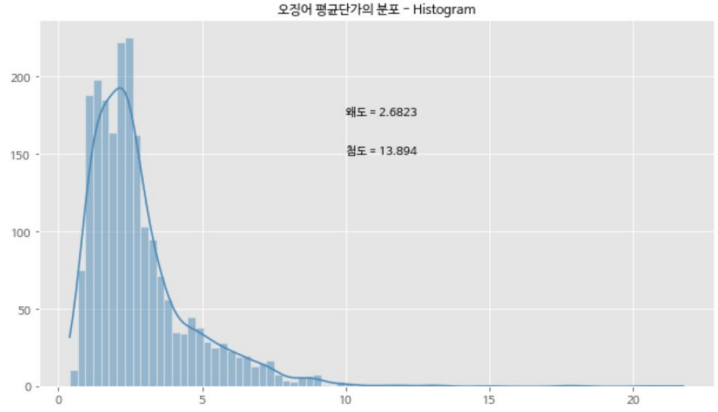
EDA_연어 이상치 제거

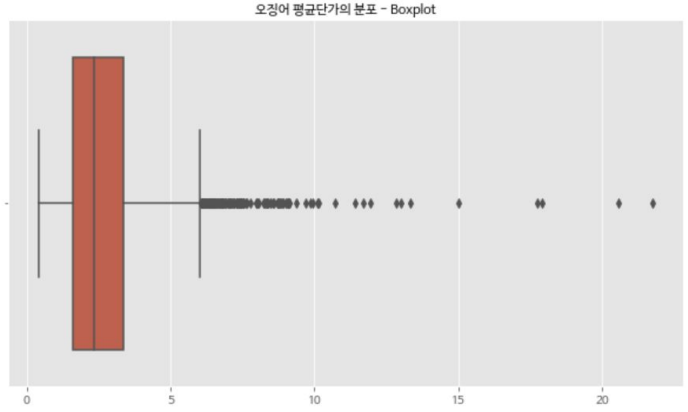
1. IQR(Interquartile range)방법을 사용하여 이상치를 NaN(Not a Number)값으로 변경한다.
2. Interpolation(보간법)을 사용하여 NaN(Not a Number)값을 채워준다.

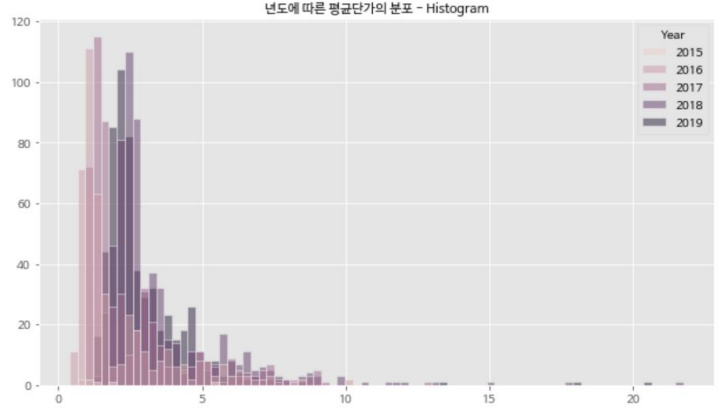


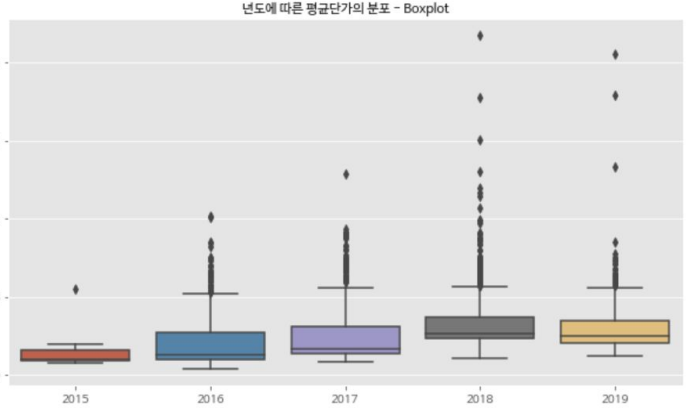


EDA_오징어

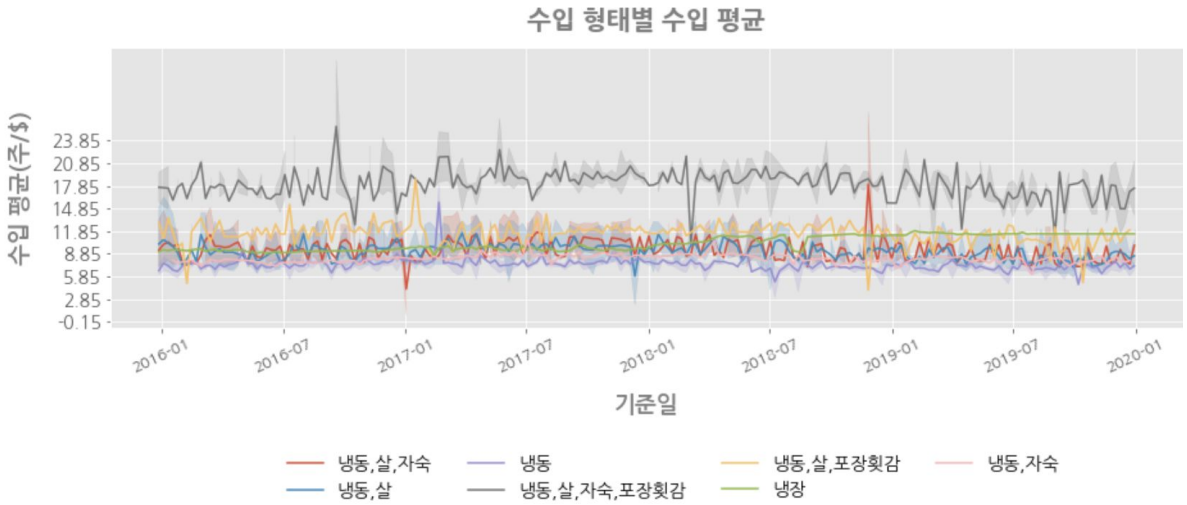

오징어 평균단가의 분포 - Histogram
왜도 = 2.6823
첨도 = 13.894

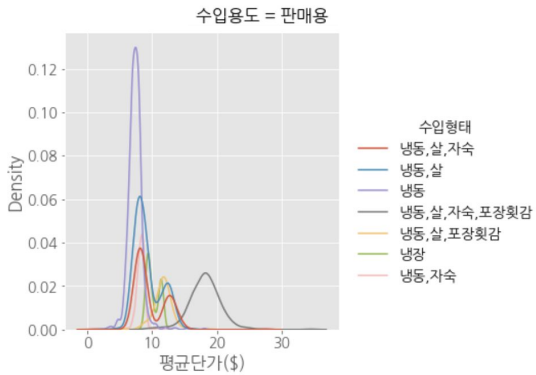

오징어 평균단가의 분포 - Boxplot

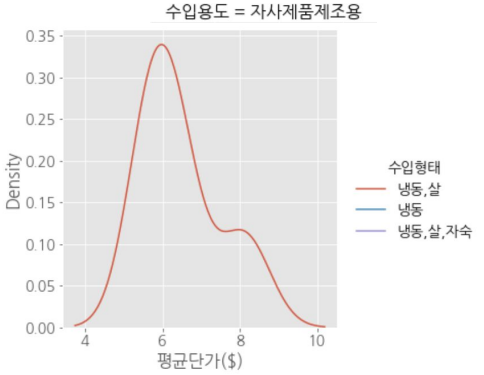

년도에 따른 평균단가의 분포 - Histogram
Year
2015
2016
2017
2018
2019


년도에 따른 평균단가의 분포 - Boxplot

EDA - 새우


수입 형태별 수입 평균
수입 평균(주/$)
23.85
20.85
17.85
14.85
11.85
8.85
5.85
2.85
-0.15
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
기준일
냉동,살,자숙
냉동
냉동,살,포장횟감
냉동,자숙
냉동,살
냉동,살,자숙,포장횟감
냉장


