Assignment 4:

# -\*- coding: utf-8 -\*-

"""assignment 4.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1xTWuS5yXH\_I6GjEKqYnW\_prdy4lCzlrU

"""

'''

Assignment IV:

Due: June 12 (this is a small assignment)

Pick a dataset from UCI under classification and build a neural network for the classification.

Try more than one architecture, for example, by having different layers. Using Dense layers is meaningful for all problems. (remember: Conv2D is relevant only for rectangular data, such as images)

Play with the number of neurons and epochs to see how the network perform.

This assignment will help you if you plan to do any neural networks for the final project.

Use the codes provided from the lecture as a starting point.

Enjoy

'''

# Commented out IPython magic to ensure Python compatibility.

from google.colab import drive

drive.mount('/content/drive')

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# %matplotlib inline

df = pd.read\_csv("/content/drive/MyDrive/Colab Notebooks/songs\_normalize.csv", sep=",")

df = df.dropna(how="any")

print("\n")

print(f"The whole dataframe looks like:\n {df}.")

print("\n")

print("The dataframe information is: ")

print(f"{df.info()}.")

print("\n")

print(f"The dataframe shape is: {df.shape}.")

X= df.iloc[:, [2,6]]

X\_train = df.iloc[:,[2,6,7]]

X\_4 = df.iloc[:,[2,6,7,8]]

X\_5 = df.iloc[:,[2,6,7,8,9]]

X\_6 = df.iloc[:,[2,6,7,8,9,10]]

X\_7 = df.iloc[:,[2,6,7,8,9,10,11]]

X\_8 = df.iloc[:,[2,6,7,8,9,10,11,12]]

X\_9 = df.iloc[:,[2,6,7,8,9,10,11,12,13]]

X\_10 = df.iloc[:,[2,6,7,8,9,10,11,12,13,14]]

X\_11 = df.iloc[:,[2,6,7,8,9,10,11,12,13,14,15]]

X\_12 = df.iloc[:,[2,6,7,8,9,10,11,12,13,14,15,16]]

y = df["popularity"]

lst = []

#keras1.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=2, activation='sigmoid', input\_dim = 2) ## change it to 3, 4, 5, 6, .. to see results

model.add(layer\_1)

# layer 2

layer\_2 = Dense(units=1, activation='sigmoid')

model.add(layer\_2)

print(model.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model.fit(X,y, epochs=2, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model.predict(X))

# good to test on new data, test data.

lst.append(model.predict(X)[0])

print(lst)

#keras2.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model2 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=3, activation='sigmoid', input\_dim = 3) ## change it to 3, 4, 5, 6, .. to see results

model2.add(layer\_1)

# layer 2

layer\_2 = Dense(units=2, activation='sigmoid')

model2.add(layer\_2)

#layer 3

layer\_3 = Dense(units=1, activation='sigmoid')

model2.add(layer\_3)

print(model2.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model2.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model2.fit(X\_train,y, epochs=3, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model2.predict(X\_train))

# good to test on new data, test data.

lst.append(model2.predict(X\_train)[0])

#keras3.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model3 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=4, activation='sigmoid', input\_dim = 4) ## change it to 3, 4, 5, 6, .. to see results

model3.add(layer\_1)

# layer 2

layer\_2 = Dense(units=3, activation='sigmoid')

model3.add(layer\_2)

#layer 3

layer\_3 = Dense(units=2, activation='sigmoid')

model3.add(layer\_3)

#layer 4

layer\_4 = Dense(units=1, activation='sigmoid')

model3.add(layer\_4)

print(model3.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model3.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model3.fit(X\_4,y, epochs=4, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model3.predict(X\_4))

# good to test on new data, test data.

lst.append(model3.predict(X\_4)[0])

#keras4.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model4 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=5, activation='sigmoid', input\_dim = 5) ## change it to 3, 4, 5, 6, .. to see results

model4.add(layer\_1)

# layer 2

layer\_2 = Dense(units=4, activation='sigmoid')

model4.add(layer\_2)

