(Y\_train, Z\_train), (Y\_test, Z\_test) = boston\_housing.load\_data()

print(Y\_train[0], Z\_train[0])

scl = StandardScaler()

scl.fit(Y\_train)

Y\_train\_scaled = scl.transform(Y\_train)

Y\_test\_scaled = scl.transform(Y\_test)

print(Y\_train\_scaled[0])

ml = models.Sequential()

ml.add(layers.Dense(8, activation='sigmoid', input\_shape=[Y\_train.shape[1]]))

ml.add(layers.Dense(16, activation='softmax'))

ml.add(layers.Dense(1))

keras.optimizers.SGD(lr=0.01)

ml.compile(optimizer=SGD(), loss='mse', metrics=['mae'])

tensorborad = TensorBoard(log\_dir="logs/**{}**".format(time()))

history = model.fit(Y\_train\_scaled, Z\_train, batch\_size=256, validation\_split=0.2, epochs=50, callbacks=[tensorborad])

print(ml.evaluate(X\_test\_scaled, y\_test))

**import** **tensorboard** **as** **tf**

dataset = pd.read\_csv('python/heart.csv',index\_col=0)

dataset.astype(float)

ds = dataset.max()

Z = ds['target']

Y = ds.drop(['target'], axis = 1)

Y\_train, Y\_test, Z\_train, Z\_test = train\_test\_split(Y, Z, test\_size = 0.33, random\_state = 0)

np.random.seed(155)

ml = Sequential()

ml.add(Dense(40, input\_dim=12, activation='relu'))

ml.add(Dense(20, input\_dim=40, activation='relu'))

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

ml.compile(loss= keras.losses.binary\_crossentropy,

optimizer=keras.optimizers.sgd(),

metrics=['accuracy'])

tensorborad = TensorBoard(log\_dir="logs/**{}**".format(time()))

his = model.fit(X\_train, Y\_train,batch\_size=256,epochs=20,verbose=1,

validation\_data=(Y\_test, Z\_test), callbacks=[tensorborad])

Z\_pred = ml.predict\_classes(Y\_test)

score = ml.evaluate(Y\_test, Z\_test, verbose=0)

print('Loss:', score[0])

print('Accuracy:', score[1])

D = pd.read\_csv("monkeylabels.txt")

print(D)

h1=150

w1=150

channels=3

size=32

seed=1337

Y\_dir = Path('training')

Z\_dir = Path('validation')

X\_datagen = ImageDataGenerator(rescale=1./255)

X\_generator = train\_datagen.flow\_from\_directory(X\_dir,

target\_size=(h1,w1),

batch\_size=size,

seed=seed,

class\_mode='categorical')

Y\_datagen = ImageDataGenerator(rescale=1./255)

Y\_generator = test\_datagen.flow\_from\_directory(Y\_dir,

target\_size=(h1,w1),

batch\_size=size,

seed=seed,

class\_mode='categorical')

ml = Sequential()

ml.add(Conv2D(32, kernel\_size=(3, 3),activation='relu',input\_shape=(150,150,3)))

ml.add(Flatten())

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

ml.add(Dropout(0.5))

ml.add(Dense(256,activation='relu'))

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

ml.compile(Adam(lr=0.0001),loss="categorical\_crossentropy", metrics=["accuracy"])

history = ml.fit\_generator(train\_generator,

steps\_per\_epoch= 1027/batch\_size,

epochs=2,

verbose=1,

validation\_data=test\_generator,

validation\_steps= 4)

ml.summary()

print(history.history.keys())

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

plt.title('Training, validation accuracy')

plt.plot(epochs, acc, 'red', label='Training acc')

plt.plot(epochs, val\_acc, 'blue', label='Validation acc')

plt.legend()

plt.figure()

plt.title('Training, validation loss')

plt.plot(epochs, loss, 'red', label='Training loss')

plt.plot(epochs, val\_loss, 'blue', label='Validation loss')

plt.legend()

plt.show()

ds = pd.read\_csv('test (1).tsv/train.tsv', sep='**\t**')

ds = pd.read\_csv('test (1).tsv/test.tsv', sep='**\t**')

ds = ds[['Phrase','Sentiment']]

ds['Phrase'] = ds['Phrase'].apply(**lambda** x: x.lower())

ds['Phrase'] = ds['Phrase'].apply((**lambda** x: re.sub('[^a-zA-z0-9\s]', '', x)))

