# 08. Neural Network

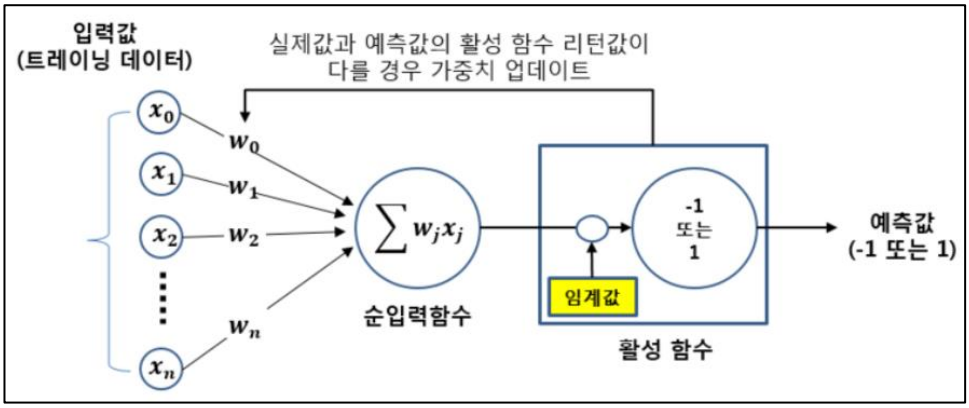
Artificial Neural Network

요조 [곽지운, 박현준, 이정현, 최동훈]

## 1. Artificial neural network : MLP, RBF와 SOM

### 1.1 MLP(Multiple Layer Perceptron, 다층 퍼셉트론)

① **Perceptron(퍼셉트론)**




퍼셉트론 알고리즘은 다수의 신호(Input)을 입력받아서 하나의 신호(Output)을 출력한다. 이는 뉴런이 여러 입력신호를 받아들여 하나의 신호로 통합하고, 통합된 신호 값이 어떤 임계값을 초과하면 하나의 단일신호를 생성하여 다른 뉴런으로 전전달하는 것에 바탕한것이다.

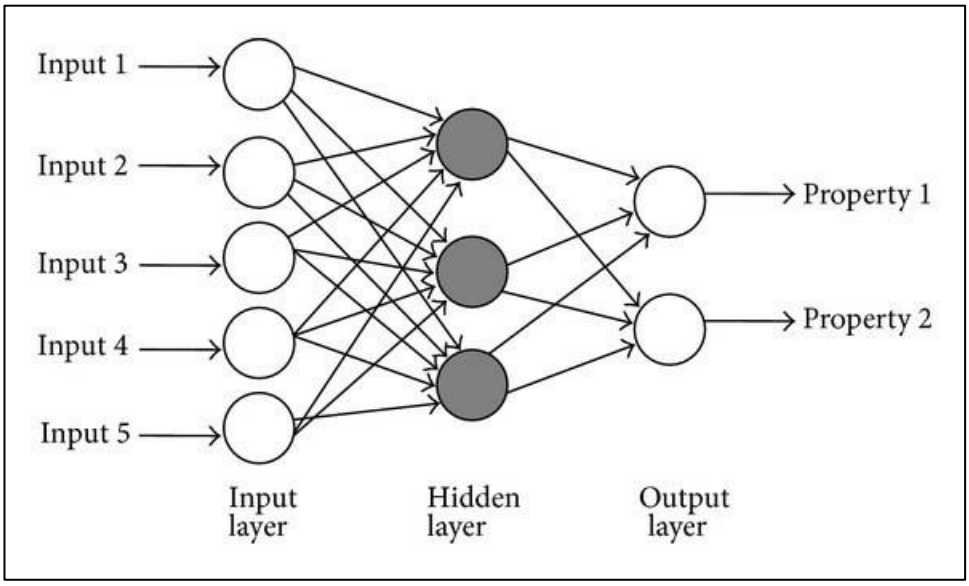
$x_0 \sim x_n$은 퍼셉트론 알고리즘으로 입력되는 값이며, $w_0 \sim w_n$은 입력값에 곱해지는 weight(가중치, parameter)로서 입력신호가 출력값에 주는 영향도를 조절하는 매개변수이다. 기계학습 과정을 통해 이 weight값을 결정하게되고 결정된 weight값이 클수록 해당 입력신호가 중요하다고 볼수있다.

활성 함수(Activation function)은 순입력 함수의 결과값이 임계값보다 크면 1 그렇지않은 경우에는 -1을 출력한다.

퍼셉트론은 training data를 사용하여 supervised learning(지도학습)을 수행하는 알고리즘으로, 입력데이터에 대응되는 실제데이터 y를 가지고있어야한다.

그리고 입력데이터로부터 퍼셉트론 알고리즘을 통해 예측데이터를 구하였을 때 실제데이터 y값과 다른 경우, 오류가 최소화될때까지 특정식을 통하여 weight $w_0 \sim w_n$을 update하게 된다.

② **Multiple Layer Perceptron(다층 퍼셉트론)**



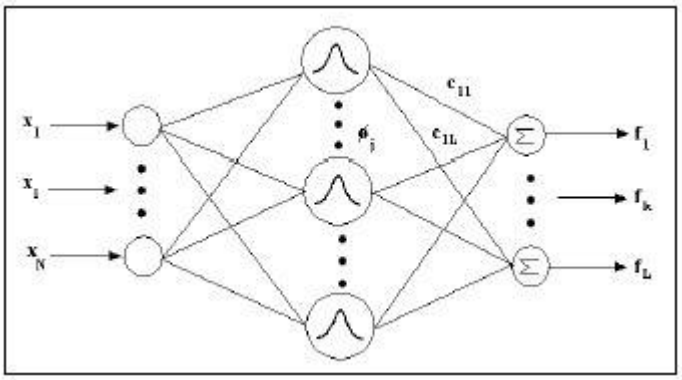

다층 퍼셉트론은 입력층과 출력층 사이에 퍼셉트론으로 이루어진 hidden layer(은닉계층)이 하나 이상 존재하는 신경망이다. 다층 퍼셉트론은 인접한 층의 퍼셉트론간의 연결은 존재해도 같은 층의 퍼셉트론끼리의 연결은 존재하지않는 feedforward neural network(순방향 신경망)이다. 또한 한번 지나간 층으로 다시 연결되는 feedback도 존재하지않는다.

다층 퍼셉트론을 사용하는 이유로, 예를들면 이미지를 고양이와 강아지로 분류하는 경우, 입력과 출력간에 매우 복잡한 관계와 이를 위한 많은 feature들 사이의 관계를 통해 특정 지어지는 가능성을 고려할수있습니다. 이 경우 선형모델을 사용할 경우, 간단한 XOR논리식의 결과도 고려하지못하는등, 정확도가 낮을것이며, 이때 여러 개의 은닉계층을 사용하여 비선형적이고, 일반적인 모델을 만들게됨으로써 정확도를 높일수있게됩니다.

다층 퍼셉트론은 입력층에서 전달되는 값이 은닉층의 모든 노드로 전달되며 모든 노드의 출력값 역시 출력층의 모든 노드로 전달된다. 이 과정을 순전파(Feedforward)라고 한다.

동작원리는 단층 퍼셉트론과 동일하게 활성화 함수가 내놓는 결과값이 실제값과 오류가 최소화되도록 가중치값을 결정하는것이다. 이때 다층 퍼셉트론은 은닉계층과 출력층에 여러 개의 활성화 함수가 존재하고, 이에 따른 가중치도 여러 개 존재하게된다. 그렇게 모든 training data에 대해서 출력층의 활성화 함수에 의한 결과값이 실제값의 허용 오차 이내가 되면 학습을 종료한다.

### 1.2 RBF(Radial Basis Function Neural Network, 방사형 기저함수 신경망)




RBF는 방사형 구조를 기본으로 하는 네트워크로, 은닉층을 1개만 가진다는 특징이 있다. RBFNN은 입력층, 은닉층, 출력층 등 3개의 다른 층으로 이루어져 있고, 네트워크의 도식은 위 그림과 같다. 입력값을 은닉층과 연결시킬 때 RBFNN은 linear function이 아닌 다음과 같은 함수를 이용한다.

$$H_i = \exp\left( -\frac{(x_1 - c_{1i})^2 + (x_2 - c_{2i})^2 + \ldots + (x_k - c_{ki})^2}{r_i^2} \right)$$

각 은닉층 노드는 반경이 r이며 종모양의 표면을 나타낸다. C는 은닉층 노드 종 모양의 가중치를 의미하고, 중점과 반경은 자율모형 추정방법에 의해 추정한다. 중점과 반경이 결정되면 다중 로지스틱 회귀분석을 이용하여 추정을 진행할 수 있다.

### 1.3 SOM(Self Organizing Map, 자기조직화 지도)

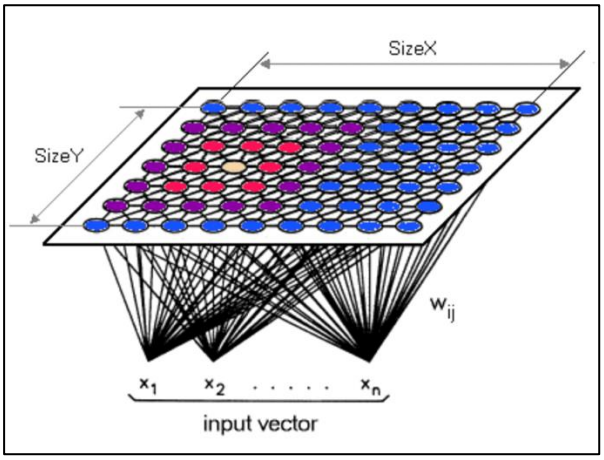
① **기본 개념**

자기조직화지도(Self-organizing map, SOM)는 대뇌피질의 시각피질을 모델화한 인공신경망의 일종이다. 또한 SOM 모델은 차원축소(dimensionality reduction)와 군집화(clustering)을 동시에 수행하는 분류기법으로 사용되는 알고리즘이다.

SOM의 최대 장점은 통합거리맥트릭스(Unified Distance Matrix, U-Matrix)를 이용하여 손쉽게 2차원 데이터의 클러스터를 생성하고 데이터가 암시하는 패턴, 흥미로운 사실을 이해할 수 있다. 하지만 어떤 유사성과 거리 함수를 선택하느냐에 따라 클러스터의 내용이 매우 크게 달라질 수 있으며, SOM 모델의 수학 연산상의 복잡성으로 인해 수천 개 이상의 데이터 Set은 분석하는 것이 불가능하다는 단점이 존재한다.

② **학습단계**

SOM이란 사람이 눈으로 볼 수 있는 저차원(2~3차원) 격자에 고차원 데이터의 개체들이 대응하도록 인공신경망과 유사한 방식의 학습을 통해 군집을 도출해내는 기법이다. 고차원의 데이터 원공간에서 유사한 개체들은 저차원에 인접한 격자들과 연결되는데, 저차원 격자에서의 유사도는 고차원 입력 공간에서의 유사도를 최대한 보존하도록 학습된다.




SOM에는 오직 숫자형 데이터만 입력할 수 있으며 모델 파라미터는 거리함수(유클리드 거리를 대표적으로 사용), 그리고 격자 또는 래티스 파라미트(폭과 높이 또는 래티스에 존재하는 셀의 수) 등을 사용한다. SOM 알고리즘 학습 방식은 초기에 가중치(그림 상의 wij)는 랜덤값으로 생성되며 다음과 같이 세 단계의 과정을 거치게 된다.

- ***경쟁단계:***

이 단계의 각 뉴런의 가중치와 입력 데이터의 벡터간의 거리를 산출 및 비교하여 뉴런간의 경쟁을 벌인다. 예를 들어 입력 데이터의 할당 여부를 결정할 때, 입력 데이터와 거리가 가까운 뉴런들을 선정한다.

- ***협력단계:***

경쟁에서 선정한 뉴런은 토폴로지 이웃(Topological neighborhood) 영역에서 가장 좋은 공간 위치를 차지하게 되며, 뉴런(j,i) 거리 Sij, 이웃크기 일 때, 승리한 뉴런의 토폴로지 이웃은 다음과 같이 정의된다.

$$T_{j,I(x)} = \exp\left(-\frac{S_{ij}^2}{2\sigma(t)^2}\right)$$

토폴로지 이웃의 크기는 지수함수 등, 감쇄 함수(decay functions)들을 이용해서 다음과 같이 정의할 수 있다. 여기서 t는 학습 횟수(time-step)이다.

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_0}\right)$$

- ***적응단계:***

이 단계에서는 승리한 뉴런의 가중치와 토폴로지 이웃 뉴런을 업데이터 하게 된다. 가중치 업데이트는 음과 같은 함수를 사용하게 된다.

$$\Delta w_{ji} = \eta(t) T_{j,I(x)}(t)\left(x_i - w_{ji}\right)$$

위의 식에서 n(t)는 t스탭에서 학습률을 뜻한다. 원하는 횟수만큼 협력단계, 적응단계를 반복하여 적용하여 가중치를 업데이트 한다.

## 2. 데이터

### 2.1 변수 설명 및 기초통계

Python sklearn dataset에 있는 보스턴의 집값 데이터를 활용하였다. 해당 데이터는 1978년 보스턴 교외의 506개의 주택에 대해 14개 범주의 수치를 정리한 것이다. 14개의 변수들은 다음과 같다.

| | |
|---|---|
| **CRIM** | 도시의 1인당 범죄율 |
| **ZN** | 25,000 평방 피트가 넘는 주택 비율 |
| **INDUS** | 타운당 비소매 사업 면적 비율 |
| **CHAS** | 길이 찰스강과 경계하면 1, 아니면 0 |
| **NOX** | 일산화질소 농도 (0.1ppm 단위) |
| **RM** | 주택 당 평균 방 수 |
| **AGE** | 1940년 이전에 건축된 주인이 거주하는 주택 비율 |
| **DIS** | 보스턴의 5개 고용센터까지 가중치 거리 |
| **RAD** | 방사형 고속도로로의 접근성 지수 |
| **TAX** | $10,000 당 최대 재산세율 |
| **PTRATIO** | 타운별 학생-교사 비율 |
| **B** | 아프리카계 미국인의 비율 |
| **LSTAT** | 저소득층 계층의 비율 |
| **MEDV (target)** | **주인이 거주하는 주택 가격의 중간값 ($1,000 단위)** |

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | MEDV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.00632 | 18.0 | 2.31 | 0.0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1.0 | 296.0 | 15.3 | 396.90 | 4.98 | 24.0 |
| 1 | 0.02731 | 0.0 | 7.07 | 0.0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2.0 | 242.0 | 17.8 | 396.90 | 9.14 | 21.6 |
| 2 | 0.02729 | 0.0 | 7.07 | 0.0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2.0 | 242.0 | 17.8 | 392.83 | 4.03 | 34.7 |
| 3 | 0.03237 | 0.0 | 2.18 | 0.0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3.0 | 222.0 | 18.7 | 394.63 | 2.94 | 33.4 |
| 4 | 0.06905 | 0.0 | 2.18 | 0.0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3.0 | 222.0 | 18.7 | 396.90 | 5.33 | 36.2 |

해당 데이터는 총 506개의 데이터로 구성되어있다. 총 14개의 변수 중 설명 변수는 13개이고, 설명 변수 중 범주형 변수 1개가 존재하고, 나머지는 연속형 변수이다. 종속변수 MEDV는 연속형 변수이며 따라서 "Regression"을 진행하게된다.