#layer 3

layer\_3 = Dense(units=3, activation='sigmoid')

model4.add(layer\_3)

#layer 4

layer\_4 = Dense(units=2, activation='sigmoid')

model4.add(layer\_4)

#layer 5

layer\_5 = Dense(units=1, activation='sigmoid')

model4.add(layer\_5)

print(model4.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_4.input\_shape, layer\_5.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model4.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model4.fit(X\_5,y, epochs=5, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model4.predict(X\_5))

# good to test on new data, test data.

lst.append(model4.predict(X\_5)[0])

#keras5.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model5 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=6, activation='sigmoid', input\_dim = 6) ## change it to 3, 4, 5, 6, .. to see results

model5.add(layer\_1)

# layer 2

layer\_2 = Dense(units=5, activation='sigmoid')

model5.add(layer\_2)

#layer 3

layer\_3 = Dense(units=4, activation='sigmoid')

model5.add(layer\_3)

#layer 4

layer\_4 = Dense(units=3, activation='sigmoid')

model5.add(layer\_4)

#layer 5

layer\_5 = Dense(units=2, activation='sigmoid')

model5.add(layer\_5)

#layer 6

layer\_6 = Dense(units=1, activation='sigmoid')

model5.add(layer\_6)

print(model5.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model5.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model5.fit(X\_6,y, epochs=6, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model5.predict(X\_6))

# good to test on new data, test data.

lst.append(model5.predict(X\_6)[0])

#keras6.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model6 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=7, activation='sigmoid', input\_dim = 7) ## change it to 3, 4, 5, 6, .. to see results

model6.add(layer\_1)

# layer 2

layer\_2 = Dense(units=6, activation='sigmoid')

model6.add(layer\_2)

#layer 3

layer\_3 = Dense(units=5, activation='sigmoid')

model6.add(layer\_3)

#layer 4

layer\_4 = Dense(units=4, activation='sigmoid')

model6.add(layer\_4)

#layer 5

layer\_5 = Dense(units=3, activation='sigmoid')

model6.add(layer\_5)

#layer 6

layer\_6 = Dense(units=2, activation='sigmoid')

model6.add(layer\_6)

#layer 7

layer\_7 = Dense(units=1, activation='sigmoid')

model6.add(layer\_7)

print(model6.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

print("\n\n layer\_7 : ", layer\_7.input\_shape, layer\_7.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model6.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model6.fit(X\_7,y, epochs=7, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model6.predict(X\_7))

# good to test on new data, test data.

lst.append(model6.predict(X\_7)[0])

#keras7.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model7 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=8, activation='sigmoid', input\_dim = 8) ## change it to 3, 4, 5, 6, .. to see results

model7.add(layer\_1)

# layer 2

layer\_2 = Dense(units=7, activation='sigmoid')

model7.add(layer\_2)

#layer 3

layer\_3 = Dense(units=6, activation='sigmoid')

model7.add(layer\_3)

#layer 4

layer\_4 = Dense(units=5, activation='sigmoid')

model7.add(layer\_4)

#layer 5

layer\_5 = Dense(units=4, activation='sigmoid')

model7.add(layer\_5)

#layer 6

layer\_6 = Dense(units=3, activation='sigmoid')

model7.add(layer\_6)

#layer 7

layer\_7 = Dense(units=2, activation='sigmoid')

model7.add(layer\_7)

#layer 8

layer\_8 = Dense(units=1, activation='sigmoid')

model7.add(layer\_8)

print(model7.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

print("\n\n layer\_7 : ", layer\_7.input\_shape, layer\_7.output\_shape)

print("\n\n layer\_8 : ", layer\_8.input\_shape, layer\_8.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model7.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model7.fit(X\_8,y, epochs=8, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model7.predict(X\_8))

# good to test on new data, test data.

lst.append(model7.predict(X\_8)[0])

#keras8.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model8 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=9, activation='sigmoid', input\_dim = 9) ## change it to 3, 4, 5, 6, .. to see results

model8.add(layer\_1)