**for** idx, row **in** ds.iterrows():

row[0] = row[0].replace('rt', ' ')

max\_fatures = 2000

tk = Tokenizer(num\_words=max\_fatures, split=' ')

tk.fit\_on\_texts(ds['Phrase'].values)

X = tk.texts\_to\_sequences(ds['Phrase'].values)

X = pad\_sequences(X)

d1 = data1[['Phrase']]

d1['Phrase'] = data1['Phrase'].apply(**lambda** x: x.lower())

d1['Phrase'] = data1['Phrase'].apply((**lambda** x: re.sub('[^a-zA-z0-9\s]', '', x)))

**for** idx, row **in** data.iterrows():

row[0] = row[0].replace('rt', ' ')

tk.fit\_on\_texts(data1['Phrase'].values)

Y = tk.texts\_to\_sequences(data1['Phrase'].values)

Y = pad\_sequences(Y)

Y = np.delete(Y, 0, 1)

embed\_dim = 64

lstm\_out = 196

**def** createmodel():

ml = Sequential()

ml.add(Embedding(max\_fatures, embed\_dim,input\_length = X.shape[1]))

ml.add(LSTM(lstm\_out, dropout=0.2, recurrent\_dropout=0.2))

ml.add(Dense(5,activation='softmax'))

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

**return** ml

labelencoder = LabelEncoder()

integer\_encoded = labelencoder.fit\_transform(data['Sentiment'])

x = to\_categorical(integer\_encoded)

y = to\_categorical(integer\_encoded)

batch\_size = 32

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, x, test\_size=0.25, random\_state=1000)

model = createmodel()

history=model.fit(X\_train,y\_train, epochs=2, verbose=**True**, validation\_data=(X\_test,y\_test), batch\_size=256)

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

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

print(history.history.keys())

plt.title('Accuracy-Loss')

plt.xlabel('accuracy')

plt.ylabel('loss')

plt.show()

ds = pd.read\_csv('/content/drive/My Drive/python/train.tsv', sep='**\t**')

ds1 = pd.read\_csv('/content/drive/My Drive/python/test.tsv', sep='**\t**')

*# Keeping only the neccessary columns*

ds = ds[['Phrase','Sentiment']]

ds['Phrase'] = ds['Phrase'].apply(**lambda** x: x.lower())

ds['Phrase'] = ds['Phrase'].apply((**lambda** x: re.sub('[^a-zA-z0-9\s]', '', x)))

**for** idx, row **in** ds.iterrows():

row[0] = row[0].replace('rt', ' ')

max\_fatures = 2000

tk = Tokenizer(num\_words=max\_fatures, split=' ')

tk.fit\_on\_texts(data['Phrase'].values)

X = tk.texts\_to\_sequences(data['Phrase'].values)

X = pad\_sequences(X)

data1 = data1[['Phrase']]

data1['Phrase'] = data1['Phrase'].apply(**lambda** x: x.lower())

data1['Phrase'] = data1['Phrase'].apply((**lambda** x: re.sub('[^a-zA-z0-9\s]', '', x)))

**for** idx, row **in** data.iterrows():

row[0] = row[0].replace('rt', ' ')

tk.fit\_on\_texts(data1['Phrase'].values)

Y = tk.texts\_to\_sequences(data1['Phrase'].values)

Y = pad\_sequences(Y)

Y = np.delete(Y, 0, 1)

embed\_dim = 64

lstm\_out = 196

**def** createmodel():

ml = Sequential()

ml.add(Embedding(max\_fatures, embed\_dim,input\_length = X.shape[1]))

ml.add(LSTM(lstm\_out, dropout=0.2, recurrent\_dropout=0.2))

ml.add(Dense(5,activation='softmax'))

