SMART FIT SHIRT – WEARABLE COACH FOR OPTIMAL EXERCISE PERFORMANCE

PROBLEM DESCRIPTION:

Spinal cord injuries (SCI) often lead to profound loss of mobility and sensation, underscoring the critical need for effective rehabilitation. However, traditional methods frequently fall short in providing personalized feedback and consistent monitoring, resulting in suboptimal outcomes. To overcome these limitations, a Spinal Cord Rehabilitation System (SCRS) is proposed, utilizing Raspberry Pi and sensor technology integrated with a machine learning classification model. This project aims to develop a portable, cost-effective, and user-friendly system to aid healthcare professionals and patients in monitoring and optimizing rehabilitation exercises more effectively, thereby improving functional capabilities and outcomes for individuals with SCI.

DATASET DESCRIPTION:

During the training phase, we utilize data from accelerometers and gyroscopes, capturing readings along the X, Y, and Z axes. These readings serve as input for our machine learning model. In the testing phase, real-time data is collected and stored in a Google Sheet using the Google API. These real-time values are then employed to assess the performance of the trained machine learning model.

CODE:

ML MODEL:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score, classification_report

data = pd.read_csv('/content/pad_dataset1.csv')

X = data.drop(columns=['output','time index'])

y = data['output']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = XGBClassifier()

model.fit(X_train, y_train)

y_proba = model.predict_proba(X_test)

y_pred_max_proba = y_proba.max(axis=1)

y_pred_classes = y_proba.argmax(axis=1)

for i, (pred_class, max_prob) in enumerate(zip(y_pred_classes, y_pred_max_proba)):
```

```
    print(f"Instance {i + 1}: Predicted class is {pred_class} with the probability {max_prob * 100:.2f}%")
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, y_pred))
new_sample = [[20.61, -25.3, -0.06, -23.93, 40.54, -48.07]]
new_df = pd.DataFrame(new_sample, columns=X.columns)
new_proba = model.predict_proba(new_df)
new_pred_max_proba = new_proba.max(axis=1)
new_pred_classes = new_proba.argmax(axis=1)
for i, (pred_class, max_prob) in enumerate(zip(new_pred_classes, new_pred_max_proba)):
    print(f"New instance {i + 1}: Predicted class is {pred_class} with the probability {max_prob * 100:.2f}%")
import pickle
# Assuming 'model' is your trained XGBoost model
with open('spi_shirt.pkl', 'wb') as f:
    pickle.dump(model, f)
with open('spi_shirt.pkl', 'rb') as f:
    loaded_model = pickle.load(f)
import numpy as np
import gspread
from oauth2client.service_account import ServiceAccountCredentials
import requests
import pickle
# Load your machine learning model using pickle
with open('/content/spi_shirt.pkl', 'rb') as f:
    model = pickle.load(f)
# Define the scope and credentials to access Google Sheets and Google Drive
scope = ['https://www.googleapis.com/auth/spreadsheets',
```

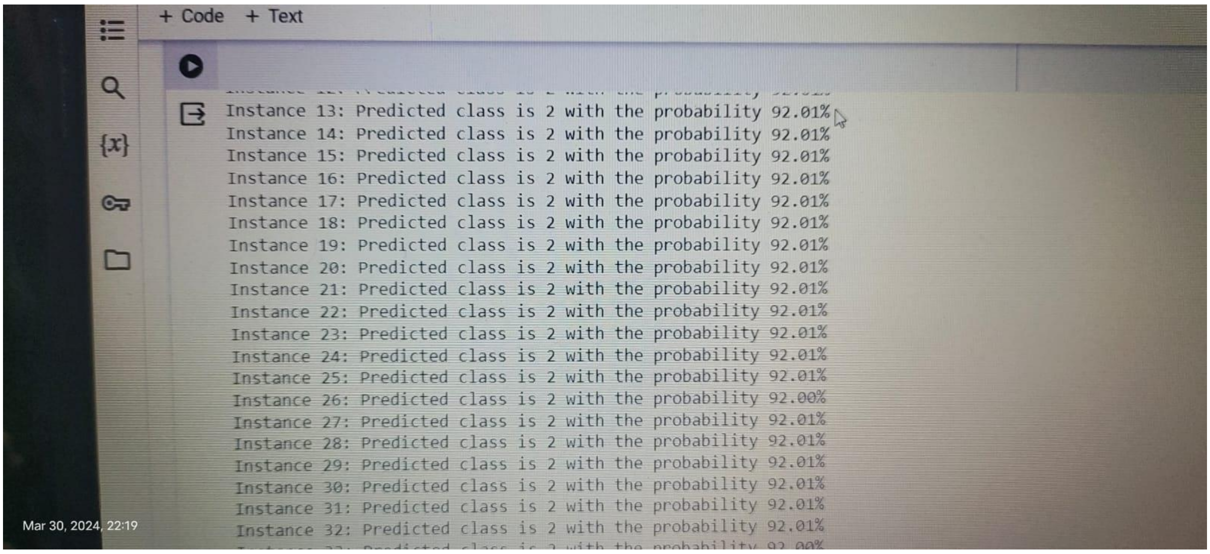
```
        'https://www.googleapis.com/auth/drive']
credentials = ServiceAccountCredentials.from_json_keyfile_name('/content/credentials.json', scope)
client = gspread.authorize(credentials)
sheet = client.open("sshirt")
worksheet = sheet.get_worksheet(0)
try:
    sensor_data = worksheet.get_all_values()
except requests.exceptions.ReadTimeout as e:
    print("Timeout while reading data from Google Sheet:", e)
    sensor_data = []
processed_data = []
# Iterate through each row in sensor_data
for row in sensor_data:
    # Check if all elements in the row are empty strings
    if all(cell == '' for cell in row):
        continue  # Skip empty rows
    # Convert each element in the row to float
    processed_row = []
    for cell in row:
        processed_row.append(float(cell))
    processed_data.append(processed_row)
if processed_data:
    y_proba = model.predict_proba(np.array(processed_data))
    y_pred_max_proba = y_proba.max(axis=1)
    y_pred_classes = y_proba.argmax(axis=1)
    for i, (pred_class, max_prob) in enumerate(zip(y_pred_classes, y_pred_max_proba)):
        print(f"Instance {i + 1}: Predicted class is {pred_class} with the probability {max_prob * 100:.2f}%")
    y_pred = model.predict(np.array(processed_data))
else:
```