# layer 2

layer\_2 = Dense(units=8, activation='sigmoid')

model8.add(layer\_2)

#layer 3

layer\_3 = Dense(units=7, activation='sigmoid')

model8.add(layer\_3)

#layer 4

layer\_4 = Dense(units=6, activation='sigmoid')

model8.add(layer\_4)

#layer 5

layer\_5 = Dense(units=5, activation='sigmoid')

model8.add(layer\_5)

#layer 6

layer\_6 = Dense(units=4, activation='sigmoid')

model8.add(layer\_6)

#layer 7

layer\_7 = Dense(units=3, activation='sigmoid')

model8.add(layer\_7)

#layer 8

layer\_8 = Dense(units=2, activation='sigmoid')

model8.add(layer\_8)

#layer 9

layer\_9 = Dense(units=1, activation='sigmoid')

model8.add(layer\_9)

print(model8.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

print("\n\n layer\_7 : ", layer\_7.input\_shape, layer\_7.output\_shape)

print("\n\n layer\_8 : ", layer\_8.input\_shape, layer\_8.output\_shape)

print("\n\n layer\_9 : ", layer\_9.input\_shape, layer\_9.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model8.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model8.fit(X\_9,y, epochs=9, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model8.predict(X\_9))

# good to test on new data, test data.

lst.append(model8.predict(X\_9)[0])

#keras9.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model9 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=10, activation='sigmoid', input\_dim = 10) ## change it to 3, 4, 5, 6, .. to see results

model9.add(layer\_1)

# layer 2

layer\_2 = Dense(units=9, activation='sigmoid')

model9.add(layer\_2)

#layer 3

layer\_3 = Dense(units=8, activation='sigmoid')

model9.add(layer\_3)

#layer 4

layer\_4 = Dense(units=7, activation='sigmoid')

model9.add(layer\_4)

#layer 5

layer\_5 = Dense(units=6, activation='sigmoid')

model9.add(layer\_5)

#layer 6

layer\_6 = Dense(units=5, activation='sigmoid')

model9.add(layer\_6)

#layer 7

layer\_7 = Dense(units=4, activation='sigmoid')

model9.add(layer\_7)

#layer 8

layer\_8 = Dense(units=3, activation='sigmoid')

model9.add(layer\_8)

#layer 9

layer\_9 = Dense(units=2, activation='sigmoid')

model9.add(layer\_9)

#layer 10

layer\_10 = Dense(units=1, activation='sigmoid')

model9.add(layer\_10)

print(model9.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

print("\n\n layer\_7 : ", layer\_7.input\_shape, layer\_7.output\_shape)

print("\n\n layer\_8 : ", layer\_8.input\_shape, layer\_8.output\_shape)

print("\n\n layer\_9 : ", layer\_9.input\_shape, layer\_9.output\_shape)

print("\n\n layer\_10 : ", layer\_10.input\_shape, layer\_10.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model9.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model9.fit(X\_10,y, epochs=10, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model9.predict(X\_10))

# good to test on new data, test data.

lst.append(model9.predict(X\_10)[0])

#keras10.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model10 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=11, activation='sigmoid', input\_dim = 11) ## change it to 3, 4, 5, 6, .. to see results

model10.add(layer\_1)

# layer 2

layer\_2 = Dense(units=10, activation='sigmoid')

model10.add(layer\_2)

#layer 3

layer\_3 = Dense(units=9, activation='sigmoid')

model10.add(layer\_3)

#layer 4

layer\_4 = Dense(units=8, activation='sigmoid')

model10.add(layer\_4)

#layer 5

layer\_5 = Dense(units=7, activation='sigmoid')

model10.add(layer\_5)

#layer 6

layer\_6 = Dense(units=6, activation='sigmoid')

model10.add(layer\_6)

#layer 7

layer\_7 = Dense(units=5, activation='sigmoid')

model10.add(layer\_7)

#layer 8

layer\_8 = Dense(units=4, activation='sigmoid')

model10.add(layer\_8)