그리고 5개의 설명변수를 임의 선택후 종속변수와 함께 새로운 데이터프레임을 만들어 진행하였다.

```
dataset_new = df.copy().loc[:,['INDUS','RM','AGE','TAX','DIS','MEDV']]
```

| | INDUS | RM | AGE | TAX | DIS | MEDV |
|---|---|---|---|---|---|---|
| **0** | 2.31 | 6.575 | 65.2 | 296.0 | 4.0900 | 24.0 |
| **1** | 7.07 | 6.421 | 78.9 | 242.0 | 4.9671 | 21.6 |
| **2** | 7.07 | 7.185 | 61.1 | 242.0 | 4.9671 | 34.7 |
| **3** | 2.18 | 6.998 | 45.8 | 222.0 | 6.0622 | 33.4 |
| **4** | 2.18 | 7.147 | 54.2 | 222.0 | 6.0622 | 36.2 |
| **...** | ... | ... | ... | ... | ... | ... |
| **501** | 11.93 | 6.593 | 69.1 | 273.0 | 2.4786 | 22.4 |
| **502** | 11.93 | 6.120 | 76.7 | 273.0 | 2.2875 | 20.6 |
| **503** | 11.93 | 6.976 | 91.0 | 273.0 | 2.1675 | 23.9 |
| **504** | 11.93 | 6.794 | 89.3 | 273.0 | 2.3889 | 22.0 |
| **505** | 11.93 | 6.030 | 80.8 | 273.0 | 2.5050 | 11.9 |

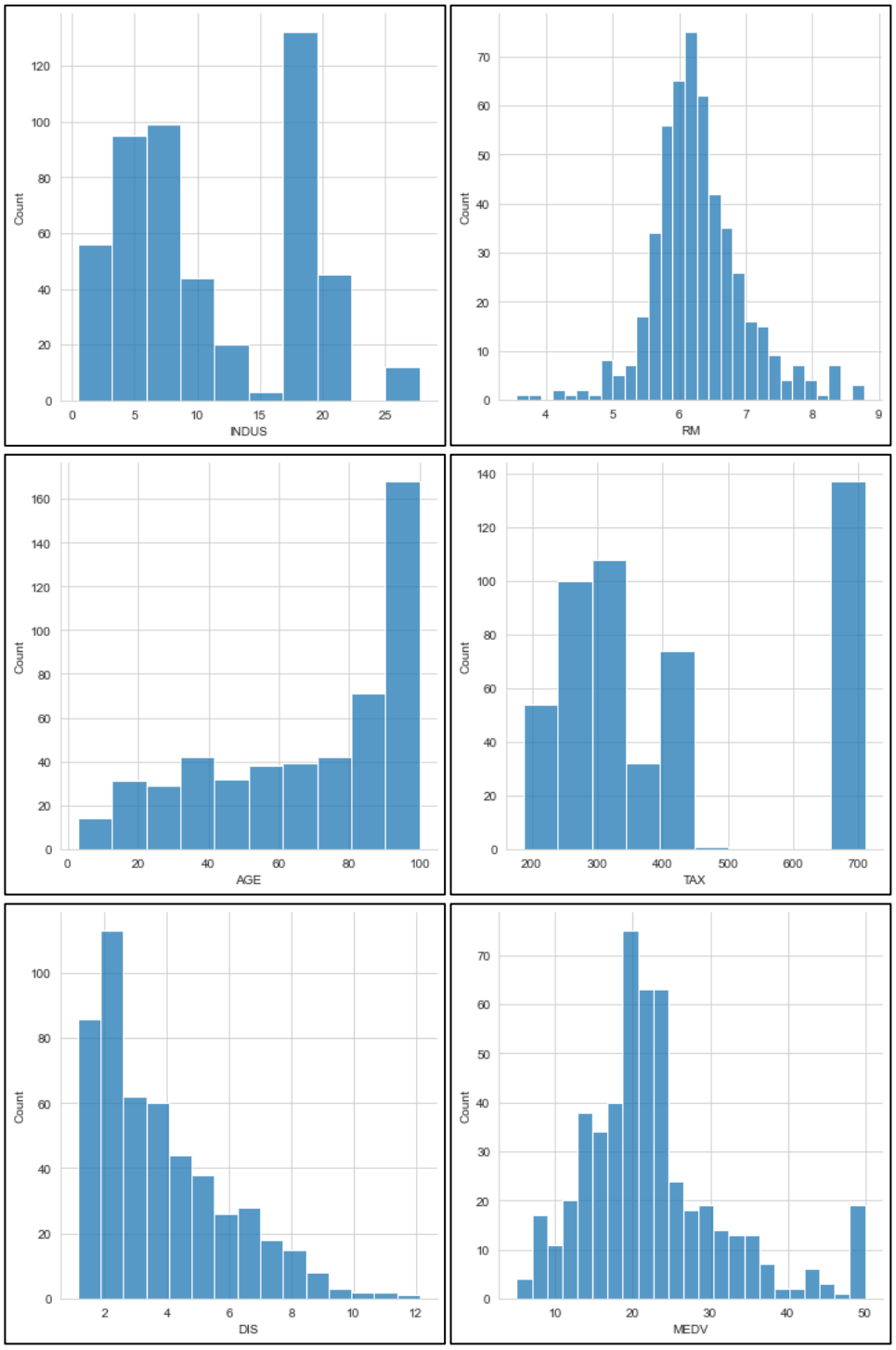
아래는 각 feature별 평균과 분산과 각 feature들에대한 분포를 시각화로 표현한것이다.

```
dataset_new.mean()
```

```
INDUS      11.136779
RM          6.284634
AGE        68.574901
TAX       408.237154
DIS         3.795043
MEDV       22.532806
dtype: float64
```

```
dataset_new.var()
```

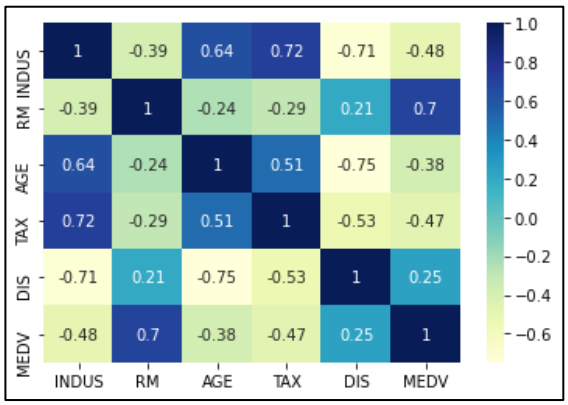
```
INDUS        47.064442
RM            0.493671
AGE         792.358399
TAX       28404.759488
DIS           4.434015
MEDV         84.586724
dtype: float64
```




다음은 변수간의 상관계수를 나타낸것이며, 시각화를 한것이다.

```
dataset_new.corr()
```

| | INDUS | RM | AGE | TAX | DIS | MEDV |
|---|---|---|---|---|---|---|
| **INDUS** | 1.000000 | -0.391676 | 0.644779 | 0.720760 | -0.708027 | -0.483725 |
| **RM** | -0.391676 | 1.000000 | -0.240265 | -0.292048 | 0.205246 | 0.695360 |
| **AGE** | 0.644779 | -0.240265 | 1.000000 | 0.506456 | -0.747881 | -0.376955 |
| **TAX** | 0.720760 | -0.292048 | 0.506456 | 1.000000 | -0.534432 | -0.468536 |
| **DIS** | -0.708027 | 0.205246 | -0.747881 | -0.534432 | 1.000000 | 0.249929 |
| **MEDV** | -0.483725 | 0.695360 | -0.376955 | -0.468536 | 0.249929 | 1.000000 |



## 2.2 데이터 전처리

### ① Feature, Target 분리

먼저 앞에서 나온 데이터프레임을 feature와 target으로 구분하였다.

```
# Features와 target 나누기
features = dataset_new[dataset_new.columns[:-1]]
target = dataset_new[dataset_new.columns[-1]]
```

### ② Training data, Test data 분리

전체 데이터를 8:2비율의 training data와 test data로 분류하였고, 다음과 같다.

```
# train: test = 8:2 분리
from sklearn.model_selection import train_test_split
train_features, test_features , train_target, test_target = train_test_split(
    features, target, test_size = 0.2, random_state = 2021)
```

```
print(train_features.shape)
print(train_target.shape)
print(test_features.shape)
print(test_target.shape)
```

```
(404, 5)
(404,)
(102, 5)
(102,)
```

| | INDUS | RM | AGE | TAX | DIS |
|---|---|---|---|---|---|
| **28** | 8.14 | 6.495 | 94.4 | 307.0 | 4.4547 |
| **498** | 9.69 | 6.019 | 65.3 | 391.0 | 2.4091 |
| **284** | 2.97 | 7.088 | 20.8 | 285.0 | 7.3073 |
| **414** | 18.10 | 4.519 | 100.0 | 666.0 | 1.6582 |
| **123** | 25.65 | 5.856 | 97.0 | 188.0 | 1.9444 |
| **...** | ... | ... | ... | ... | ... |
| **109** | 8.56 | 6.229 | 91.2 | 384.0 | 2.5451 |
| **128** | 21.89 | 6.431 | 98.8 | 437.0 | 1.8125 |
| **57** | 1.32 | 6.816 | 40.5 | 256.0 | 8.3248 |
| **341** | 1.52 | 7.241 | 49.3 | 284.0 | 7.0379 |
| **116** | 10.01 | 6.176 | 72.5 | 432.0 | 2.7301 |

404 rows × 5 columns

| | MEDV |
|---|---|
| **28** | 18.4 |
| **498** | 21.2 |
| **284** | 32.2 |
| **414** | 7.0 |
| **123** | 17.3 |
| **...** | ... |
| **109** | 19.4 |
| **128** | 18.0 |
| **57** | 31.6 |
| **341** | 32.7 |
| **116** | 21.2 |

404 rows × 1 columns

### ③ Standardization(표준화)

신경망 모델을 fitting하기전 training data를 기준으로 Z-score표준화를 하였다.

```
# Z-score Standardization

mean = train_features.mean(axis=0)
train_features -= mean
std = train_features.std(axis=0)
train_features /= std
```

```
# test data (Features)도 train의 기준으로 표준화
test_features -= mean
test_features /= std
```

### 2.3 Parameter개수에 따른 Sample size 평가

Layer의 Node수를 증가시키면 parameter(weight)의 개수가 증가하며 Node의 수를 크게 늘린다면 , 이에 비례하는 충분한 데이터수가 필요하다.

아래의 표는 뒷부분 **3. Model**에서 진행한 18개의 신경망 모델들과 각 경우에서 도출된 parameter의 개수를 종합한것이다. Layer, Node, Learning Rate에 따라 18개로 분류해 학습을 진행한 모델들에서 parameter의 개수, 즉 weight의 개수는 [57, 113, 129, 201, 209, 337]개가 나오게된다.

현재의 경우 404개의 training data를 사용하여 학습을 진행하였고 parameter의 개수를 고려하였을 때 적절히 node를 설정한것으로 판단된다.

| # | Layer 개수 | Node | Learning Rate | parameter |
|---|---|---|---|---|
| 1 | 1 | 8 | 0.1 | 57 |
| 2 | 1 | 16 | 0.1 | 113 |
| 3 | 2 | [8,16] | 0.1 | 209 |
| 4 | 2 | [8,8] | 0.1 | 129 |
| 5 | 3 | [8,8,8] | 0.1 | 201 |
| 6 | 3 | [8,16,8] | 0.1 | 337 |
| 7 | 1 | 8 | 1 | 57 |
| 8 | 1 | 16 | 1 | 113 |
| 9 | 2 | [8,16] | 1 | 209 |
| 10 | 2 | [8,8] | 1 | 129 |
| 11 | 3 | [8,8,8] | 1 | 201 |
| 12 | 3 | [8,16,8] | 1 | 337 |
| 13 | 1 | 8 | 0.001 | 57 |
| 14 | 1 | 16 | 0.001 | 113 |
| 15 | 2 | [8,16] | 0.001 | 209 |
| 16 | 2 | [8,8] | 0.001 | 129 |
| 17 | 3 | [8,8,8] | 0.001 | 201 |
| 18 | 3 | [8,16,8] | 0.001 | 337 |

## 3. Modeling

### 3.1. 모델링 전 코드 설정

```
# Network 구조 형성
import tensorflow as tf
from tensorflow.keras import models
from tensorflow.keras import layers

# tf.set_random_seed(2021) <- 이건 tensorflow 1 version
tf.random.set_seed(2021)

# 1 #
model1 = models.Sequential() # ANN 형성
model1.add(layers.Dense(8, activation = 'relu', input_shape=(train_features.shape[1],)))
model1.add(layers.Dense(1)) # Output layer - regression은 node가 1개

# 2 #
# loss function과 optimizer 설정
adam = tf.keras.optimizers.Adam(learning_rate=0.1)

# 3 #
model1.compile(optimizer=adam, loss='mse', metrics=['mape'])

# 콜백 함수 #
es = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, verbose=1, restore_best_weights=True)
lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.1, min_lr=0.001) # learning rate 관련


# 4 #
# model.fit(train_data, train_targets, epochs=80, batch_size=16)
reg_history1 = model1.fit(train_features, train_target, epochs=300, batch_size=1024, validation_split=0.2,
                          callbacks=[es,lr])
```

본 과제를 위해 데이터에 ANN 모델 적용을 총 18가지 모델에 진행을 하였다. Hidden Layer의 개수를 3가지, 각각 Hidden node 수를 2가지 방식으로 다르게 하였고, Learning Rate를 0,1 / 1 /0.01로 나누어서 총 18가지의 모델이 나왔다. 총 18가지 모델에서 optimizer의 loss function, metric와 콜백 함수는 공통적으로 지정을 했다.

Optimizer는 Adam(Adaptive Momentum Estimation)을 선택했다. Adam은 RMSprop과 Momentum을 섞은 것이다. Adam에서 RMSprop과 Momentum에서 v(Momentum 계수), h(AdaGrad 계수)가 각각 최소 0으로 설정되어 학습 초반에 0으로 biased 되는 문제를 해결한 방법이다. 여기서 우리는 총18가지 모델에 동일하게 적용을 했고, loss function으로 'mse', metrics는 ['mape']로 지정을 해줬다. 이때, loss function(손실함수)과 metrics(측정항목 함수)은 모델의 성능을 평가하는 데 사용되는 함수다. 둘의 차이는 Metrics는 평가한 결과로 모델을 학습시키는 데 사용되지 않는다는 점이다.

$$
\begin{aligned}
m_1 &\leftarrow \beta_1 m_0 + (1-\beta_1) g_1 \\
\widehat{m_1} &\leftarrow \frac{m_1}{1-\beta_1^1} = \frac{\beta_1 m_0}{1-\beta_1^1} + \frac{(1-\beta_1) g_1}{1-\beta_1^1} \\
&= 0 + g_1 (\because m_0 = 0)
\end{aligned}
$$

콜백 함수는 EarlyStopping과 ReducLROnPlateau에 대해 지정해줬다. 콜백 함수란 어떤 이벤트가 발생했거나 특정 시점에 도달했을 때, 시스템에서 호출하는 함수를 말한다. 즉, 모델링에서는 더이상의 큰 개선이 없을 때, 학습률을 변화시키거나 멈추게 하는 등의 작업을 할 수 있는 함수 말한다. 여기서 사용한 콜백 함수에 대해 설명하면, EarlyStopping은 모델을 더 이상 학습을 못할 경우(loss, metric의 개선이 없을 경우), 학습 도중 미리 학습을 종료시키는 것이다.
ReducLROnPlateau은 모델의 개선이 없을 경우, Learning Rate를 조절해 모델의 개선을 유도하는 것이다. 각각 다음과 같이 정해줬다.

```
# 콜백 함수 #
es = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, verbose=1, restore_best_weights=True) # Early stopping
lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.1, min_lr=0.001) # learning rate 관련
```

이때, 적용한 parameter를 살펴보자, monitor는 콜백함수의 기준이 되는 값이다. 'val_loss'를 적용하여, 이 값이 더 이상 감소되지 않는 경우 콜백함수가 일어나게 했다. patience는 training이 진행되지만, 더 이상 monitor되는 값의 개선이 없을 경우, 최적의 monitor 값을 기준으로 몇 번의 epoch을 진행할 지정하는 값이다. verbose는 0이면 진행되는 과정을 안 보여주고, 1일 경우는 각 경우가 적용되는 과정을 보여준다. factor는 learning rate를 얼마나 감소시킬지 정하는 인자값이다. min_lr은 learning rate의 하한선을 나타난다. restore_best_weigths는 True일 때는 가장 좋은 weight값을 저장하고, False는 끝난 후의 weight로 두는 것이다. Early Stopping 모델의 경우에는 Patience=10을 지정해 주었다.