```
    print("No valid sensor data found in the Google Sheet.")
```

RASPBERRY PI CODE:

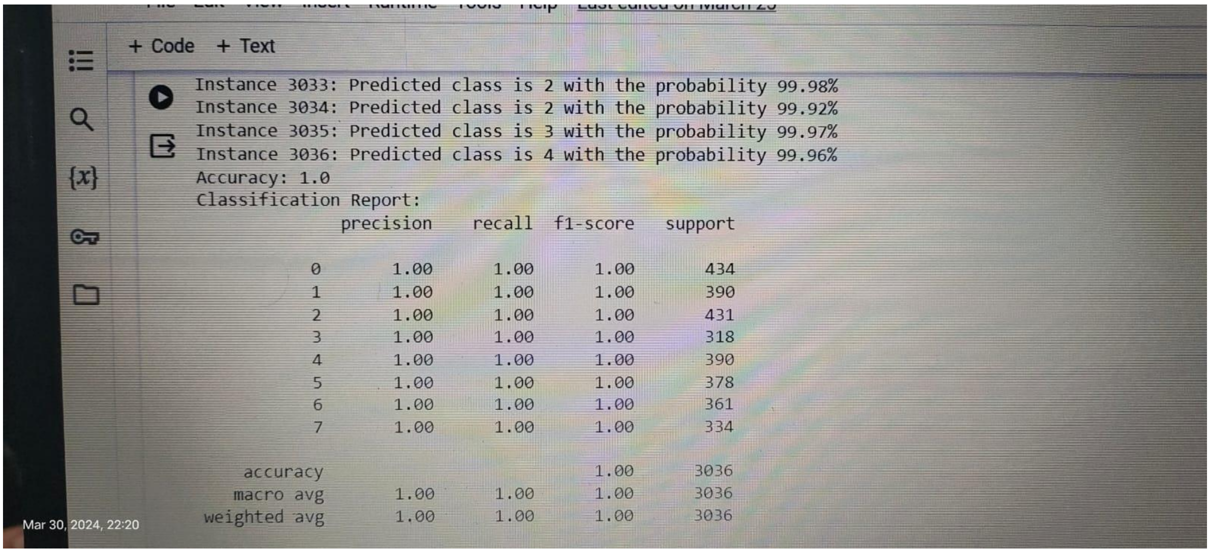
```
import smbus
# Create MPU6050 object
mpu6050_sensor = mpu6050.mpu6050(0x68)
# Set up Google Sheets
scope = ['https://spreadsheets.google.com/feeds',
         'https://www.googleapis.com/auth/drive']
creds = Credentials.from_service_account_file('credentials.json', scopes=scope)
client = gspread.authorize(creds)
sheet = client.open('sshirt').sheet1  # Replace 'Your Google Sheet Name' with your actual sheet name
# Main loop
while True:
    # Read MPU6050 sensor data
    sensor_data = mpu6050_sensor.get_all_data()
    accelerometer_data = sensor_data[0]
    gyroscope_data = sensor_data[1]
    # Write sensor values to Google Sheet
    row = [accelerometer_data['x'], accelerometer_data['y'], accelerometer_data['z'],
           gyroscope_data['x'], gyroscope_data['y'], gyroscope_data['z']]
    sheet.append_row(row)
    # Output all sensor values
    print(f"Accelerometer data: {accelerometer_data}")
    print(f"Gyroscope data: {gyroscope_data}")
time.sleep(1)
```

OUTPUT:



```
+ Code   + Text
Instance 13: Predicted class is 2 with the probability 92.01%
Instance 14: Predicted class is 2 with the probability 92.01%
Instance 15: Predicted class is 2 with the probability 92.01%
Instance 16: Predicted class is 2 with the probability 92.01%
Instance 17: Predicted class is 2 with the probability 92.01%
Instance 18: Predicted class is 2 with the probability 92.01%
Instance 19: Predicted class is 2 with the probability 92.01%
Instance 20: Predicted class is 2 with the probability 92.01%
Instance 21: Predicted class is 2 with the probability 92.01%
Instance 22: Predicted class is 2 with the probability 92.01%
Instance 23: Predicted class is 2 with the probability 92.01%
Instance 24: Predicted class is 2 with the probability 92.01%
Instance 25: Predicted class is 2 with the probability 92.01%
Instance 26: Predicted class is 2 with the probability 92.00%
Instance 27: Predicted class is 2 with the probability 92.01%
Instance 28: Predicted class is 2 with the probability 92.01%
Instance 29: Predicted class is 2 with the probability 92.01%
Instance 30: Predicted class is 2 with the probability 92.01%
Instance 31: Predicted class is 2 with the probability 92.01%
Instance 32: Predicted class is 2 with the probability 92.01%
```

Mar 30, 2024, 22:19

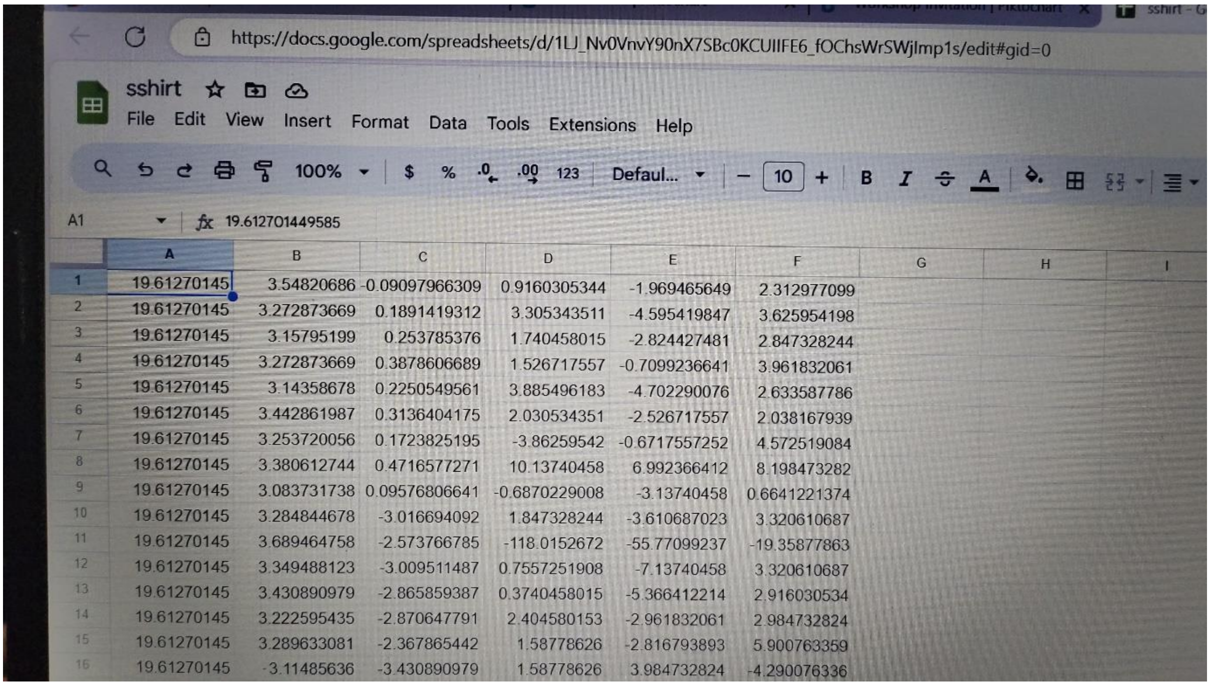


