공공데이터를 활용한
해외국가의 국내품목 수입액 예측

2021 공공데이터 활용 빅데이터 분석 경진대회

PRESENTATION
AGENDA

소개

데이터 설명

분석 및 결과

결론 및 사업화 가능성

개선점

1. Introduction

1.주제 및 목표

KOTRA의 특정 품목, 특정 국가의 데이터를 분석해 다음 해에 어떠한 국가가 어떤 품목을 한국으로부터 얼마만큼 수입할지에 대한 예측하는 과제

2. 배경 및 필요성

KOTRA의 기존 역할인 중소, 중견 기업의 해외 진출과 수출을 지원하기 위해 데이터를 활용해 어떤 국가가 특정품목을 한국에서 얼마나 수입할지 예측해 중소, 중견 기업이 수출할 시장을 탐색하는데 도움을 주고자 함

3. 활용 데이터

전년도 한국 수입액이 포함된 최근 2개년 KOTRA에서 제공하는 무역 데이터와 World Bank Group에서 제공하는 WDI(World Development Indicators) 데이터

2. DATA_데이터 설명 및 추가변수

목적: 추가 변수 사용 및 파생변수

차년도에 해당국가 해당품목을 한국에서 얼만큼 수입할지 예측하는 과제이므로 해당 국가의 수입과 무역에 관련된 데이터를 추가 및 변형하여 활용하고자 한다.

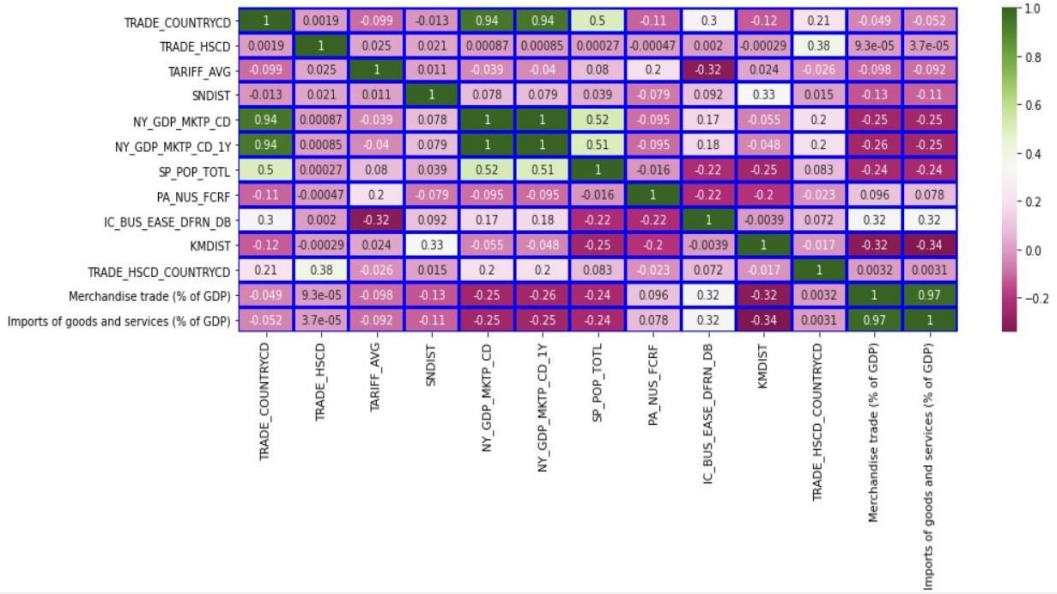
추가변수 (1) Imports of goods and services (% of GDP)

Imports of goods and services는 전체 GDP에서 해당국가가 수입한 제품 혹은 서비스의 비율을 일컫는다. 이 수치는 해당 국가가 전세계에서 받은 제품과 그 이외의 시장 서비스의 가치를 반영한다. 또한, 상품, 화물, 보험, 교통, 관광, 저작권, 자격 비용 그리고 통신, 건축, 금융, 정보, 비즈니스, 개인 혹은 정부 서비스와 같은 그 이외의 서비스의 가치를 포함한다. 고용인의 보상과 투자 이익과 이전 지출을 배제하는 수치이다.

추가변수 (2) Merchandise trade (% of GDP)

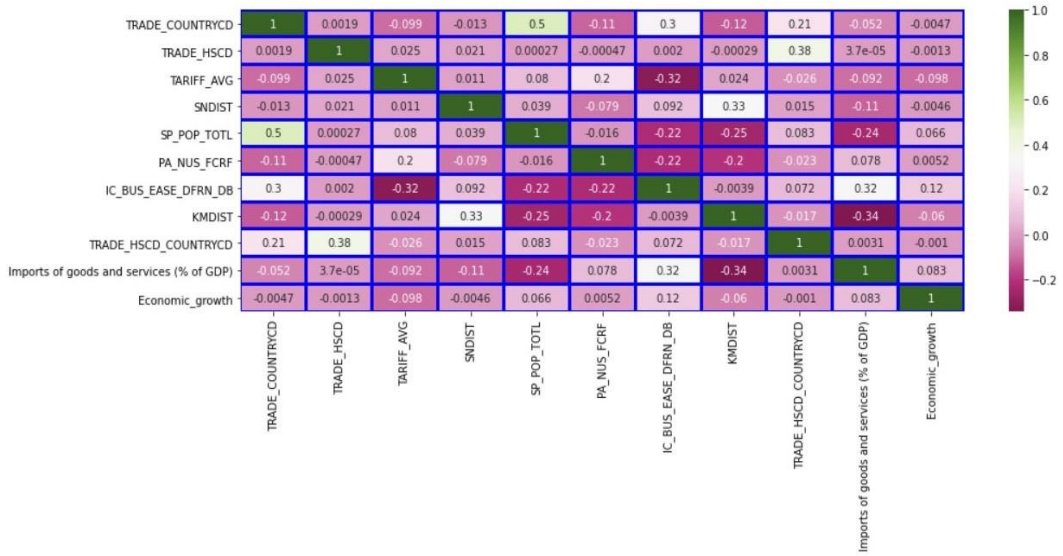
Merchandise trade는 U.S. dollar로 상품 수출과 수입을 GDP로 나눈 것의 총합을 나타내는 수치이다. 집계 방법은 가중 평균을 이용하였으며, 연간 주기로 설명된 데이터이다.

2. DATA_파생변수 생성 및 다중공선성 문제



- ✓ 두 변수 추가 후, 전체 변수에 대한 상관표를 그린 결과, NY_GDP_MKTP와 NY_GDP_MKTP_1Y 그리고 추가된 변수인 Imports of goods and services와 Merchandise trade가 서로 강한 양의 상관관계를 보임.
- ✓ NY_GDP_MKTP와 NY_GDP_MKTP_1Y는 TRADE_COUNRYCD와도 강한 양의 상관관계를 보임.
- ✓ 이와 같이 독립변수들 간의 높은 선형관계를 보여주면 다중공선성 문제를 일으켜 최종적으로 해당 국가의 해당 품목 수입금액을 예측하는데 문제가 생길 수 있음.