수입용도 = 판매용
Density
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
10
20
30
평균단가($)
수입형태
냉동,살,자숙
냉동,살
냉동
냉동,살,자숙,포장횟감
냉동,살,포장횟감
냉장
냉동,자숙

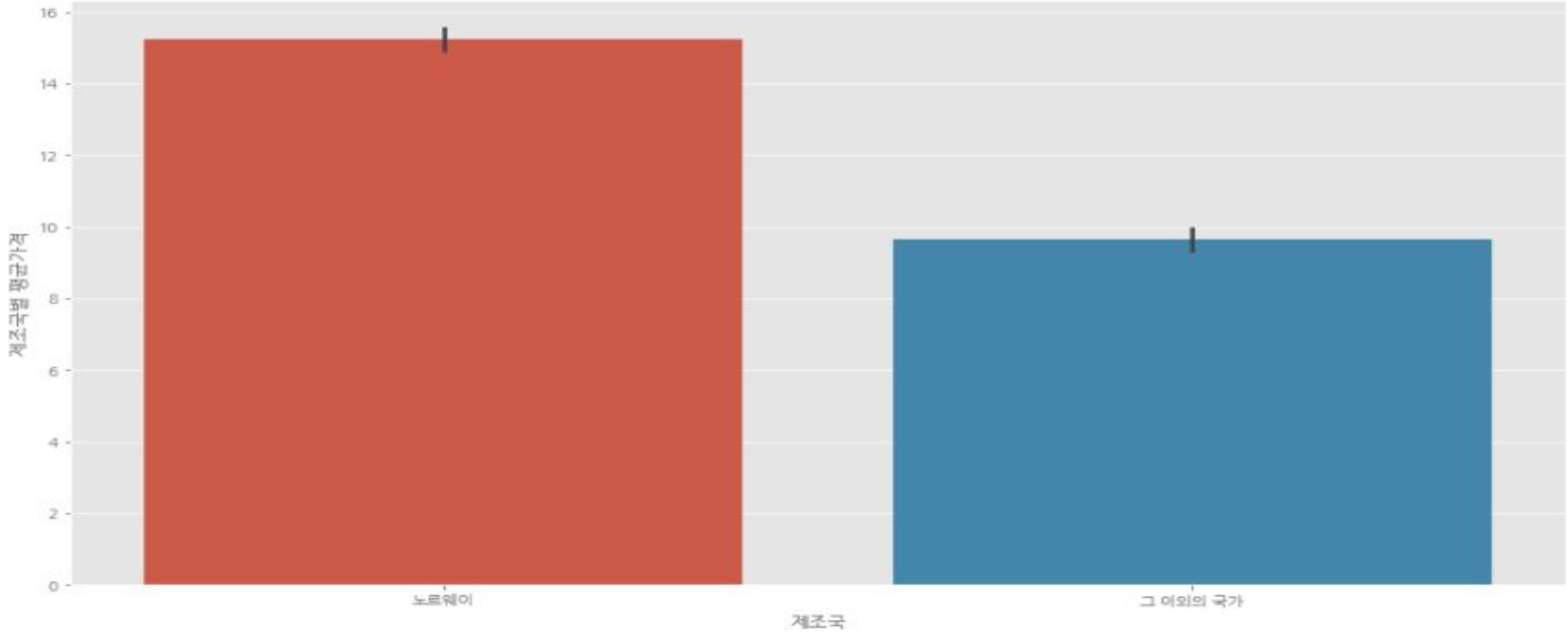

수입용도 = 자사제품제조용
Density
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
4
6
8
10
평균단가($)
수입형태
냉동,살
냉동
냉동,살,자숙

1.

Feature Engineering(FE, 특성 공학)

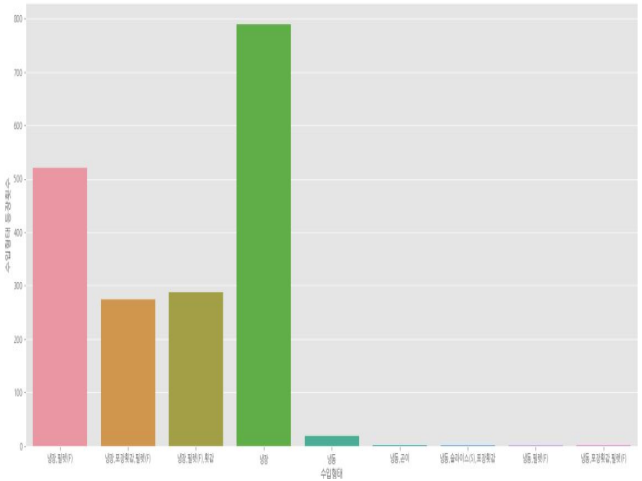
FE_연어

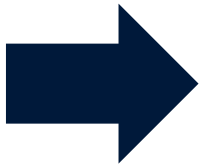
- 노르웨이를 제외한 데이터들이 많이 없으므로 노르웨이와 그 이외의 국가로 바꿔준다.

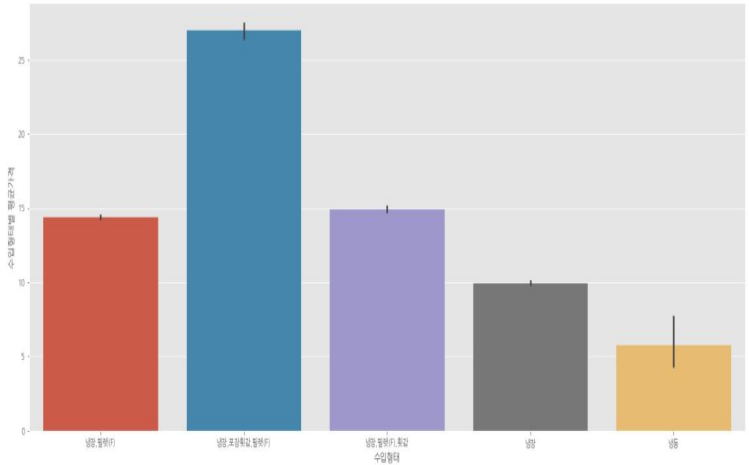


FE_연어

- 냉장의 데이터보다 냉동의 데이터가 적으므로 냉동으로 시작하는 모든 변수를 냉동으로 묶어준다.



