Main.py

import pandas as pd

import numpy as np

from sklearn.metrics import accuracy\_score

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

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

from sklearn.preprocessing import MinMaxScaler

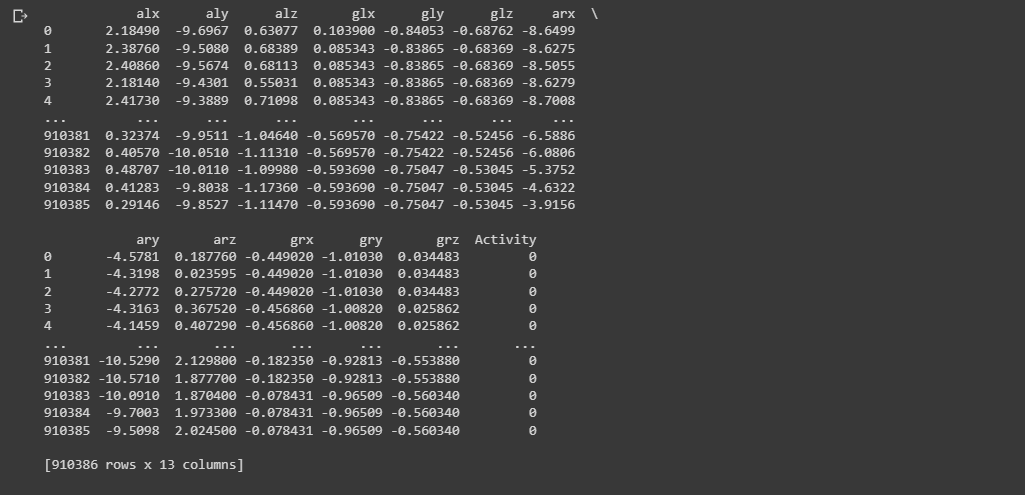
import pickle

import warnings

warnings.filterwarnings('ignore')

a=pd.read\_csv("Gait\_Data.csv")

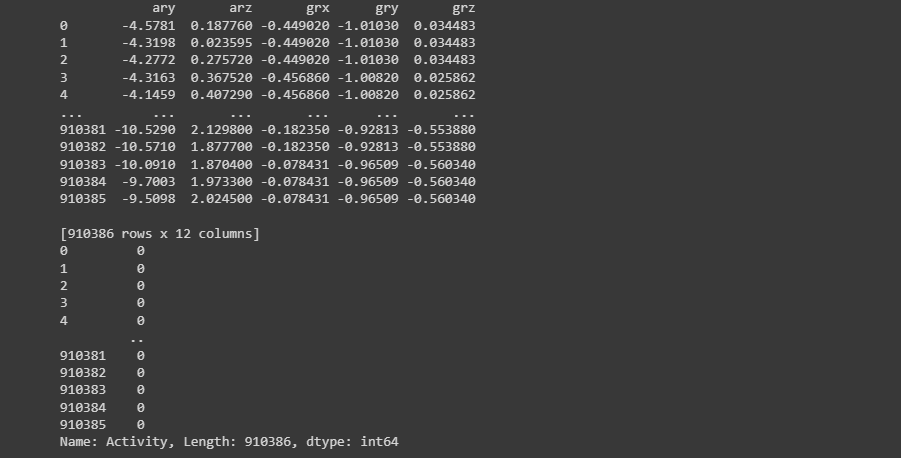
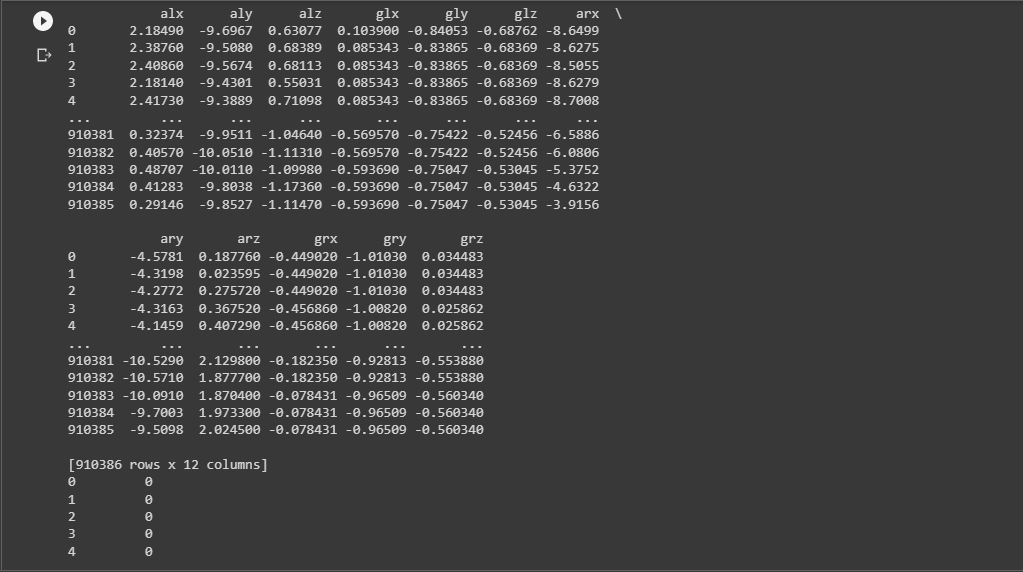
print(a)



