**1.First it mounts Google Drive to the Colab notebook using the Google Colab library:**

from google.colab import drive

drive.mount("/content/drive/")

**2.The code then imports necessary libraries for data preprocessing and model training:**

import numpy as np

import scipy

from sklearn.preprocessing import normalize

from sklearn import svm

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

import keras

from keras.models import Sequential

from keras.layers import LSTM, Dense

**3.Next, the code loads various data files using np.load() function from NumPy:**

train\_path = "/content/drive/MyDrive/Colab Notebooks/bbh/training/"

test\_path = "/content/drive/MyDrive/Colab Notebooks/bbh/testing/"

train\_labels = np.load(train\_path + "trainLabels.npy")

train\_ms\_acc = np.load(train\_path + "trainMSAccelerometer.npy")

train\_ms\_gyro = np.load(train\_path + "trainMSGyroscope.npy")

train\_jins\_acc = np.load(train\_path + "trainJinsAccelerometer.npy")

train\_jins\_gyro = np.load(train\_path + "trainJinsGyroscope.npy")

train\_acc = np.load(train\_path + "trainAccelerometer.npy")

train\_gravity = np.load(train\_path + "trainGravity.npy")

train\_gyro = np.load(train\_path + "trainGyroscope.npy")

train\_lin\_acc = np.load(train\_path + "trainLinearAcceleration.npy")

train\_mag = np.load(train\_path + "trainMagnetometer.npy")

test\_labels = np.load(test\_path + "testLabels.npy")

test\_ms\_acc = np.load(test\_path + "testMSAccelerometer.npy")

test\_ms\_gyro = np.load(test\_path + "testMSGyroscope.npy")

test\_jins\_acc = np.load(test\_path + "testJinsAccelerometer.npy")

test\_jins\_gyro = np.load(test\_path + "testJinsGyroscope.npy")

test\_acc = np.load(test\_path + "testAccelerometer.npy")

test\_gravity = np.load(test\_path + "testGravity.npy")

test\_gyro = np.load(test\_path + "testGyroscope.npy")

test\_lin\_acc = np.load(test\_path + "testLinearAcceleration.npy")

test\_mag = np.load(test\_path + "testMagnetometer.npy")

**4.The code then defines functions for data normalization and feature extraction.**

**Next, the code performs normalization on the loaded data arrays:**

norm\_train\_ms\_acc = normalization(train\_ms\_acc)

norm\_train\_ms\_gyro = normalization(train\_ms\_gyro)

norm\_train\_jins\_acc = normalization(train\_jins\_acc)

norm\_train\_jins\_gyro = normalization(train\_jins\_gyro)

norm\_train\_acc = normalization(train\_acc)

norm\_train\_gravity = normalization(train\_gravity)

norm\_train\_gyro = normalization(train\_gyro)

norm\_train\_lin\_acc = normalization(train\_lin\_acc)

norm\_train\_mag = normalization(train\_mag)

norm\_test\_ms\_acc = normalization(test\_ms\_acc)

norm\_test\_ms\_gyro = normalization(test\_ms\_gyro)

norm\_test\_jins\_acc = normalization(test\_jins\_acc)

norm\_test\_jins\_gyro = normalization(test\_jins\_gyro)

norm\_test\_acc = normalization(test\_acc)

norm\_test\_gravity = normalization(test\_gravity)

norm\_test\_gyro = normalization(test\_gyro)

norm\_test\_lin\_acc = normalization(test\_lin\_acc)

norm\_test\_mag = normalization(test\_mag)

)