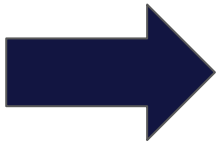




FE_연어

- 수입형태를 냉장, 필렛(F), 포장횟감, 횟감, 냉동으로 데이터를 가공 한다.

기준일	수입형태
2015-12-28	냉장,필렛(F)
2015-12-28	냉장,포장횟감,필렛(F)
2015-12-28	냉장,필렛(F),횟감
2015-12-28	냉장,필렛(F)
2015-12-28	냉장



	기준일	필렛(F)	냉동	냉장	횟감	포장횟감
0	2015-12-28	1	0	1	0	0
1	2015-12-28	1	0	1	0	1
2	2015-12-28	1	0	1	1	0
3	2015-12-28	1	0	1	0	0
4	2015-12-28	0	0	1	0	0

FE_연어

수입형태, 수입용도, 제조국_노르웨이의 week별 등장 횟수의 평균을 최종 데이터의 week에 맞춰서 넣어준다.

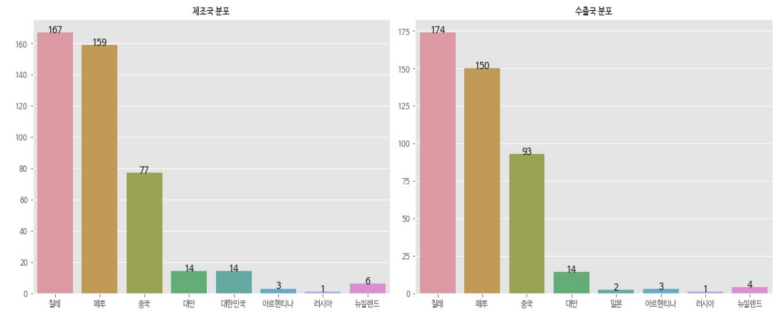
```
# 수입용도로 데이터를 week별로 groupby를 해주자.
수입용도_df = pd.get_dummies(data = df, columns =
['수입용도']).reset_index()[['기준일','week','수입용도_판매용','수입용도_자사제
품제조용','수입용도_외화획득용 원료']]
수입용도_df = 수입용도_df.groupby('week').mean().reset_index()

# 제조국을 week과 합쳐주기 위해서 데이터를 만들어 준다.
제조국_df = pd.get_dummies(df, columns = ['제조국'], drop_first =
True)[['기준일','제조국_노르웨이','week']]
제조국_df = 제조국_df.groupby('week').mean().reset_index()

# 수입형태의 빈도수를 계산한다.
수입형태_df_mean = df.groupby('week').mean()[['냉장', '필렛(F)', '횟감',
'포장횟감', '냉동']].reset_index()

# 최종 데이터에 넣어준다.
data = pd.merge(data, 수입형태_df_mean ,how ='left', on = 'week')

data = pd.merge(data, 제조국_df, how = 'left', on = 'week')

data = pd.merge(data, 수입용도_df, how = 'left', on = 'week')
```

week	냉장	필렛(F)	횟감	포장횟감	냉동	week	제조국_노르웨이
1	1.0	0.545455	0.121212	0.121212	0.0	1	0.848485
2	1.0	0.600000	0.120000	0.160000	0.0	2	0.920000
3	1.0	0.606061	0.151515	0.151515	0.0	3	0.909091
4	1.0	0.677419	0.193548	0.161290	0.0	4	0.967742
5	1.0	0.600000	0.171429	0.142857	0.0	5	0.885714

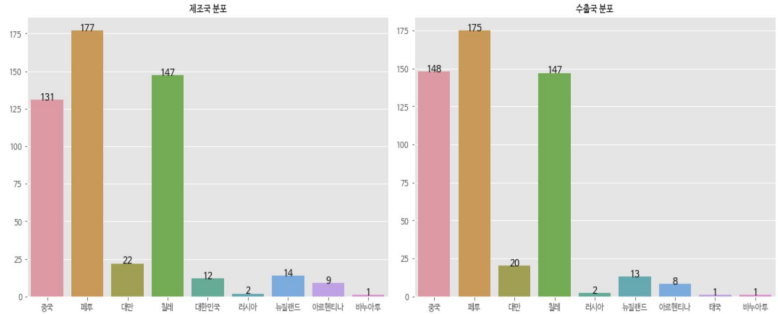
week	수입용도_판매용	수입용도_자사제품제조용	수입용도_외화획득용 원료
1	0.696970	0.303030	0.0
2	0.640000	0.360000	0.0
3	0.696970	0.303030	0.0
4	0.677419	0.322581	0.0
5	0.685714	0.314286	0.0