2. DATA_파생변수 생성 및 다중공선성 문제



- ✓ 다중공선성 문제를 해결하기 위해서 우선 NY_GDP_MKTP와 NY_GDP_MKTP_1Y를 삭제
- ✓ 그 다음, 경제성장률 지표의 공식은 *{(금년도 실질 GDP - 전년도 실질 GDP) ÷ 전년도 실질 GDP} × 100* 이기 때문에 NY_GDP_MKTP와 NY_GDP_MKTP_1Y를 활용하여 경제성장률(Economic_growth)라는 파생변수 생성.
- ✓ Imports of goods and services와 Merchandise trade의 다중공선성 문제를 해결하기 위해 두개 변수 중에 더 좋은 예측 정확도를 보인 것을 선택
- ✓ Imports of goods and services를 추가한 모델이 더 정확한 예측을 보여줬기 때문에 최종적으로 Imports of goods and services의 변수를 추가 (Merchandise trade 12.78 MAPE, Imports of goods and services 12.49 MAPE, 두 변수 다 사용 13.40 MAPE)

3. Analysis & Result

(1) 모듈 import 및 데이터 추가

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import seaborn as sns
%matplotlib inline

from sklearn.preprocessing import StandardScaler

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.optimizers import Adam
```

```
train_df = pd.read_csv('/content/drive/My Drive/Colab Notebooks/kotra/공모전데이터_분석용_KOTRA_0525.csv')
test_df = pd.read_csv('/content/drive/My Drive/Colab Notebooks/kotra/공모전데이터_예측용_KOTRA_0525.csv')
```

```
train_df.head()
```

	UNC_YEAR	HSCD	COUNTRYCD	COUNTRYNM	TRADE_COUNTRYCD	TRADE_HSCD	TARIFF_AVG	SNDIST	NY_GDP_MKTP_CD	NY_GDP_MKTP_CI
0	2017	190590	12	Algeria	46052990973	19480986257	30.0	3878.238437	170163165961	15999483
1	2017	190590	36	Australia	228441691195	19480986257	0.0	12203.155980	1329188475752	120884699
2	2017	190590	40	Austria	166475020975	19480986257	0.0	4403.247293	417237869116	39556864
3	2017	190590	56	Belgium	406412223480	19480986257	0.0	3980.375563	502698069367	47573958
4	2017	190590	76	Brazil	150749493921	19480986257	18.0	9644.206941	2062831045936	179570016

```
test_df.head()
```

	UNC_YEAR	HSCD	COUNTRYCD	COUNTRYNM	TRADE_COUNTRYCD	TRADE_HSCD	TARIFF_AVG	SNDIST	NY_GDP_MKTP_CD	NY_GDP_MKTP_CI
0	2018	190590	12	Algeria	42196119729	2.223464e+10	27.0	3735.047389	175405660377	17016316
1	2018	190590	36	Australia	240422685574	2.223464e+10	1.2	11947.511360	1432881172002	132918847
2	2018	190590	40	Austria	186965232670	2.223464e+10	2.8	4005.020029	455094861902	41723786
3	2018	190590	56	Belgium	461444842911	2.223464e+10	2.8	4501.782826	543734366831	50269806
4	2018	190590	76	Brazil	185290138433	2.223464e+10	17.1	10015.769070	1885482534238	206283104

```
In [9]: im_2017
```

Out [9]:

	index	Series_Code	Country_Name	2017
0	0	NE.IMP.GNFS.ZS	Afghanistan	..
1	1	NE.IMP.GNFS.ZS	Albania	46.62445444
2	2	NE.IMP.GNFS.ZS	Algeria	32.68912917
3	3	NE.IMP.GNFS.ZS	American Samoa	102.2875817
4	4	NE.IMP.GNFS.ZS	Andorra	..
...	...	...	...	...
212	212	NE.IMP.GNFS.ZS	Virgin Islands (U.S.)	90.57282302
213	213	NE.IMP.GNFS.ZS	West Bank and Gaza	52.72445437
214	214	NE.IMP.GNFS.ZS	Yemen, Rep.	..
215	215	NE.IMP.GNFS.ZS	Zambia	36.59283689
216	216	NE.IMP.GNFS.ZS	Zimbabwe	28.08565493

217 rows × 4 columns

```
In [10]: im_2018
```

Out [10]:

	index	Series_Code	Country_Name	2018
0	0	NE.IMP.GNFS.ZS	Afghanistan	..
1	1	NE.IMP.GNFS.ZS	Albania	45.26431362
2	2	NE.IMP.GNFS.ZS	Algeria	32.11179183
3	3	NE.IMP.GNFS.ZS	American Samoa	103.276131
4	4	NE.IMP.GNFS.ZS	Andorra	..
...	...	...	...	...
212	212	NE.IMP.GNFS.ZS	Virgin Islands (U.S.)	101.4809237
213	213	NE.IMP.GNFS.ZS	West Bank and Gaza	55.439711
214	214	NE.IMP.GNFS.ZS	Yemen, Rep.	..
215	215	NE.IMP.GNFS.ZS	Zambia	36.93089354
216	216	NE.IMP.GNFS.ZS	Zimbabwe	18.94328155

217 rows × 4 columns

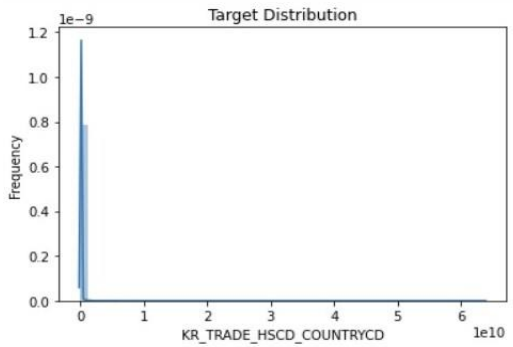
1) 데이터 분석에 활용한 언어는 python이며, 툴은 colab을 활용하며 분석을 실시

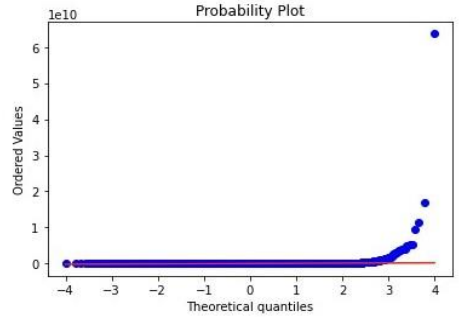
2) 데이터 분석에 필요한 파이썬 모듈을 import (pandas, numpy, sklearn, tensorflow 등)

3) 2017년도 코트라 분석용 데이터, 2018년도 코트라 훈련용 데이터 csv파일을 추가

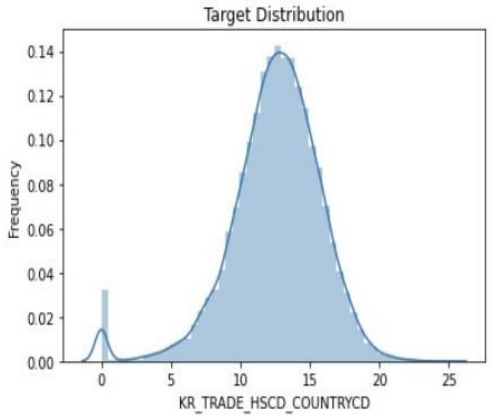
3. Analysis & Result

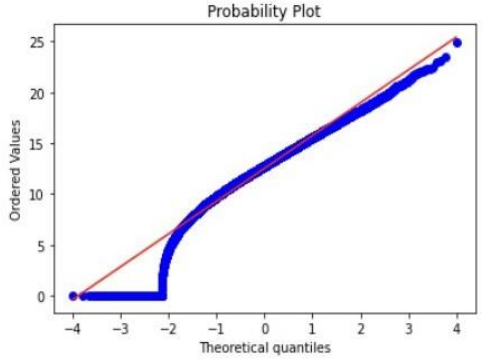
(2) 데이터 전처리 및 정규성





Skewness: 112.976





Skewness: -0.908

1) 타깃 값인 KR_TRADE_HSCD_COUNTRYCD의 정규성 검정 결과 왜도 값이 112로 정규분포를 안 따름

2) 타깃 값이 정규분포를 가질 수 있도록 log(x+1)로 변환 (0값을 가진 데이터에 대응하기 위해)

3) 그 결과 타깃 값이 정규분포를 따르게 되고 왜도 값도 -0.908의 값으로 크게 줄어듦.

3. Analysis & Result

(2) 데이터 전처리 및 결측 값 처리

In [21]:
```
#년도, 국가명 제거
#국가코드는 국가명이랑 중복된 정보이므로 제거
train_df.drop(['UNC_YEAR', 'COUNTRYCD'], axis=1, inplace=True)
test_df.drop(['UNC_YEAR', 'COUNTRYCD'], axis=1, inplace=True)
```

