BIG DATA ANALYTICS PROJECT

GROUP MEMBERS
HUSNAIN BUKHARI 20I-0626
ABDULLAH KHAN 20I-0894
ANUSHA ZUBAIR 20I-2454

This project had three phases starting with generating sensor data from the Accelerometer and Gyroscope of our phones and labeling this data.This was to be done by creating a basic Android or IOS app that collects the required data via these sensors and upload it to our Laptop via Flask Api or Bluetooth.After that this labeled data was to be stored in a database whether that be MongoDB or MYSQL.The next phase included ingesting the data in to the Kafka Environment and then processing and classifying the Labels by using ML Classification model.The last stage included implementing a frontend website using flask or any-other framework which would take Live data from our Phone and predict the position or state of the Phone and later show the readings of Accelerometer and Gyroscope on the Website along with the Predicted State.

For the collection of data we developed an app using flutter and some opensource flutter api code for the implementation(refrences given at the end).The app records the X,Y and Z coordinates of both gyroscopeand accelerometer.The app takes a server address from the user and hits the data onto that server , meanwhile a Flask api has been implemented that takes the lives data being hit on that server as both the laptop and the flutter application are on the same network (in our case samew wifi connection). This is is processed , group name is added and the result is stored is a json file where the parameters of both the sensors are in a json string list.The user just had to hardcode the file name which will later be used as a label.

Once we had the data in json files we had the job to upload the data to our choosen database that in our case was MongoDB. We used the concept of json file parsing as taught to us during the semester to parse the json file extract our required information and then post in on the database along with its assigned label.Below is a snippet of the code we used on each of our files to upload our data to MongoDB

```
import json
import pandas as pd
import pymongo
from pymongo import MongoClient
cluster=MongoClient()
db=cluster["project"]
collection=db["data"]

with open("walking.json", "r") as read_file:
        data = json.load(read_file)
prop=["accelerometer","gyroscope"]
a=json.loads(data[0]["accelerometer"])
acc_x=[]
acc_y=[]
acc_z=[]
gyr_x=[]
gyr_y=[]
gyr_z=[]
lab=[]

for i in data:
    temp=json.loads(i["accelerometer"])
    temp2=json.loads(i["gyroscope"])
    a1=temp[0]
    a2=temp[1]
    a3=temp[2]
    b1=temp2[0]
    b2=temp2[1]
    b3=temp2[2]

    acc_x.append(a1)
    acc_y.append(a2)
    acc_z.append(a3)
```

```
    gyr_x.append(b1)
    gyr_y.append(b2)
    gyr_z.append(b3)
    lab.append(1)
    dict1={"acc_x":a1,"acc_y":a2,"acc_z":a3,"gyr_x":b1,"gyr_y":b2,"gyr_z":b3,"Lable":1}
    collection.insert_one(dict1)
    dict1.clear()

df1 = pd.DataFrame(list(zip(acc_x,acc_y,acc_z, gyr_x,gyr_y,gyr_z,lab)), columns
=["acc_x","acc_y","acc_z", "gyr_x","gyr_y","gyr_z","lable"])
```

Once the data was successfully uploaded on Mongodb we read the data from our database to a panda dataframe using the following code

```
#load data from mongodb directly
import pymongo
import pandas as pd
from pymongo import MongoClient
client = MongoClient()
db = client.project
collection = db.data
finaldf = pd.DataFrame(list(collection.find()))
```

Once data was retrieved from the database we then split the data into testing and training data and applied standard scaler so that it is easier for the model to learn and understand the problem

```
from sklearn.model_selection import train_test_split
X = finaldf.drop("Lable", axis = 1)
y = finaldf["Lable"]
X_train, X_test, y_train, y_test =train_test_split(X, y, test_size=0.2, random_state=51)
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
sc = StandardScaler()
sc.fit(X_train)
X_train_sc = sc.transform(X_train)
X_test_sc = sc.transform(X_test)
```

After scaling our data we trained many ml classification models like KNN, Decision Tree,Naive Bayes,Logistic Regression,SGDC,SVM ,Random Forest, and Adaboost

But after checking all these models we saw that the best performing classifier was Decision tree and so we continued with it.We saved the trained model in a pickel file to be later used in the kafka environment

```
from sklearn import tree
import pickle
dcf = tree.DecisionTreeClassifier()
dcf = dcf.fit(X_train, y_train)
p=dcf.predict(X_test)
pickle.dump(dcf, open("dcf.pkl", "wb"))
dcf.score(X_test,y_test)
```

1. Getting individual sensor values from the flask API implemented in Producer and sending it to the Consumer using json.dump.
2. Looping through the data in the consumer.
3. Formatting data to store into different variables to be used for prediction.
4. Creating a data frame using the data and sending the data through the model trained beforehand (imported using pickle.load).
5. Mapping the result into labels using labels list created in the start.
6. Sending the sensor values and labels to the Flask API as a dictionary.
7. First thread is initiated into the (before first request) function.
8. Calling the inject function to start getting values from the consumer.
9. Update load function implemented using turbo which sends the updated data to the template sensors.html file.
10. Index.html file inherits the sensors.html file and is updated itself when turbo.push is called infinitely in update load function.
11. Live sensor values and predicted labels are displayed in the flask web app.
12. Styling is done within the HTML pages as well as the css stylesheet in the static folder.
13. Updating load with sleep time of 1 second to see the actual output and not letting it change constantly so that it isn't even readable.
14. Index.html file displays static message and inherits sensors.html which is a table of values dynamically updating constantly.