#layer 9

layer\_9 = Dense(units=3, activation='sigmoid')

model10.add(layer\_9)

#layer 10

layer\_10 = Dense(units=2, activation='sigmoid')

model10.add(layer\_10)

#layer 11

layer\_11 = Dense(units=1, activation='sigmoid')

model10.add(layer\_11)

print(model10.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

print("\n\n layer\_7 : ", layer\_7.input\_shape, layer\_7.output\_shape)

print("\n\n layer\_8 : ", layer\_8.input\_shape, layer\_8.output\_shape)

print("\n\n layer\_9 : ", layer\_9.input\_shape, layer\_9.output\_shape)

print("\n\n layer\_10 : ", layer\_10.input\_shape, layer\_10.output\_shape)

print("\n\n layer\_11 : ", layer\_11.input\_shape, layer\_11.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model10.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model10.fit(X\_11,y, epochs=11, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model10.predict(X\_11))

# good to test on new data, test data.

lst.append(model10.predict(X\_11)[0])

#keras11.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model11 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=12, activation='sigmoid', input\_dim = 12) ## change it to 3, 4, 5, 6, .. to see results

model11.add(layer\_1)

# layer 2

layer\_2 = Dense(units=11, activation='sigmoid')

model11.add(layer\_2)

#layer 3

layer\_3 = Dense(units=10, activation='sigmoid')

model11.add(layer\_3)

#layer 4

layer\_4 = Dense(units=9, activation='sigmoid')

model11.add(layer\_4)

#layer 5

layer\_5 = Dense(units=8, activation='sigmoid')

model11.add(layer\_5)

#layer 6

layer\_6 = Dense(units=7, activation='sigmoid')

model11.add(layer\_6)

#layer 7

layer\_7 = Dense(units=6, activation='sigmoid')

model11.add(layer\_7)

#layer 8

layer\_8 = Dense(units=5, activation='sigmoid')

model11.add(layer\_8)

#layer 9

layer\_9 = Dense(units=4, activation='sigmoid')

model11.add(layer\_9)

#layer 10

layer\_10 = Dense(units=3, activation='sigmoid')

model11.add(layer\_10)

#layer 11

layer\_11 = Dense(units=2, activation='sigmoid')

model11.add(layer\_11)

#layer 12

layer\_12 = Dense(units=1, activation='sigmoid')

model11.add(layer\_12)

print(model11.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

print("\n\n layer\_7 : ", layer\_7.input\_shape, layer\_7.output\_shape)

print("\n\n layer\_8 : ", layer\_8.input\_shape, layer\_8.output\_shape)

print("\n\n layer\_9 : ", layer\_9.input\_shape, layer\_9.output\_shape)

print("\n\n layer\_10 : ", layer\_10.input\_shape, layer\_10.output\_shape)

print("\n\n layer\_11 : ", layer\_11.input\_shape, layer\_11.output\_shape)

print("\n\n layer\_12 : ", layer\_12.input\_shape, layer\_12.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model11.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model11.fit(X\_12,y, epochs=12, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model11.predict(X\_12))

# good to test on new data, test data.

lst.append(model11.predict(X\_12)[0])

#keras1.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=2, activation='sigmoid', input\_dim = 2) ## change it to 3, 4, 5, 6, .. to see results

model.add(layer\_1)

# layer 2

layer\_2 = Dense(units=1, activation='sigmoid')

model.add(layer\_2)

print(model.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model.fit(X,y, epochs=2, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model.predict(X))

# good to test on new data, test data.

lst.append(model.predict(X)[0])

#keras2.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model2 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=1, activation='sigmoid', input\_dim = 3) ## change it to 3, 4, 5, 6, .. to see results

model2.add(layer\_1)

# layer 2

layer\_2 = Dense(units=2, activation='sigmoid')

model2.add(layer\_2)

#layer 3

layer\_3 = Dense(units=1, activation='sigmoid')

model2.add(layer\_3)

print(model2.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model2.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model2.fit(X\_train,y, epochs=3, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model2.predict(X\_train))

