#read the data

import pandas as pd

data = pd.read\_csv(r'C:\Users\lahar\Desktop\NN06\diabetes.csv')

path\_to\_csv = r'C:\Users\lahar\Desktop\NN06\diabetes.csv'

import keras

import pandas

from keras.models import Sequential

from keras.layers.core import Dense, Activation

# load dataset

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

import pandas as pd

import numpy as np

dataset = pd.read\_csv(path\_to\_csv, header=None).values

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(dataset[:,0:8], dataset[:,8],

test\_size=0.25, random\_state=87)

np.random.seed(155)

my\_first\_nn = Sequential() # create model

my\_first\_nn.add(Dense(20, input\_dim=8, activation='relu')) # hidden layer

my\_first\_nn.add(Dense(4, activation='relu')) # hidden layer

my\_first\_nn.add(Dense(1, activation='sigmoid')) # output layer

my\_first\_nn.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['acc'])

my\_first\_nn\_fitted = my\_first\_nn.fit(X\_train, Y\_train, epochs=100,

initial\_epoch=0)

print(my\_first\_nn.summary())

print(my\_first\_nn.evaluate(X\_test, Y\_test))

#read the data

data = pd.read\_csv(r'C:\Users\lahar\Desktop\NN06\breastcancer.csv')

path\_to\_csv = r'C:\Users\lahar\Desktop\NN06\breastcancer.csv'

import keras

import pandas as pd

import numpy as np

from keras.models import Sequential

from keras.layers.core import Dense, Activation

from sklearn.datasets import load\_breast\_cancer

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

# load dataset

cancer\_data = load\_breast\_cancer()

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(cancer\_data.data, cancer\_data.target,

test\_size=0.25, random\_state=87)

np.random.seed(155)

my\_nn = Sequential() # create model

my\_nn.add(Dense(20, input\_dim=30, activation='relu')) # hidden layer 1

my\_nn.add(Dense(1, activation='sigmoid')) # output layer

my\_nn.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['acc'])

my\_nn\_fitted = my\_nn.fit(X\_train, Y\_train, epochs=100,

initial\_epoch=0)

print(my\_nn.summary())

print(my\_nn.evaluate(X\_test, Y\_test))

#read the data

data = pd.read\_csv(r'C:\Users\lahar\Desktop\NN06\breastcancer.csv')

path\_to\_csv = r'C:\Users\lahar\Desktop\NN06\breastcancer.csv'

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

import keras

import pandas as pd

import numpy as np

from keras.models import Sequential

from keras.layers.core import Dense, Activation

from sklearn.datasets import load\_breast\_cancer

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

# load dataset

cancer\_data = load\_breast\_cancer()

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(cancer\_data.data, cancer\_data.target,

test\_size=0.25, random\_state=87)

np.random.seed(155)

my\_nn = Sequential() # create model

my\_nn.add(Dense(20, input\_dim=30, activation='relu')) # hidden layer 1

my\_nn.add(Dense(1, activation='sigmoid')) # output layer

my\_nn.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['acc'])

my\_nn\_fitted = my\_nn.fit(X\_train, Y\_train, epochs=100,

initial\_epoch=0)

print(my\_nn.summary())

print(my\_nn.evaluate(X\_test, Y\_test))

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout

import matplotlib.pyplot as plt

# load MNIST dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# normalize pixel values to range [0, 1]

x\_train = x\_train.astype('float32') / 255

x\_test = x\_test.astype('float32') / 255

# convert class labels to binary class matrices

num\_classes = 10

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

# create a simple neural network model

model = Sequential()

model.add(Dense(512, activation='relu', input\_shape=(784,)))

model.add(Dropout(0.2))

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

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

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

# train the model and record the training history

history = model.fit(x\_train.reshape(-1, 784), y\_train, validation\_data=(x\_test.reshape(-1, 784), y\_test),

epochs=20, batch\_size=128)

# plot the training and validation accuracy and loss curves

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

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

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

plt.title('Model Accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Validation'], loc='lower right')

plt.subplot(1, 2, 2)

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

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

plt.title('Model Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Validation'], loc='upper right')

plt.show()

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout

import matplotlib.pyplot as plt

import numpy as np

# load MNIST dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# normalize pixel values to range [0, 1]

x\_train = x\_train.astype('float32') / 255

x\_test = x\_test.astype('float32') / 255

# convert class labels to binary class matrices

num\_classes = 10

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

# create a simple neural network model

model = Sequential()

model.add(Dense(512, activation='relu', input\_shape=(784,)))

model.add(Dropout(0.2))

