NNDL ASSIGNMENT \_ 6

# \*\*Use the use case in the class: a. Add more Dense layers to the existing code and check how the accuracy changes\*\*

import pandas as pd

import numpy as np

from keras.models import Sequential

from keras.layers import Dense

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

dataset = pd.read\_csv("diabetes.csv", header=None).values

X = dataset[:,0:8]

Y = dataset[:,8]

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.25, random\_state=87)

np.random.seed(155)

model = Sequential()

model.add(Dense(20, 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\_train, Y\_train, epochs=100, initial\_epoch=0)

loss, accuracy = model.evaluate(X\_test, Y\_test)

print("Test Loss:", loss)

print("Test Accuracy:", accuracy)

#2. Change the data source to Breast Cancer dataset \* available in the source code folder and make required

changes. Report accuracy of the model.

import pandas as pd

import numpy as np

from keras.models import Sequential

from keras.layers import Dense

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

from sklearn.datasets import load\_breast\_cancer

data = load\_breast\_cancer()

X = data.data

Y = data.target

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.25, random\_state=87)

np.random.seed(155)

model = Sequential()

model.add(Dense(20, input\_dim=30, 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\_train, Y\_train, epochs=100, initial\_epoch=0)

loss, accuracy = model.evaluate(X\_test, Y\_test)

print("Test Loss:", loss)

print("Test Accuracy:", accuracy)

#3. Normalize the data before feeding the data to the model and check how the normalization change your

accuracy (code given below).

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

import pandas as pd

import numpy as np

from keras.models import Sequential

from keras.layers import Dense

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

from sklearn.datasets import load\_breast\_cancer

from sklearn.preprocessing import StandardScaler

data = load\_breast\_cancer()

X = data.data

Y = data.target

scaler = StandardScaler()

X = scaler.fit\_transform(X)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.25, random\_state=87)

np.random.seed(155)

model = Sequential()

model.add(Dense(20, input\_dim=30, 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\_train, Y\_train, epochs=100, initial\_epoch=0)

loss, accuracy = model.evaluate(X\_test, Y\_test)

print("Test Loss:", loss)

print("Test Accuracy:", accuracy)

PROBLEM \_02

#1. Plot the loss and accuracy for both training data and validation data using the history object in the sourcecode.

import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.show()

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.show()

#2. Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model

on that single image.

import matplotlib.pyplot as plt

plt.imshow(test\_images[0], cmap='gray')

image = test\_data[0].reshape(1, dimData)

prediction = model.predict(image)

print('Prediction:', np.argmax(prediction))

#3. We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the

activation to tanh or sigmoid and see what happens

model = Sequential()

model.add(Dense(512, activation='tanh', input\_shape=(dimData,)))

model.add(Dense(256, activation='tanh'))

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

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

model.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(train\_data, train\_labels\_one\_hot, batch\_size=256, epochs=10, verbose=1,

                   validation\_data=(test\_data, test\_labels\_one\_hot))

                   model = Sequential()

model.add(Dense(512, activation='sigmoid', input\_shape=(dimData,)))

model.add(Dense(256, activation='sigmoid'))

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

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

model.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(train\_data, train\_labels\_one\_hot, batch\_size=256, epochs=10, verbose=1,

                   validation\_data=(test\_data, test\_labels\_one\_hot))

#4. Run the same code without scaling the images and check the performance?

train\_data = train\_images.reshape(train\_images.shape[0],dimData)

test\_data = test\_images.reshape(test\_images.shape[0],dimData)

train\_labels\_one\_hot = to\_categorical(train\_labels)

test\_labels\_one\_hot = to\_categorical(test\_labels)

model = Sequential()

model.add(Dense(512, activation='relu', input\_shape=(dimData,)))

model.add(Dense(512, activation='relu'))

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

model.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(train\_data, train\_labels\_one\_hot, batch\_size=256, epochs=10, verbose=1,

                   validation\_data=(test\_data, test\_labels\_one\_hot))

GIT HUB LINK: https://github.com/Goli18/NNDL\_ASS6.git

VIDEO LINK : https://files.fm/f/x29t8drtj