FE_오징어

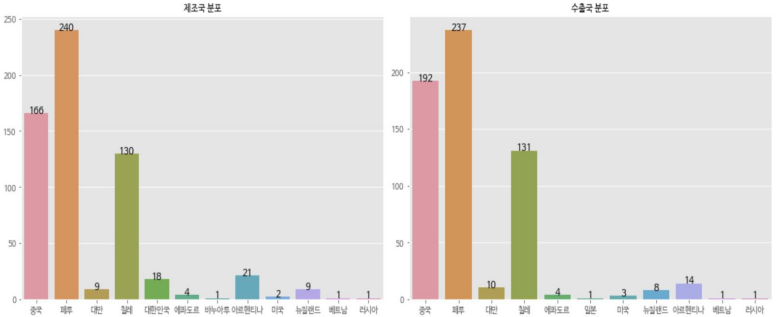
2016년도



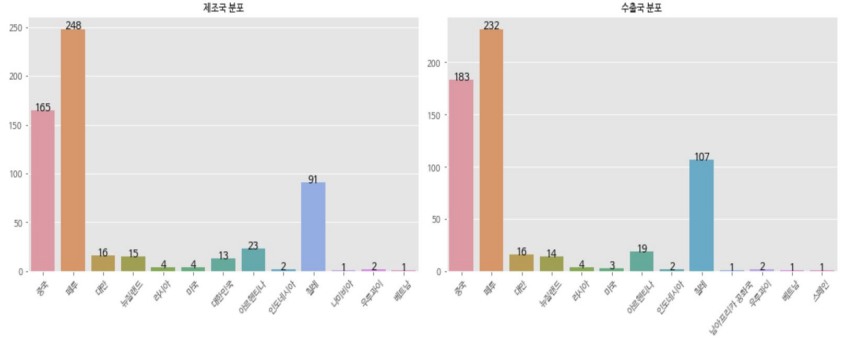
2017년도



2018년도

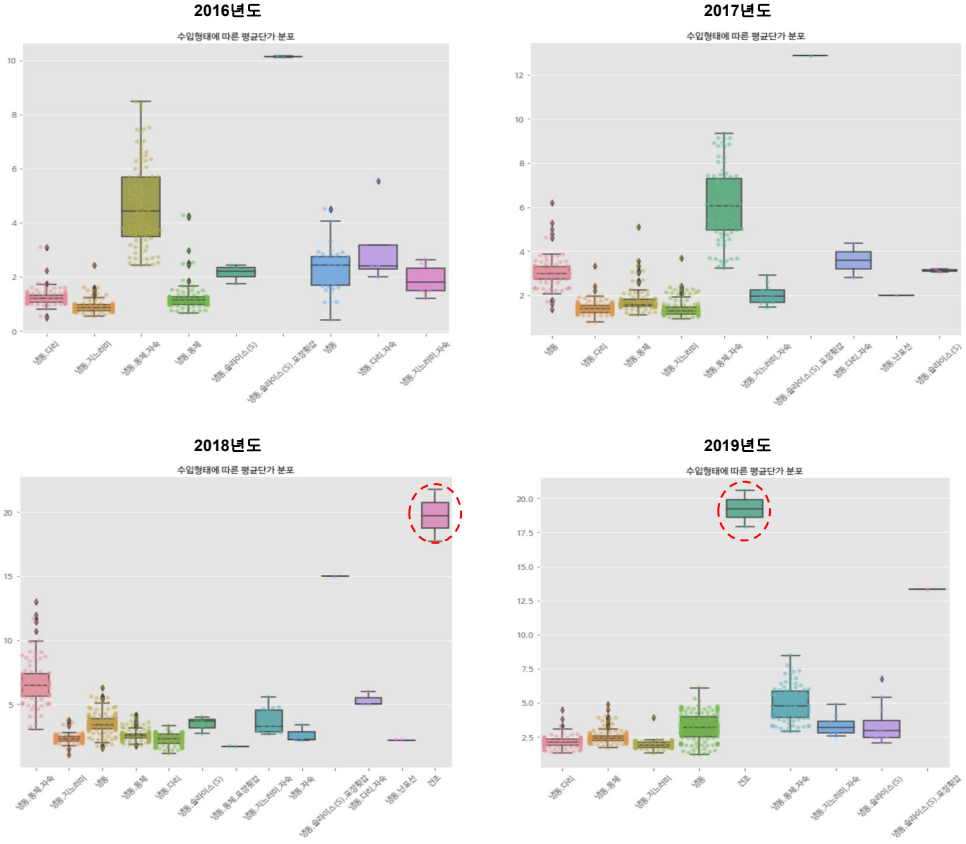


2019년도



- 중국, 칠레, 페루 3개의 국가가 모든 연도에서 자주 관측됨을 확인
- 중국, 칠레, 페루, 그 외 국가 4개의 범주로 축소

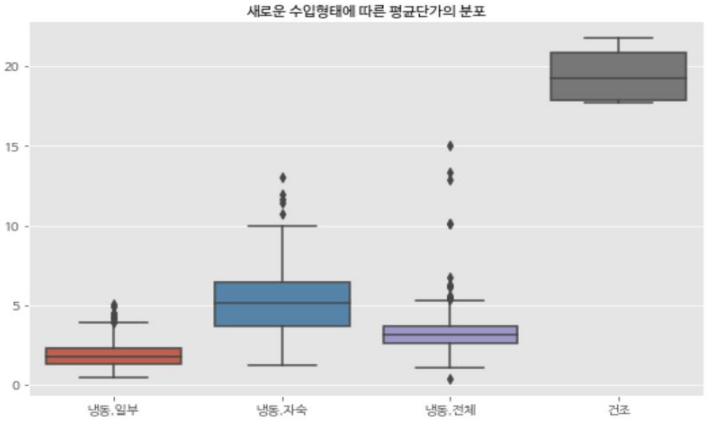
FE_오징어



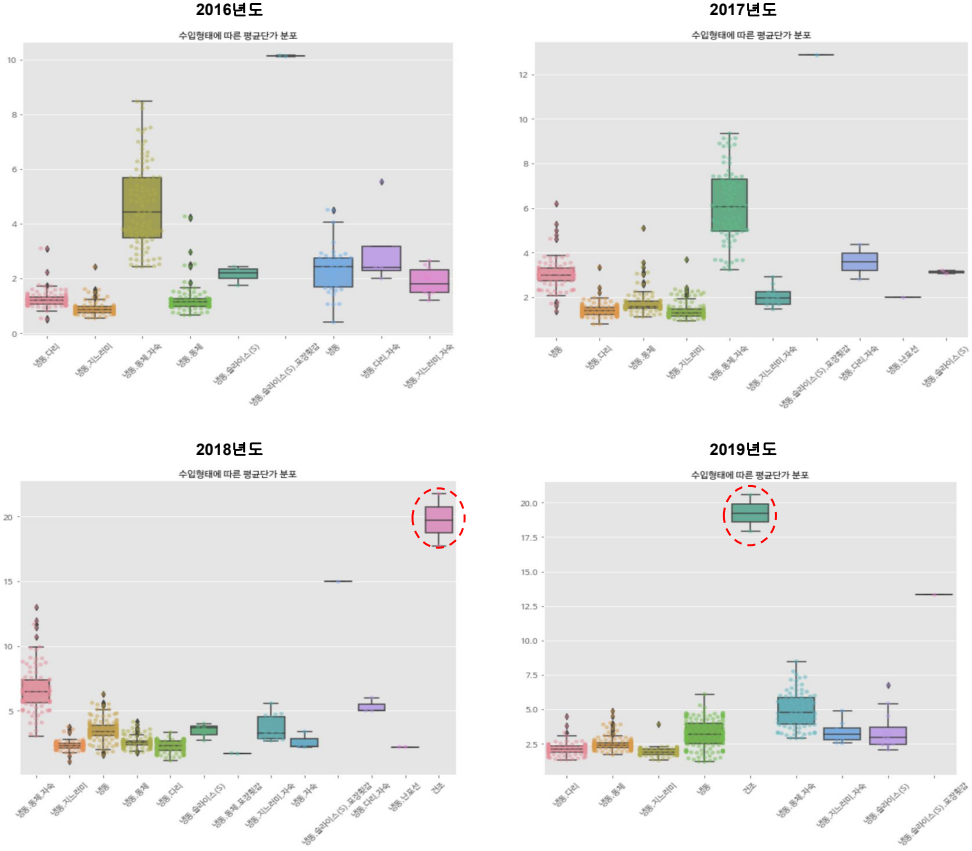
연도마다 수입형태에 따른 평균단가 분포를 통해 비슷한 패턴 발견

자숙이 포함되어있을 때 가격이 높음을 확인

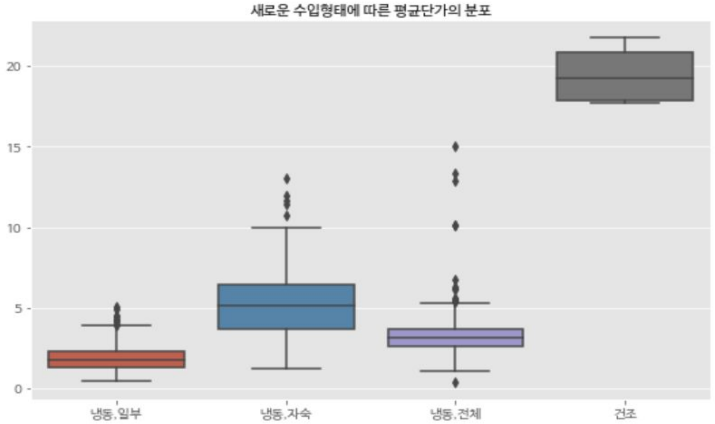
수입형태 변수를 4개의 범주로 축소



FE_오징어

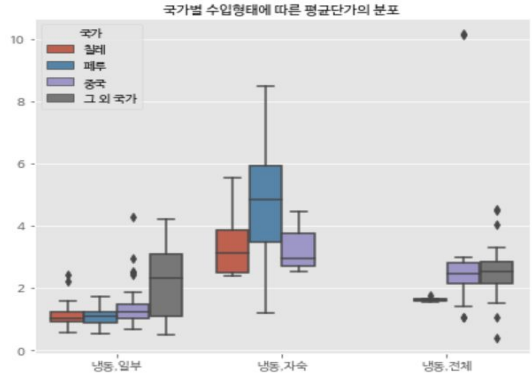


연도마다 수입형태에 따른 평균단가 분포를 통해 비슷한 패턴 발견

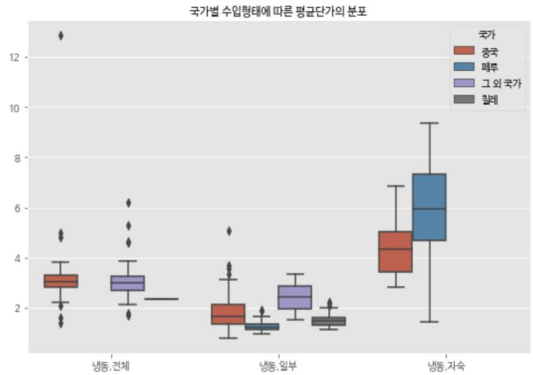


FE_오징어_이상치 처리

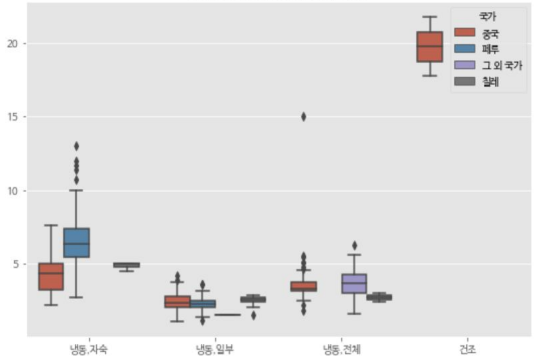
2016년도



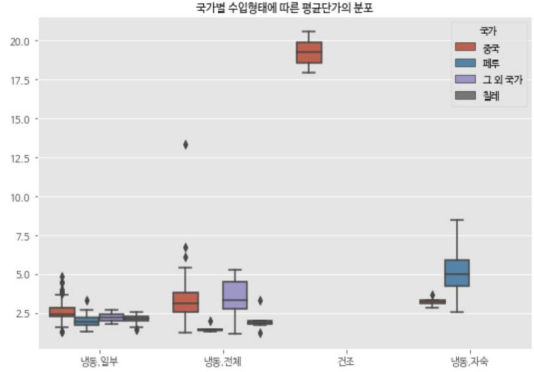
2017년도



2018년도



2019년도



새로 생성한 수입형태, 국가와 연도별로 이상치가 존재함을 확인

수입형태가 건조인 경우 관측치 제거

IQR 방법을 사용하여 이상치를 상한, 하한값으로 대체

1.

추가 데이터

추가 데이터 사용 이유

USD/KRW

한국은 직접표시환율을 사용해 표시하며, 자유변동환율제도를 따르고 있음. 평균 단가의 표시가 USD기준이기 때문에, 수입업자의 비용적 측면을 고려하면 USD/KRW를 기간별로 평균을 구한다면 기준통화국인 미국의 화폐로 표기된 각 나라의 국제 수지도 일정 비율로 영향을 줄 수 있다고 판단.