# good to test on new data, test data.

lst.append(model2.predict(X\_train)[0])

#keras3.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model3 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=1, activation='sigmoid', input\_dim = 4) ## change it to 3, 4, 5, 6, .. to see results

model3.add(layer\_1)

# layer 2

layer\_2 = Dense(units=2, activation='sigmoid')

model3.add(layer\_2)

#layer 3

layer\_3 = Dense(units=2, activation='sigmoid')

model3.add(layer\_3)

#layer 4

layer\_4 = Dense(units=1, activation='sigmoid')

model3.add(layer\_4)

print(model3.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model3.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model3.fit(X\_4,y, epochs=4, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model3.predict(X\_4))

# good to test on new data, test data.

lst.append(model3.predict(X\_4)[0])

#keras4.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model4 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=1, activation='sigmoid', input\_dim = 5) ## change it to 3, 4, 5, 6, .. to see results

model4.add(layer\_1)

# layer 2

layer\_2 = Dense(units=2, activation='sigmoid')

model4.add(layer\_2)

#layer 3

layer\_3 = Dense(units=3, activation='sigmoid')

model4.add(layer\_3)

#layer 4

layer\_4 = Dense(units=2, activation='sigmoid')

model4.add(layer\_4)

#layer 5

layer\_5 = Dense(units=1, activation='sigmoid')

model4.add(layer\_5)

print(model4.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_4.input\_shape, layer\_5.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model4.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model4.fit(X\_5,y, epochs=5, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model4.predict(X\_5))

# good to test on new data, test data.

lst.append(model4.predict(X\_5)[0])

#keras5.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model5 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=1, activation='sigmoid', input\_dim = 6) ## change it to 3, 4, 5, 6, .. to see results

model5.add(layer\_1)

# layer 2

layer\_2 = Dense(units=2, activation='sigmoid')

model5.add(layer\_2)

#layer 3

layer\_3 = Dense(units=3, activation='sigmoid')

model5.add(layer\_3)

#layer 4

layer\_4 = Dense(units=3, activation='sigmoid')

model5.add(layer\_4)

#layer 5

layer\_5 = Dense(units=2, activation='sigmoid')

model5.add(layer\_5)

#layer 6

layer\_6 = Dense(units=1, activation='sigmoid')

model5.add(layer\_6)

print(model5.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model5.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model5.fit(X\_6,y, epochs=6, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model5.predict(X\_6))

# good to test on new data, test data.

lst.append(model5.predict(X\_6)[0])

#keras6.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model6 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=1, activation='sigmoid', input\_dim = 7) ## change it to 3, 4, 5, 6, .. to see results

model6.add(layer\_1)

# layer 2

layer\_2 = Dense(units=2, activation='sigmoid')

model6.add(layer\_2)

#layer 3

layer\_3 = Dense(units=3, activation='sigmoid')

model6.add(layer\_3)

#layer 4

layer\_4 = Dense(units=4, activation='sigmoid')

model6.add(layer\_4)

#layer 5

layer\_5 = Dense(units=3, activation='sigmoid')

model6.add(layer\_5)

#layer 6

layer\_6 = Dense(units=2, activation='sigmoid')

model6.add(layer\_6)

#layer 7

layer\_7 = Dense(units=1, activation='sigmoid')

model6.add(layer\_7)

print(model6.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

print("\n\n layer\_7 : ", layer\_7.input\_shape, layer\_7.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model6.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model6.fit(X\_7,y, epochs=7, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model6.predict(X\_7))

# good to test on new data, test data.

lst.append(model6.predict(X\_7)[0])

#keras7.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model7 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=1, activation='sigmoid', input\_dim = 8) ## change it to 3, 4, 5, 6, .. to see results

model7.add(layer\_1)

# layer 2

layer\_2 = Dense(units=2, activation='sigmoid')

model7.add(layer\_2)

#layer 3

layer\_3 = Dense(units=3, activation='sigmoid')

model7.add(layer\_3)