```
mc = pd.DataFrame(df.isnull().sum(), columns=["Missing Count"])
mc = mc[mc['Missing Count']!=0] #불리언 인덱싱
#새로운 컬럼 추가
mc['Missing %'] = (mc['Missing Count'] / df.shape[0]) * 100
mc.sort_values('Missing %',ascending=False)
```

	Missing Count	Missing %
PA_NUS_FCRF	6976	16.461371
TARIFF_AVG	754	1.779225
SNDIST	25	0.058993
TRADE_HSCD_COUNTRYCD	24	0.056633

```
#결측값 처리 어떻게 할지 다시 고민 필요
df['PA_NUS_FCRF'] = df['PA_NUS_FCRF'].interpolate(method = 'linear' , limit_direction = 'forward')
df['TARIFF_AVG'] = df['TARIFF_AVG'].fillna(df['TARIFF_AVG'].min())
df['SNDIST'] = df['SNDIST'].fillna(df['SNDIST'].min())
df['TRADE_HSCD_COUNTRYCD'] = df['TRADE_HSCD_COUNTRYCD'].fillna(df['TRADE_HSCD_COUNTRYCD'].min())
```

```
#KR_TRADE_HSCD_COUNTRYCD 0인 값을 mean으로 대체
target_df.replace(0, target_df.mean(), inplace=True)
```

```
target_df.describe()
```

	KR_TRADE_HSCD_COUNTRYCD
count	21189.000000
mean	12.725472
std	2.890775
min	0.693147
25%	10.957242
50%	12.764688
75%	14.651716
max	24.877377

1) 국가명과 국가코드는 중복된 정보이므로 국가코드를 제거

2) 데이터 중 년도 데이터는 수입액 예측 분석에 관련성이 없다고 간주하여 제거

3) 결측 값이 10% 이상인 PA_NUS_FCRF 같은 경우에는 pandas의 interpolate 메소드를 사용해 선형적으로 동일한 간격으로 채워줌

4) 나머지 독립변수들은 결측 값 비율이 10% 미만이기 때문에 결측 값의 영향을 최소화하기 위해 최소값으로 치환

5) 0값을 가진 타깃 값은 평균으로 치환

3. Analysis & Result

(2) 데이터 전처리 및 결측 값 처리

결측치 처리

Pandas에서 제공하는 누락 데이터에 특정 값을 채우는 함수로, 각 함수의 기준에 맞게 NaN 값을 지정한 값으로 바꾸는 역할 제공

Interpolate()

인덱스를 무시하고 값들을 선형적으로 같은 간격으로 결측 값을 처리하는 방법

결측치 처리 가이드라인

결측치 비율	처리방법
10% 미만	삭제 or 대치
10% ~ 50%	regression or model based imputation
50% 이상	해당 컬럼(변수) 자체 제거

3. Analysis & Result

(2) 데이터 전처리_정규성 및 원 핫 인코딩

	skewness
TRADE_HSCD_COUNTRYCD	30.100387
TARIFF_AVG	20.037292
TRADE_HSCD	6.440118
PA_NUS_FCRF	4.149168
SP_POP_TOTL	3.897122
TRADE_COUNTRYCD	2.924412
Imports of goods and services (% of GDP)	2.546442
Economic_growth	-1.875440
SNDIST	0.923553
IC_BUS_EASE_DFRN_DB	-0.698085
KMDIST	0.601935

In [41]:
```
#스케일 범위가 크고 skew 값이 0.5 보다 큰 것들은 log(x+1)로 변환
#Economic_growth는 0보다 작은 값이 있어서 변환 안함
#ICBUS_EASE_DFRN_DB는 0~100점 사이의 값을 가지고 있기 때문에 변환 안함
df['TRADE_COUNTRYCD'] = np.log1p(df['TRADE_COUNTRYCD'])
df['TRADE_HSCD'] = np.log1p(df['TRADE_HSCD'])
df['TARIFF_AVG'] = np.log1p(df['TARIFF_AVG'])
df['SNDIST'] = np.log1p(df['SNDIST'])
df['SP_POP_TOTL'] = np.log1p(df['SP_POP_TOTL'])
df['PA_NUS_FCRF'] = np.log1p(df['PA_NUS_FCRF'])
df['KMDIST'] = np.log1p(df['KMDIST'])
df['TRADE_HSCD_COUNTRYCD'] = np.log1p(df['TRADE_HSCD_COUNTRYCD'])
df['Imports of goods and services (% of GDP)'] = np.log1p(df['Imports of goods and services (% of GDP)'])
```