```
+ Code   + Text
Instance 3033: Predicted class is 2 with the probability 99.98%
Instance 3034: Predicted class is 2 with the probability 99.92%
Instance 3035: Predicted class is 3 with the probability 99.97%
Instance 3036: Predicted class is 4 with the probability 99.96%
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       434
           1       1.00      1.00      1.00       390
           2       1.00      1.00      1.00       431
           3       1.00      1.00      1.00       318
           4       1.00      1.00      1.00       390
           5       1.00      1.00      1.00       378
           6       1.00      1.00      1.00       361
           7       1.00      1.00      1.00       334

    accuracy                           1.00      3036
   macro avg       1.00      1.00      1.00      3036
weighted avg       1.00      1.00      1.00      3036
```

Mar 30, 2024, 22:20


https://docs.google.com/spreadsheets/d/1LJ_Nv0VnvY90nX7SBc0KCUIIFE6_fOChsWrSWjlmp1s/edit#gid=0

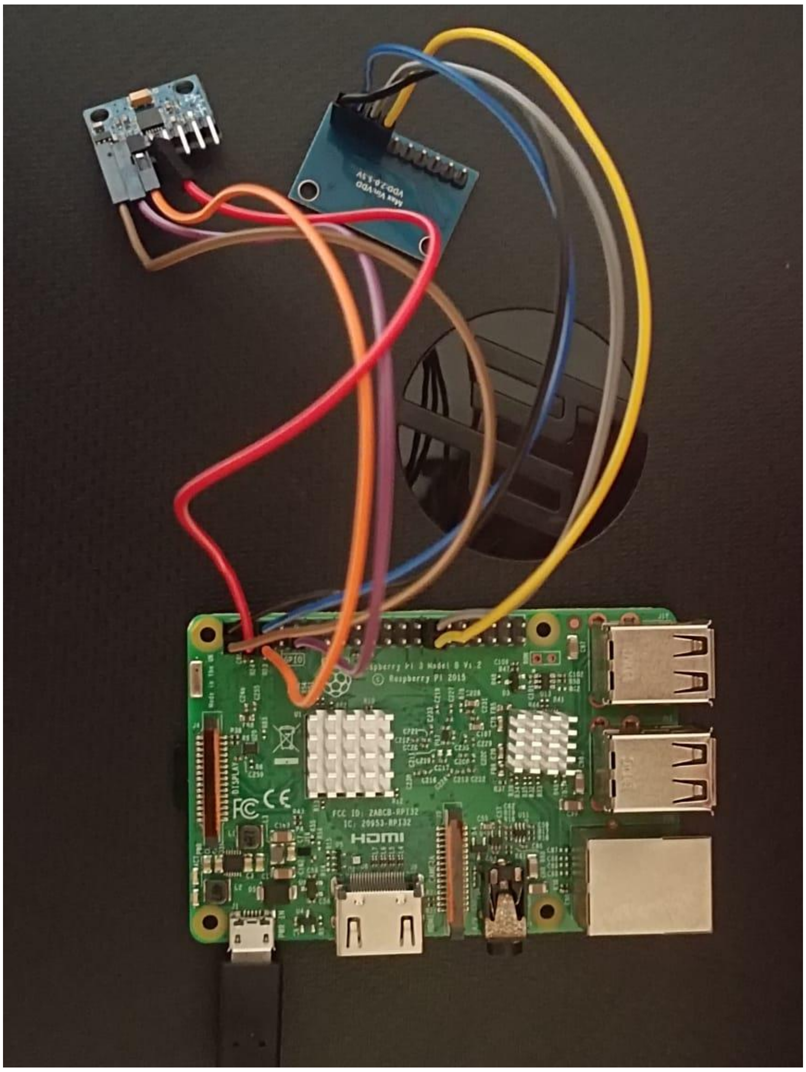
sshirt

File Edit View Insert Format Data Tools Extensions Help

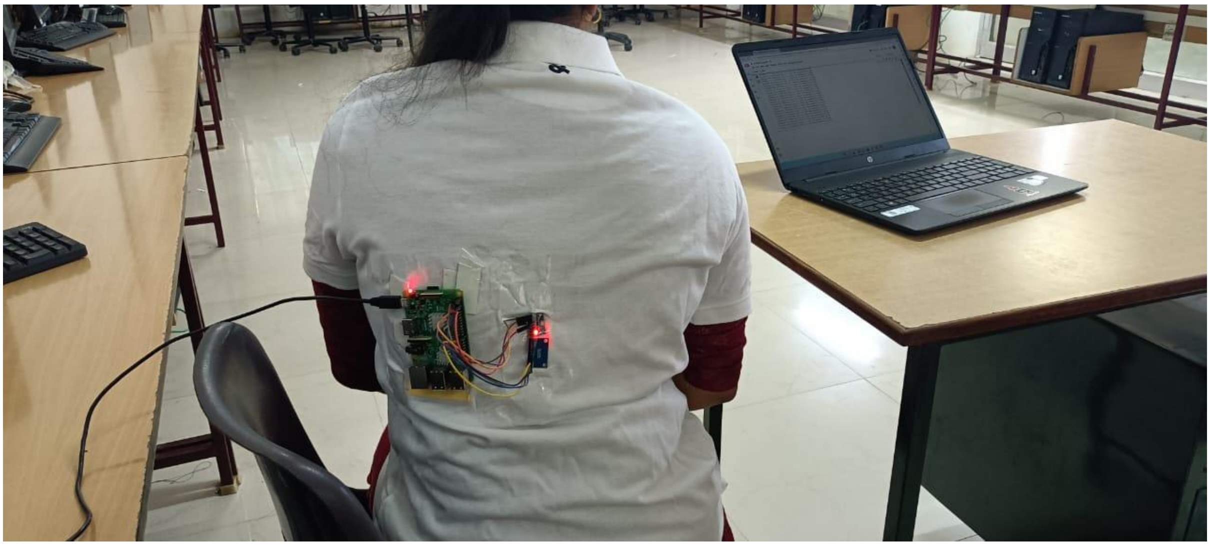
A1 fx 19.612701449585

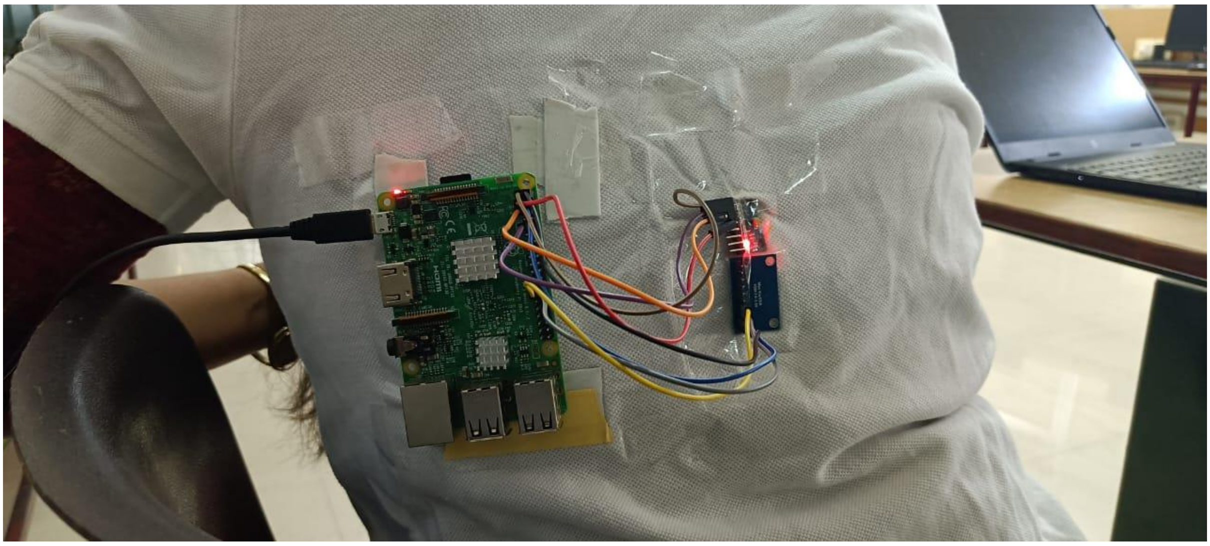