#layer 4

layer\_4 = Dense(units=4, activation='sigmoid')

model7.add(layer\_4)

#layer 5

layer\_5 = Dense(units=4, activation='sigmoid')

model7.add(layer\_5)

#layer 6

layer\_6 = Dense(units=3, activation='sigmoid')

model7.add(layer\_6)

#layer 7

layer\_7 = Dense(units=2, activation='sigmoid')

model7.add(layer\_7)

#layer 8

layer\_8 = Dense(units=1, activation='sigmoid')

model7.add(layer\_8)

print(model7.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

print("\n\n layer\_7 : ", layer\_7.input\_shape, layer\_7.output\_shape)

print("\n\n layer\_8 : ", layer\_8.input\_shape, layer\_8.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model7.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model7.fit(X\_8,y, epochs=8, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model7.predict(X\_8))

# good to test on new data, test data.

lst.append(model7.predict(X\_8)[0])

#keras8.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model8 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=1, activation='sigmoid', input\_dim = 9) ## change it to 3, 4, 5, 6, .. to see results

model8.add(layer\_1)

# layer 2

layer\_2 = Dense(units=2, activation='sigmoid')

model8.add(layer\_2)

#layer 3

layer\_3 = Dense(units=3, activation='sigmoid')

model8.add(layer\_3)

#layer 4

layer\_4 = Dense(units=4, activation='sigmoid')

model8.add(layer\_4)

#layer 5

layer\_5 = Dense(units=5, activation='sigmoid')

model8.add(layer\_5)

#layer 6

layer\_6 = Dense(units=4, activation='sigmoid')

model8.add(layer\_6)

#layer 7

layer\_7 = Dense(units=3, activation='sigmoid')

model8.add(layer\_7)

#layer 8

layer\_8 = Dense(units=2, activation='sigmoid')

model8.add(layer\_8)

#layer 9

layer\_9 = Dense(units=1, activation='sigmoid')

model8.add(layer\_9)

print(model8.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

print("\n\n layer\_7 : ", layer\_7.input\_shape, layer\_7.output\_shape)

print("\n\n layer\_8 : ", layer\_8.input\_shape, layer\_8.output\_shape)

print("\n\n layer\_9 : ", layer\_9.input\_shape, layer\_9.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model8.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model8.fit(X\_9,y, epochs=9, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model8.predict(X\_9))

# good to test on new data, test data.

lst.append(model8.predict(X\_9)[0])

#keras9.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model9 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=1, activation='sigmoid', input\_dim = 10) ## change it to 3, 4, 5, 6, .. to see results

model9.add(layer\_1)

# layer 2

layer\_2 = Dense(units=2, activation='sigmoid')

model9.add(layer\_2)

#layer 3

layer\_3 = Dense(units=3, activation='sigmoid')

model9.add(layer\_3)

#layer 4

layer\_4 = Dense(units=4, activation='sigmoid')

model9.add(layer\_4)

#layer 5

layer\_5 = Dense(units=5, activation='sigmoid')

model9.add(layer\_5)

#layer 6

layer\_6 = Dense(units=5, activation='sigmoid')

model9.add(layer\_6)

#layer 7

layer\_7 = Dense(units=4, activation='sigmoid')

model9.add(layer\_7)

#layer 8

layer\_8 = Dense(units=3, activation='sigmoid')

model9.add(layer\_8)

#layer 9

layer\_9 = Dense(units=2, activation='sigmoid')

model9.add(layer\_9)

#layer 10

layer\_10 = Dense(units=1, activation='sigmoid')

model9.add(layer\_10)

print(model9.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

print("\n\n layer\_7 : ", layer\_7.input\_shape, layer\_7.output\_shape)

print("\n\n layer\_8 : ", layer\_8.input\_shape, layer\_8.output\_shape)

print("\n\n layer\_9 : ", layer\_9.input\_shape, layer\_9.output\_shape)

print("\n\n layer\_10 : ", layer\_10.input\_shape, layer\_10.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model9.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model9.fit(X\_10,y, epochs=10, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model9.predict(X\_10))