```
df = pd.get_dummies(df)
print(df.shape)
```

```
(42378, 554)
```

COUNTRYNM_Belgium	COUNTRYNM_Brazil	COUNTRYNM_Canada	COUNTRYNM_Chile	COUNTRYNM_China	COUNTRYNM_China, Hong Kong SAR	COUNTRYNM_Czechia
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
1	0	0	0	0	0	0
0	1	0	0	0	0	0
...	...	...	...	...	...	...
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

1) 독립변수의 정규 분포를 따르기 위하여 왜도 값이 0.5 보다 큰 값으로 log(x+1)로 변환.

2) 스케일의 범위가 큰 변수도 log(x+1)로 변환.

3) ICBUS_EASE_DFRN_DB는 0에서 100 점 사의 값을 가지고 있어 스케일의 범위가 크지 않고 왜도 값이 0.5 보다 크지만 0.5와 차이가 크지 않기 때문에 변환 안함.

4) Economic_growth는 0보다 작은 값이 있기 때문에 log(x+1)로 변환할 수 없음.

5) HSCD, COUNTRYNM은 범주형 변수이기 때문에 이를 DNN 회귀 모델에 적용하기 위해 더미함수를 통하여 원 핫 인코딩으로 변환.

3. Analysis & Result

(2) 데이터 전처리_데이터 분리 및 스케일링

In [51]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X_train, target_df, test_size=0.2)
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_train = pd.DataFrame(X_train, columns = df.columns)

X_test = scaler.transform(X_test)
X_test = pd.DataFrame(X_test, columns = df.columns)
```

1) 타깃변수: KR_TRADE_HSCD_COUNTRYCD

2) train 데이터로 학습을 진행하고 test 데이터로 성능 평가하기 위해 분리

3) test 데이터는 전체 훈련 데이터의 20%, train 데이터는 80% 비율로 분리

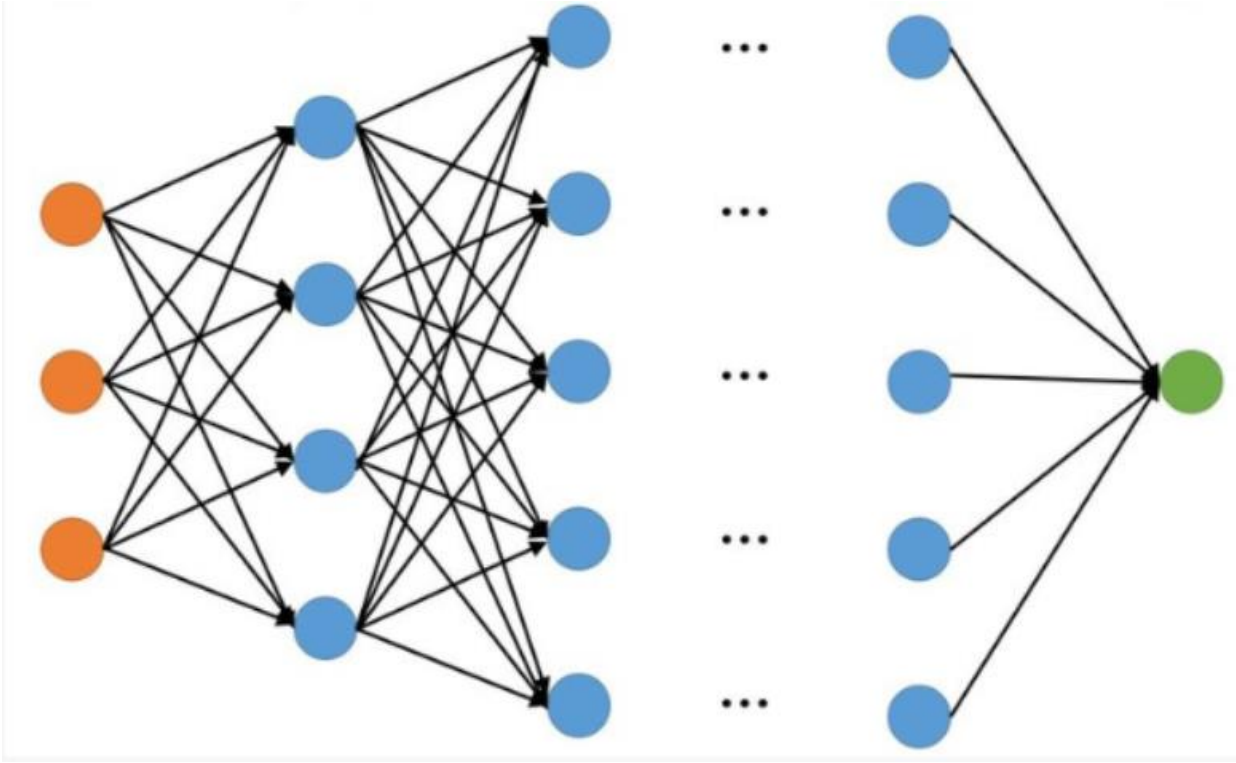
4) StandardScaler로 평균 0, 분산 1의 분포를 가지도록 데이터 스케일링

3. Analysis & Result

(3) 모델 소개 _ DNN

DNN

- ✓ 심층 신경망은 입력층과 출력층 사이에 여러 개의 은닉층들로 이루어진 인공 신경망이다.
- ✓ 예측하려는 타깃 값은 연속형 변수이기 때문에 심층 신경망을 이용한 회귀분석을 실시
- ✓ 심층 신경망은 일반적인 인공 신경망과 마찬가지로 복잡한 비선형 관계들을 모델링 할 수 있기 때문에 독립변수가 많은 이 모델에 적합.



3. Analysis & Result

(3) 모델 정의 및 모델 구축

```
# 모델 구조 정의하기
model=keras.models.Sequential([

    keras.layers.Dense(1024, input_dim = x_train.shape[1], kernel_initializer="he_normal"),
    keras.layers.LeakyReLU(),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(0.4),

    keras.layers.Dense(512, kernel_initializer="he_normal"),
    keras.layers.LeakyReLU(),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(0.3),
    keras.layers.Dense(512, kernel_initializer="he_normal"),
    keras.layers.LeakyReLU(),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(0.3),

    keras.layers.Dense(units=256, kernel_initializer="he_normal"),
    keras.layers.LeakyReLU(),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(0.2),

    keras.layers.Dense(units=256, kernel_initializer="he_normal"),
    keras.layers.LeakyReLU(),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(0.01),
    keras.layers.Dense(units=128, kernel_initializer="he_normal"),
    keras.layers.LeakyReLU(),
    keras.layers.Dropout(0.05),
    keras.layers.Dense(units=1, activation="linear")])

optimizer = Adam(learning_rate=0.005, decay=5e-4)
# 모델 구축하기
model.compile(
        loss='mse',          # mean_squared_error(평균제곱오차)의 alias
        optimizer=optimizer,   # 최적화 기법 중 하나
        metrics=['mape'])    # 실험 후 관찰하고 싶은 metric 들을 나열함.
```

```
model.summary()
```