WTI(유가 데이터)

유가는 물류에서 배제할 수 없는 한정된 자원이며, 특정국과 미국간 관계에 따라 등락이 정해지기도 하기 때문에 환율과 항상 일치하는 방향을 가지고 있지 않음

은행업무일

주 별 근무일(미국, 한국, 양국) 비율, 특정 기간에 수요가 높은 수산물이 존재(설날, 추석 등, 기사 자료)하는데 선적가능일에 따라 하선 시기가 달라짐

추가 데이터 사용 이유

- 수출업계에서는 원유와 통화에 의해서 가격 변동이 있다.

연어 가격에 웃고 울고...노르웨이·칠레 희비교차

머니투데이 김신회 기자
2016.01.14 10:27

의견 남기기 가

유가 하락 환율 영향…노르웨이산 연어 가격 사상 최고, 칠레산은 3년 만에 최저

세계 1, 2위 연어 수출국인 노르웨이와 칠레의 희비가 엇갈리고 있다. 노르웨이는 연어 가격이 사상 최고 수준으로 올라 환호하지만 칠레는 3년 만에 최저 수준으로 떨어진 연어 가격에 울상이다.

파이낸셜타임스(FT)는 13일(현지시간) 노르웨이와 칠레 연어 수출업계의 희비가 교차하게 된 건 원유를 비롯한 원자재 가격 하락과 이에 따른 원자재 수출국들의 통화 절하 탓이라고 지적했다.

산유국 가운데 하나인 노르웨이에서는 저유가의 영향으로 현지 통화인 크로네화 가치가 급락했다. 지난해 초부터 달러 대비로 16%, 유로화 대비로는 6% 하락했다. 러시아가 2014년에 취한 서방산 식품 수입 금지 조치에 맞서 노르웨이가 연어 가격을 낮춘 가운데 크로네화가 약세를 띠면서 노르웨이산 연어의 수출 경쟁력이 부쩍 높아졌다.

출처 : https://m.mt.co.kr/renew/view.html?no=2016011409262065338&googleamp

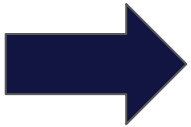
추가데이터

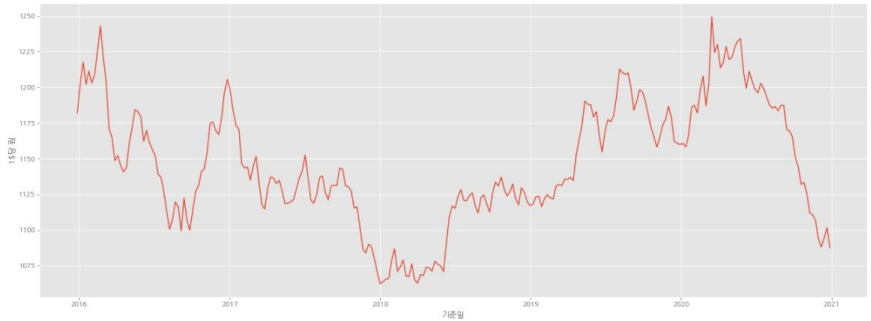
구글 스프레드 시트에서 GOOGLE FINANCE를 이용하여 주별 환율 데이터 생성

