1. DNN 문제

### Keras DNN으로 집값 예측하기

1. 패키지 수입 및 파라미터 설정

# 수입

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from time import time

from keras.models import Sequential

from keras.layers import Dense

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

# 파라미터

MY\_HIDDEN = 1000

MY\_EPOCH = 500

pd.set\_option('display.max\_columns', None)

2. 데이터 처리

# 데이터 읽기

raw = pd.read\_csv('/content/drive/MyDrive/data/housing.csv')

# 데이터 샘플 10개 출력

print(raw.head(10))

# 데이터 통계 출력

print(raw.describe())

plt.figure(figsize=(15, 8))

sns.set(font\_scale=1.5)

sns.boxplot(data=raw, palette='dark')

# z-점수 정규화

scaler = StandardScaler()

z\_data = scaler.fit\_transform(raw)

# 데이터 형식 출력

print(type(raw))

print(type(z\_data))

# numpy를 pandas로 전환

z\_data = pd.DataFrame(z\_data)

print(type(z\_data))

# z-점수 정규화 된 데이터 샘플 출력

print(z\_data.head(10))

print(z\_data.describe())

plt.figure(figsize=(15, 8))

sns.set(font\_scale=1.5)

sns.boxplot(data=z\_data, palette='dark')

# 데이터 모양 출력

print('분리 전:', z\_data.shape)

# 데이터를 입력과 출력으로 분리

x\_data = z\_data.drop(6, axis=1)

y\_data = z\_data[6]

#데이터 모형 출력

print('분리 후:')

print(x\_data.shape)

print(y\_data.shape)

# 데이터 4분할

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(x\_data,y\_data,train\_size=0.7)

# 최종 데이터 모양 출력

print('학습용 입력 데이터 모양:', X\_train.shape)

print('학습용 출력 데이터 모양:', Y\_train.shape)

print('평가용 입력 데이터 모양:', X\_test.shape)

print('평가용 출력 데이터 모양:', Y\_test.shape)

3. 인공 신경망 구현

model = Sequential()

input = X\_train.shape[1]

# 입력층과 은닉층 1 추가

model.add(Dense(units=MY\_HIDDEN,

input\_dim=input,

activation='relu'))

# 은닉층 2 추가

model.add(Dense(units=MY\_HIDDEN,

activation='relu'))

# 출력층 추가

model.add(Dense(units=1))

print('DNN 요약')

model.summary()

4. 인공신경망 학습

# 학습 환경 설정

model.compile(optimizer='sgd',

loss='mse')

print('학습 시작')

begin = time()

model.fit(x=X\_train,

y=Y\_train,

epochs=MY\_EPOCH,

verbose=0)

end = time()

print('총 학습 시간: {:.2f}초'.format(end - begin))

5. 인공신경망 평가

# 신경망 평가

loss = model.evaluate(x=X\_test,

y=Y\_test,

verbose=0)

print('최종 손실값: {:.2f}'.format(loss))

# DNN으로 예측

pred = model.predict(x=X\_test)

plt.figure(figsize=(10, 10))

sns.regplot(x=Y\_test, y=pred)

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.show()

첫번째 문제는 집값을 예측하는 것에서, 집의 나이를 추측하는 것으로 코딩을 바꾸어 보세요. 어떤 문제가 난이도가 높은가요? 입니다. 정확히 모르겠지만 제 생각에는 집값의 예측하는 부분이 #데이터를 입력과 출력으로 분리 파트에서 숫자 12를 6으로 변경하여 집의 나이를 추측할 수 있을 것 같아서 실행을 해보았는데 집값을 구했을 땐 총 학습시간은 10.82초, 최종 손실값은 0.15가 나왔습니다. 제 생각으로 바꾼 집의 나이를 추측하는 것의 총 학습시간은 9.25초, 최종 손실값은 0.22가 나왔습니다.

2번째 문제는 잘 모르겠습니다.

3번째 문제는 손실값을 최소화하는 optimizer 와 loss 함수를 찾아보세요. 어떤 combination인가요? 입니다. 일단 optimizer는 SGD입니다. SGD는 매개변수 값을 조정 시 전체 데이터가 아니라 랜덤으로 선택한 하나의 데이터에 대해서만 계산하는 방법입니다. 더 적은 데이터를 사용하므로 더 빠르게 계산할 수 있습니다. Loss 함수는 MSE 입니다. MSE는 평균 제곱 오차로 연속형 변수를 예측할 때 사용합니다.

4번째 문제는 가능한 방법을 총동원해서 손실값을 개선해 보는 문제입니다. 저는 MY\_EPOCH 값을 줄이고 은닉층을 2개를 추가했습니다.

### Keras DNN으로 집값 예측하기

1. 패키지 수입 및 파라미터 설정

# 수입

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from time import time

from keras.models import Sequential

from keras.layers import Dense

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

# 파라미터

MY\_HIDDEN = 1000

MY\_EPOCH = 150

pd.set\_option('display.max\_columns', None)

2. 데이터 처리

# 데이터 읽기

raw = pd.read\_csv('/content/drive/MyDrive/data/housing.csv')

# 데이터 샘플 10개 출력

print(raw.head(10))

# 데이터 통계 출력

print(raw.describe())

plt.figure(figsize=(15, 8))

sns.set(font\_scale=1.5)

sns.boxplot(data=raw, palette='dark')

# z-점수 정규화

scaler = StandardScaler()

z\_data = scaler.fit\_transform(raw)

# 데이터 형식 출력

print(type(raw))

print(type(z\_data))

# numpy를 pandas로 전환

z\_data = pd.DataFrame(z\_data)

print(type(z\_data))

# z-점수 정규화 된 데이터 샘플 출력

print(z\_data.head(10))

print(z\_data.describe())

plt.figure(figsize=(15, 8))

sns.set(font\_scale=1.5)

sns.boxplot(data=z\_data, palette='dark')

# 데이터 모양 출력

print('분리 전:', z\_data.shape)

# 데이터를 입력과 출력으로 분리

x\_data = z\_data.drop(12, axis=1)

y\_data = z\_data[12]

#데이터 모형 출력

print('분리 후:')

print(x\_data.shape)

print(y\_data.shape)

# 데이터 4분할

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(x\_data,y\_data,train\_size=0.7)

# 최종 데이터 모양 출력

print('학습용 입력 데이터 모양:', X\_train.shape)

print('학습용 출력 데이터 모양:', Y\_train.shape)

print('평가용 입력 데이터 모양:', X\_test.shape)

print('평가용 출력 데이터 모양:', Y\_test.shape)

3. 인공 신경망 구현

model = Sequential()

input = X\_train.shape[1]

# 입력층과 은닉층 1 추가

model.add(Dense(units=MY\_HIDDEN,

input\_dim=input,

activation='relu'))

# 은닉층 2 추가

model.add(Dense(units=MY\_HIDDEN,

activation='relu'))

# 은닉층 3 추가

model.add(Dense(units=MY\_HIDDEN,

activation='relu'))

# 은닉층 4 추가

model.add(Dense(units=MY\_HIDDEN,

activation='relu'))

# 출력층 추가

model.add(Dense(units=1))

print('DNN 요약')

model.summary()

4. 인공신경망 학습

# 학습 환경 설정

model.compile(optimizer='sgd',

loss='mse')

print('학습 시작')

begin = time()

model.fit(x=X\_train,

y=Y\_train,

epochs=MY\_EPOCH,

verbose=0)

end = time()

print('총 학습 시간: {:.2f}초'.format(end - begin))

5. 인공신경망 평가

# 신경망 평가

loss = model.evaluate(x=X\_test,

y=Y\_test,

verbose=0)

print('최종 손실값: {:.2f}'.format(loss))

# DNN으로 예측

pred = model.predict(x=X\_test)

plt.figure(figsize=(10, 10))

sns.regplot(x=Y\_test, y=pred)

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.show()