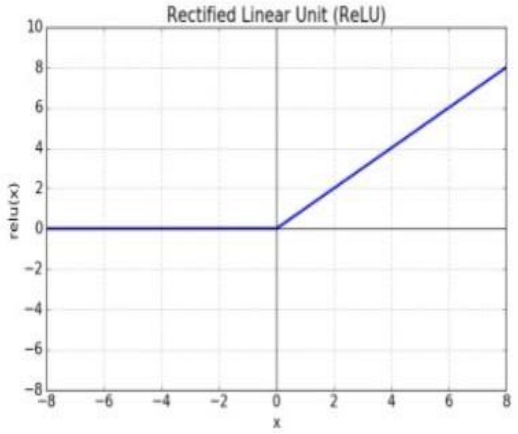
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1024)              568320
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 1024)              0
_________________________________________________________________
batch_normalization (BatchNo (None, 1024)              4096
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 512)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 512)               2048
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 512)               0
_________________________________________________________________
batch_normalization_2 (Batch (None, 512)               2048
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               131328
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
batch_normalization_3 (Batch (None, 256)               1024
```

- ✓ 200개 뉴런과 100개의 뉴런을 가진 두개의 은닉층과 하나의 출력층으로 구성
- ✓ 'LeakyRelu' 활성화 함수를 통하여 layer 전달
- ✓ 은닉층의 가중치 초기화는 He 초기화(kernel_initializer="he_normal") 사용
- ✓ 배치 정규화: 각 가중치의 결과값의 스케일을 조정하고 이동시킴
- ✓ Dropout : 뉴런의 개수가 많을수록 비율을 높게 지정(과대적합 규제 방법)
- ✓ 손실함수: mean_squared_error(평균 제곱근 오차)
- ✓ 최적화: Adam 학습률 0.005, decay_rate=5e-4로 지정
- ✓ Metrics: mape를 통한 성능 확인

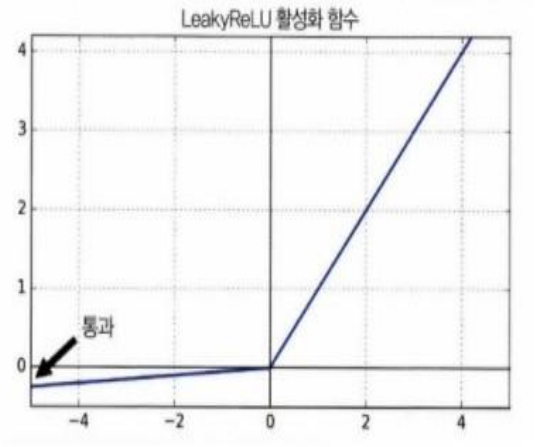
3. Analysis & Result

(3) 모델 정의 및 모델 구축

$$f(x) = max(0, x)$$



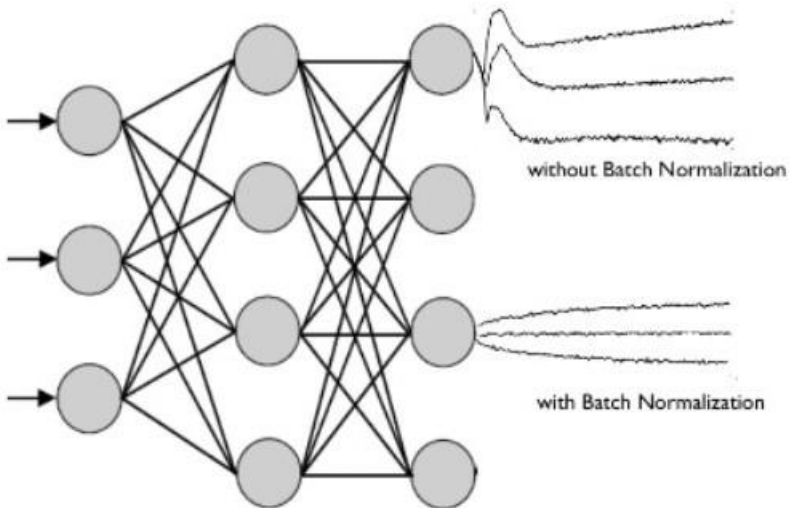