### 3.2. 모델링 진행

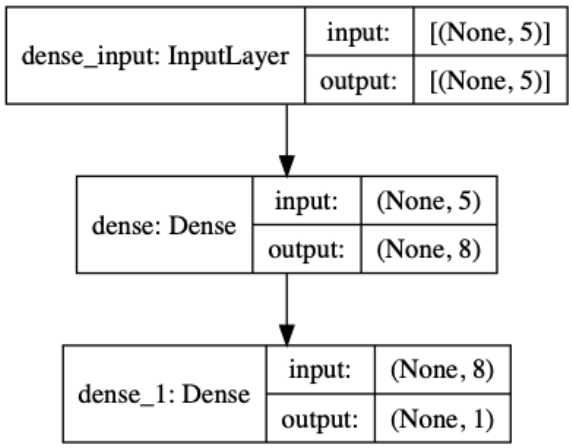
**1) Learning Rate = 0.1 인 경우**

**- Model 1 : Hidden layer = 1, Hidden Node = 8, Learning rate = 0.1, epoch = 300, batch_size=1024**

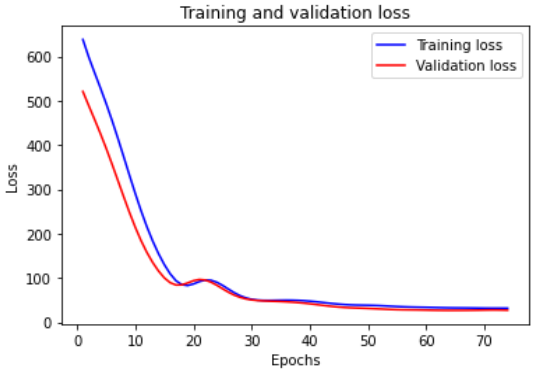
```
Epoch 74/300
1/1 [==============================] - 0s 36ms/step - loss: 31.6130 - mape: 20.4322 - val_loss: 27.1271 - val_mape: 24.0755
Restoring model weights from the end of the best epoch.
Epoch 00074: early stopping
```

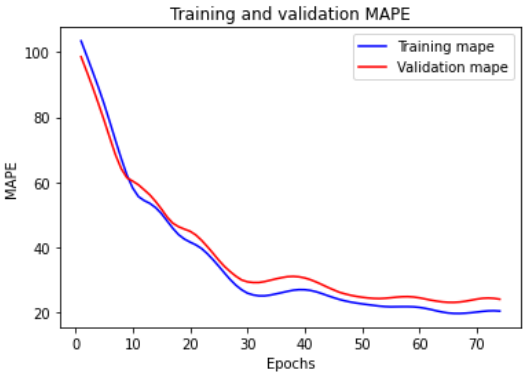
- Hidden layer = 1, activation = 'relu', Hidden Node = 8, Learning rate = 0.1, epoch = 300, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 74번째에 Early Stop했다. 그 결과, 최적의 Epoch는 74였다.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 8)                 48
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 9
=================================================================
Total params: 57
Trainable params: 57
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input노드와 hidden layer(1) node들과는 48개의 연결된 간선의 수가 있었고, hidden layer의 node 8개에 bias(b*)노드가 하나 더 추가되어, hidden layer(1) node와 output node와는 9개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 74일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 31.6130, Validation loss가 27.1271인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한MAPE 그래프의 경우 Training mape가 20.4322, Validation mape가 24.0755 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다

```
test R squared: 0.578
test MSE: 26.930
tset MAPE: 0.220
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.578, MSE는 26.930, MAPE는 0.220 값이 나온 것을 확인했다.

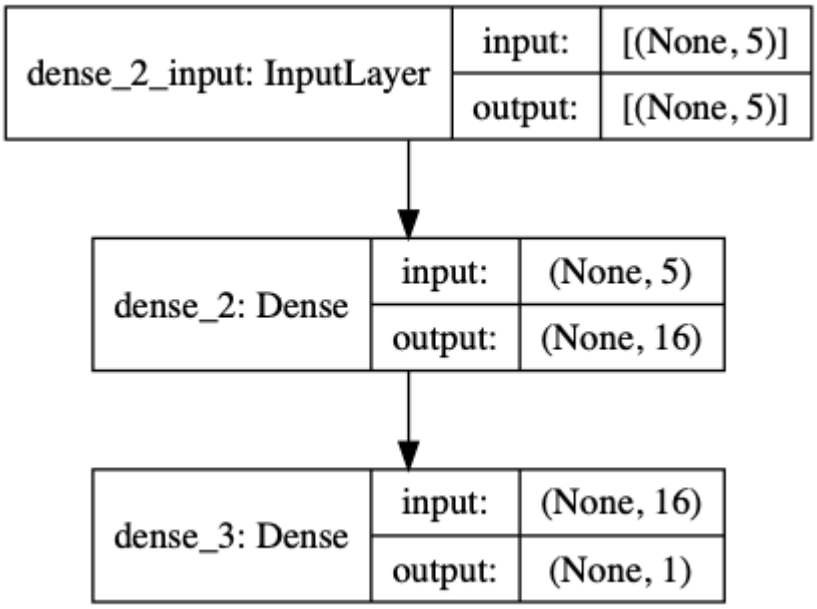
**2) Learning Rate = 0.1 인 경우**

**- Model 2 : Hidden layer = 1, Hidden Node = 8, Learning rate = 0.1, epoch = 300, batch_size=1024**

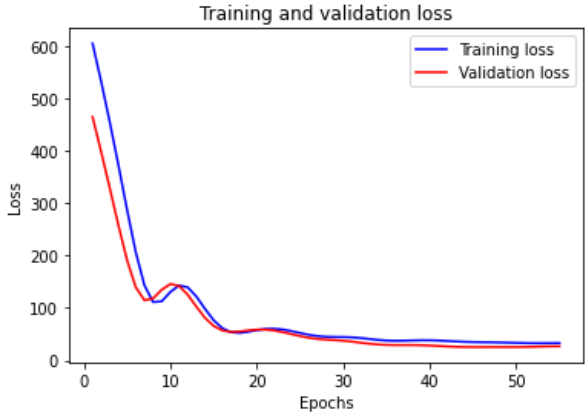
```
Epoch 55/300
1/1 [==============================] - 0s 32ms/step - loss: 32.9741 - mape: 21.0366 - val_loss: 26.8819 - val_mape: 24.6194
Restoring model weights from the end of the best epoch.
Epoch 00055: early stopping
```

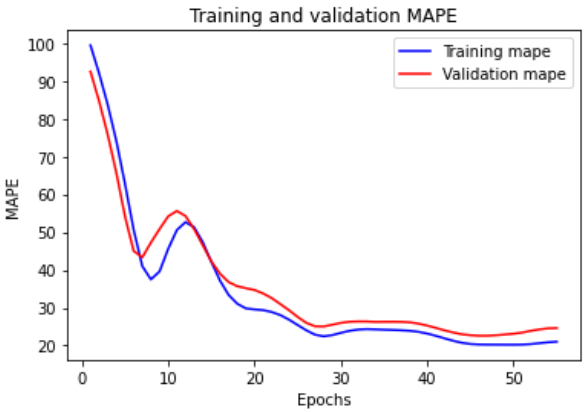
- Hidden layer = 1, activation = 'relu', Hidden Node = 16, Learning rate = 0.1, epoch = 300, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 55번째에 Early Stop했다. 그 결과, 최적의 Epoch는 55였다.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 16)                96
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
=================================================================
Total params: 113
Trainable params: 113
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input 노드와 hidden layer(1) node들과는 96개의 연결된 간선의 수가 있었고, hidden layer(1)의 node 16개에 bias(b*)노드가 하나 더 추가되어, hidden layer(1)과 output node와는 17개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 55일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 32.9741, Validation loss가 26.8819인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 21.0336, Validation mape가 24.6194 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다.

```
test R squared: 0.583
test MSE: 26.630
tset MAPE: 0.221
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.583, MSE는 26.630, MAPE는 0.221 값이 나온 것을 확인했다.

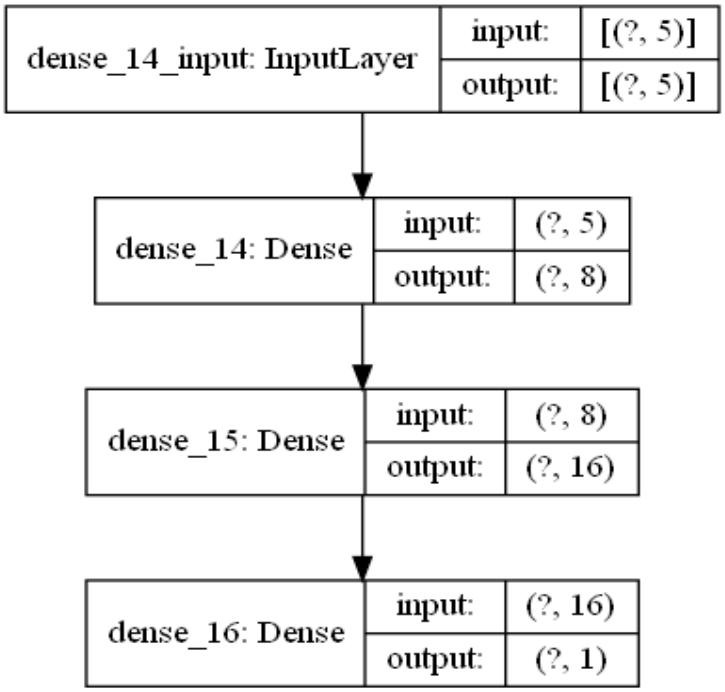
**3) Learning Rate = 0.1 인 경우**

**- Model 3 : Hidden layer = 2, Hidden Node = [8,16], Learning rate = 0.1, epoch = 300, batch_size=1024**

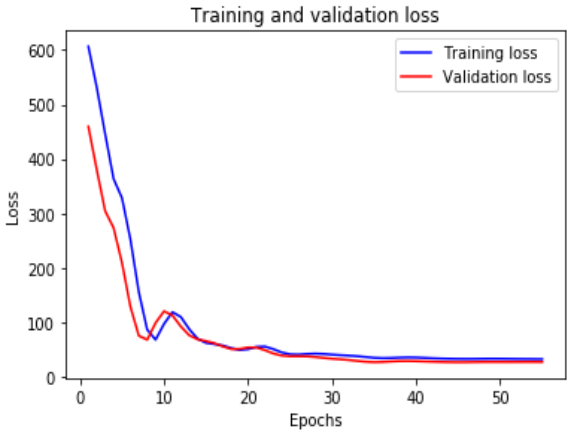
```
Epoch 55/300
1/1 [==============================] - 0s 37ms/step - loss: 32.4919 - mape: 20.3011 - val_loss: 27.3270 - val_mape: 24.7220
Restoring model weights from the end of the best epoch.
Epoch 00055: early stopping
```

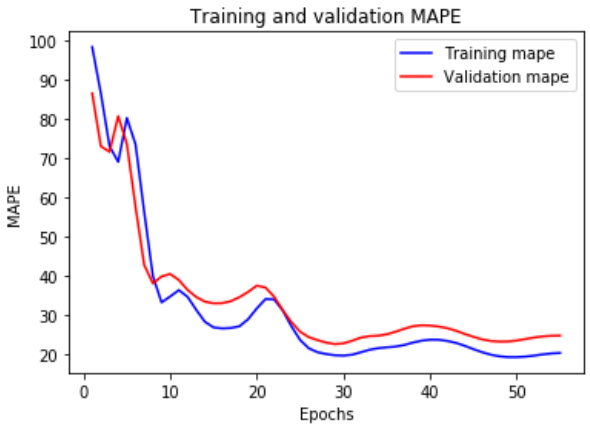
- Hidden layer = 2, activation = 'relu', Hidden Node = [8,16], Learning rate = 0.1, epoch = 300, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 55번째에 Early Stop했다. 그 결과, 최적의 Epoch는 55였다.

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_14 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_15 (Dense)             (None, 16)                144
_________________________________________________________________
dense_16 (Dense)             (None, 1)                 17
=================================================================
Total params: 209
Trainable params: 209
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input 노드와 hidden layer(1) node들과는 48개의 연결된 간선의 수가 있었고, hidden layer(1)의 node 8개에 bias(b*)노드가 추가되어 9개의 node와 hidden layer(2)의 node 16개와는 144 개의 에 bias(b*)노드가 하나 더 추가되어, hidden layer(2)과 output node와는 17개의 간선이 연결되어 있었던 것을 확인했다.



- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 74일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 32.4919, Validation loss가 27.3270인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 20.3011, Validation mape가 24.7220 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다

```
test R squared: 0.565
test MSE: 27.767
tset MAPE: 0.225
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.565, MSE는 27.767, MAPE는 0.225 값이 나온 것을 확인했다.

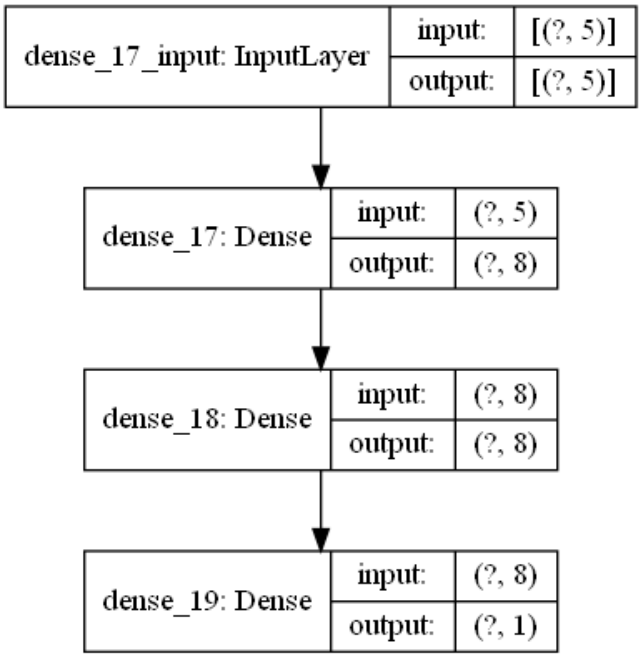
**4) Learning Rate = 0.1 인 경우**

**- Model 4 : Hidden layer = 2, Hidden Node = [8,8], Learning rate = 0.1, epoch = 300, batch_size=1024**

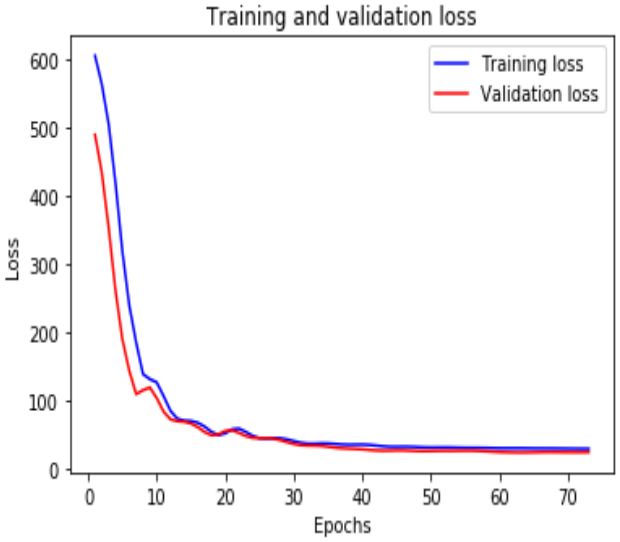
```
Restoring model weights from the end of the best epoch.
323/323 [==============================] - 0s 28us/sample - loss: 29.1232 - mape: 18.4067 - val_loss: 24.1967 - val_mape: 22.3415
Epoch 00073: early stopping
```

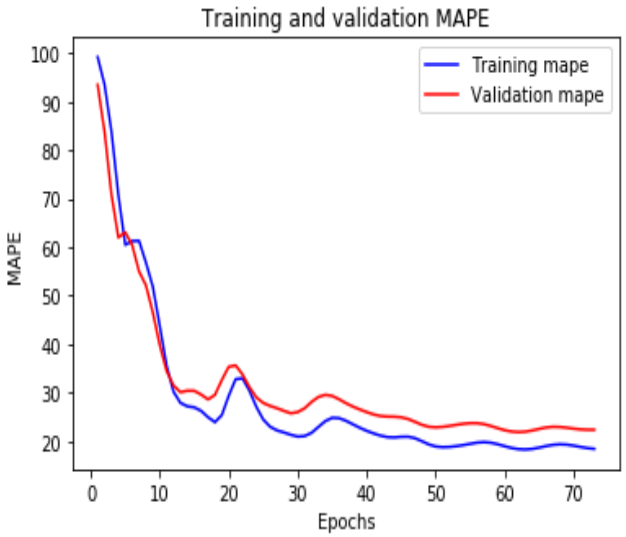
- Hidden layer = 2, activation = 'relu', Hidden Node = [8,8], Learning rate = 0.1, epoch = 300, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 73번째에 Early Stop했다. 그 결과, 최적의 Epoch는 73였다.

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_17 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_18 (Dense)             (None, 8)                 72
_________________________________________________________________
dense_19 (Dense)             (None, 1)                 9
=================================================================
Total params: 129
Trainable params: 129
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input 노드 6개와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었고, hidden layer(1)의 node 8개와 bias(b*)노드가 추가 되어 9개의 node와 hidden layer(2)의 node 8개와는 72개의 연결된 간선의 수가 존재한다. hidden layer(2)의 node 8개에 bias(b**)노드가 하나 더 추가되어, hidden layer(2)과 output node와는 9개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 73일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 39.1232, Validation loss가 24.1967인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 28.4067, Validation mape가 22.3415 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다.

```
test R squared: 0.607
test MSE: 25.080
tset MAPE: 0.205
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.607, MSE는 25.080, MAPE는 0.205 값이 나온 것을 확인했다.

**5) Learning Rate = 0.1 인 경우**

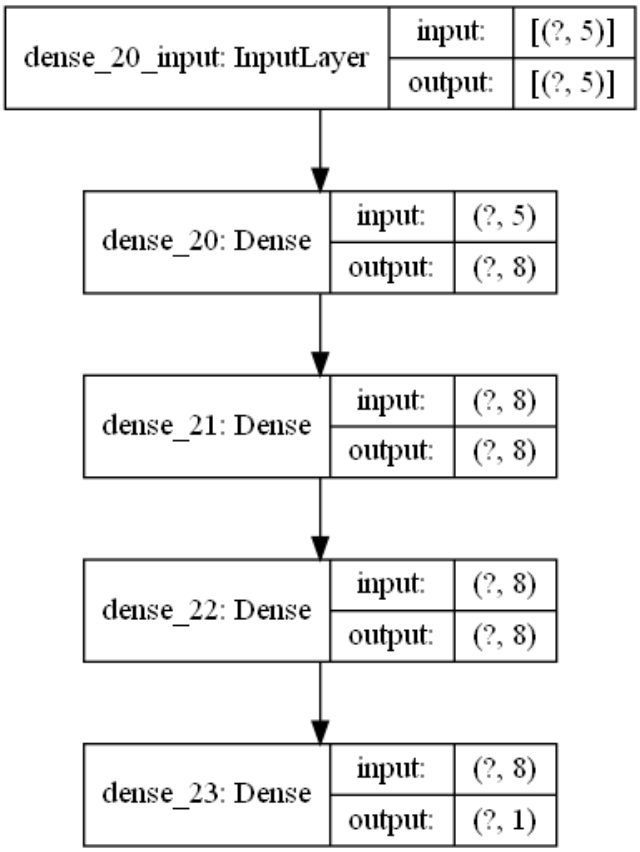
**- Model 5 : Hidden layer = 3, Hidden Node = [8,8,8], Learning rate = 0.1, epoch = 300, batch_size=1024**

```
Restoring model weights from the end of the best epoch.
323/323 [==============================] - 0s 31us/sample - loss: 29.3930 - mape: 18.8181 - val_loss: 23.0694 - val_mape: 22.1841
Epoch 00065: early stopping
```

- Hidden layer = 3, activation = 'relu', Hidden Node = [8,8,8], Learning rate = 0.1, epoch = 300, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을

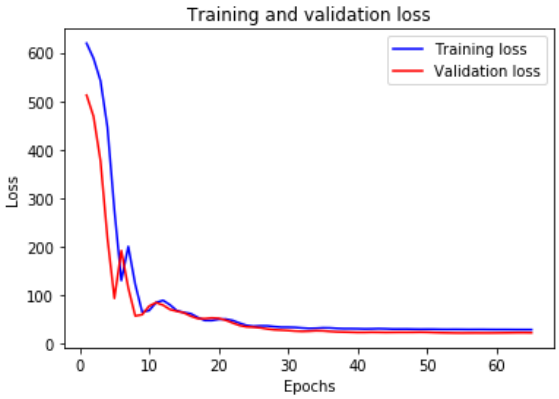
했다. 이때, 이 Model은 65번째에 Early Stop했다. 그 결과, 최적의 Epoch는 65였다.

