DEEP LEARNING

Assign 1

import pandas as pd

df=pd.read\_csv("Boston.csv")

df

df.isnull().sum()

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

y=df.iloc[:,-1]

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3)

from sklearn.preprocessing import MinMaxScaler

mms=MinMaxScaler()

mms.fit(x\_train)

x\_train=mms.transform(x\_train)

x\_test=mms.transform(x\_test)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

model=Sequential()

model.add(Dense(128, activation='relu', name='Input'))

model.add(Dense(64,activation='relu', name='hidden'))

model.add(Dense(1,activation='linear', name='dense\_output'))

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

model.summary()

history = model.fit(x\_train, y\_train, epochs=10, validation\_split=0.05, verbose = 1) history = model.fit(x\_train, y\_train, epochs=10, validation\_split=0.05, verbose = 1)

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

Assign3(imbd)

from tensorflow.keras.datasets import imdb

import numpy as np

(x\_train, y\_train), (x\_test, y\_test) = imdb.load\_data(num\_words = 10000)

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense,Embedding, Flatten, Dense, Dropout

max\_length = 200

x\_train = pad\_sequences(x\_train, maxlen=max\_length)

x\_test = pad\_sequences(x\_test, maxlen=max\_length)

model = Sequential([

Embedding(input\_dim=10000, output\_dim=128),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.5),

Dense(64, activation='relu'),

Dropout(0.5),

Dense(1, activation='sigmoid')

])

model.summary()

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

history = model.fit(x\_train, y\_train, epochs=5, batch\_size=32, validation\_split=0.2)

loss, accuracy = model.evaluate(x\_test, y\_test)

loss

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

Assign 2(multiclass)

import pandas as pd

df=pd.read\_csv("letter-recognition.data")

x=df.iloc[:,1:17]

y=df.iloc[:,0]

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3,random\_state=10)

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

x\_train=sc.fit\_transform(x\_train)

x\_test=sc.transform(x\_test)

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

y\_train = le.fit\_transform(y\_train)

y\_test = le.transform(y\_test)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

model = Sequential()

model.add(Dense(128, activation='relu', input\_dim=x\_train.shape[1]))

model.add(Dense(64, activation='relu'))

model.add(Dense(26, activation='softmax'))

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

model.fit(x\_train, y\_train, epochs=10, batch\_size=32, validation\_data=(x\_test, y\_test))

test\_loss, test\_accuracy = model.evaluate(x\_test, y\_test)

print(f'Test Accuracy: {test\_accuracy}')

import numpy as np

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

y\_pred\_classes = [np.argmax(element) for element in y\_pred]

y\_pred\_letters = le.inverse\_transform(y\_pred\_classes)

from tensorflow.keras.datasets import fashion\_mnist

import os

os.environ['TF\_ENABLE\_ONEDNN\_OPTS'] = '0'

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

import matplotlib.pyplot as plt

plt.imshow(x\_train[1])

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense,Flatten,MaxPooling2D,Conv2D

model=Sequential()

model.add(Conv2D(filters=64,activation='relu',kernel\_size=(3,3),input\_shape=(28,28,1)))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Flatten())

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

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

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

model.fit(x\_train, y\_train, epochs=15, batch\_size=64, validation\_split=0.2)

labels = ['t\_shirt', 'trouser', 'pullover', 'dress', 'coat', 'sandal', 'shirt', 'sneaker', 'bag', 'ankle\_boots']

predictions = model.predict(x\_test[:1])

import numpy as np

label = labels[np.argmax(predictions)]

import matplotlib.pyplot as plt

print(label)

plt.imshow(x\_test[:1][0])

plt.show()

Assign 3(plant)

import tensorflow as tf

import os

import cv2

import pickle

from tqdm import tqdm

import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

def extract\_images\_from\_folder(root\_folder, target\_size=(256, 256)):

image\_data = []

labels = []

label\_to\_index = {} # Dictionary to map label names to numerical indices

index = 0

for subdir, dirs, files in os.walk(root\_folder):

for file in tqdm(files):

filepath = os.path.join(subdir, file)

img = cv2.imread(filepath)

img = cv2.resize(img, target\_size)

label = os.path.basename(subdir)

if label not in label\_to\_index:

label\_to\_index[label] = index

index += 1

image\_data.append(img)

labels.append(label\_to\_index[label])

image\_data = np.array(image\_data)

labels = np.array(labels)

return image\_data, labels, label\_to\_index

x\_train = x\_train / 255.0

x\_test = x\_test / 255.0

x\_val = x\_val / 255.0

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(256, 256, 3)),

MaxPooling2D((2, 2)),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D((2, 2)),

Conv2D(128, (3, 3), activation='relu'),

MaxPooling2D((2, 2)),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.5),

Dense(len(label\_reverse\_train), activation='softmax')

])

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

history = model.fit(x\_train, y\_train, epochs=2, batch\_size=32, validation\_data=(x\_val, y\_val))

Assign 4 (rnn)

import numpy as np

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Dropout

import matplotlib.pyplot as plt

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

data = df['Close'].values.reshape(-1, 1)

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(data)

def create\_sequences(data, seq\_length):

X, y = [], []

for i in range(len(data) - seq\_length):

X.append(data[i:i+seq\_length])

y.append(data[i+seq\_length])

return np.array(X), np.array(y)

sequence\_length = 20

X, y = create\_sequences(scaled\_data, sequence\_length)

train\_size = int(len(X) \* 0.8)

X\_train, X\_test = X[:train\_size], X[train\_size:]

y\_train, y\_test = y[:train\_size], y[train\_size:]

model = Sequential([

LSTM(50, input\_shape=(X\_train.shape[1], X\_train.shape[2])),

Dropout(0.2),

Dense(1)

])

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

history = model.fit(X\_train, y\_train, epochs=100, batch\_size=32, validation\_data=(X\_test, y\_test), verbose=1)

loss = model.evaluate(X\_test, y\_test)

print('Test Loss:', loss)

predictions = model.predict(X\_test)

predictions = scaler.inverse\_transform(predictions)

y\_test = scaler.inverse\_transform(y\_test)

plt.figure(figsize=(12, 6))

plt.plot(predictions, label='Predicted')

plt.plot(y\_test, label='Actual')

plt.xlabel('Time')

plt.ylabel('Stock Price')

plt.title('Google Stock Price Prediction')

plt.legend()

plt.show()