############################ features ####################################

X=a.drop(['Activity'],axis=1)

print(X)



############################ labels ######################################

Y=a['Activity']

print(Y)

############################# traing and testing part #######################

x\_train,x\_test,y\_train,y\_test = train\_test\_split(X,Y,shuffle=True,test\_size=0.25, random\_state=0)

############################# Algorithm Implementation #######################

############################# Gaussian Naive Bayes #######################

from sklearn.naive\_bayes import GaussianNB

NB = GaussianNB()

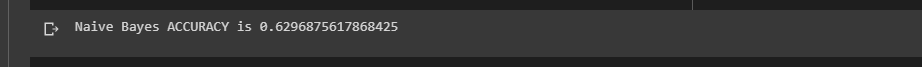
NB.fit(x\_train, y\_train) #train the data

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

##print(y\_pred)

##print(y\_test)

print('Naive Bayes ACCURACY is', accuracy\_score(y\_test,y\_pred))



############################# RandomForestClassifier #######################

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(max\_depth=2, random\_state=0)

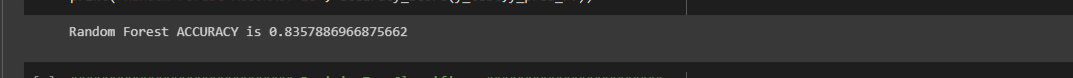
clf.fit(x\_train, y\_train) #train the data

y\_pred\_rf=clf.predict(x\_test)

##print(y\_pred)

##print(y\_test)

print('Random Forest ACCURACY is', accuracy\_score(y\_test,y\_pred\_rf))



############################### KNN ############################################

from sklearn.neighbors import KNeighborsClassifier

# Create KNN classifier with k=3

knn = KNeighborsClassifier(n\_neighbors=3)

# Train the classifier

knn.fit(X\_train, y\_train)

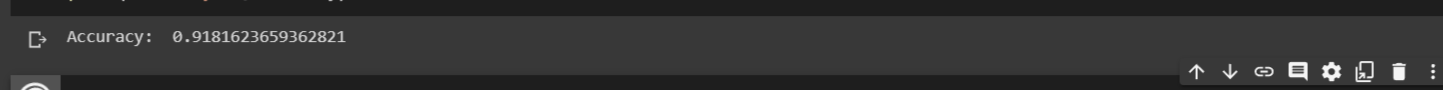
# Make predictions on the test set

y\_pred = knn.predict(X\_test)

# Calculate accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy: ", accuracy)



##############################3#### SVM ####################################

from sklearn import svm

# Create a SVM classifier with a linear kernel

clf = svm.SVC(kernel='linear', C=1)

# Train the classifier on the training data

clf.fit(x\_train, y\_train)

# Make predictions on the testing data

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

# Evaluate the accuracy of the classifier

accuracy = accuracy\_score(y\_test, y\_pred)

print('SVM Accuracy:', accuracy)

############################# DecisionTreeClassifier #######################

from sklearn.tree import DecisionTreeClassifier

DT = DecisionTreeClassifier(random\_state=0)

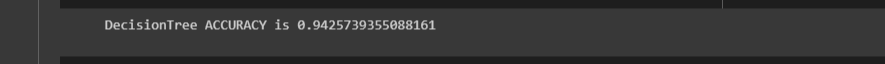
DT.fit(x\_train, y\_train) #train the data

y\_pred\_DT=DT.predict(x\_test)

##print(y\_pred)

##print(y\_test)

print('DecisionTree ACCURACY is', accuracy\_score(y\_test,y\_pred\_DT))



from keras.models import Sequential

from keras.layers import LSTM, Dense

model\_LSTM = Sequential()

model\_LSTM.add(LSTM(32, input\_shape=(12,1)))

model\_LSTM.add(Dense(16, activation='relu'))

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

# Compile the model

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

# Train the model

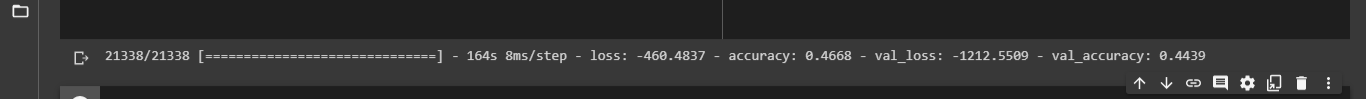
history = model\_LSTM.fit(x\_train, y\_train, epochs=1, batch\_size=32, validation\_data=(x\_test, y\_test))

# Evaluate the model

test\_loss, test\_acc = model\_LSTM.evaluate(x\_train, Y\_test)

##print("Test Loss:", test\_loss)

print("Test Accuracy:", test\_acc)



filename = 'model.pkl'

#pickle.dump(DT, open(filename, 'wb'))