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

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

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

# train the model

model.fit(x\_train.reshape(-1, 784), y\_train, validation\_data=(x\_test.reshape(-1, 784), y\_test),

epochs=20, batch\_size=128)

# plot one of the images in the test data

plt.imshow(x\_test[0], cmap='gray')

plt.show()

# make a prediction on the image using the trained model

prediction = model.predict(x\_test[0].reshape(1, -1))

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

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout

import matplotlib.pyplot as plt

import numpy as np

# load MNIST dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# normalize pixel values to range [0, 1]

x\_train = x\_train.astype('float32') / 255

x\_test = x\_test.astype('float32') / 255

# convert class labels to binary class matrices

num\_classes = 10

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

# create a list of models to train

models = []

# model with 1 hidden layer and tanh activation

model = Sequential()

model.add(Dense(512, activation='tanh', input\_shape=(784,)))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

models.append(('1 hidden layer with tanh', model))

# model with 1 hidden layer and sigmoid activation

model = Sequential()

model.add(Dense(512, activation='sigmoid', input\_shape=(784,)))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

models.append(('1 hidden layer with sigmoid', model))

# model with 2 hidden layers and tanh activation

model = Sequential()

model.add(Dense(512, activation='tanh', input\_shape=(784,)))

model.add(Dropout(0.2))

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

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

models.append(('2 hidden layers with tanh', model))

# model with 2 hidden layers and sigmoid activation

model = Sequential()

model.add(Dense(512, activation='sigmoid', input\_shape=(784,)))

model.add(Dropout(0.2))

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

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

models.append(('2 hidden layers with sigmoid', model))

# train each model and plot loss and accuracy curves

for name, model in models:

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

history = model.fit(x\_train.reshape(-1, 784), y\_train, validation\_data=(x\_test.reshape(-1, 784), y\_test),

epochs=20, batch\_size=128, verbose=0)

# plot loss and accuracy curves

plt.plot(history.history['loss'], label='train\_loss')

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

plt.plot(history.history['accuracy'], label='train\_accuracy')

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

plt.title(name)

plt.xlabel('Epoch')

plt.legend()

plt.show()

# evaluate the model on test data

loss, accuracy = model.evaluate(x\_test.reshape(-1, 784), y\_test, verbose=0)

print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout

import matplotlib.pyplot as plt

import numpy as np

# load MNIST dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# convert class labels to binary class matrices

num\_classes = 10

y\_train = keras.utils.to\_categorical(y\_train, num\_classes)

y\_test = keras.utils.to\_categorical(y\_test, num\_classes)

# create a list of models to train

models = []

# model with 1 hidden layer and tanh activation

model = Sequential()

model.add(Dense(512, activation='tanh', input\_shape=(784,)))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

models.append(('1 hidden layer with tanh', model))

# model with 1 hidden layer and sigmoid activation

model = Sequential()

model.add(Dense(512, activation='sigmoid', input\_shape=(784,)))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

models.append(('1 hidden layer with sigmoid', model))

# model with 2 hidden layers and tanh activation

model = Sequential()

model.add(Dense(512, activation='tanh', input\_shape=(784,)))

model.add(Dropout(0.2))

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

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

models.append(('2 hidden layers with tanh', model))

# model with 2 hidden layers and sigmoid activation

model = Sequential()

model.add(Dense(512, activation='sigmoid', input\_shape=(784,)))

model.add(Dropout(0.2))

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

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax'))

models.append(('2 hidden layers with sigmoid', model))

# train each model and plot loss and accuracy curves

for name, model in models:

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

history = model.fit(x\_train.reshape(-1, 784), y\_train, validation\_data=(x\_test.reshape(-1, 784), y\_test),

epochs=20, batch\_size=128, verbose=0)

# plot loss and accuracy curves

plt.plot(history.history['loss'], label='train\_loss')

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

plt.plot(history.history['accuracy'], label='train\_accuracy')

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

plt.title(name)

plt.xlabel('Epoch')

plt.legend()

plt.show()

# evaluate the model on test data

loss, accuracy = model.evaluate(x\_test.reshape(-1, 784), y\_test, verbose=0)

print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))