$$LeakyReLU_{\alpha}(x) = max(\alpha x, x)$$



- ✓ RELU 활성화 함수 일부 뉴런이 0 이외의 값을 출력하지 않음 (가중치 합이 음수인 뉴런은 쓸 수 없게 됨)
- ✓ 이러한 문제를 해결하기 위해 LeakyReLu를 사용
- ✓ LeakyReLU 함수의 ReLU 함수의 변종
- ✓ 하이퍼파리미터 α가 이 함수가 새는 정도를 결정
- ✓ 새는 정도란 x < 0 일 때 이 함수의 기울기 (일반적으로 0.01로 설정)
- ✓ 0보다 작은 가중치의 값을 가진 뉴런을 사용할 수 없는 문제를 해결

3. Analysis & Result

(3) 모델 정의 및 모델 구축



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

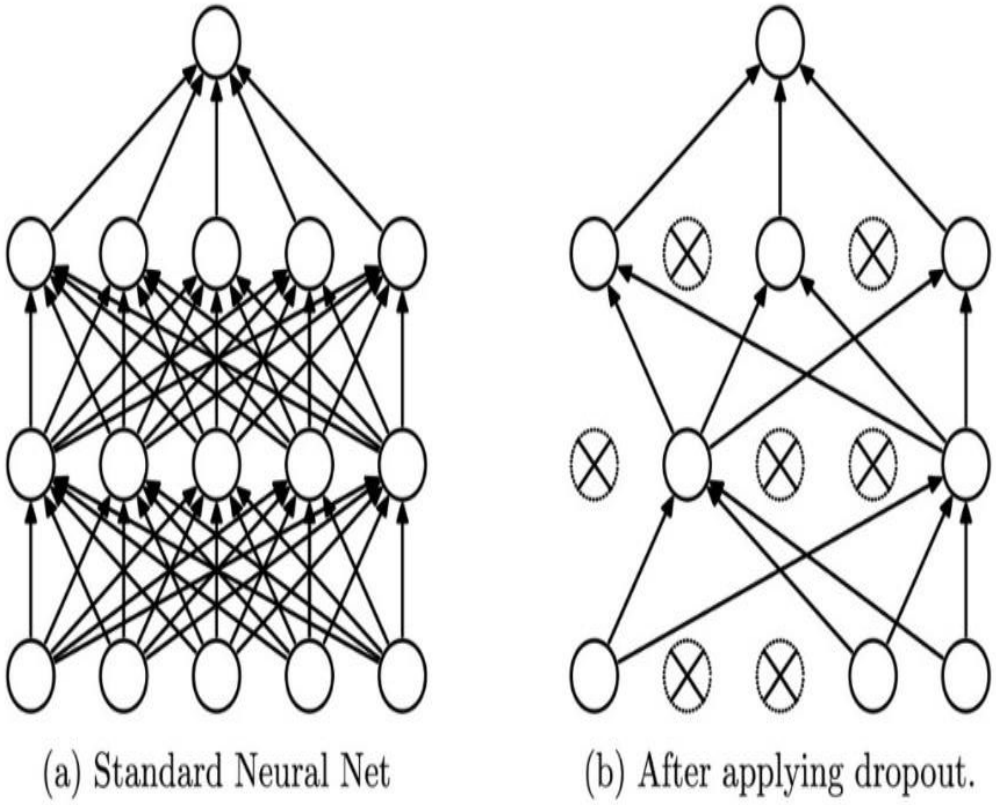
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

✓ 배치 정규화

- 각 입력을 미니배치의 평균과 분산을 이용해 정규화한 다음에 가중치와 편향값을 학습하는 방식
- 레이어를 통과할 때마다 변화하는 결과값의 분포가 바뀌는 현상으로 생기는 문제를 해결 (층이 깊어지면서 생기는 문제를 해결)
- 가중치 초기화에 대한 민감도를 감소
- 모델의 일반화 효과가 생김

3. Analysis & Result

(3) 모델 정의 및 모델 구축



(a) Standard Neural Net

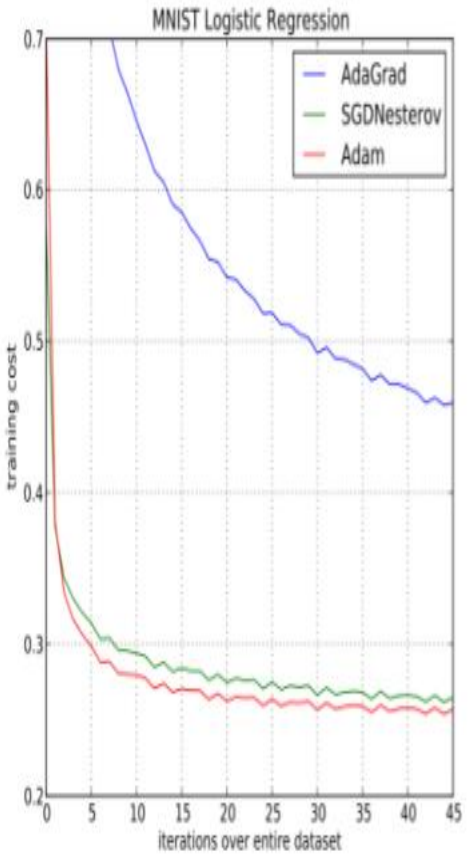
(b) After applying dropout.

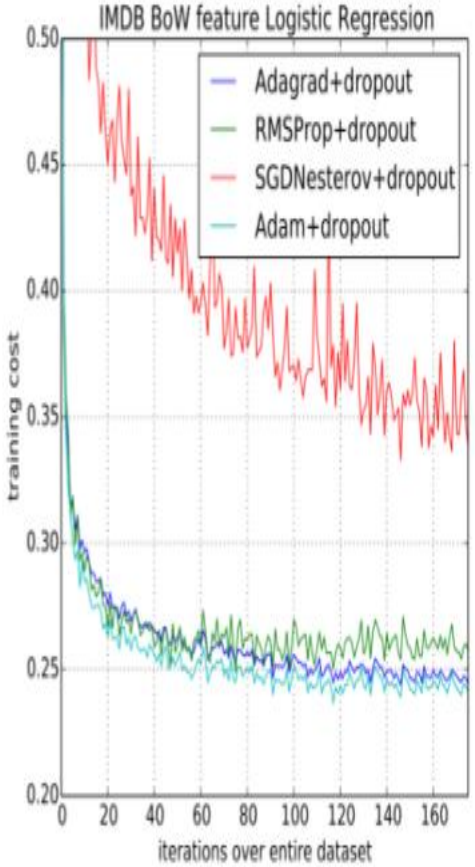
✓ **Dropout**

- 임의 노드를 지정된 확률만큼 제거해 학습에 참여하지 않도록 하는 방법
- 피드 포워드 (feed forward) 과정과 오류 역전파 과정에서 제거된 노드는 학습에 참여하지 않음
- 매 미니배치마다 랜덤으로 설정됨
- 하이퍼파라미터 드랍아웃 비율 p를 지정
- 과대적합 규제하는 방법 중에 하나

3. Analysis & Result

(3) 모델 정의 및 모델 구축





Adam optimizer

- ✓ 기존 경사하강법(SGD)은 local minimum에 빠질 위험이 있음
- ✓ 각 파라미터의 기울기 값에 따라 다르게 계산하는 적응적 학습률 알고리즘이 필요함
- ✓ 따라서 모멘텀 최적화와 RMSProp을 합친 Adam 최적화 기법을 사용 (딥러닝에서 많이 사용됨)
- ✓ 최적점에 갈수록 지정된 비율에 따라 학습률을 줄이기 위해 하이퍼파라미터 decay_rate 지정
- ✓ 드롭 아웃 규제를 적용하였을 때, Adam이 가장 빠르게 수렴

3. Analysis & Result

(3) 모델 훈련