**5.The code defines functions for segmentation and feature extraction.**

**Segmentation is performed on the normalized data arrays:**

feature\_train\_ms\_acc = get\_features(norm\_train\_ms\_acc, window\_size, stride\_size)

feature\_train\_ms\_gyro = get\_features(norm\_train\_ms\_gyro, window\_size, stride\_size)

feature\_train\_jins\_acc = get\_features(norm\_train\_jins\_acc, window\_size, stride\_size)

feature\_train\_jins\_gyro = get\_features(norm\_train\_jins\_gyro, window\_size, stride\_size)

feature\_train\_acc = get\_features(norm\_train\_acc, window\_size, stride\_size)

feature\_train\_gravity = get\_features(norm\_train\_gravity, window\_size, stride\_size)

feature\_train\_gyro = get\_features(norm\_train\_gyro, window\_size, stride\_size)

feature\_train\_lin\_acc = get\_features(norm\_train\_lin\_acc, window\_size, stride\_size)

feature\_train\_mag = get\_features(norm\_train\_mag, window\_size, stride\_size)

feature\_test\_ms\_acc = get\_features(norm\_test\_ms\_acc, window\_size, stride\_size)

feature\_test\_ms\_gyro = get\_features(norm\_test\_ms\_gyro, window\_size, stride\_size)

feature\_test\_jins\_acc = get\_features(norm\_test\_jins\_acc, window\_size, stride\_size)

feature\_test\_jins\_gyro = get\_features(norm\_test\_jins\_gyro, window\_size, stride\_size)

feature\_test\_acc = get\_features(norm\_test\_acc, window\_size, stride\_size)

feature\_test\_gravity = get\_features(norm\_test\_gravity, window\_size, stride\_size)

feature\_test\_gyro = get\_features(norm\_test\_gyro, window\_size, stride\_size)

feature\_test\_lin\_acc = get\_features(norm\_test\_lin\_acc, window\_size, stride\_size)

feature\_test\_mag = get\_features(norm\_test\_mag, window\_size, stride\_size)

**6.Finally, the extracted features are combined into a single feature matrix:**

feature\_train = feature\_train\_ms\_acc

feature\_train = np.hstack((feature\_train, feature\_train\_ms\_gyro))

feature\_train = np.hstack((feature\_train, feature\_train\_jins\_acc))

feature\_train = np.hstack((feature\_train, feature\_train\_jins\_gyro))

feature\_train = np.hstack((feature\_train, feature\_train\_acc))

feature\_train = np.hstack((feature\_train, feature\_train\_gravity))

feature\_train = np.hstack((feature\_train, feature\_train\_gyro))

feature\_train = np.hstack((feature\_train, feature\_train\_lin\_acc))

feature\_train = np.hstack((feature\_train, feature\_train\_mag))

feature\_test = feature\_test\_ms\_acc

feature\_test = np.hstack((feature\_test, feature\_test\_ms\_gyro))

feature\_test = np.hstack((feature\_test, feature\_test\_jins\_acc))

feature\_test = np.hstack((feature\_test, feature\_test\_jins\_gyro))

feature\_test = np.hstack((feature\_test, feature\_test\_acc))

feature\_test = np.hstack((feature\_test, feature\_test\_gravity))

feature\_test = np.hstack((feature\_test, feature\_test\_gyro))

feature\_test = np.hstack((feature\_test, feature\_test\_lin\_acc))

feature\_test = np.hstack((feature\_test, feature\_test\_mag))

print(feature\_train.shape)

print(feature\_test.shape)

**7.Then if performs classification on it**

from sklearn import svm

from sklearn.impute import SimpleImputer

# Create an imputer object

imputer = SimpleImputer(strategy='mean')

# Fit the imputer on the training data

imputer.fit(feature\_train)

# Transform the training data

feature\_train\_imputed = imputer.transform(feature\_train)

# Create an SVM classifier with linear kernel

classification = svm.SVC(kernel='linear')

# Fit the classifier on the imputed training data

classification.fit(feature\_train\_imputed, trainLabels)

**8.and at the end it checks the model how accurate and good it is working**

**# Transform the testing data using the imputer**

feature\_test\_imputed = imputer.transform(feature\_test)

# Evaluate the classifier on the imputed testing data

measurement\_of\_standard = classification.score(feature\_test\_imputed, testLabels)

print("How far it is good?:", measurement\_of\_standard)