A	B	C	D	E
2015-12-23	=index(GOOGLEFINANCE("USDKRW", "CLOSE", "2015-12-23"),2,2)			
2015-12-24	1167.88			
2015-12-25	1167.88			
2015-12-26	1192.321			

2015-12-23	1172.16
2015-12-24	1167.88
2015-12-25	1167.88
2015-12-26	1192.321

기준일	대한민국
2015-12-28	1182.143143
2016-01-04	1202.685571
2016-01-11	1217.597714
2016-01-18	1202.050571
2016-01-25	1211.344429

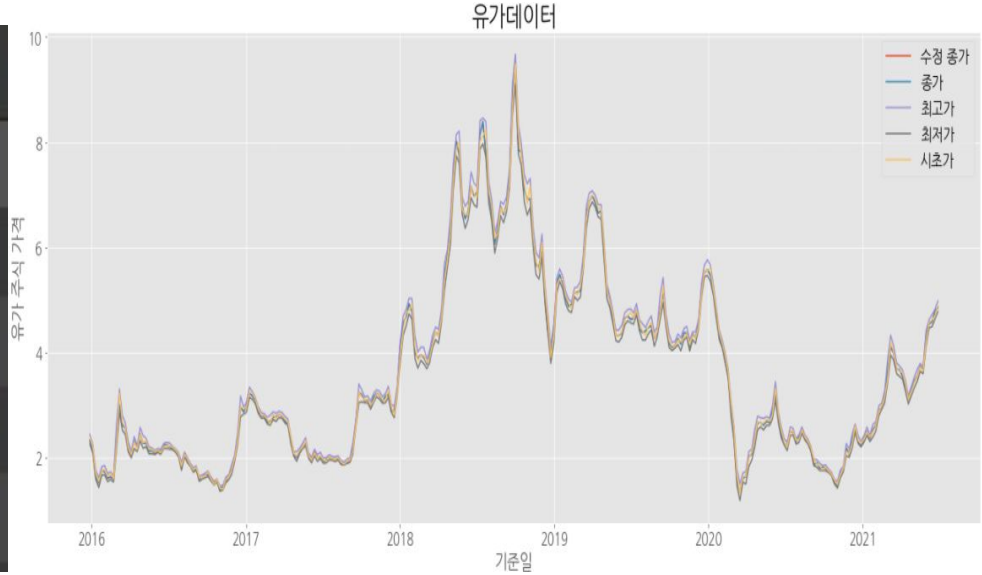




추가 데이터

- 유가 데이터는 yfinance툴을 사용하여 WTI 증권을 사용했습니다.

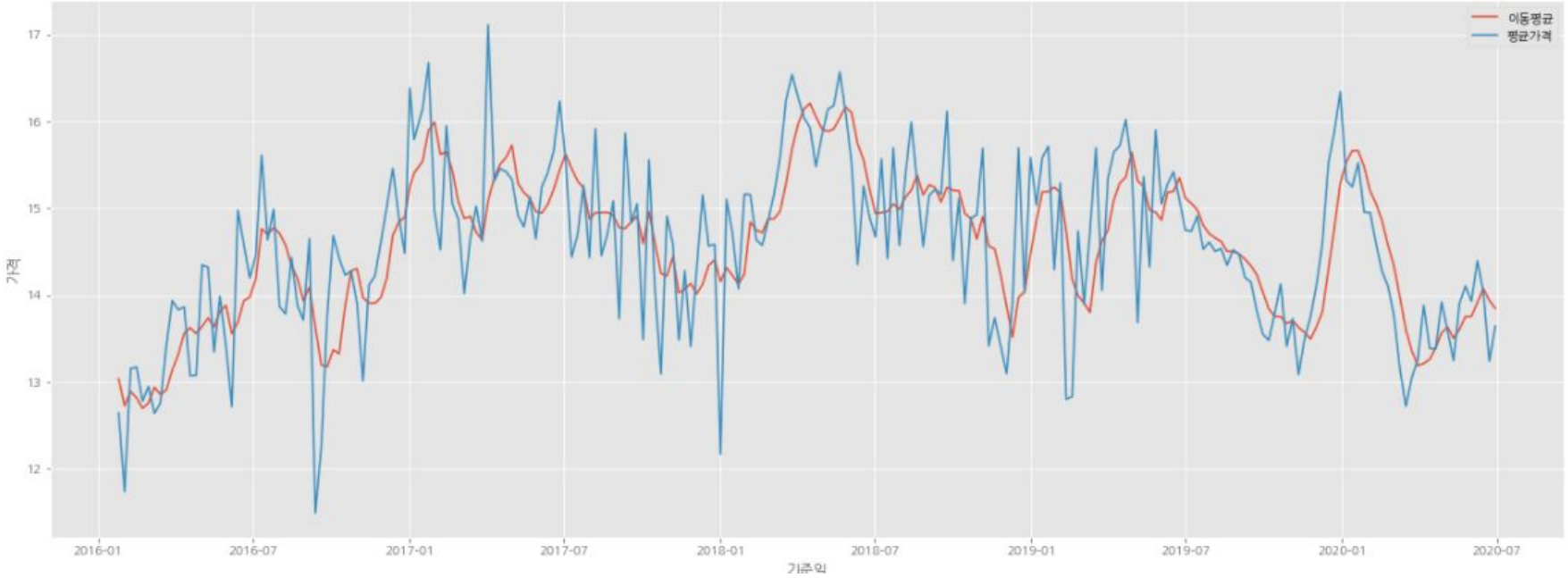
기준일	Adj Close	Close	High	Low	Open
2015-12-28	2.315	2.315	2.4375	2.250	2.3800
2016-01-04	2.112	2.112	2.2320	2.066	2.1860
2016-01-11	1.626	1.626	1.7520	1.578	1.6940
2016-01-18	1.510	1.510	1.6125	1.425	1.5425
2016-01-25	1.752	1.752	1.8200	1.660	1.7460



FE_연어

- Voting Regressor로 예측한 값을 test의 종속변수로 놓고 train의 종속변수와 시간 순서대로 Moving average를 구한다.

  - Moving average 와 target을 비교한 그래프



FE_연어 변수 설명

환율 데이터	대한민국
수입형태의 빈도	냉장, 필렛(F), 횟감, 포장횟감, 냉동
노르웨이 빈도	제조국_노르웨이
수입용도의 빈도	수입용도_판매용, 수입용도_자사제품제조용, 수입용도_외화획득용 원료
유가 데이터	Adj Close
이동평균	moving_average

대한민국	냉장	필렛(F)	횟감	포장횟감	냉동	제조국_노르웨이	수입용도_판매용	수입용도_자사제품제조용	수입용도_외화획득용 원료	Adj Close	moving_average
7.100311	1.000000	0.677419	0.193548	0.161290	0.000000	0.967742	0.677419	0.322581	0.000000	1.7520	13.037535
7.093393	1.000000	0.600000	0.171429	0.142857	0.000000	0.885714	0.685714	0.314286	0.000000	1.7240	12.727598
7.098180	1.000000	0.645161	0.161290	0.161290	0.000000	0.967742	0.677419	0.322581	0.000000	1.6040	12.892301
7.111960	0.939394	0.606061	0.151515	0.151515	0.060606	0.909091	0.666667	0.303030	0.030303	1.6275	12.819329
7.125989	0.972222	0.555556	0.138889	0.138889	0.027778	0.833333	0.694444	0.277778	0.027778	1.5860	12.675802