# good to test on new data, test data.

lst.append(model9.predict(X\_10)[0])

#keras10.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model10 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=1, activation='sigmoid', input\_dim = 11) ## change it to 3, 4, 5, 6, .. to see results

model10.add(layer\_1)

# layer 2

layer\_2 = Dense(units=2, activation='sigmoid')

model10.add(layer\_2)

#layer 3

layer\_3 = Dense(units=3, activation='sigmoid')

model10.add(layer\_3)

#layer 4

layer\_4 = Dense(units=4, activation='sigmoid')

model10.add(layer\_4)

#layer 5

layer\_5 = Dense(units=5, activation='sigmoid')

model10.add(layer\_5)

#layer 6

layer\_6 = Dense(units=6, activation='sigmoid')

model10.add(layer\_6)

#layer 7

layer\_7 = Dense(units=5, activation='sigmoid')

model10.add(layer\_7)

#layer 8

layer\_8 = Dense(units=4, activation='sigmoid')

model10.add(layer\_8)

#layer 9

layer\_9 = Dense(units=3, activation='sigmoid')

model10.add(layer\_9)

#layer 10

layer\_10 = Dense(units=2, activation='sigmoid')

model10.add(layer\_10)

#layer 11

layer\_11 = Dense(units=1, activation='sigmoid')

model10.add(layer\_11)

print(model10.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

print("\n\n layer\_7 : ", layer\_7.input\_shape, layer\_7.output\_shape)

print("\n\n layer\_8 : ", layer\_8.input\_shape, layer\_8.output\_shape)

print("\n\n layer\_9 : ", layer\_9.input\_shape, layer\_9.output\_shape)

print("\n\n layer\_10 : ", layer\_10.input\_shape, layer\_10.output\_shape)

print("\n\n layer\_11 : ", layer\_11.input\_shape, layer\_11.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model10.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model10.fit(X\_11,y, epochs=11, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model10.predict(X\_11))

# good to test on new data, test data.

lst.append(model10.predict(X\_11)[0])

#keras11.py

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras import optimizers

from tensorflow.keras.utils import plot\_model

import numpy as np

# define first architecture of the model

#

model11 = Sequential() # creates an empty sequential model

# layer 1

layer\_1 = Dense(units=1, activation='sigmoid', input\_dim = 12) ## change it to 3, 4, 5, 6, .. to see results

model11.add(layer\_1)

# layer 2

layer\_2 = Dense(units=2, activation='sigmoid')

model11.add(layer\_2)

#layer 3

layer\_3 = Dense(units=3, activation='sigmoid')

model11.add(layer\_3)

#layer 4

layer\_4 = Dense(units=4, activation='sigmoid')

model11.add(layer\_4)

#layer 5

layer\_5 = Dense(units=5, activation='sigmoid')

model11.add(layer\_5)

#layer 6

layer\_6 = Dense(units=6, activation='sigmoid')

model11.add(layer\_6)

#layer 7

layer\_7 = Dense(units=6, activation='sigmoid')

model11.add(layer\_7)

#layer 8

layer\_8 = Dense(units=5, activation='sigmoid')

model11.add(layer\_8)

#layer 9

layer\_9 = Dense(units=4, activation='sigmoid')

model11.add(layer\_9)

#layer 10

layer\_10 = Dense(units=3, activation='sigmoid')

model11.add(layer\_10)

#layer 11

layer\_11 = Dense(units=2, activation='sigmoid')

model11.add(layer\_11)