```
'''
20개에 한 번씩 업데이터 실행
0:미출력, 1:진행상황출력, 2:에포크당 출력
'''
checkpoint_cb = keras.callbacks.ModelCheckpoint('dnn_kotra.h5')
early_stopping_cb = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
hist = model.fit(
    x_train, y_train,
    batch_size=20,
    epochs=500,
    validation_data=(x_test, y_test),
    callbacks=[checkpoint_cb, early_stopping_cb],
    verbose=1)

# 테스트 데이터 입력
scores = model.evaluate(x_test, y_test)
print('test_loss: ', scores[0])
print('test_mape: ', scores[1])
```

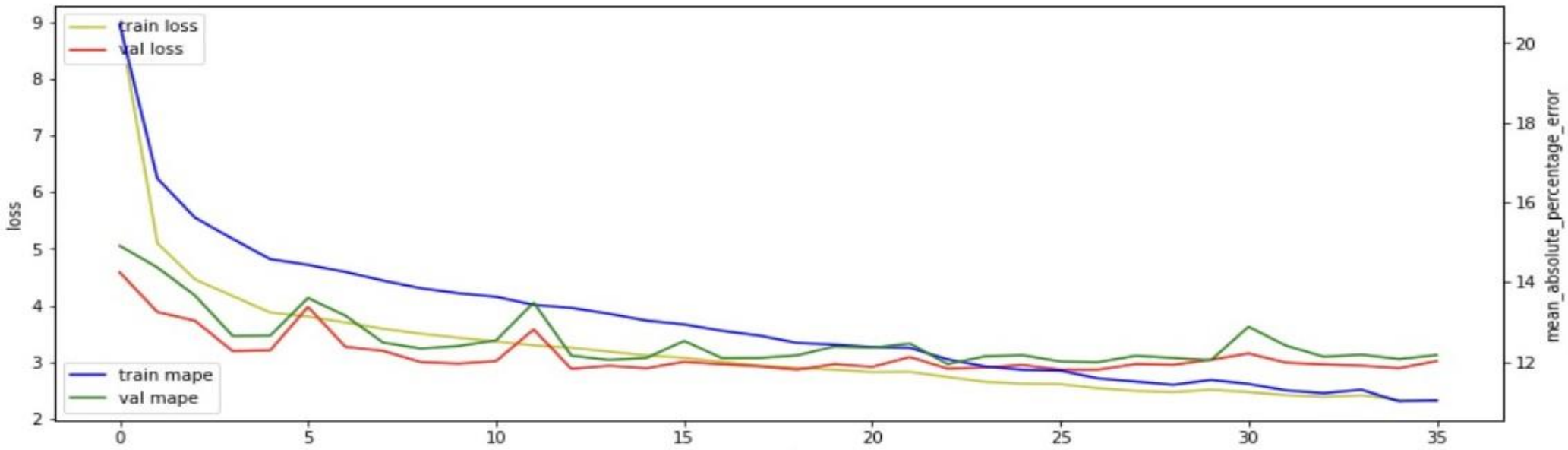
```
Epoch 33/500
848/848 [==============================] - 8s 9ms/step - loss: 2.3834 - mape: 11.2157 - val_loss: 2.9563 - val_mape: 12.1287
Epoch 34/500
848/848 [==============================] - 8s 9ms/step - loss: 2.4121 - mape: 11.2987 - val_loss: 2.9340 - val_mape: 12.1806
Epoch 35/500
848/848 [==============================] - 8s 9ms/step - loss: 2.3297 - mape: 11.0113 - val_loss: 2.8806 - val_mape: 12.0724
Epoch 36/500
848/848 [==============================] - 8s 9ms/step - loss: 2.3059 - mape: 11.0335 - val_loss: 3.0163 - val_mape: 12.1706
133/133 [==============================] - 0s 3ms/step - loss: 2.8611 - mape: 12.0100
test_loss:  2.86114239692688
test_mape:  12.009997367858887
```

- Batch_size (전체 트레이닝 데이터 셋을 여러 작은 그룹을 나누었을 때 batch size는 하나의 소그룹에 속하는 데이터 수) = 10
- Epochs(전체 트레이닝 셋이 신경망을 통과한 횟수) = 100
- EarlyStopping(조기종료) patience = 10

➢ 과대적합을 막기 위해 10 에포크 동안 훈련이 진행되는 동안 정확도 개선이 없으면 조기 종료.

3. Analysis & Result

(4) 훈련 결과



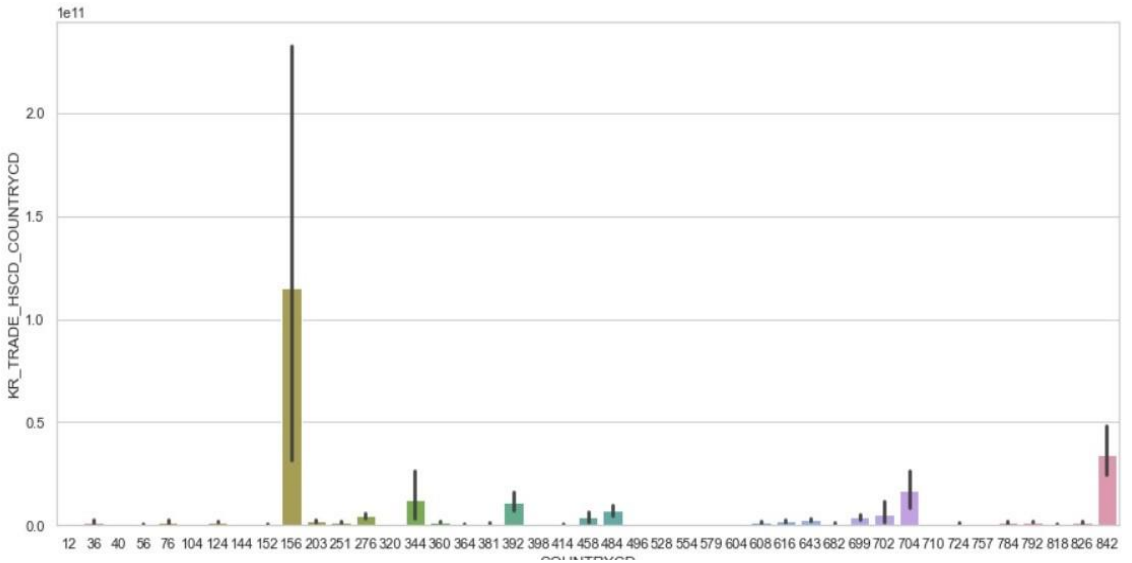
- ✓ 훈련 결과 최종 loss 값은 2.86 , 최종 mape 값은 12.01의 결과 도출
- ✓ 2개의 은닉층을 가진 간단한 구조의 DNN 모델보다 더 좋은 성능을 보여줌 (12.49 MAPE)
- ✓ 검증 데이터 손실이 훈련 데이터보다 높아지기 전 조기 종료되는 것을 볼 수 있음

4. Conclusion

4. Conclusion

- ✓ 해당 예측 모델을 통해 해외 시장에 수출하려는 기업들에게 가이드라인을 제시할 수 있음
- ✓ 2019년에는 중국, 미국, 베트남 순으로 한국으로부터 수입액이 가장 많을 것으로 예측
- ✓ Ex) 중국이 한국에 가장 많이 수입할 품목코드를 예측해 중국시장에 진출하려는 기업에게 어떠한 품목이 수출 전망이 좋을지 제시할 수 있음

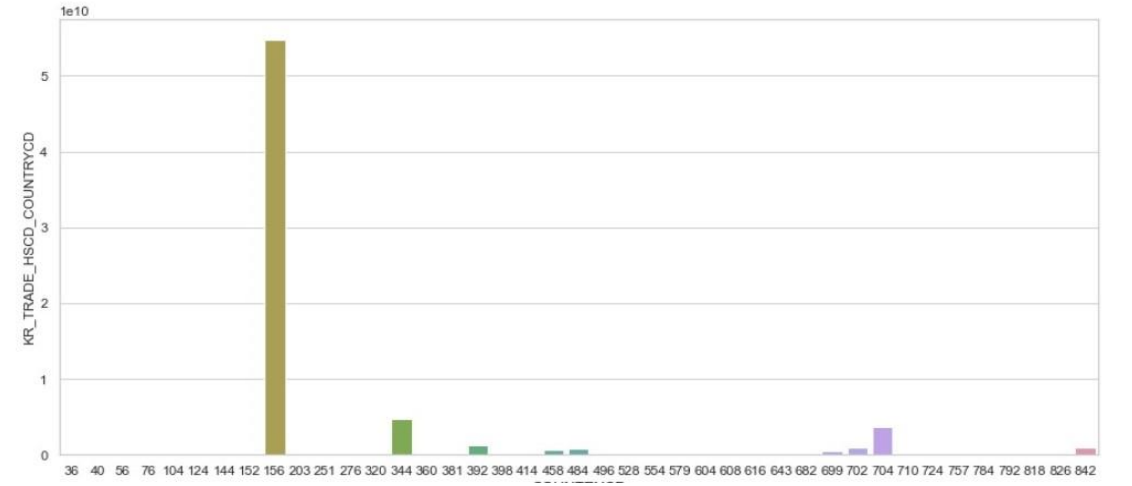
```
#2019년에는 중국(156), 미국(842), 베트남(704) 순으로 총 수입액이 많을 것이라고 예측
plt.figure(figsize=(15, 7))
sns.barplot(x='COUNTRYCD', y='KR_TRADE_HSCD_COUNTRYCD', data=data, estimator=np.sum)
plt.show()
```




```
#중국이 한국에 제일 많이 수입하는 품목코드
data.iat[China['KR_TRADE_HSCD_COUNTRYCD'].idxmax(),1]
```

```
854232
```

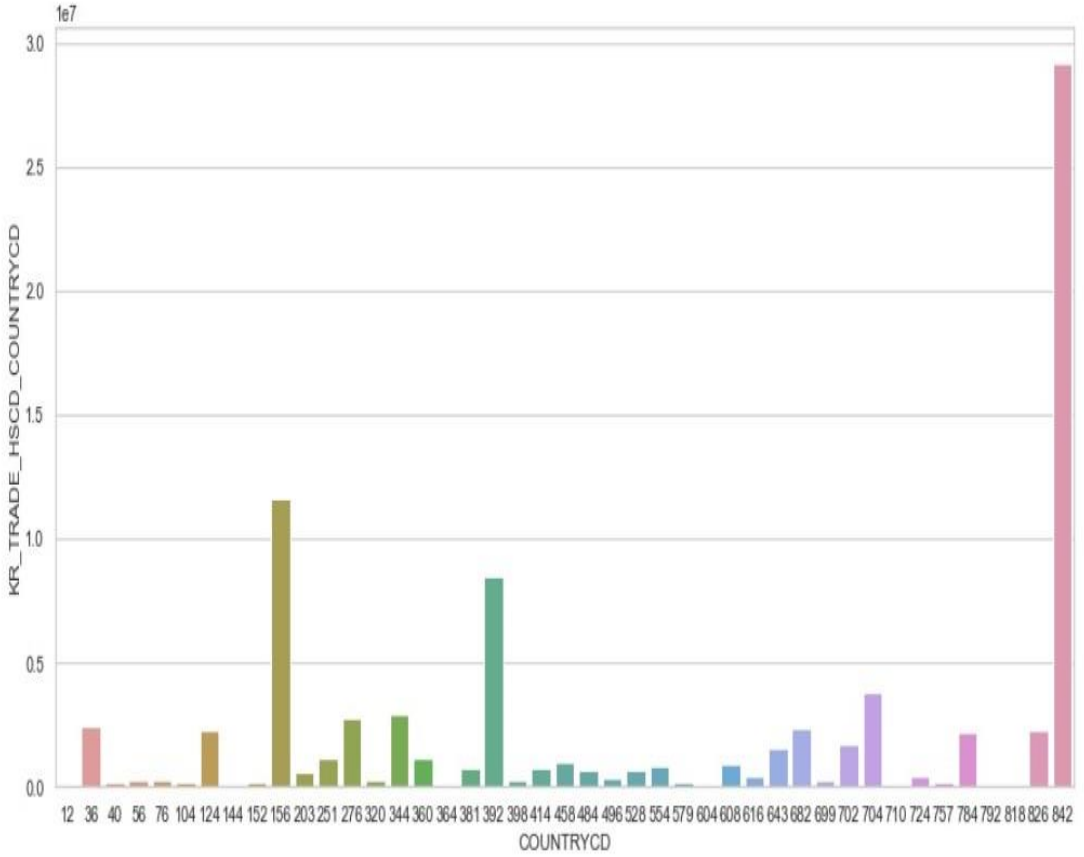
```
#중국이 한국에 제일 많이 수입하는 품목코드
plt.figure(figsize=(15, 7))
sns.barplot(x='COUNTRYCD', y='KR_TRADE_HSCD_COUNTRYCD', data=data[data['HSCD'] == 854232])
plt.show()
```



4. Conclusion

- ✓ 특정 품목코드를 어느 나라가 제일 한국에 수입을 많이 할지 예측할 수 있음
- ✓ 특정 품목을 수출하려는 기업에게 전망이 좋을 해외 시장을 예측해 수출할 국가를 제시할 수 있음
- ✓ Ex) 품목코드가 190590인 품목을 수출하려는 기업에게 미국이 제일 많이 수입할 것이라고 예측함으로써 진출할 해외 시장을 미국으로 제시할 수 있음

```
#특정 품목코드를 어느 나라가 제일 한국에 수입을 많이 할지 예측할 수 있음
#품목코드가 190590인 품목은 미국(국가코드 842)이 제일 많이 한국에 수입할 것이라고 예측
plt.figure(figsize=(15, 7))
sns.barplot(x='COUNTRYCD', y='KR_TRADE_HSCD_COUNTRYCD', data=data[data['HSCD'] == 190590])
plt.show()
```



5. 개선점

✓ **결측 값 처리에 대해서 더 고민이 필요**

- 결측 값이 10퍼센트 미만인 데이터들은 최소값으로 대치했지만 어떻게 처리해야 예측 정확도를 높일 수 있는지 더 연구가 필요
- TARIFF_AVG에 관해 결측 값이 존재할 뿐만 아니라 0값을 가진 데이터도 너무 많음

```
#전체 데이터에서 0값을 가진 TARIFF_AVG의 비율
tariff = df[df['TARIFF_AVG'] == 0]
(len(tariff.index) / df.shape[0]) * 100
```

```
46.502902449384116
```

- 이러한 데이터는 타깃 데이터를 예측하는데 방해를 줄 수 있음
- 타깃 값 중에 0인 값들도 존재 -> 그대로 두면 예측 정확도에 큰 영향을 줌

✓ **타깃 값에 대해 유의미한 영향을 줄 다른 독립변수에 대한 조사가 필요**

✓ **모델 구조와 하이퍼 파라미터에 대해 더 연구가 필요**

- 일반화 성능을 높일 수 있는 모델 구조가 필요 (활성화 함수, 최적화 함수의 학습률)
- 훈련 데이터 뿐만 아니라 검증 데이터의 정확도를 높여야 함 (과적합을 피해야함)

참고자료

- 권철민, *파이썬 머신러닝 완벽가이드*, 파주:위키북스, 2020, 353 ~ 374
- 오일석, *MACHINE LEARNING 기계학습*, 서울: 한빛아카데미
- “Training Neural Network for price prediction with Tensorflow”, *towards data science* https://towardsdatascience.com/training-neural-networks-for-price-prediction-with-tensorflow-8aafe0c55198
- 오렐리앙 제롱, *핸즈온 머신러닝 사이킷런, 케라스, 텐서플로 2를 활용한 머신러닝, 딥러닝 완벽 실무*, 서울:한빛미디어, 2020, 412 ~ p.459
- Sergey Ioffe and Christian Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, Proceedings of the 32nd International Conference on Machine Learning (2015): 448 - 456
- Diederik P. Kingma, Jimmy Ba, “Adam: A Method for Stochastic Optimization”, International Conference on Learning Representations, 2015

THANK YOU

HAVE A NICE DAY!