FE - 새우

```
# 기준일, 어종, 상세어종으로만 입력받는 평가 데이터이기 때문에 글자로 들어간 변수는 수치화

pivoted_train = (train[train["상세어종"].apply(lambda x: 1 if x in spec_kor else 0)==1]).\
                pivot_table(index=["기준일", "어종", "상세어종"], values=["제품구분", "제조국", "수입용도",
"수입형태",\
                "평균단가($)"], aggfunc={"제품구분":"count", "제조국":"nunique", "수입용도":"nunique",\
                "수입형태":"nunique", "평균단가($)":"mean"}).reset_index()
```

Date	US_Bday	KR_Bday	Both_Bday
2015-12-28	0.571429	0.571429	0.571429
2016-01-04	0.714286	0.714286	0.714286
2016-01-11	0.714286	0.714286	0.714286
2016-01-18	0.571429	0.714286	0.571429
2016-01-25	0.714286	0.714286	0.714286

- 평가 데이터와 다른 변수를 반영할 방법 중 하나로 기준일별 unique 수를 입력했으나, 결과가 좋지 않았기에 시간 변수만 반영하기로 결정
- 미국과 한국의 공휴일을 바탕으로 생성한 **은행업무일**

1.

Modeling

Modeling_연어

Voting Regressor

```
def run_submit_model(train_data = None, test_data =  None):
  train_data = train_data.reset_index(drop = True)
  test_data = test_data.reset_index(drop = True)

  # train과 test데이터를 X_train,y_train,X_test 로 나눠준다.
  X_train, y_train = train_data.drop(['기준일','target'], axis = 1), train_data['target']
  X_test = test_data.drop(['기준일'],axis = 1)

  # 위의 모델중에 가장 잘 나온 randomforest를 가지고 모델을 돌려준다.
  lr = LinearRegression()
  lgb = LGBMRegressor(random_state = 42)
  cat = CatBoostRegressor(random_state = 42,verbose = False, task_type = 'GPU')
  xgb = XGBRegressor(random_state = 42,tree_method = 'gpu_hist')
  ridge = Ridge(random_state = 42)
  rfg = RandomForestRegressor(random_state = 42)
  gb = GradientBoostingRegressor(random_state = 42)
  print(X_train.columns)
  print(X_test.columns)


  vo_reg = VotingRegressor(estimators = [('lr',lr),('lgb',lgb),('cat',cat),('xgb',xgb),('ridge',ridge),('rfg',rfg),('gb',gb)])
  vo_reg.fit(X_train, y_train)
  y_pred = vo_reg.predict(X_test)

  return y_pred

y_pred = run_submit_model(train_data = data, test_data = test_data)
```

Moving Average 생성

```
# y_pred를 target값에 넣어줌
test_data['target'] = y_pred

# 전체 데이터와 test데이터를 합친다.
final_data = pd.concat([data, test_data], axis = 0)

# Moving Average 변수를 생성한다.
final_data['moving_average'] = final_data['target'].rolling(5).mean()
```

moving_average
13.037535
12.727598
12.892301
12.819329
12.675802

Modeling_연어

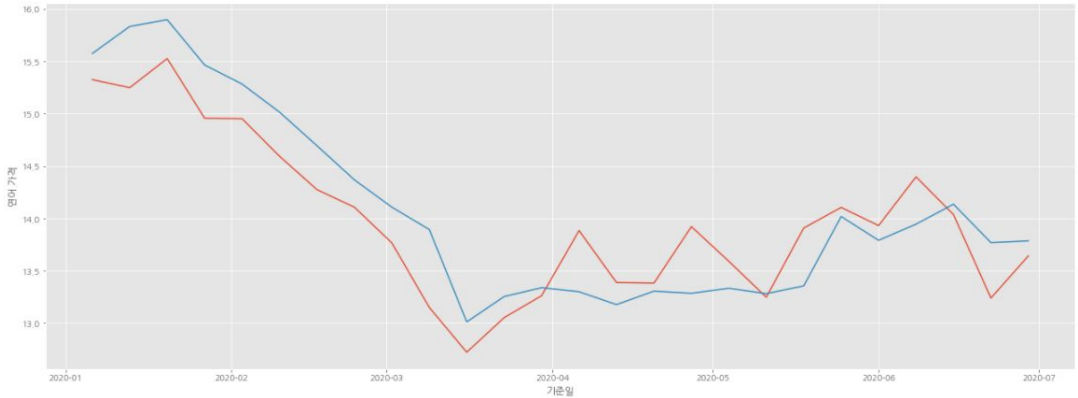
Moving Average 추가후 Voting Regressor 진행

```
y_pred = run_submit_model(train_data = data, test_data = test_data)
```

```
Index(['대한민국', '냉장', '필렛(F)', '횟감', '포장횟감', '냉동', '제조국_노르웨이', '수입용도_판매용',
       '수입용도_자사제품제조용', '수입용도_외화획득용 원료', 'Adj Close'],
      dtype='object')
Index(['대한민국', '냉장', '필렛(F)', '횟감', '포장횟감', '냉동', '제조국_노르웨이', '수입용도_판매용',
       '수입용도_자사제품제조용', '수입용도_외화획득용 원료', 'Adj Close'],
      dtype='object')
```

검증 데이터

RMSE : 0.7822



Modeling_오징어

RandomForest
(Parameter optimization)

XGBoost
(Parameter optimization)

LightGBM
(Parameter optimization)