#layer 12

layer\_12 = Dense(units=1, activation='sigmoid')

model11.add(layer\_12)

print(model11.summary()) # to verify model structure

print("\n\n layer\_1 : ", layer\_1.input\_shape, layer\_1.output\_shape)

print("\n\n layer\_2 : ", layer\_2.input\_shape, layer\_2.output\_shape)

print("\n\n layer\_3 : ", layer\_3.input\_shape, layer\_3.output\_shape)

print("\n\n layer\_4 : ", layer\_4.input\_shape, layer\_4.output\_shape)

print("\n\n layer\_5 : ", layer\_5.input\_shape, layer\_5.output\_shape)

print("\n\n layer\_6 : ", layer\_6.input\_shape, layer\_6.output\_shape)

print("\n\n layer\_7 : ", layer\_7.input\_shape, layer\_7.output\_shape)

print("\n\n layer\_8 : ", layer\_8.input\_shape, layer\_8.output\_shape)

print("\n\n layer\_9 : ", layer\_9.input\_shape, layer\_9.output\_shape)

print("\n\n layer\_10 : ", layer\_10.input\_shape, layer\_10.output\_shape)

print("\n\n layer\_11 : ", layer\_11.input\_shape, layer\_11.output\_shape)

print("\n\n layer\_12 : ", layer\_12.input\_shape, layer\_12.output\_shape)

# tell Keras what loss,optimizer

opt = keras.optimizers.Adam(learning\_rate=0.01)

model11.compile(loss='mean\_squared\_error', optimizer =opt)

# at this stage, the model is not done yet.

# now for training data

# np.random.seed(9)

# X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])

# y = np.array([[0],[1],[1],[0]])

# training

model11.fit(X\_12,y, epochs=12, verbose=2, max\_queue\_size = 40)

# now the model is ready. you can use it for prediction

# we check it on training data

print(model11.predict(X\_12))

# good to test on new data, test data.

lst.append(model11.predict(X\_12)[0])

# regression analysis

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

X\_train = [2,3,4,5,6,7,8,10,11,12,13]

y\_train = [0.94913673, 0.9801116, 0.9924147, 0.9930438, 0.9895524, 0.99626696, 0.9925206, 0.99731386, 0.9982563, 0.97462773, 0.9958976]

X\_test = [2,3,4,5,6,7,8,10,11,12,13]

y\_test = [0.76626456, 0.84791243, 0.9827498, 0.988194, 0.99446905, 0.99014926, 0.98769546, 0.9972669, 0.9947778, 0.9973445, 0.99845743]

poly1 = np.poly1d(np.polyfit(X\_train, y\_train, 6))

poly2 = np.poly1d(np.polyfit(X\_test, y\_test, 6))

line = np.linspace(1, 22, 100)

plt.figure(figsize=(6,10))

plt.scatter(X\_train, y\_train, label='True Points')

plt.scatter(X\_test, y\_test, label='True Points')

plt.plot(line, poly1(line))

plt.plot(line, poly2(line))

plt.title('Comparison between Accuracies of 2 Types of Neuro-network Architecture')

plt.xlabel('Inputs')

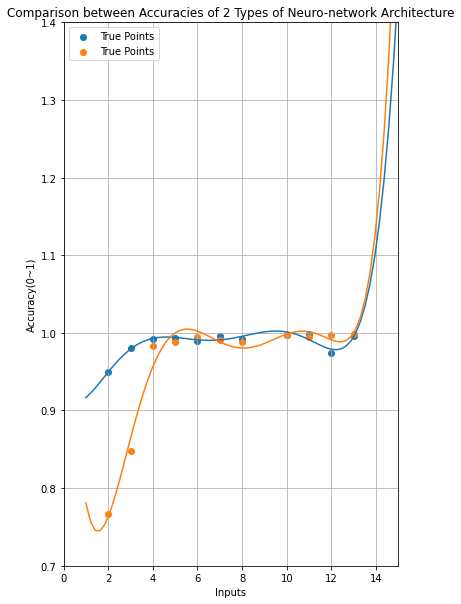
plt.ylabel('Accuracy(0~1)')

plt.axis([0, 15, 0.7, 1.4])

plt.grid(True)

plt.legend()

plt.show()



Epoch

So in this assignment I experimented with many kinds of architectures and make all the results on a graph with regression. The general trend is that with more inputs and epoch, the accuracy increase, but there is small flucturations in between.