	A	B	C	D	E	F
1	19.61270145	3.54820686	-0.09097966309	0.9160305344	-1.969465649	2.312977099
2	19.61270145	3.272873669	0.1891419312	3.305343511	-4.595419847	3.625954198
3	19.61270145	3.15795199	0.253785376	1.740458015	-2.824427481	2.847328244
4	19.61270145	3.272873669	0.3878606689	1.526717557	-0.7099236641	3.961832061
5	19.61270145	3.14358678	0.2250549561	3.885496183	-4.702290076	2.633587786
6	19.61270145	3.442861987	0.3136404175	2.030534351	-2.526717557	2.038167939
7	19.61270145	3.253720056	0.1723825195	-3.86259542	-0.6717557252	4.572519084
8	19.61270145	3.380612744	0.4716577271	10.13740458	6.992366412	8.198473282
9	19.61270145	3.083731738	0.09576806641	-0.6870229008	-3.13740458	0.6641221374
10	19.61270145	3.284844678	-3.016694092	1.847328244	-3.610687023	3.320610687
11	19.61270145	3.689464758	-2.573766785	-118.0152672	-55.77099237	-19.35877863
12	19.61270145	3.349488123	-3.009511487	0.7557251908	-7.13740458	3.320610687
13	19.61270145	3.430890979	-2.865859387	0.3740458015	-5.366412214	2.916030534
14	19.61270145	3.222595435	-2.870647791	2.404580153	-2.961832061	2.984732824
15	19.61270145	3.289633081	-2.367865442	1.58778626	-2.816793893	5.900763359
16	19.61270145	-3.11485636	-3.430890979	1.58778626	3.984732824	-4.290076336

SENSOR CONNECTION:



RESULT ANALYSIS:





The analysis of the classification report alongside the pattern analysis indicates a highly successful outcome for your spinal cord exercise recognition system. The model achieved perfect accuracy (1.00) on all eight exercise classifications, which is reflected in the precision, recall, and F1-score of 1.00 for each class. This suggests that the system can effectively differentiate between the various exercises based on the patterns learned from the accelerometer and gyroscope data.

The pattern analysis likely played a crucial role in achieving this accuracy. By identifying key features like peak accelerations, dominant movement axes, and gyroscope patterns specific to each exercise, the model was able to accurately classify them based on the unique movement signatures captured by the sensors. This highlights the importance of understanding these patterns for feature selection and model training in such a system.

Overall, the combination of a well-designed machine learning model and in-depth pattern analysis resulted in a system that can remarkably recognize and classify different spinal cord exercises with perfect accuracy.