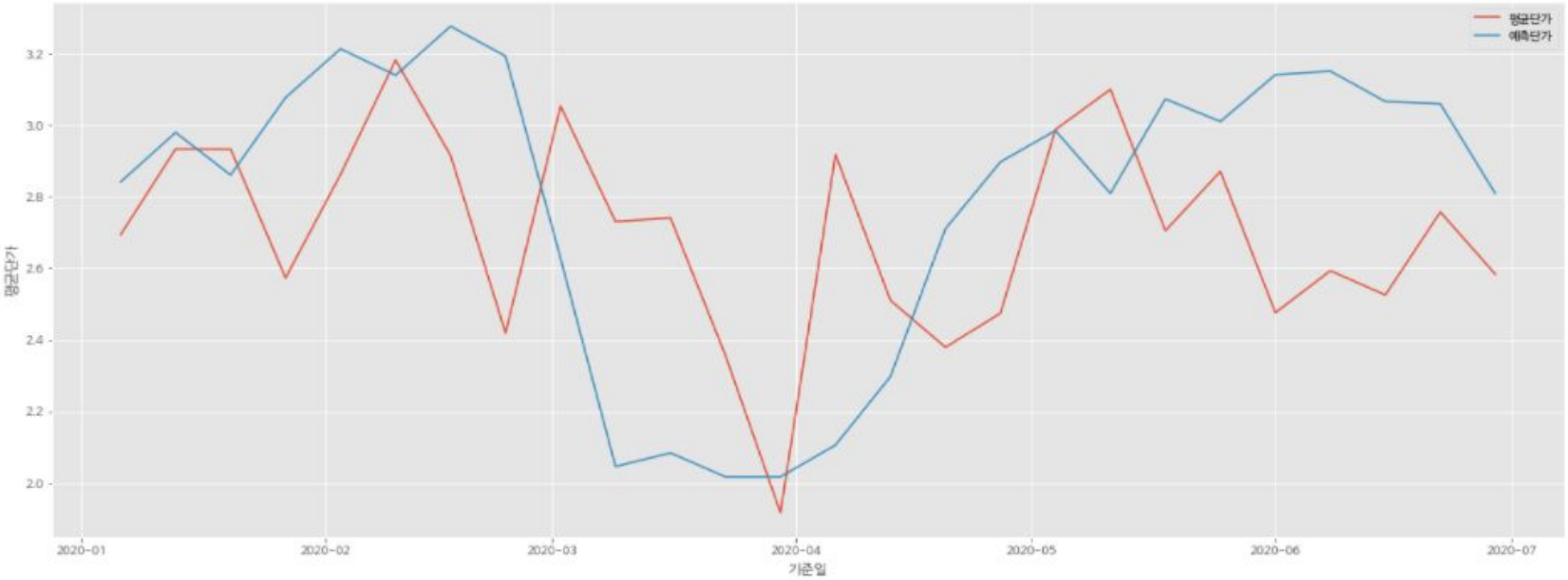
Gradient Boosting
(Parameter optimization)

Stacking Regressor
(Linear Regression)

평균 단가 예측값

Modeling_오징어

RMSE : 0.4282



추가 데이터 및 모델 선별

	CatBoost	GB	Gamma	Lasso	LightGBM	RForest	Ridge	TheilSen	Tweedie	XGBoost	XGBoostRF
min_comp_0	3.988562	3.991258	3.993284	4.006288	3.979931	3.977417	4.008177	4.013165	3.993678	3.999788	3.988096
mean_comp_0	3.995512	3.997250	4.003475	4.006288	3.991849	3.992420	4.023846	4.191394	4.003219	4.001998	3.990510
max_comp_0	4.002791	4.003242	4.011124	4.006288	4.003766	4.007384	4.040028	4.786746	4.010488	4.004208	3.992757
min_comp_1	4.002766	3.989988	3.996537	4.006288	3.995046	3.996152	4.011163	4.016482	3.996990	3.986070	3.989059
mean_comp_1	4.014546	4.000283	4.004516	4.006288	4.012953	4.020380	4.011800	4.121667	4.004882	3.997847	3.992815
max_comp_1	4.026147	4.010577	4.011124	4.006288	4.030859	4.043106	4.012276	4.343263	4.012297	4.008928	3.996471
min_comp_2	4.000882	4.012389	3.997414	4.011124	4.003984	3.995083	4.005622	4.012815	3.997432	4.011577	3.985026
mean_comp_2	4.026359	4.025166	4.006870	4.011124	4.019978	4.025444	4.011437	4.066904	4.006887	4.026052	3.996929
max_comp_2	4.047649	4.037943	4.010762	4.011124	4.036454	4.061330	4.017736	4.191962	4.010770	4.040528	4.008225
min_comp_3	3.988562	3.991258	3.993284	4.006288	3.979931	3.977417	4.008177	4.013165	3.993678	3.999788	3.988096
mean_comp_3	3.999748	3.997250	4.004320	4.006288	3.994996	3.997550	4.022810	4.179985	4.004094	4.003355	3.992753
max_comp_3	4.033640	4.003242	4.011124	4.006288	4.026462	4.038587	4.040028	4.786746	4.011093	4.016924	4.010696
min_comp_4	3.983207	4.015883	3.989924	4.005973	3.993827	3.987082	4.074430	4.005514	3.990145	3.998935	3.982202
mean_comp_4	3.987576	4.019584	4.000743	4.005973	3.997476	3.994076	4.151914	4.180864	3.999531	4.007585	3.983069
max_comp_4	3.989663	4.023286	4.011124	4.005973	4.001097	3.998704	4.248324	4.503174	4.010138	4.016320	3.983317
min_comp_5	3.987815	3.990049	3.989760	4.005973	3.991166	3.984802	4.070047	4.049746	3.989950	3.979935	3.982616
mean_comp_5	3.988564	3.995090	4.000572	4.005973	3.994180	3.988761	4.157818	4.233302	3.999345	3.983849	3.983166
max_comp_5	3.989676	4.000131	4.011124	4.005973	3.997263	3.997497	4.271892	4.587575	4.010125	3.987762	3.983471

```
1 # 데이터 형태 다름 무시하고 확인한 min
2 (mmm_dscb.describe()).loc["min", :].sort_values()
```

```
RForest       3.977417
LightGBM      3.979931
XGBoost       3.979935
XGBoostRF     3.982202
CatBoost      3.983207
rfg           3.984057
catboost      3.986362
Gamma         3.989760
Tweedie       3.989950
GB            3.989988
```

comp_0 : 기준일

comp_1 : 기준일 + 환율

comp_2 : 기준일 + 유가

comp_3 : 기준일 + 은행업무일

comp_4 : 기준일 + 환율 + 유가

comp_5 : 기준일 + 은행업무일 + 환율 + 유가

Modeling with Hyperparameter Tuning_세우

```
pd.DataFrame({'GridSearchCV(RandomForestRegressor)': pd.Series(rfr_best_param_g), 'Random
```

	GridSearchCV(RandomForestRegressor)	RandomizedSearchCV(RandomForestRegressor)
max_depth	3	3
max_features	sqrt	sqrt
n_estimators	150	150
n_jobs	-1	3
oob_score	True	False
random_state	42	777

```
# Used StandsardScaler
pd.DataFrame({'GridSearchCV(RandomForestRegressor)': pd.Series(rfr_best_param_g_std), 'Ra
```

	GridSearchCV(RandomForestRegressor)	RandomizedSearchCV(RandomForestRegressor)
max_depth	3	3
max_features	log2	log2
n_estimators	150	150
n_jobs	-1	-1
oob_score	False	True
random_state	42	42

RandomForest Regressor

	GridSearchCV(XGBRegressor)	RandomizedSearchCV(XGBRegressor)
booster	gblinear	gblinear
eval_metric	rmse	rmse
feature_selector	cyclic	cyclic
importance_type	weight	weight
learning rate	0.01	0.2
max_depth	3	10
n_estimators	300	100
objective	reg:linear	reg:linear
predictor	auto	auto
reg_alpha	0	1
reg_lambda	0.3	0.5
seed	0	777
updater	shotgun	shotgun

XGBoost Regressor

	RandomizedSearchCV(LGBMRegressor)
boosting_type	gbdt
learning rate	0.001
n_estimators	100
n_jobs	-1
nthread	3
num_leaves	10
random_state	0
reg_alpha	1
reg_lambda	2
silent	True

LightGBM Regressor

	RForest	XGBoost	CatBoost	LightGBM	Voting
RMSE	0.94551	0.987504	1.000093	0.958926	0.942686


감사합니다