COMP6733 - IoT Design Studio

The development of a construction worker safety detection and monitoring system

Project Report Final

Group Members:

z5353871 Kinto Wan

z3405715 Chin Chai

z5137765 Xiyang Shi

z5263967 Guy Chilcott

Contents

Background:

Construction workers are undeniably one of the most important members of our society. However, they are at high risks of injuries and health problems, as construction is one of the most hazardous work-place industries in the world [1,2]. One reason for this is the highly physically demanding nature of the job, which exposes construction workers to work-related musculoskeletal disorder (WMSD). WMSD is the painful disorder of soft tissues (muscles, tendons, ligaments, nerves, joints and cartilage) caused by reasons such as poor posture, high exertions, vibrations, during work [4]. The most common types of WMSD include carpal tunnel syndrome, sciatica, herniated discs, lower back pain, etc. This includes strains, sprains and traumas [3]. WMSD is extremely common, particularly in the construction work community. Infact, between 2020-2021, 470,000 workers have suffered from WMSD just in the UK [4].

It is found that one of the main reasons for WMSD in construction workers is caused by poor working posture and extreme working conditions/environments [3]. Statistically, given poor working posture, the top 3 ergonomic problems in construction are [5]:

1. working in same position for long time
2. bending/twisting
3. working in awkward/cramped positions

Additionally, it is also found that another factor to WMSD is the harsh climatic conditions can increase the risk of physical fatigue and even potential death [3].

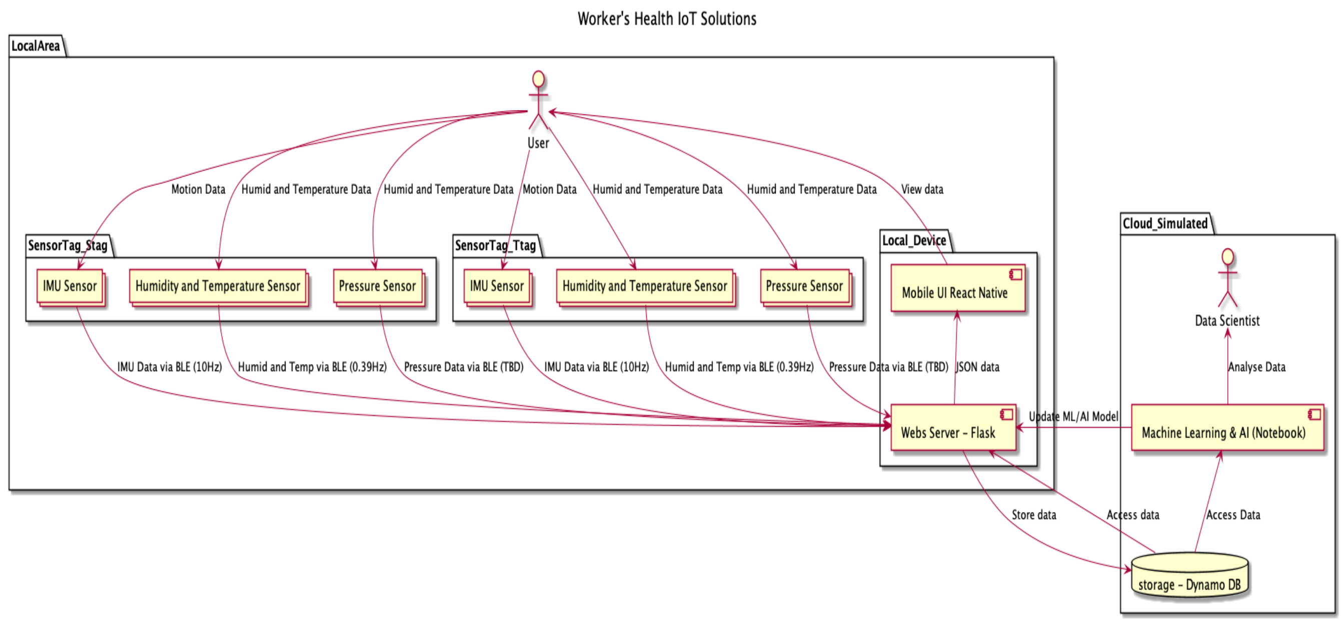
Aims/Goals:

Develop a system, which can help detect and monitor the work-related musculoskeletal disorder (WMSD) risk of construction or office workers. We will aim to use two sensor tags to determine and analyse the dynamic posture, as well as track the safety of the surrounding environment, by measuring the linear and angular acceleration, magnetic field strength, humidity, temperature and pressure.

Methods:

Solution Architect

The solution architect the team has achieved is shown below.


Worker's Health IoT Solutions
LocalArea
User
Motion Data
Humid and Temperature Data
Humid and Temperature Data
Motion Data
Humid and Temperature Data
Humid and Temperature Data
View data
SensorTag_Stag
IMU Sensor
Humidity and Temperature Sensor
Pressure Sensor
SensorTag_Ttag
IMU Sensor
Humidity and Temperature Sensor
Pressure Sensor
Local_Device
Mobile UI React Native
IMU Data via BLE (10Hz)
Humid and Temp via BLE (0.39Hz)
Pressure Data via BLE (TBD)
IMU Data via BLE (10Hz)
Humid and Temp via BLE (0.39Hz)
Pressure Data via BLE (TBD)
JSON data
Webs Server - Flask
Update ML/AI Model
Cloud_Simulated
Data Scientist
Analyse Data
Machine Learning & AI (Notebook)
Store data
Access data
Access Data
storage - Dynamo DB

Data Acquisition

To enable data acquisition from Sensortag to data storage, Bluetooth Low Energy wireless technology has been chosen as BLE consumes low energy with high data rate when it is optimised.

BLE Connection

Two Sensortags have been programmed with BLE functionalities using TI-RTOS BLE Stack. The device name has been programmed as “StillThinking STag” and “StillThinking TTag” which are side thigh and trunk sensortag respectively.

Python Flask

Python Flask is a lightweight web server library. It is chosen as it provides great flexibility in integrating with databases (NoSQL), serving REST API, and communicating with Sensortag using python library - Bleak. Following list some tasks that are performed by the Python flask.

- Connect to STag and TTag and display connection status
- Insert IMU, humidity and temperature data to DynamoDB

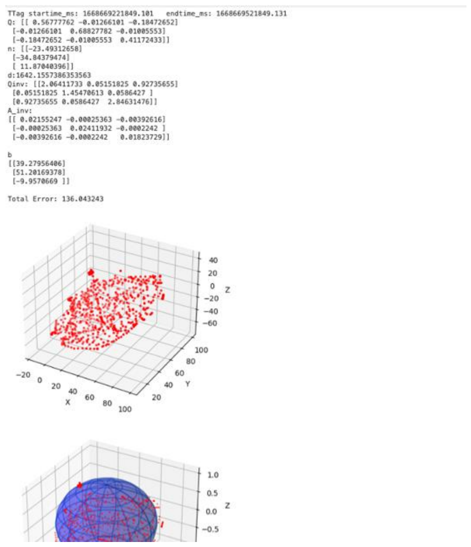
- Response to REST API call with JSON data

Dynamo DB

Dynamo DB structure has been designed with both tag-name (TTag/STag) and sensor type (IMU/Humidity/Pressure) as partition key. Sort key that is applied to the database is timestamp in milliseconds.

Data Pre-processing - Magnetometer Magnetic Field (in Notebook)

Calibration is essential for magnetometer and Sensortag's magnetometer data is notoriously noisy. Hard iron and soft iron calibration have been introduced to the model as per method provided at reference [6]. Also, scale correction has been introduced in the model to have a magnetic field at range +/-57µT. However, a satisfactory result is not yet achieved. Following gives a calibration output using reference [6], which 'b' is hard iron and 'A_inv' is the soft iron to give a unit vector of magnetic field. To have usable data from the magnetometer, IMU data rate has to be significantly increased.



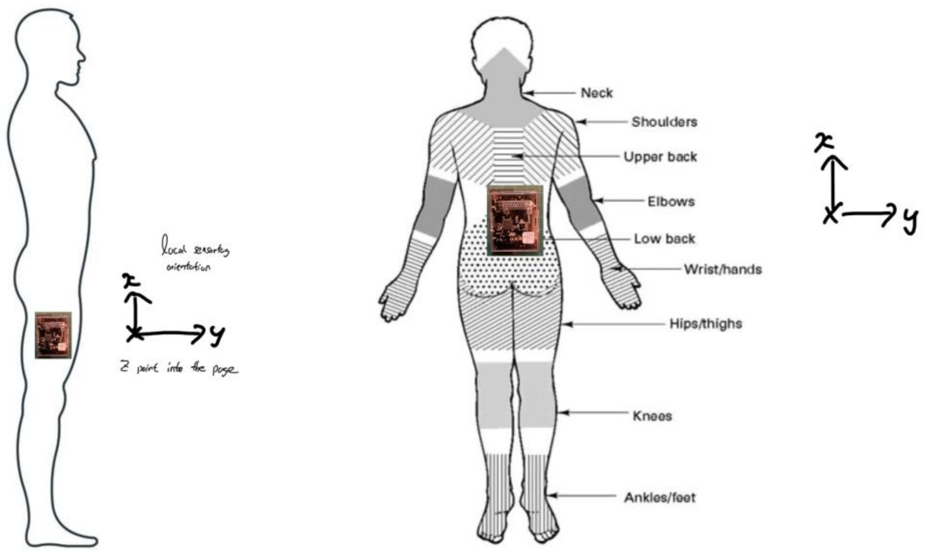
```
TTag startime_ms: 1668669221849.101   endtime_ms: 1668669521849.131
Q: [[ 0.56777762 -0.01266101 -0.18472652]
 [-0.01266101  0.68827782 -0.01005553]
 [-0.18472652 -0.01005553  0.41172433]]
n: [[-23.49312658]
 [-34.84379474]
 [ 11.87040396]]
d:1642.1557386353563
Qinv: [[2.06411733 0.05151825 0.92735655]
 [0.05151825 1.45470613 0.0586427 ]
 [0.92735655 0.0586427  2.84631476]]
A_inv:
[[ 0.02155247 -0.00025363 -0.00392616]
 [-0.00025363  0.02411932 -0.0002242 ]
 [-0.00392616 -0.0002242   0.01823729]]

b
[[39.27956406]
 [51.20169378]
 [-9.9570669 ]]

Total Error: 136.043243
```

Data Collection

For our data collection process, we followed a strict protocol to collect the data. Our protocol follows as such:

- There are 3 order of actions for collection : Standing, Bending, