1. 회귀분석을 위한 fullly\_connected\_network(Dense Layer)를 생성하고 컴파일하고 모델을 학습시키시오.

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

import numpy

import tensorflow as tf

data\_set = np.loadtxt("../dataset/ThoraricSurgery.csv", delimiter=",")

X = Data\_set[:,0:17]

Y = Data\_set[:,17]

“””

model = Sequential()

model.add(Dense(12, input\_dim=17, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(1))

model = Sequential()

model.add(Dense(12, input\_dim=17, activation='linear'))

model.add(Dense(8, activation='linear'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mse',

optimizer='adam',

metrics=['mse'])

model.fit(X, Y, epochs=200, batch\_size=10)”””

2. 입력데이터 shape은 (8,)이고 은닉층의 node 수는 10인 은닉층의 수는 2인 binary classification을 위한 fullly\_connected\_network를 생성하시오.

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

dataset = numpy.loadtxt("../dataset/pima-indians-diabetes.csv", delimiter=",")

X = dataset[:,0:8]

Y = dataset[:,8]

“””

model = Sequential()

model.add(Dense(15, input\_dim=8, activation='relu'))

model.add(Dense(10, activation='relu'))

model.add(Dense(10, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary\_crossentropy',

optimizer='adam',

metrics=['accuracy'])

model.fit(X, Y, epochs=200, batch\_size=10)”””

3. 입력데이터 shape은 (4,)이고 입력층의 node는 30개 노드, 은닉층의 node 수는 10이고 분류할 label의 class 수가 3인 multi classification을 위한 fullly\_connected\_network를 생성하시오.

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn.preprocessing import LabelEncoder

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

import tensorflow as tf

np.random.seed(3)

tf.random.set\_seed(3)

df = pd.read\_csv('../dataset/iris.csv', names = ["sepal\_length", "sepal\_width", "petal\_length", "petal\_width", "species"])

dataset = df.values

X = dataset[:,0:4].astype(float)

Y\_obj = dataset[:,4]

# 문자열을 숫자로 변환

e = LabelEncoder() # 정수인덱싱 : 0, 1, 2, …

e.fit(Y\_obj)

Y = e.transform(Y\_obj)

Y\_encoded = tf.keras.utils.to\_categorical(Y) # one\_hot : 1,0,0/ 0,1,0/ 0,0,1 …

“”” model = Sequential()

model.add(Dense(30, input\_dim=4, activation='relu'))

model.add(Dense(10, activation='relu'))

model.add(Dense(3, activation='softmax')) # muti\_cls activation function

model.compile(loss='categorical\_crossentropy',

optimizer='adam',

metrics=['accuracy'])

model.fit(X, Y\_encoded, epochs=50, batch\_size=1)

model.evaluate(X, Y\_encoded)”””

4. MNIST 손글씨 데이터를 구별하는 cnn network를 생성하고 학습시키시오..

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D

from keras.callbacks import ModelCheckpoint,EarlyStopping

import matplotlib.pyplot as plt

import numpy

import os

import tensorflow as tf

(X\_train, Y\_train), (X\_test, Y\_test) = mnist.load\_data()

X\_train = X\_train.reshape(X\_train.shape[0], 28, 28, 1).astype('float32') / 255

X\_test = X\_test.reshape(X\_test.shape[0], 28, 28, 1).astype('float32') / 255

Y\_train = np\_utils.to\_categorical(Y\_train)

Y\_test = np\_utils.to\_categorical(Y\_test)

“””

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3), input\_shape=(28, 28, 1), activation='relu'))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=2))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(10, activation='softmax'))

model.compile(loss='categorical\_crossentropy',

optimizer='adam',

metrics=['accuracy'])

history = model.fit(X\_train, Y\_train, validation\_data=(X\_test, Y\_test), epochs=30, batch\_size=200, verbose=0)”””

MODEL\_DIR = './model/'

if not os.path.exists(MODEL\_DIR):

os.mkdir(MODEL\_DIR)

modelpath="./model/{epoch:02d}-{val\_loss:.4f}.hdf5"

checkpointer = ModelCheckpoint(filepath=modelpath, monitor='val\_loss', verbose=1, save\_best\_only=True)

early\_stopping\_callback = EarlyStopping(monitor='val\_loss', patience=10)

history = model.fit(X\_train, Y\_train, validation\_data=(X\_test, Y\_test), epochs=30, batch\_size=200, verbose=0, callbacks=[early\_stopping\_callback,checkpointer])

print("\n Test Accuracy: %.4f" % (model.evaluate(X\_test, Y\_test)[1]))

5. LSTM 모델을 사용해서 reuter 뉴스 분류모델을 생성하고 학습시키시오..

import numpy

import tensorflow as tf

import matplotlib.pyplot as plt

from keras.datasets import reuters # 11258기사, 46카테고리

from keras.models import Sequential

from keras.layers import Dense, LSTM, Embedding

from keras.preprocessing import sequence

from keras.utils import np\_utils

(X\_train, Y\_train), (X\_test, Y\_test) = reuters.load\_data(num\_words=1000, test\_split=0.2) # 단어 사용 빈도 상위 1000개만 불러옴

category = numpy.max(Y\_train) + 1

# 데이터 전처리

x\_train = sequence.pad\_sequences(X\_train, maxlen=100) # 단어수를 100으로 통일

x\_test = sequence.pad\_sequences(X\_test, maxlen=100)

y\_train = np\_utils.to\_categorical(Y\_train) # onehot으로 변환

y\_test = np\_utils.to\_categorical(Y\_test)

# 모델의 설정

“””

model = Sequential()

model.add(Embedding(1000, 100)) # length of words, embedding vector size

model.add(LSTM(100, activation='tanh')) # hidden state shape

model.add(Dense(46, activation='softmax'))

model.summary()

model.compile(loss='categorical\_crossentropy',

optimizer='adam',

metrics=['accuracy'])

history = model.fit(x\_train, y\_train, batch\_size=10, epochs=20, validation\_data=(x\_test, y\_test))

preds = model.predict(x\_test[5:6])

”””