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

**return** ml

labelencoder = LabelEncoder()

integer\_encoded = labelencoder.fit\_transform(data['Sentiment'])

x = to\_categorical(integer\_encoded)

y = to\_categorical(integer\_encoded)

batch\_size = 32

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, x, test\_size=0.25, random\_state=1000)

ml = createmodel()

history=model.fit(X\_train,y\_train, epochs=2, verbose=**True**, validation\_data=(X\_test,y\_test), batch\_size=256)

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

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

print(history.history.keys())

plt.title('Accuracy-Loss')

plt.xlabel('accuracy')

plt.ylabel('loss')

plt.show()

print(ml.metrics\_names)

encoding\_dim = 32

img = Input(shape=(784,))

encoded = Dense(encoding\_dim, activation='relu')(img)

decoded = Dense(784, activation='sigmoid')(encoded)

autoencoder = Model(input\_img, decoded)

encoder = Model(img, encoded)

encoded\_input = Input(shape=(encoding\_dim,))

decoder\_layer = autoencoder.layers[-1]

decoder = Model(encoded\_input, decoder\_layer(encoded\_input))

autoencoder.compile(optimizer='adadelta', loss='binary\_crossentropy', metrics=['accuracy'])

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

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

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

Y\_train = Y\_train.reshape((len(Y\_train), np.prod(Y\_train.shape[1:])))

Y\_test = Y\_test.reshape((len(Y\_test), np.prod(Y\_test.shape[1:])))

noise\_factor = 0.5

Y\_train\_noisy = Y\_train + noise\_factor \* np.random.normal(loc=0.0, scale=1.0, size=Y\_train.shape)

Y\_test\_noisy = Y\_test + noise\_factor \* np.random.normal(loc=0.0, scale=1.0, size=x\_test.shape)

tensorboard = TensorBoard(log\_dir='2', histogram\_freq=0, write\_graph=**True**, write\_images=**False**)

history = autoencoder.fit(Y\_train\_noisy, Y\_train,

epochs=20,

batch\_size=256,

shuffle=**True**,

validation\_data=(Y\_test\_noisy, Y\_test\_noisy), callbacks=[tensorboard])

encoded\_imgs = encoder.predict(Y\_test)

decoded\_imgs = decoder.predict(encoded\_imgs)

n = 10

plt.figure(figsize=(20, 4))

**for** i **in** range(n):

ax = plt.subplot(3, n, i + 1)

plt.imshow(Y\_test[i].reshape(28, 28))

plt.gray()

ax.get\_xaxis().set\_visible(**False**)

ax.get\_yaxis().set\_visible(**False**)

ax = plt.subplot(3, n, i + 1 + n)

plt.imshow(Y\_test\_noisy[i].reshape(28, 28))

plt.gray()

ax.get\_xaxis().set\_visible(**False**)

ax.get\_yaxis().set\_visible(**False**)

ax = plt.subplot(3, n, i + 1 + n + n)

plt.imshow(decoded\_imgs[i].reshape(28, 28))

plt.gray()

ax.get\_xaxis().set\_visible(**False**)

ax.get\_yaxis().set\_visible(**False**)

plt.show()

fig1 = plt.figure()

plt.plot(history.history['loss'], 'r', linewidth=3.0)

plt.plot(history.history['val\_loss'], 'b', linewidth=3.0)

plt.legend(['Training loss', 'Validation Loss'], fontsize=18)

plt.xlabel('Epochs ', fontsize=16)

plt.ylabel('Loss', fontsize=16)

plt.title('Loss Curves : ', fontsize=16)

fig1.savefig('loss\_lstm.png')

fig2 = plt.figure()

plt.plot(history.history['accuracy'], 'r', linewidth=3.0)

plt.plot(history.history['val\_accuracy'], 'b', linewidth=3.0)

plt.legend(['Training acc', 'Validation acc'], fontsize=18)

plt.xlabel('Epochs ', fontsize=16)

plt.ylabel('Accuracy', fontsize=16)

plt.title('Accuracy Curves : ', fontsize=16)

fig2.savefig('acc\_lstm.png')