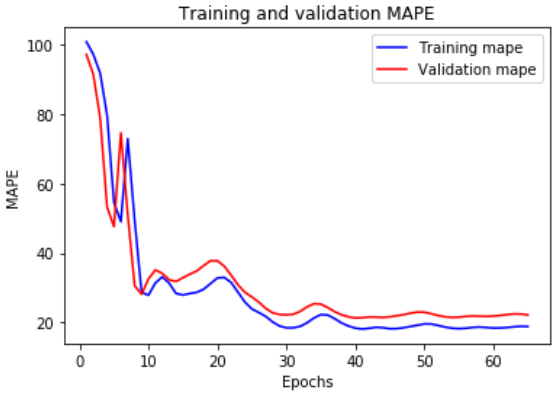
```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_20 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_21 (Dense)             (None, 8)                 72
_________________________________________________________________
dense_22 (Dense)             (None, 8)                 72
_________________________________________________________________
dense_23 (Dense)             (None, 1)                 9
=================================================================
Total params: 201
Trainable params: 201
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input 노드 6개와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었고, hidden layer(1)의 node 8개와 bias(b*)노드가 추가 되어 9개의 node와 hidden layer(2)의 node 8개와는 72개의 연결된 간선의 수가 존재한다. hidden layer(2)의 node 8개와 bias(b**)노드가 추가 되어 9개의 node와 hidden layer(3)의 node 8개와는 72개의 연결된 간선의 수가

존재한다. hidden layer(3)의 node 8개에 bias(b***)노드가 하나 더 추가되어, hidden layer(3)의 node 9개와 output node와는 9개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 65일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 29.3930, Validation loss가 23.0694인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 18.8181, Validation mape가 22.1841 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다.

```
test R squared: 0.593
test MSE: 26.018
tset MAPE: 0.202
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.593, MSE는 26.018, MAPE는 0.202 값이 나온 것을 확인했다.

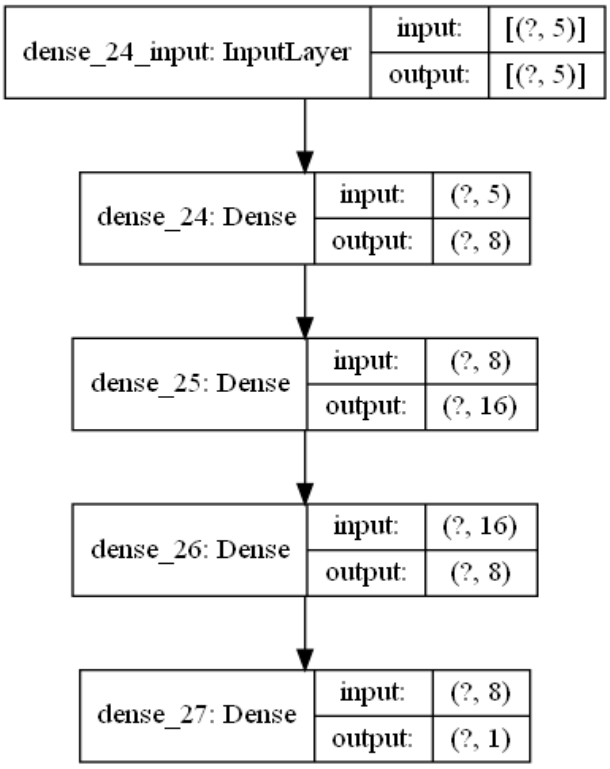
**6) Learning Rate = 0.1 인 경우**

**- Model 6 : Hidden layer = 3, Hidden Node = [8,16,8], Learning rate = 0.1, epoch = 300, batch_size=1024**

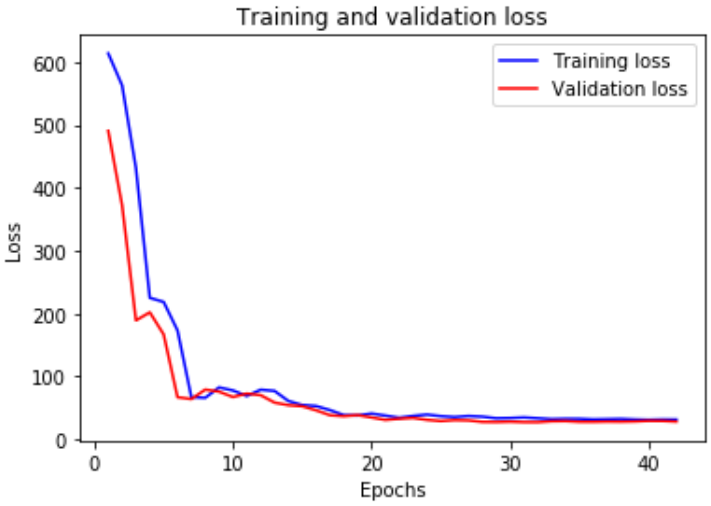
```
Restoring model weights from the end of the best epoch.
323/323 [==============================] - 0s 31us/sample - loss: 31.4835 - mape: 20.6751 - val_loss: 28.2602 - val_mape: 24.0456
Epoch 00042: early stopping
```

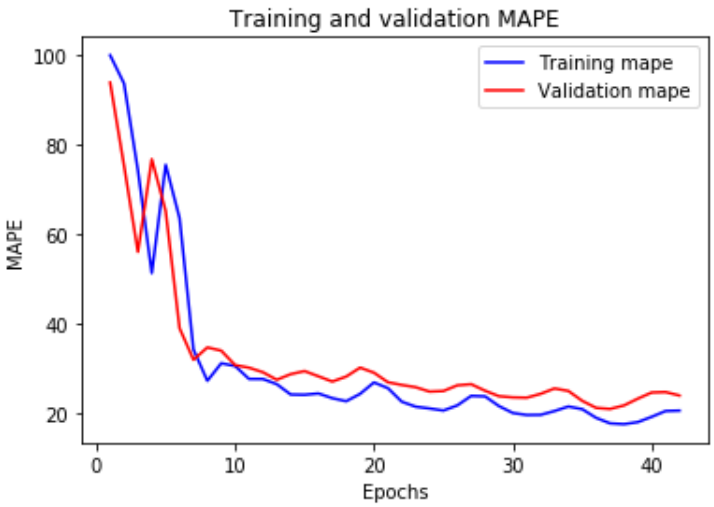
- Hidden layer = 1, activation = 'relu', Hidden Node = [8,16,8], Learning rate = 0.1, epoch = 300, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 42번째에 Early Stop했다. 그 결과, 최적의 Epoch는 42였다.

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_24 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_25 (Dense)             (None, 16)                144
_________________________________________________________________
dense_26 (Dense)             (None, 8)                 136
_________________________________________________________________
dense_27 (Dense)             (None, 1)                 9
=================================================================
Total params: 337
Trainable params: 337
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input노드 6개와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었고, hidden layer(1)의 node 8개와 bias(b*)노드가 추가 되어 9개의 node와 hidden layer(2)의 node 16개와는 144개의 연결된 간선의 수가 존재한다. hidden layer(2)의 node 16개와 bias(b**)노드가 추가 되어 17개의 node와 hidden layer(3)의 node 8개와는 136개의 연결된 간선의 수가 존재한다. hidden layer(3)의 node 8개에 bias(b***)노드가 하나 더 추가되어, hidden layer(3)의 node 9개와 output node와는 9개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 42일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 31.4835, Validation loss가 28.2602인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 20.6751, Validation mape가 24.0456 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다.

```
test R squared: 0.585
test MSE: 26.480
tset MAPE: 0.222
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.585, MSE는 26.480, MAPE는 0.222 값이 나온 것을 확인했다.

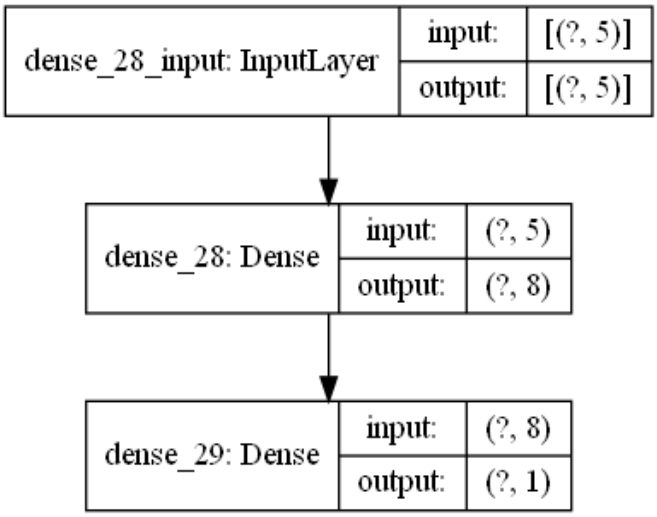
**7) Learning Rate = 1 인 경우**

**- Model 7 : Hidden layer = 1, Hidden Node = 8, Learning rate = 1, epoch = 300, batch_size=1024**

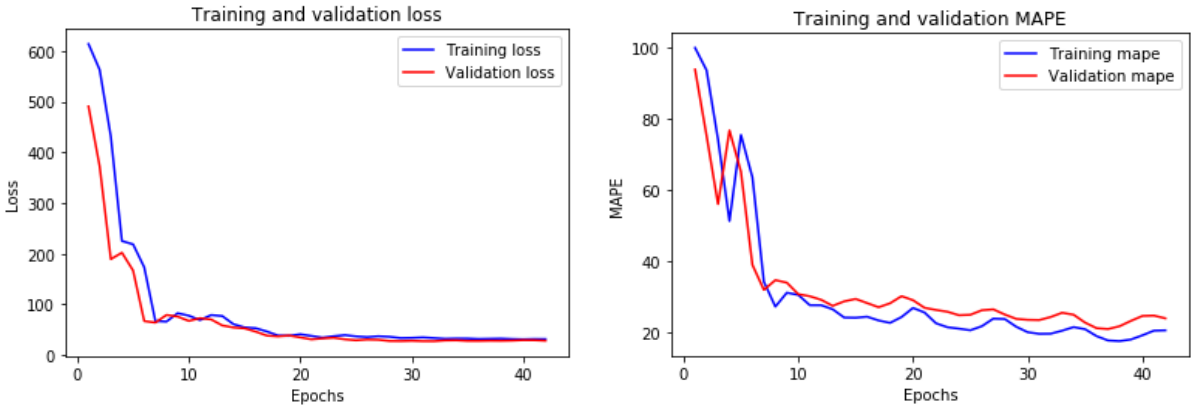
```
Restoring model weights from the end of the best epoch.
323/323 [==============================] - 0s 31us/sample - loss: 35.1710 - mape: 22.4774 - val_loss: 28.7355 - val_mape: 24.3947
Epoch 00029: early stopping
```

- Hidden layer = 1, activation = 'relu', Hidden Node = 8, Learning rate = 1, epoch = 300, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 29번째에 Early Stop했다. 그 결과, Epoch는 29였다.

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_28 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_29 (Dense)             (None, 1)                 9
=================================================================
Total params: 57
Trainable params: 57
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input 노드 6개와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었다. hidden layer(1)의 node 8개에 bias(b*)노드가 하나 더 추가되어, hidden layer(1)과 output node와는 9개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 29

일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 35.1710, Validation loss가 28.7355 인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 22.4744, Validation mape가 24.3947 지점인

```
test R squared: 0.634
test MSE: 23.381
tset MAPE: 0.197
```

부근부터 것을 그래프를 통해 시각적으로 확인했다

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.634, MSE는 23.381, MAPE는 0.197 값이 나온 것을 확인했다.

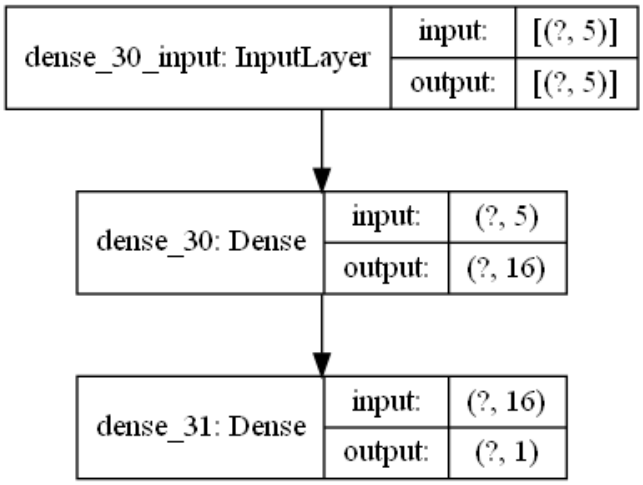
**8) Learning Rate = 1 인 경우**

**- Model 8 : Hidden layer = 1, Hidden Node = 16, Learning rate = 1, epoch = 300, batch_size=1024**

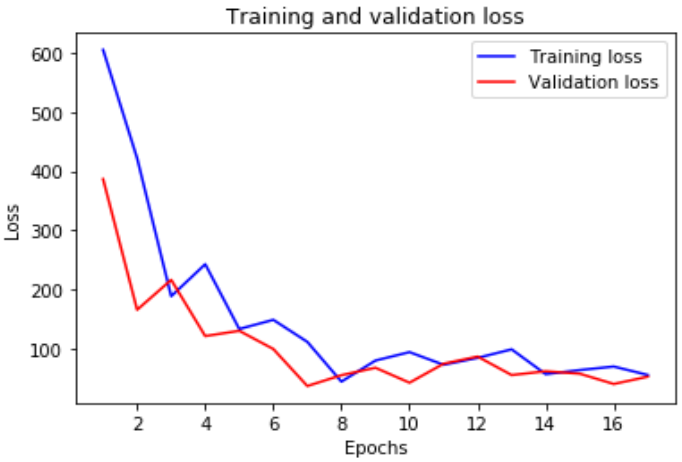
```
Restoring model weights from the end of the best epoch.
323/323 [==============================] - 0s 28us/sample - loss: 55.0672 - mape: 25.0177 - val_loss: 52.1936 - val_mape: 39.7918
Epoch 00017: early stopping
```

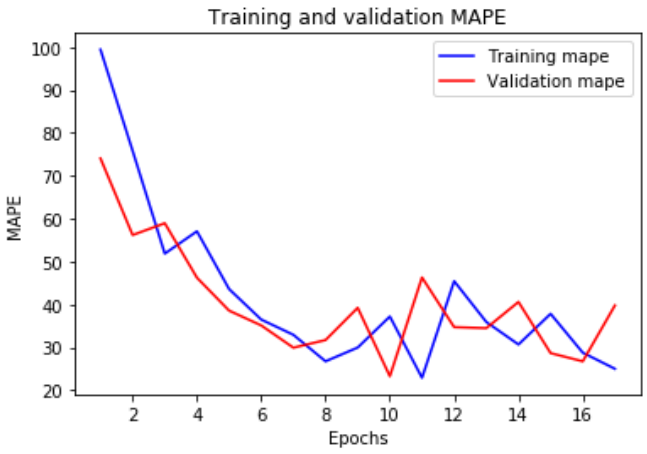
- Hidden layer = 1, activation = 'relu', Hidden Node = 16, Learning rate = 1, epoch = 300, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 17번째에 Early Stop했다. 그 결과, 최적의 Epoch는 17였다.

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_30 (Dense)             (None, 16)                96
_________________________________________________________________
dense_31 (Dense)             (None, 1)                 17
=================================================================
Total params: 113
Trainable params: 113
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input 노드 6개와 hidden layer(1) node 16개와는 96개의 연결된 간선의 수가 있었다. hidden layer(1)의 node 16개에 bias(b*)노드가 하나 더 추가되어, hidden layer(1)과 output node와는 17개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 17일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 55.0672, Validation loss가 52.1936인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 25.0177, Validation mape가 39.7918 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다.

```
test R squared: 0.372
test MSE: 40.122
tset MAPE: 0.276
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.372, MSE는 40.122, MAPE는 0.276 값이 나온 것을 확인했다.

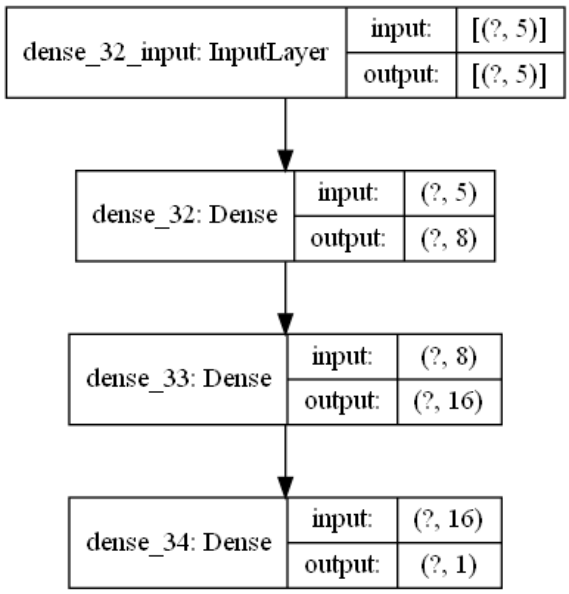
**9) Learning Rate = 1 인 경우**

**- Model 9 : Hidden layer = 2, Hidden Node = [8,16], Learning rate = 1, epoch = 300, batch_size=1024**

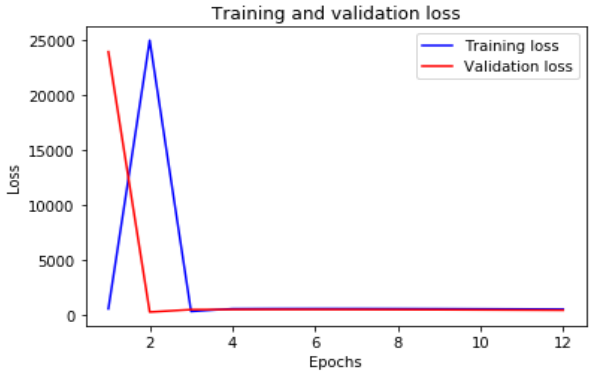
```
Restoring model weights from the end of the best epoch.
323/323 [==============================] - 0s 31us/sample - loss: 573.4240 - mape: 95.2608 - val_loss: 482.2869 - val_mape: 93.0995
Epoch 00012: early stopping
```

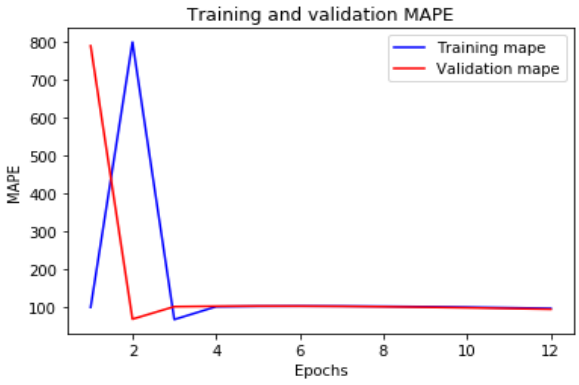
- Hidden layer = 2, activation = 'relu', Hidden Node = [8,16], Learning rate = 1, epoch = 300, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 12번째에 Early Stop했다. 그 결과, 최적의 Epoch는 12였다.

```
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_32 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_33 (Dense)             (None, 16)                144
_________________________________________________________________
dense_34 (Dense)             (None, 1)                 17
=================================================================
Total params: 209
Trainable params: 209
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input 노드 6개와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었고, hidden layer(1)의 node 8개와 bias(b*)노드가 추가 되어 9개의 node와 hidden layer(2)의 node 16개와는 144개의 연결된 간선의 수가 존재한다. hidden layer(2)의 node 16개에 bias(b*)노드가 하나 더 추가 되어, hidden layer(2)의 17node와 output node와는 17개의 간선이 연결되어 있었던 것을 확인했다.



- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 74일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 573.4240, Validation loss가 482.2869인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 93.0995, Validation mape가 95.2608 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다

```
test R squared: -4.486
test MSE: 350.398
tset MAPE: 0.694
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 -4.486, MSE는 350.398, MAPE는 0.694 값이 나온 것을 확인했다.

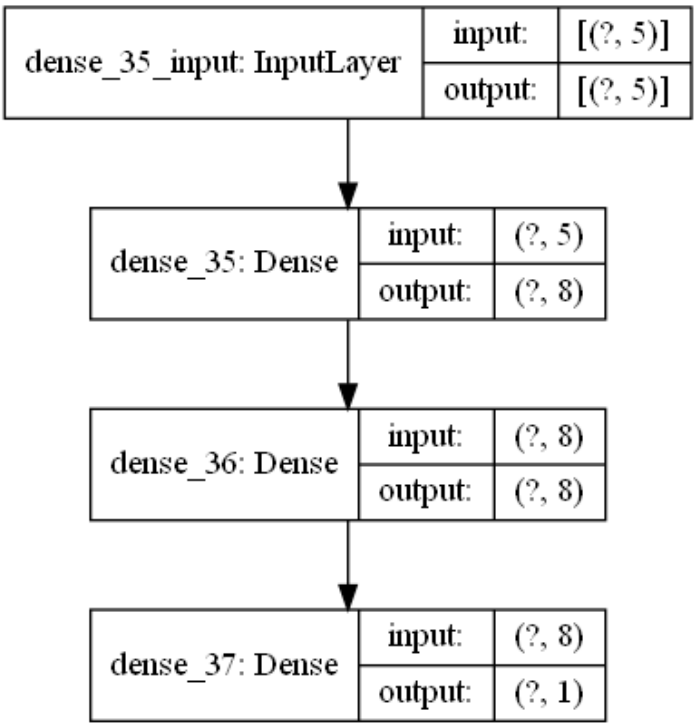
**10) Learning Rate = 1 인 경우**

**- Model 10 : Hidden layer = 2, Hidden Node = [8,8], Learning rate = 1, epoch = 300, batch_size=1024**

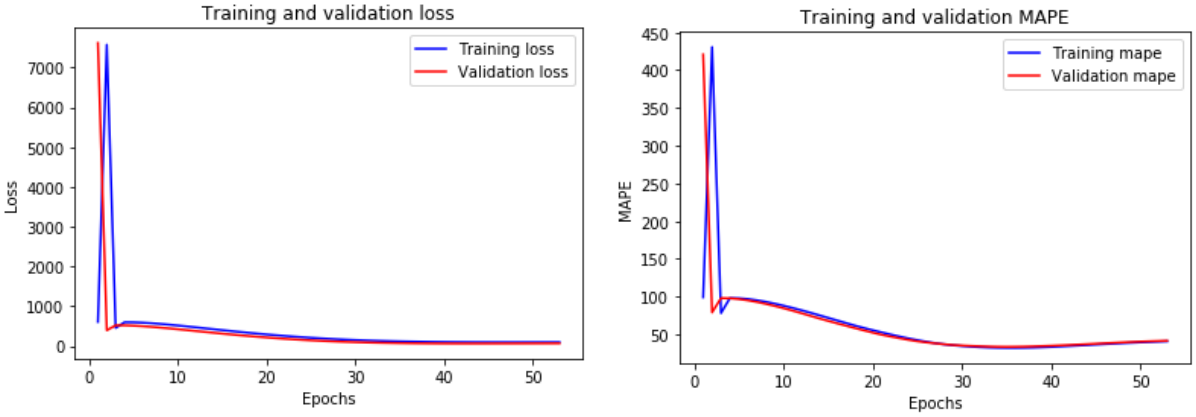
```
Restoring model weights from the end of the best epoch.
323/323 [==============================] - 0s 31us/sample - loss: 96.6426 - mape: 40.9885 - val_loss: 67.8286 - val_mape: 41.7044
Epoch 00053: early stopping
```

- Hidden layer = 2, activation = 'relu', Hidden Node = [8,16], Learning rate = 1, epoch = 300, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 53번째에 Early Stop했다. 그 결과, 최적의 Epoch는 53였다.

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_35 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_36 (Dense)             (None, 8)                 72
_________________________________________________________________
dense_37 (Dense)             (None, 1)                 9
=================================================================
Total params: 129
Trainable params: 129
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input 노드 6개와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었고, hidden layer(1)의 node 8개와 bias(b*)노드가 추가 되어 9개의 node와 hidden layer(2)의 node 8개와는 72개의 연결된 간선의 수가 존재한다. hidden layer(2)의 node 8개에 bias(b*)노드가 하나 더 추가되어, hidden layer(2)과 output node와는 9개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 53일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 96.6426, Validation loss가 67.8286인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 40.9885, Validation mape가 41.7044 지점인

부근부터 것을 그래프를 통해 시각적으로 확인했다.

```
test R squared: -0.008
test MSE: 64.383
tset MAPE: 0.289
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 -0.008, MSE는64.383, MAPE는 0.289 값이 나온 것을 확인했다.

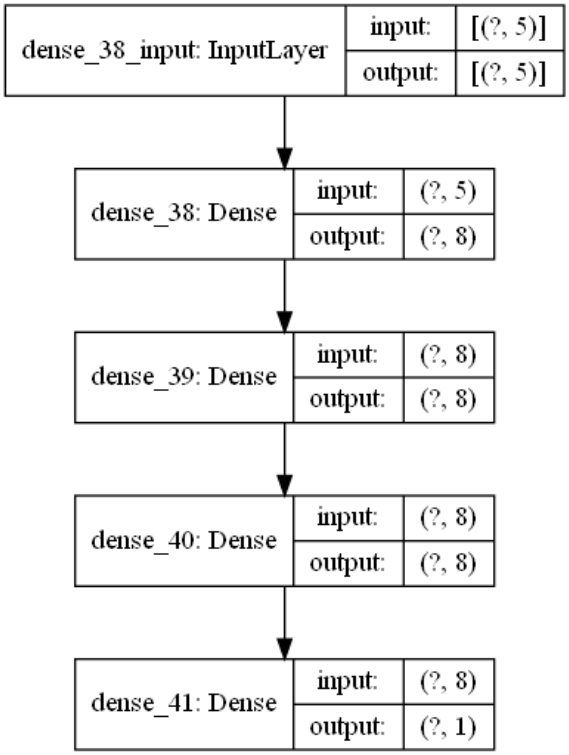
**11) Learning Rate = 1 인 경우**

**- Model 11 : Hidden layer = 3, Hidden Node = [8,8,8], Learning rate = 1, epoch = 300, batch_size=1024**

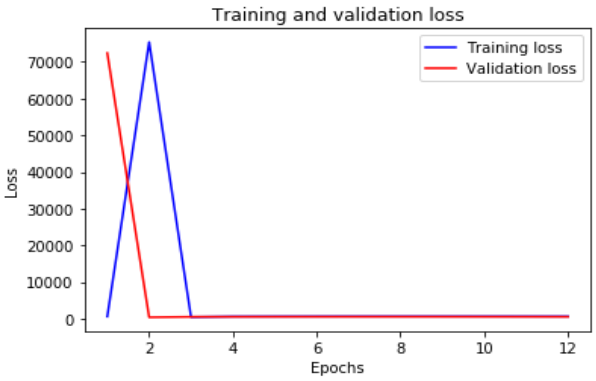
```
Restoring model weights from the end of the best epoch.
323/323 [==============================] - 0s 34us/sample - loss: 642.1314 - mape: 103.2554 - val_loss: 556.1001 - val_mape: 102.4981
Epoch 00012: early stopping
```

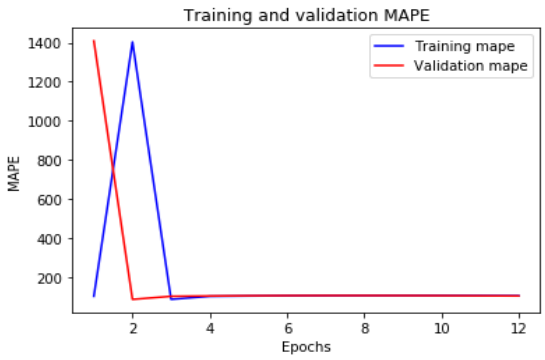
- Hidden layer = 3, activation = 'relu', Hidden Node = [8,8,8], Learning rate = 1, epoch = 300, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 12번째에 Early Stop했다. 그 결과, 최적의 Epoch는 12였다.

```
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_38 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_39 (Dense)             (None, 8)                 72
_________________________________________________________________
dense_40 (Dense)             (None, 8)                 72
_________________________________________________________________
dense_41 (Dense)             (None, 1)                 9
=================================================================
Total params: 201
Trainable params: 201
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input 노드 6개와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었고, hidden layer(1)의 node 8개와 bias(b*)노드가 추가 되어 9개의 node와 hidden layer(2)의 node 8개와는 72개의 연결된 간선의 수가 존재한다. hidden layer(2)의 node 8개와 bias(b**)노드가 추가 되어 9개의 node와 hidden layer(3)의 node 8개와는 72개의 연결된 간선의 수가 존재한다. hidden layer(3)의 node 8개에 bias(b***)노드가 하나 더 추가되어, hidden layer(3)의 node 9개와 output node와는 9개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 12

일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 642.1314, Validation loss가 556.1001인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 103.2554, Validation mape가 102.4981 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다

```
test R squared: -6.089
test MSE: 452.818
tset MAPE: 0.865
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 -6.089, MSE는 452.818, MAPE는 0.865 값이 나온 것을 확인했다.

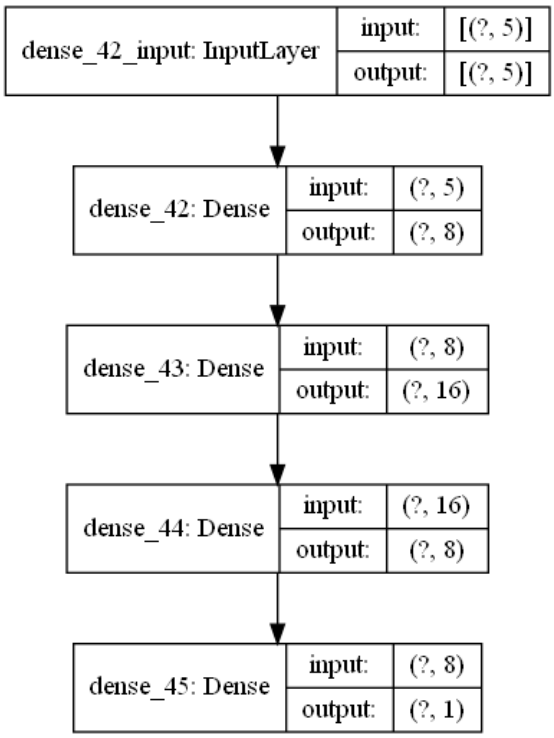
**12) Learning Rate = 1 인 경우**

**- Model 12 : Hidden layer = 3, Hidden Node = [8,16,8], Learning rate = 1, epoch = 300, batch_size=1024**

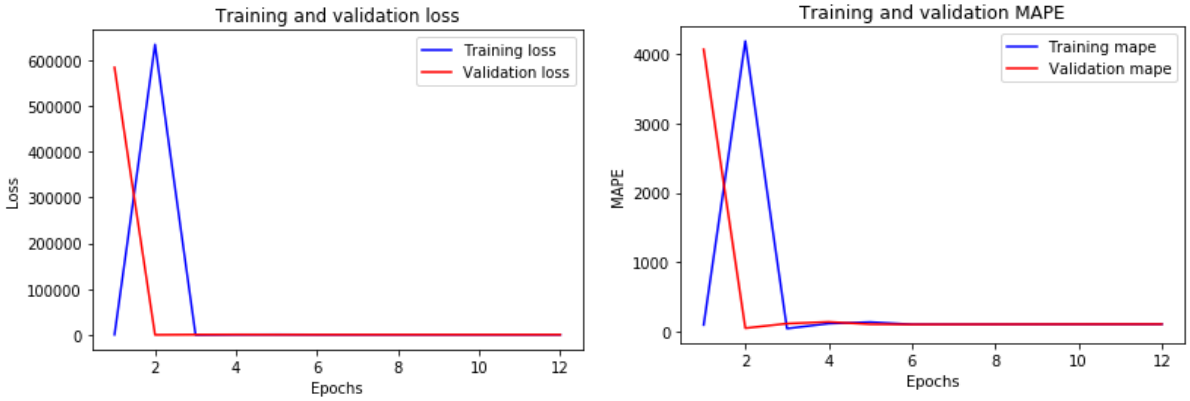
```
Restoring model weights from the end of the best epoch.
323/323 [==============================] - 0s 31us/sample - loss: 710.3408 - mape: 110.7083 - val_loss: 631.0757 - val_mape: 111.3474
Epoch 00012: early stopping
```

- Hidden layer = 3, activation = 'relu', Hidden Node = [8,16,8], Learning rate = 1, epoch = 300, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 12번째에 Early Stop했다. 그 결과, 최적의 Epoch는 12였다.

```
Model: "sequential_16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_42 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_43 (Dense)             (None, 16)                144
_________________________________________________________________
dense_44 (Dense)             (None, 8)                 136
_________________________________________________________________
dense_45 (Dense)             (None, 1)                 9
=================================================================
Total params: 337
Trainable params: 337
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input 노드 6개와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었고, hidden layer(1)의 node 8개와 bias(b*)노드가 추가 되어 9개의 node와 hidden layer(2)의 node 16개와는 144개의 연결된 간선의 수가 존재한다. hidden layer(2)의 node 16개와 bias(b**)노드가 추가 되어 17개의 node와 hidden layer(3)의 node 8개와는 136개의 연결된 간선의 수가 존재한다. hidden layer(3)의 node 8개에 bias(b***)노드가 하나 더 추가되어, hidden layer(3)의 node 9개와 output node와는 9개의 간선이 연결되어 있었던 것을 확인했다.




- 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 12때부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 710.3408, Validation loss가 631.0757인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정

한 MAPE 그래프의 경우 Training mape가 110.7083, Validation mape가 111.3474 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다.

```
test R squared: -2.150
test MSE: 201.238
tset MAPE: 0.523
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 -2.150, MSE는201.238, MAPE는 0.523 값이 나온 것을 확인했다.

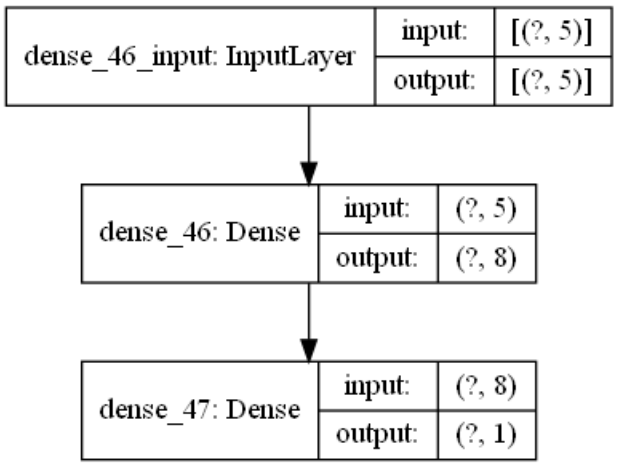
**13) Learning Rate = 0.001 인 경우**

**- Model 13 : Hidden layer = 1, Hidden Node = 8, Learning rate = 0.001, epoch = 300, batch_size=1024**

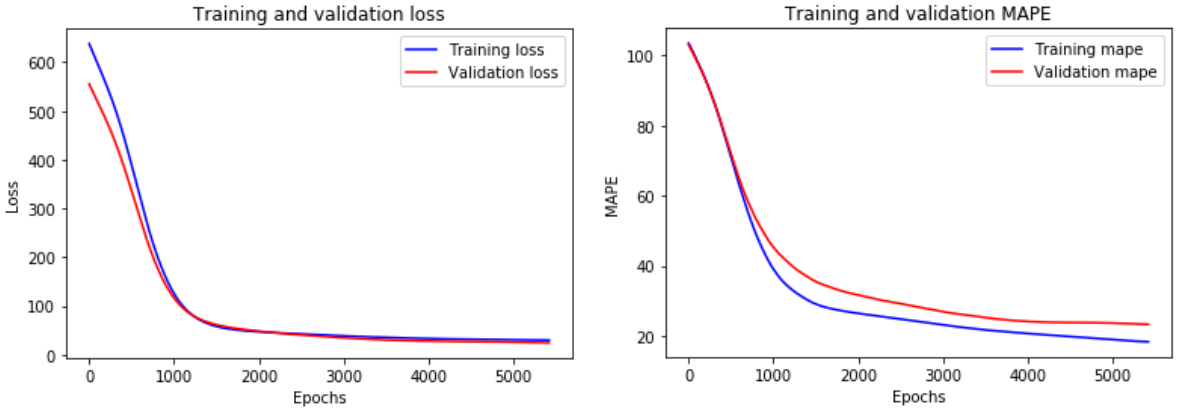
```
Restoring model weights from the end of the best epoch.
323/323 [==============================] - 0s 46us/sample - loss: 29.5977 - mape: 18.3416 - val_loss: 24.3385 - val_mape: 23.3347
Epoch 05420: early stopping
```

- Hidden layer = 1, activation = 'relu', Hidden Node = 8, Learning rate = 0.001, epoch = 6000, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 5420번째에 Early Stop했다. 그 결과, 최적의 Epoch는 5420였다.

```
Model: "sequential_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_46 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_47 (Dense)             (None, 1)                 9
=================================================================
Total params: 57
Trainable params: 57
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input노드와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었고, hidden layer의 node 8개에 bias(b*)노드가 하나 더 추가되어, hidden layer(1) node 9개와 output node와는 9개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 5420일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 29.5977, Validation loss가 24.3385인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 18.3416, Validation mape가 23.3347 지점

인 부근부터 것을 그래프를 통해 시각적으로 확인했다.

```
test R squared: 0.626
test MSE: 23.865
tset MAPE: 0.200
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.626, MSE는 23.865, MAPE는 0.200 값이 나온 것을 확인했다.

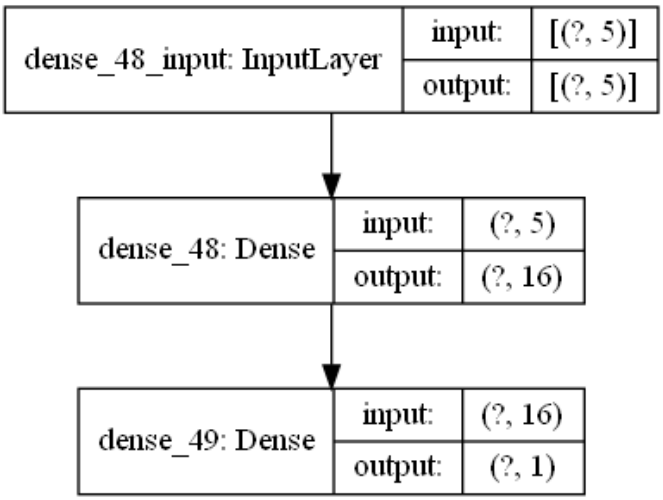
**14) Learning Rate = 0.001 인 경우**

**- Model 14 : Hidden layer = 1, Hidden Node = 16, Learning rate = 0.001, epoch = 6000, batch_size=1024**

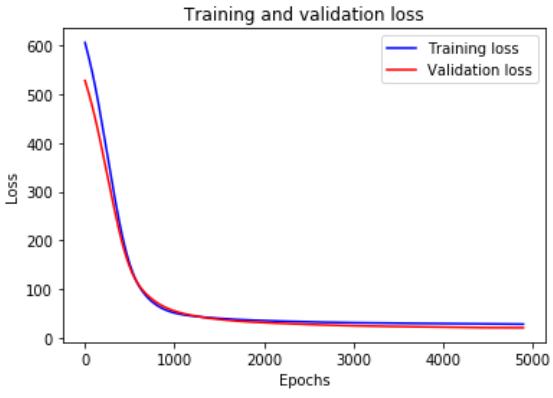
```
Restoring model weights from the end of the best epoch.
323/323 [==============================] - 0s 50us/sample - loss: 28.7366 - mape: 18.3840 - val_loss: 21.3476 - val_mape: 21.8358
Epoch 04896: early stopping
```

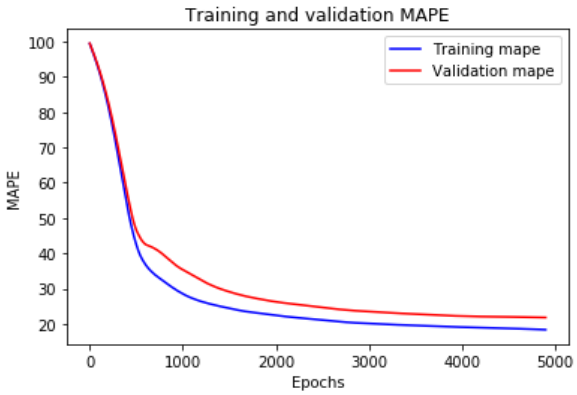
- Hidden layer =1 , activation = 'relu', Hidden Node = 16, Learning rate = 0.001, epoch = 6000, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 4896번째에 Early Stop했다. 그 결과, 최적의 Epoch는 4896였다.

```
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_48 (Dense)             (None, 16)                96
_________________________________________________________________
dense_49 (Dense)             (None, 1)                 17
=================================================================
Total params: 113
Trainable params: 113
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input노드와 hidden layer(1) node 16개와는 96개의 연결된 간선의 수가 있었고, hidden layer의 node 16개에 bias(b*)노드가 하나 더 추가되어, hidden layer(1) node 17개와 output node와는 17개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 4896일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 28.7366, Validation loss가 21.3476인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로

설정한 MAPE 그래프의 경우 Training mape가 18.3840, Validation mape가 21.8358 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다.

```
test R squared: 0.624
test MSE: 23.995
tset MAPE: 0.197
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.624, MSE는 22.995, MAPE는 0.197 값이 나온 것을 확인했다.

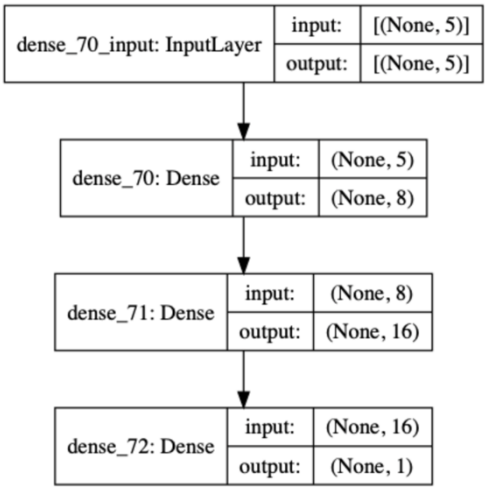
**15) Learning Rate = 0.001 인 경우**

**- Model 15 : Hidden layer = 2, Hidden Node = [8,16], Learning rate = 0.001, epoch = 6000, batch_size=1024**

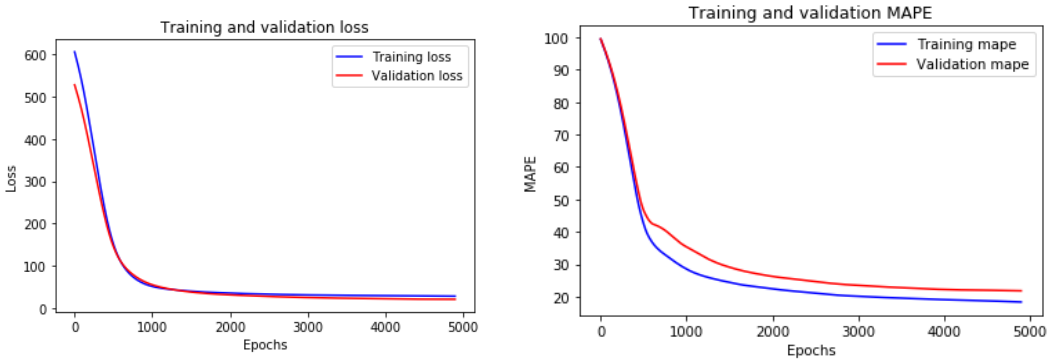
```
Epoch 2644/6000
1/1 [==============================] - 0s 36ms/step - loss: 27.7594 - mape: 17.9013 -
val_loss: 22.9792 - val_mape: 22.2104
Restoring model weights from the end of the best epoch.
Epoch 02644: early stopping
```

- Hidden layer =2 , activation = 'relu', Hidden Node = [8,16], Learning rate = 0.001, epoch = 6000, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 2644번째에 Early Stop했다. 그 결과, 최적의 Epoch는 2644였다.

```
Model: "sequential_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_50 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_51 (Dense)             (None, 16)                144
_________________________________________________________________
dense_52 (Dense)             (None, 1)                 17
=================================================================
Total params: 209
Trainable params: 209
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input노드 6개와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었고, hidden layer(1)의 node 8개와 bias(b*)노드가 추가 되어 9개의 node와 hidden layer(2)의 node 16개와는 144개의 연결된 간선의 수가 존재한다. hidden layer(2)의 node 16개에 bias(b*)노드가 하나 더 추가되어, hidden layer(2)과 output node와는 17개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 2644일 때 부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 27.7594, Validation loss가 22.9792인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 17.9013, Validation mape가 22.2104 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다.

```
test R squared: 0.504
test MSE: 31.697
tset MAPE: 0.209
```

- Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.504, MSE는 31.697, MAPE는 0.209 값이 나온 것을 확인했다.

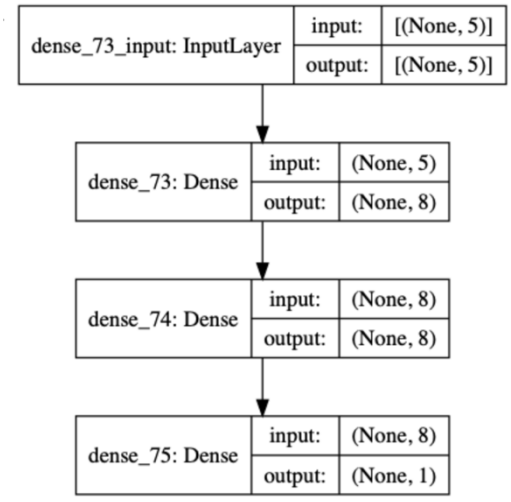
**16) Learning Rate = 0.001 인 경우**

**- Model 16 : Hidden layer = 2, Hidden Node = [8,8], Learning rate = 0.001, epoch = 6000, batch_size=1024**

```
Epoch 2540/6000
1/1 [==============================] - 0s 30ms/step - loss: 29.7651 - mape: 18.8353 -
val_loss: 25.3481 - val_mape: 23.0034
Restoring model weights from the end of the best epoch.
Epoch 02540: early stopping
```

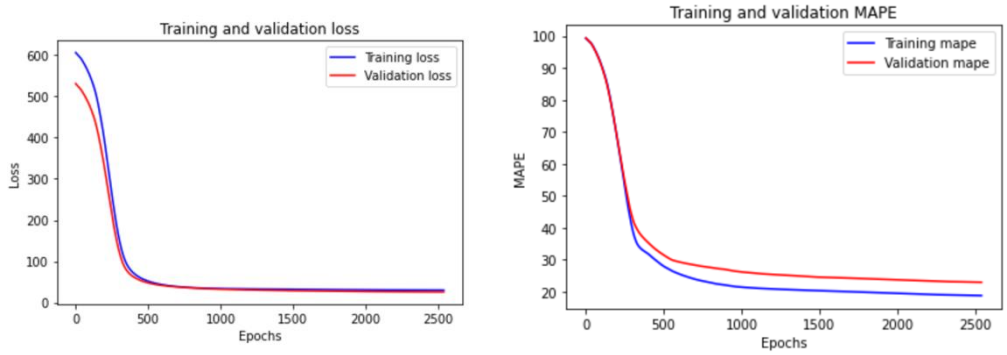
- Hidden layer =2 , activation = 'relu', Hidden Node = [8,8], Learning rate = 0.001, epoch = 6000, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 2540번째에 Early Stop했다. 그 결과, 최적의 Epoch는 2540였다.

```
Model: "sequential_20"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_53 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_54 (Dense)             (None, 8)                 72
_________________________________________________________________
dense_55 (Dense)             (None, 1)                 9
=================================================================
Total params: 129
Trainable params: 129
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input노드 6개와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었고,

hidden layer(1)의 node 8개와 bias(b*)노드가 추가 되어 9개의 node와 hidden layer(2)의 node 8개와는 72개의 연결된 간선의 수가 존재한다. hidden layer(2)의 node 8개에 bias(b*)노드가 하나 더 추가되어, hidden layer(2)과 output node와는 9개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 2540일 때부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 29.7651, Validation loss가 25.3481인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 18.8353, Validation mape가 23.0034 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다.

```
test R squared: 0.557
test MSE: 28.295
tset MAPE: 0.209
```

Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.557, MSE는 28.295, MAPE는 0.209 값이 나온 것을 확인했다.

**17) Learning Rate = 0.001 인 경우**

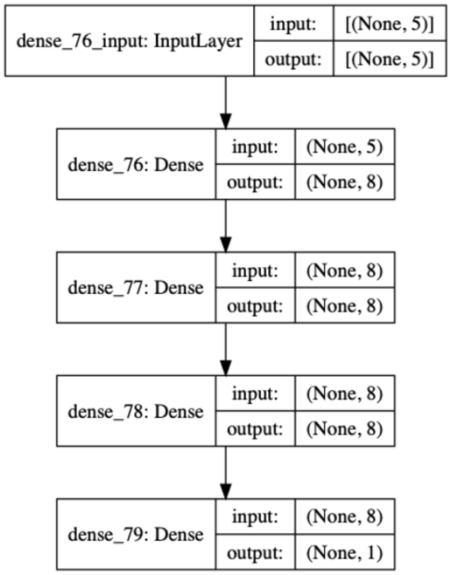
**- Model 17 : Hidden layer = 3, Hidden Node = [8,8,8], Learning rate = 0.001, epoch = 6000, batch_size=1024**

```
Epoch 2036/6000
1/1 [==============================] - 0s 33ms/step - loss: 28.3114 - mape: 17.7596 -
val_loss: 23.2183 - val_mape: 21.6689
Restoring model weights from the end of the best epoch.
Epoch 02036: early stopping
```

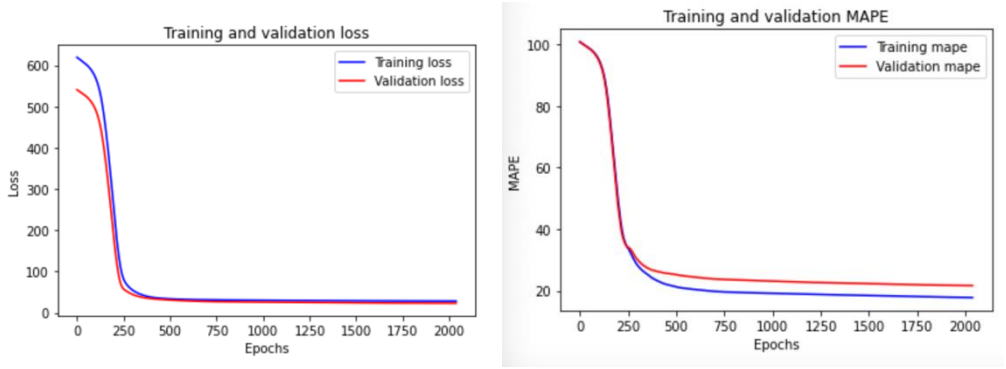
- Hidden layer =3 , activation = 'relu', Hidden Node = [8,8,8], Learning rate = 0.001, epoch = 6000, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행

을 했다. 이때, 이 Model은 2036번째에 Early Stop했다. 그 결과, 최적의 Epoch는 2036였다.

```
Model: "sequential_21"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_56 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_57 (Dense)             (None, 8)                 72
_________________________________________________________________
dense_58 (Dense)             (None, 8)                 72
_________________________________________________________________
dense_59 (Dense)             (None, 1)                 9
=================================================================
Total params: 201
Trainable params: 201
Non-trainable params: 0
_________________________________________________________________
```




- 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input노드 6개와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었고, hidden layer(1)의 node 8개와 bias(b*)노드가 추가 되어 9개의 node와 hidden layer(2)의 node 8개와는 72개의 연결된 간선의 수가 존재한다. hidden layer(2)의 node 8개와 bias(b**)노드가 추가 되어 9개의 node와 hidden layer(3)의 node 8개와는 72개의 연결된 간선의 수가 존재한다. hidden layer(3)의 node 8개에 bias(b***)노드가 하나 더 추가되어, hidden layer(3)의 node 9개와 output node와는 9개의 간선이 연결되어 있었던 것을 확인했다.




- 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 2036일 때부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 28.3114, Validation loss가 23.2183인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 17.7596, Validation mape가 21.6689 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다.

```
test R squared: 0.537
test MSE: 29.562
tset MAPE: 0.202
```

Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.537, MSE는 29.562, MAPE는 0.202 값이 나온 것을 확인했다.

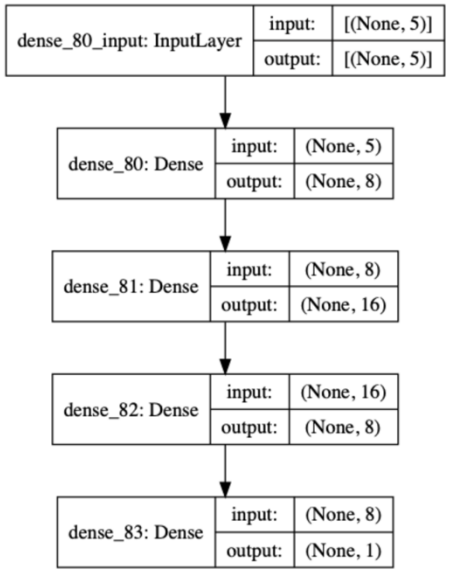
**18) Learning Rate = 0.001 인 경우**

**- Model 18 : Hidden layer = 3, Hidden Node = [8,16,8], Learning rate = 0.001, epoch = 6000, batch_size=1024**

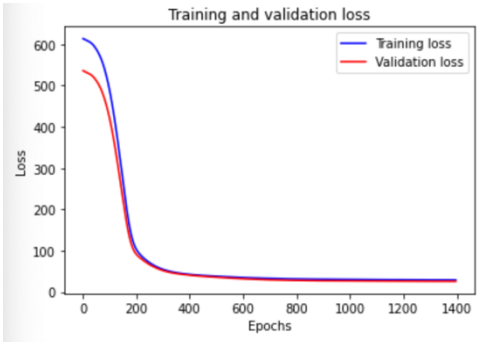
```
Epoch 1396/6000
1/1 [==============================] - 0s 43ms/step - loss: 28.8744 - mape: 18.2830 -
val_loss: 25.6002 - val_mape: 22.6990
Restoring model weights from the end of the best epoch.
Epoch 01396: early stopping
```

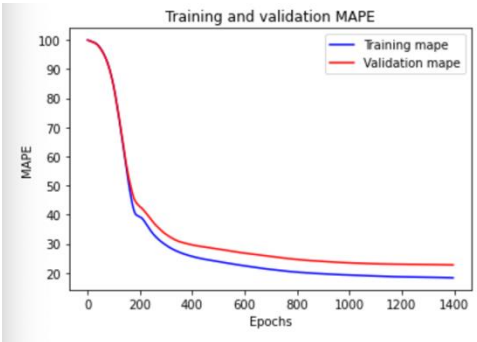
- Hidden layer =3 , activation = 'relu', Hidden Node = [8,16,8], Learning rate = 0.001, epoch = 6000, batch_size=1024, train과 validation data는 8:2, validation_split으로 나눠줘서 진행을 했다. 이때, 이 Model은 1396번째에 Early Stop했다. 그 결과, 최적의 Epoch는 1396였다.

```
Model: "sequential_22"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_60 (Dense)             (None, 8)                 48
_________________________________________________________________
dense_61 (Dense)             (None, 16)                144
_________________________________________________________________
dense_62 (Dense)             (None, 8)                 136
_________________________________________________________________
dense_63 (Dense)             (None, 1)                 9
=================================================================
Total params: 337
Trainable params: 337
Non-trainable params: 0
_________________________________________________________________
```




➢ 모델의 구조는 위 그림 다음과 같았다. 총 input node 5개에 bias(b)노드가 추가되어 input노드 6개와 hidden layer(1) node 8개와는 48개의 연결된 간선의 수가 있었고, hidden layer(1)의 node 8개와 bias(b*)노드가 추가 되어 9개의 node와 hidden layer(2)의 node 16개와는 144개의 연결된 간선의 수가 존재한다. hidden layer(2)의 node 16개와 bias(b**)노드가 추가 되어 17개의 node와 hidden layer(3)의 node 8개와는 136개의 연결된 간선의 수가 존재한다. hidden layer(3)의 node 8개에 bias(b***)노드가 하나 더 추가되어, hidden layer(3)의 node 9개와 output node와는 9개의 간선이 연결되어 있었던 것을 확인했다.



➢ 위 그림은 train데이터와 validaion 데이터를 그래프로 비교 분석한 결과다. 왼쪽 그래프는 MSE를 나타냈고, 오른쪽은 MAPE 지표를 나타났다. 그리고 수치상으로는 Epoch = 2036일 때부터 EarlyStop한 결과, Loss 그래프의 경우 Training loss가 28.8744, Validation loss가 25.6002인 지점 부근부터 수렴하는 것을 시각적으로 확인할 수 있었고, Metrics로 설정한 MAPE 그래프의 경우 Training mape가 18.2830, Validation mape가 22.6990 지점인 부근부터 것을 그래프를 통해 시각적으로 확인했다.

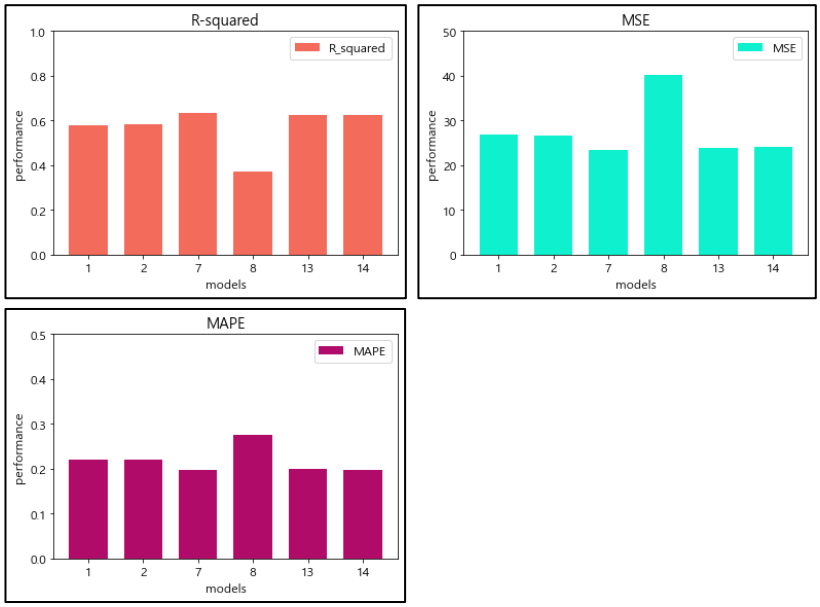
```
test R squared: 0.525
test MSE: 30.328
tset MAPE: 0.211
```

Test데이터를 넣고, Modeling 한 결과 위와 같은 지표가 나왔다. 알 스퀘어는 0.525, MSE는 30.328, MAPE는 0.211 값이 나온 것을 확인했다.

### 3.3. 성능 비교

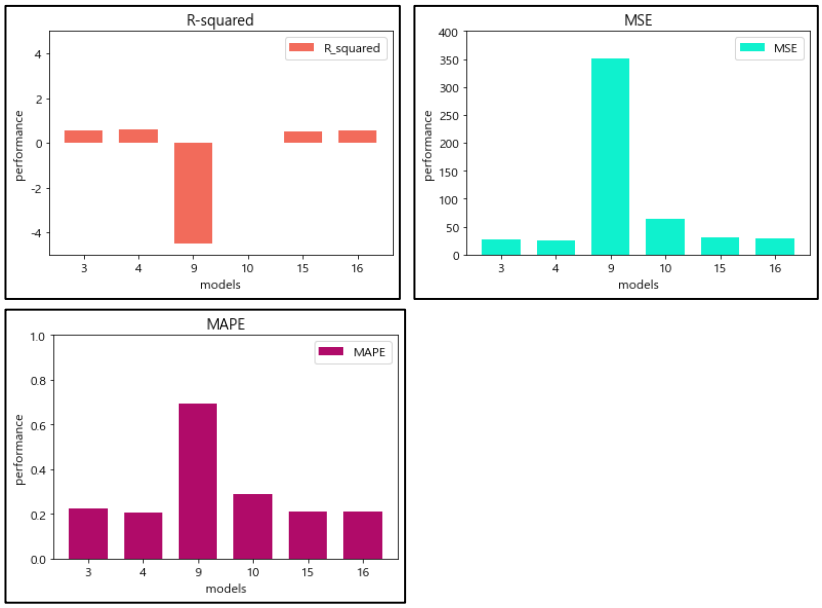
#### ① 같은 hidden layer수 모델간의 비교

##### A. 1-layer모델간 비교



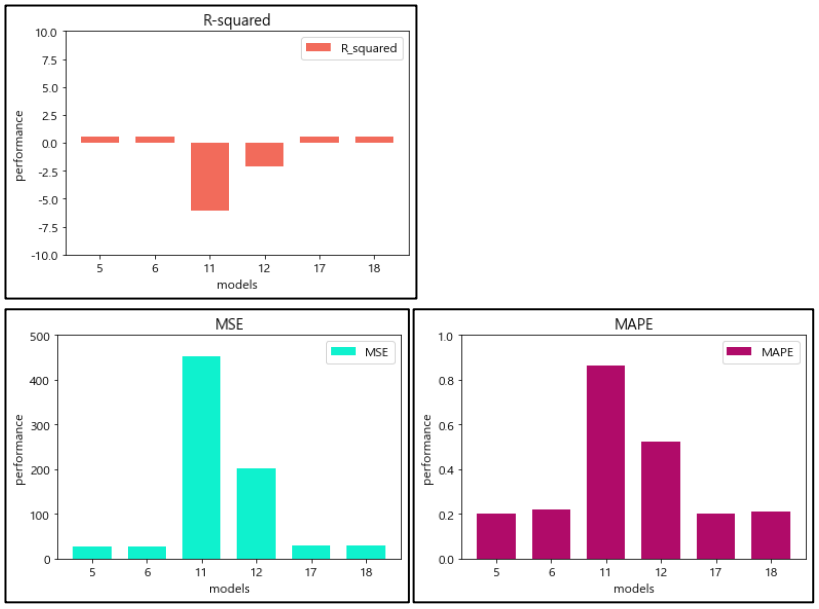

: 1-layer모델들의 R_squared, MSE, MAPE를 시각화하고 비교한 결과, Layer 1개, Node 8개, Learning rate가 1인 7번 신경망 모델이 1-layer모델중에서 준수한 성능을 내는것으로 판단된다.

##### B. 2-layer모델간 비교




: 2-layer모델들의 R_squared, MSE, MAPE를 시각화하고 비교한 결과, Layer 2개, Node [8, 8]개, Learning rate가 0.1인 4번 신경망 모델이 2-layer모델중에서 준수한 성능을 내는것으로 판단된다.

### C. 3-layer모델간 비교

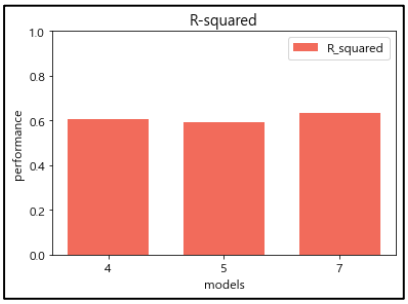


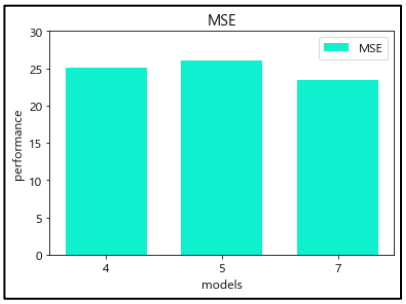

: 3-layer모델들의 R_squared, MSE, MAPE를 시각화하고 비교한 결과, Layer 3개, Node [8, 8, 8]개, Learning rate가 0.1인 5번 신경망 모델이 3-layer모델중에서 준수한 성능을 내는것으로 판단된다.

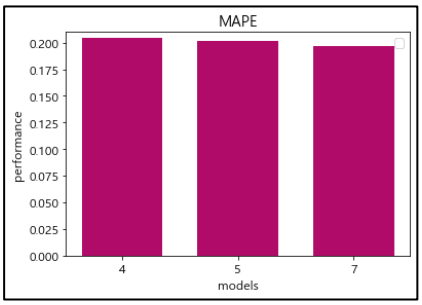
## ② 각 hidden layer 최적 모델간의 비교

앞서 1-layer 모델중에서 7번 모델, 2-layer 모델중에서 4번 모델, 3-

layer 모델중에서 5번 모델이 준수한 성능을 내는것으로 판단했다.




4, 5, 7번 모델간의 R_squared, MSE, MAPE값을 비교하고 시각화한 결과, Layer 1개, Node 8개, Learning rate가 1인 7번 신경망 모델이 가장 준수 준수한 것 판단되어, 18개의 신경망 모델중 7번 신경망 모델이 가장 최적의 성능을 내는것으로 판단된다.

### 3.4. 최종 성능 평가

| # | Layer 개수 | Node | Learning Rate | R-squared | MSE | MAPE |
|---|---|---|---|---|---|---|
| 1 | 1 | 8 | 0.1 | 0.578 | 26.930 | 0.220 |
| 2 | 1 | 16 | 0.1 | 0.583 | 26.630 | 0.221 |
| 3 | 2 | [8,16] | 0.1 | 0.565 | 27.767 | 0.225 |
| **4** | **2** | **[8,8]** | **0.1** | **0.607** | **25.080** | **0.205** |
| **5** | **3** | **[8,8,8]** | **0.1** | **0.593** | **26.018** | **0.202** |
| 6 | 3 | [8,16,8] | 0.1 | 0.585 | 26.480 | 0.222 |
| **7** | **1** | **8** | **1** | **0.634** | **23.381** | **0.197** |
| 8 | 1 | 16 | 1 | 0.372 | 40.122 | 0.276 |
| 9 | 2 | [8,16] | 1 | -4.486 | 350.398 | 0.694 |
| 10 | 2 | [8,8] | 1 | -0.008 | 64.383 | 0.289 |
| 11 | 3 | [8,8,8] | 1 | -6.089 | 452.818 | 0.865 |
| 12 | 3 | [8,16,8] | 1 | -2.150 | 201.238 | 0.523 |
| 13 | 1 | 8 | 0.001 | 0.626 | 23.872 | 0.200 |
| 14 | 1 | 16 | 0.001 | 0.624 | 24.000 | 0.197 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 15 | 2 | [8,16] | 0.001 | 0.504 | 31.697 | 0.209 |
| 16 | 2 | [8,8] | 0.001 | 0.557 | 28.295 | 0.209 |
| 17 | 3 | [8,8,8] | 0.001 | 0.537 | 29.562 | 0.202 |
| 18 | 3 | [8,16,8] | 0.001 | 0.525 | 30.328 | 0.211 |

위 표는 총 18개의 모델에 대한 각각의 Test R-squared, Test-MSE, Test-MAPE를 정리한 내용이다. 위 같은 layer간의 비교 분석 결과 각각 1개 ayer일 때는 7번, 2개 layer일때는 4번, 3개 layer일 때는 5번이 각 layer 개수 분류에서 가장 좋은 성능 모델임이 나왔다 (Bold체로 표시) 위 세개 중에서는 가장 R-squared가 높고 MSE, MAPE가 낮은 7번 모델이 가장 좋은 성능임으로 확인이 되어서 최종적으로 본 데이터에서는 7번 모델, 즉, Hiddnen Layer 1개, Hidden Node 8개, Learning rate가 가장 최적의 성능을 내는것으로 판단된다.

## 4. 신경망모델에서 각 노드의 변수가중치(weight)에 대한 해석

각 노드의 변수가중치 설명은 총 18개 모델 중 2개의 모델의 설명을 진행하였다. 2개의 모델은 위 성능비교를 통해 얻어진 상위 2개의 신경망 모델을 대상으로 하였고, 선정 이후에 해석을 구체적으로 진행하였다.

### 4.1 상위 2개 신경망 모델에 대한 노드 변수가중치(weight)의 의미해석

상위 2개의 신경망 모델은 MAPE(Mean Absolute Percentage Error)를 통해 선정하였다. MAPE란 MAE(Mean Absolute Error)를 퍼센트로 변환한 것으로, MAE는 예측값과 실제값의 차이를 절댓값을 취해 합산하는 것이다. 그 식은 다음과 같다.

$$MAE = \frac{\sum |y - \hat{y}|}{n}$$

이를 퍼센트로 변환하는 식은 다음과 같다.

$$MAPE = \frac{\sum |\frac{y - \hat{y}}{y}|}{n} * 100\%$$

해당 지표의 값이 작을수록 그 신경망 모델이 우수하다는 것을 의미한다. 따라서 상위 2개의 신경망으로 7번 모델과 14번 모델을 선정했다.

| # | Layer 개수 | Node | Learning Rate | R-squared | MSE | MAPE |
|---|---|---|---|---|---|---|
| 7 | 1 | 8 | 1 | 0.634 | 23.381 | 0.197 |
| 14 | 1 | 16 | 0.001 | 0.624 | 24.000 | 0.197 |

두 모델 모두 공통적으로 layer의 개수는 1개였으며 MAPE는 0.197로 동일했다.

#### 4.1.1 7번 모델

우선 7번 모델은 1개의 Hidden layer만 가졌으며 layer 안의 node수는 8개였고 learning rate는 1로 설정하였다. 해당 모델의 Input layer에서 hidden layer로 가는 weight는 다음과 같다.

