1. Py

df = pd.read\_csv('crimedata.csv')

data = pd.DataFrame(df, columns=["crim","zn","indus","chas","nox","rm","age","dis","rad","tax","ptratio","b","lstat","medv"])

label\_col = 'medv'

print(data.describe())

x=data.iloc[:,0:13]

y=data.iloc[:,13]

x\_train, x\_valid, y\_train, y\_valid = train\_test\_split(x, y, test\_size=0.3, random\_state=87)

np.random.seed(155)

def model1(x\_size, y\_size):

model = Sequential()

model.add(Dense(100, activation="sigmoid", input\_shape=(x\_size,)))

model.add(Dropout(0.1))

model.add(Dense(50, activation="relu"))

model.add(Dense(20, activation="relu"))

model.add(Dense(y\_size))

print(model.summary())

keras.optimizers.SGD(lr=0.1)

model.compile(loss='mean\_squared\_error', optimizer=SGD(), metrics=[metrics.mae])

return(model)

Model = model1(x\_train.shape[1], 1)

Model.summary()

hist = Model.fit(x\_train, y\_train, batch\_size=64, epochs=5, shuffle=True, verbose=0, validation\_data=(x\_valid, y\_valid), callbacks=[TensorBoardColabCallback(tbc)])

train\_score = Model.evaluate(x\_train, y\_train, verbose=0)

valid\_score = Model.evaluate(x\_valid, y\_valid, verbose=0)

print('Train MAE: ', round(train\_score[1], 4), ', Train Loss: ', round(train\_score[0], 4))

print('Val MAE: ', round(valid\_score[1], 4), ', Val Loss: ', round(valid\_score[0], 4))

2.py

df = pd.read\_csv("heart.csv", sep=',')

df.astype(float)

# Normalize values to range [0:1]

df /= df.max()

# split data into features & target columns

x= df.drop(columns = 'target')

y = df['target']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, train\_size = 0.75, test\_size = 0.25)

y\_train = np\_utils.to\_categorical(y\_train, 2)

y\_test = np\_utils.to\_categorical(y\_test, 2)

# Creating the model

model = Sequential()

model.add(Dense(1024, input\_shape=(13,), kernel\_regularizer = regularizers.l2( 0.01)))

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(1024))

model.add(Activation('tanh'))

model.add(Dropout(0.5))

model.add(Dense(1024))

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(2))

model.add(Activation('sigmoid'))

model.summary()

# compile the model

model.compile(loss='categorical\_crossentropy', optimizer=SGD(), metrics=['accuracy', keras\_metrics.precision(), keras\_metrics.recall()])

# train the model

history = model.fit(x\_train, y\_train,batch\_size=256, epochs=50,verbose=0, validation\_split=0.25,callbacks=[TensorBoardColabCallback(tbc)])

# make prediction

y\_pred = model.predict\_classes(x\_test)

score = model.evaluate(x\_test, y\_test, verbose=0)

3.py

DATA\_FILE = 'Spam\_Data.csv'

df = pd.read\_csv(DATA\_FILE,encoding='latin-1')

print(df.head())

tags = df.Category

texts = df.Message

num\_max = 2000

# preprocess

le = LabelEncoder()

cat = le.fit\_transform(df.Category)

tok = Tokenizer(num\_words=num\_max)

tok.fit\_on\_texts(df.Message)

mat\_texts = tok.texts\_to\_matrix(texts,mode='count')

print(cat[:5])

print(mat\_texts[:5])

print(tags.shape,mat\_texts.shape)

max\_len = 100

cnn\_texts\_seq = tok.texts\_to\_sequences(texts)

print(cnn\_texts\_seq[0])

cnn\_texts\_mat = sequence.pad\_sequences(cnn\_texts\_seq,maxlen=max\_len)

print(cnn\_texts\_mat[0])

print(cnn\_texts\_mat.shape)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(cnn\_texts\_mat, cat, random\_state=42, test\_size=.1)

model = Sequential()

model.add(Embedding(2000,20,input\_length=max\_len))

model.add(Dropout(0.2))

model.add(Conv1D(128, 3, padding='same', activation='relu', kernel\_constraint=maxnorm(3)))

model.add(GlobalMaxPooling1D())

model.add(Dense(128))

model.add(Dropout(0.2))

model.add(Activation('relu'))

# model.add(Dense(1))

# model.add(Activation('sigmoid'))

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

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

model.fit(X\_train, y\_train, epochs = 5, batch\_size=32, verbose = 1)

# model.fit(X\_train, Y\_train, epochs = 3, batch\_size=batch\_size, verbose = 2, callbacks=[TensorBoardColabCallback(tbc)])

score,acc = model.evaluate(X\_test,y\_test,verbose=1,batch\_size=32)

print(score)

print(acc)

print(model.metrics\_names)

4.py

data = pd.read\_csv('Spam\_Data.csv',encoding='latin-1')

# Keeping only the neccessary columns

data = data[['Message','Category']]

data['Message'] = data['Message'].apply(lambda x: x.lower())

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

for idx, row in data.iterrows():

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

max\_fatures = 2000

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

tokenizer.fit\_on\_texts(data['Message'].values)

X = tokenizer.texts\_to\_sequences(data['Message'].values)

X = pad\_sequences(X)

embed\_dim = 128

lstm\_out = 196

def createmodel():

model = Sequential()

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

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

model.add(Dense(2,activation='softmax'))

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

return model

# print(model.summary())

labelencoder = LabelEncoder()

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

y = to\_categorical(integer\_encoded)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X,y, test\_size = 0.33, random\_state = 42)

batch\_size = 32

model = createmodel()

model.fit(X\_train, Y\_train, epochs = 3, batch\_size=batch\_size, verbose = 2)

# model.fit(X\_train, Y\_train, epochs = 3, batch\_size=batch\_size, verbose = 2, callbacks=[TensorBoardColabCallback(tbc)])

score,acc = model.evaluate(X\_test,Y\_test,verbose=2,batch\_size=batch\_size)

print(score)

print(acc)

print(model.metrics\_names)