```
In [30]: pd.DataFrame(model.get_weights()[0])
```

Out[30]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | -2.076026 | -1.257015 | -2.162390 | 0.150514 | -0.983251 | 4.988212 | 0.495522 | -6.974717 |
| 1 | 0.818258 | 2.034670 | 3.164726 | 3.963000 | 0.814939 | 0.863415 | 0.562755 | -2.369919 |
| 2 | 0.048568 | -1.759760 | -2.756717 | 0.819765 | -0.481640 | 1.350500 | 1.186973 | 1.649738 |
| 3 | -3.151246 | -3.861568 | -2.760843 | -2.327465 | -2.136896 | 5.571667 | -0.474956 | -3.843774 |
| 4 | 0.645698 | -0.927398 | 2.925219 | -2.631753 | 0.570735 | -2.487990 | -2.985180 | -0.414323 |

각 input값의 의미는 다음과 같다.

| Input | 의미 |
|---|---|
| 0 | INDUS(타운당 비소매 사업 면적 비율) |
| 1 | RM(주택 당 평균 방 수) |
| 2 | AGE(1940년 이전에 건축된 주인이 거주하는 주택 비율) |
| 3 | TAX($10,000당 최대 재산세율) |
| 4 | DIS(보스턴의 5개의 고용센터까지의 가중치 거리) |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **INDUS** | -2.07603 | -1.25702 | -2.16239 | 0.150514 | -0.98325 | 4.988212 | 0.495522 | -6.97472 |
| **RM** | 0.818258 | 2.03467 | 3.164726 | 3.963 | 0.814939 | 0.863415 | 0.562755 | -2.36992 |
| **AGE** | 0.048568 | -1.75976 | -2.75672 | 0.819765 | -0.48164 | 1.3505 | 1.186973 | 1.649738 |
| **TAX** | -3.15125 | -3.86157 | -2.76084 | -2.32746 | -2.1369 | 5.571667 | -0.47496 | -3.84377 |
| **DIS** | 0.645698 | -0.9274 | 2.925219 | -2.63175 | 0.570735 | -2.48799 | -2.98518 | -0.41432 |
| **BIAS** | 0.209647 | 5.810978 | -7.02047 | 2.905868 | -7.24855 | -5.82132 | -5.84618 | -6.28524 |

각 노드의 가중치가 의미하는 바를 쉽게 파악하기 위해 다음과 같이 표를 그리고 색조로 나타냈다.

0번 노드의 경우 INDUS와 TAX의 가중치가 두드러지게 큰 음의 숫자를 가진 것을 확인할 수 있었다. 따라서 타운당 비소매 사업 면적 비율과 최대 재산세율이 주요한 Input 변수이며 그 값이 클수록 0번 노드의 값은 작아진다. 편차(BIAS)는 0.209647로 7개의 노드 중 가장 작다. 그러므로 해당 노드의 성능이 가장 뛰어나다고 결론지을 수 있었다.

1번 노드의 경우 INDUS, AGE와 TAX의 가중치는 큰 음의 숫자를, RM의 가중치는 큰 양의 숫자를 가진 것을 확인할 수 있었다. 상대적으로 DIS의 가중치는 값이 작았으며 따라서 4가지 변수, 타운당 비소매 사업 면적 비율, 주택 당 평균 방 수, 1940년 이전에 건축된 주인이 거주하는 주택 비율, 최대 재산세율이 주요한 Input 변수이다. 타운당 비소매 사업 면적 비율, 1940년 이전에 건축된 주인이 거주하는 주택 비율, 최대 재산세율이 클수록 1번 노드의 값은 작아지고 주택 당 평균 방수가 커질수록 노드의 값이 커진다. 편차(BIAS)는 5.810978로 0번 노드에 비해 현격히 성능이 떨어진다.

2번 노드의 경우 INDUS, AGE와 TAX의 가중치는 큰 음의 숫자를, RM과 DIS의 가중치는 큰 양의 숫자를 가진 것을 확인할 수 있었다. 따라서 5가지 Input 변수 모두 주요하다. 타운당 비소매 사업 면적 비율, 1940년 이전에 건축된 주인이 거주하는 주택 비율, 최대 재산세율이 클수록 2번 노드의 값은 작아지고 주택 당 평균 방수와 보스턴의 5개의 고용센터까지의 가중치 거리가 커질수록 노드의 값이 커진다. 편차(BIAS)는 -7.02047로 2번째로 절댓값이 크다. 해당 노드의 성능은 1번보다도 떨어진다고 판단할 수 있다.

3번 노드의 경우 INDUS와 AGE의 가중치는 상대적으로 작게 나왔고, RM의 가중치는 큰 양의 숫자, TAX와 DIS의 가중치는 큰 음의 숫자를 가진 것을 확인할 수 있었다. 따라서 주택 당 평균 방 수, 최대 재산세율, 보스턴의 5개의 고용센터까지의 가중치 거리가 3번 노드에서 중요한 Input 변수라 판단할 수 있었다. 주택 당 평균 방수가 클수록 3번 노드의 값은 커지고 최대 재산세율과 고용센터까지의 가중치 거리가 클수록 3번 노드의 값은 작아진다. 편차(BIAS)는 2.905868로 7개의 노드 중 2번째로 절댓값이 작다. 그러므로 해당 노드의 성능은 다른 노드들에 비해서 양호하다 판단할 수 있다.

4번 노드의 경우 INDUS, RM, AGE, DIS의 가중치는 상대적으로 작게 나왔고, TAX의 가중치는 큰 음의 숫자를 가진 것을 확인할 수 있었다. 따라서 최대 재산세율이 4번 노드에서 중요한 Input 변수라 판단할 수 있었다. 해당 값이 클수록 4번 노드의 값은 작아진다. 편차(BIAS)는 -7.24855로 7개의 노드 중 절댓값이 가장 크다. 그러므로 해당 노드의 성능은 다른 노드들에 비해 낮다고 판단할 수 있다.

5번 노드의 경우 RM와 AGE의 가중치는 상대적으로 작게 나왔고, INDUS와 TAX의 가중치는 큰 양의 숫자, DIS의 가중치는 큰 음의 숫자를 가진 것을 확인할 수 있었다. 따라서 타운당 비소매 사업 면적 비율, 최대 재산세율, 보스턴의 5개의 고용센터까지의 가중치 거리가 5번 노드에서 중요한 Input 변수라 판단할 수 있었다. 타운당 비소매 사업 면적 비율, 최대 재산세율이 클수록 5번 노드의 값이 커지고 고용센터까지의 가중치 거리가 클수록 5번 노드의 값은 작아진다. 편차(BIAS)는 -5.82132로 0번과 3번 노드에 비해 상대적으로 크다. 그러므로 해당 노드의 성능 역시 상대적으로 낮다고 판단할 수 있다.

6번 노드의 경우 DIS를 제외한 나머지 값들의 가중치는 작게 나왔다. 따라서 해당 노드에서 가장 중요한 변수는 보스턴 5개의 고용센터까지의 거리라고 판단할 수 있었다. 해당

input값이 클수록 6번 노드의 값은 작아진다. 편차(BIAS)는 -5.84618로 7개의 노드 중 절댓값이 큰 편에 속한다. 그러므로 해당 노드의 성능 역시 상대적으로 낮다고 판단할 수 있다.

7번 노드의 경우 DIS의 가중치는 상대적으로 작게 나왔고, AGE의 가중치는 큰 양의 숫자, INDUS, TAX와 RM의 가중치는 큰 음의 숫자를 가진 것을 확인할 수 있었다. 특히 INDUS의 값이 굉장히 큰 음의 숫자를 가진 것을 확인했다. 따라서 타운당 비소매 사업 면적 비율, 주택 당 평균 방 수, 최대 재산세율이 클수록 7번 노드는 작아지고 1940년 이전에 건축된 주인이 거주하는 주택 비율이 클수록 7번 노드의 크기는 커진다. 편차(BIAS)는 -6.28524로 7개의 노드 중 큰 편에 속한다. 그러므로 해당 노드의 성능 역시 상대적으로 낮다고 판단할 수 있다.

#### 4.1.2 14번 모델

14번 모델 역시 모델은 1개의 Hidden layer만 가졌으며 layer 안의 node수는 16개였고 learning rate는 0.001로 설정하였다. 해당 모델의 Input layer에서 hidden layer로 가는 weight와 bias는 다음과 같다.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| INDUS | -0.42358 | -0.53997 | 0.428866 | -0.25057 | -0.98625 | 0.215058 | 0.203656 | 0.659141 |
| RM | 1.383114 | 0.863151 | -0.75145 | 1.175466 | -1.77089 | 0.562725 | -0.04659 | -0.28546 |
| AGE | -0.3486 | 0.026947 | 0.442627 | -0.59578 | 0.213827 | -0.1395 | -0.4053 | -0.0753 |
| TAX | -0.88131 | -0.6089 | 0.028437 | -1.01908 | -0.31579 | -0.14126 | -0.12022 | -0.1198 |
| DIS | -0.48289 | 0.209552 | -0.2669 | 0.113472 | -0.57007 | -0.11503 | 0.337996 | 0.027215 |
| BIAS | 1.27586 | 0.995314 | 1.815301 | 1.199194 | -0.11235 | -0.24274 | -0.47183 | -0.54175 |

| | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|
| INDUS | -0.48331 | -0.3089 | 0.911262 | -0.12912 | 0.047629 | -0.75995 | -0.2982 | 0.9782 |
| RM | 0.622248 | 0.108317 | 0.136686 | 1.102657 | 0.551896 | -0.35329 | 0.902467 | 1.593938 |
| AGE | 0.037685 | -0.15514 | -0.28493 | 0.362167 | 0.242245 | -1.02354 | -0.00094 | -1.56453 |
| TAX | -1.80591 | -0.29159 | 0.688788 | -0.2231 | -1.10813 | -0.09684 | -0.52197 | 1.428342 |
| DIS | -2.98998 | 0.447635 | -0.3697 | -1.09229 | 0.171017 | 0.083502 | 0.573677 | -0.4962 |
| BIAS | 0.654409 | 1.220138 | 1.310248 | 1.13728 | 1.149235 | 1.073221 | 1.045737 | 2.017766 |

편의상 각 변수를 한글이 아닌 영문으로 가중치 해석을 진행하였다.

0번 노드의 경우 RM과 TAX가 주요한 Input 변수임을 확인할 수 있었다. RM의 값이 크면 0번 노드의 값이 커지고 TAX의 값이 커지면 0번 노드의 값이 작아진다. 편차는 7번 모델에 비해 상대적으로 굉장히 작게 나왔다. 이는 모든 노드에 해당된다. 따라서 노드들의 성능 측면에선 해당 모델이 7번 모델보다 우수하다 판단할 수 있다.

1번 노드의 경우 RM과 INDUS가 주요한 Input 변수임을 확인할 수 있었다. RM의 값이

크면 노드의 값이 커지고 INDUS의 값이 커지면 노드의 값이 작아진다.

2번 노드의 경우 RM이 주요한 Input 변수임을 확인할 수 있었다. RM의 값이 크면 노드의 값이 커진다. 해당 노드의 편차 값은 16개의 변수 중 가장 크다. 그러므로 해당 노드의 성능이 16가지의 노드 중 가장 떨어진다.

3번 노드의 경우 RM과 TAX가 주요한 Input 변수임을 확인할 수 있었다. RM의 값이 크면 노드의 값이 커지고 INDUS의 값이 커지면 노드의 값이 작아진다.

4번 노드의 경우 RM과 INDUS가 주요한 Input 변수임을 확인할 수 있었다. RM의 값과 INDUS의 값이 커지면 노드의 값이 작아진다. 해당 노드의 편차는 16개의 변수 중 가장 작았다. 따라서 4번 노드의 성능이 가장 뛰어나다 판단할 수 있다.

5번 노드의 경우 RM이 주요한 Input 변수임을 확인할 수 있었다. RM의 값이 크면 노드의 값이 커진다.

6번 노드의 경우 뚜렷하게 큰 가중치를 가지는 변수가 발견되지 않았다.

7번 노드의 경우 INDUS가 주요한 Input 변수임을 확인할 수 있었다. INDUS의 값이 크면 노드의 값이 커진다.

8번 노드의 경우 RM과 TAX, DIS가 주요한 Input 변수임을 확인할 수 있었다. RM의 값이 크면 노드의 값이 커지고 TAX, DIS의 값이 커지면 노드의 값이 작아진다.

9 번 노드의 역시 뚜렷하게 큰 가중치를 가지는 변수가 발견되지 않았다.

10 번 노드의 경우 TAX 와 INDUS 가 주요한 Input 변수임을 확인할 수 있었다. 두 Input 의 값이 크면 노드의 값이 커진다.

11 번 노드의 경우 RM 과 DIS 가 주요한 Input 변수임을 확인할 수 있었다. RM 의 값이 크면 노드의 값이 커지고 DIS 의 값이 커지면 노드의 값이 작아진다.

12 번 노드의 경우 RM 와 TAX 가 주요한 Input 변수임을 확인할 수 있었다. RM 의 값이 크면 노드의 값이 커지고 TAX 의 값이 커지면 노드의 값이 작아진다.

13 번 노드의 경우 INDUS 와 AGE 가 주요한 Input 변수임을 확인할 수 있었다. 두 변수의 값이 크면 노드의 값이 작아진다.

14 번 노드의 경우 RM 과 DIS 가 주요한 Input 변수임을 확인할 수 있었다. RM 과 DIS 의 값이 크면 노드의 값이 커진다.

15 번 노드의 경우 INDUS, RM, AGE, TAX 가 주요한 Input 변수임을 확인할 수 있었다. INDUS, RM, TAX 의 값이 크면 노드의 값이 커지고 AGE 의 값이 커지면 노드의 값이 작아진다.

## 5. Reference

본 과제의 데이터는 sklearn 의 boston housing data 를 import 하여 진행하였다.

```
import pandas as pd
from sklearn.datasets import load_boston

boston = load_boston()
dir(boston)
dfX = pd.DataFrame(boston.data, columns = boston.feature_names)
dfy = pd.DataFrame(boston.target, columns = ['MEDV'])

df = pd.concat([dfX,dfy], axis=1)
df
```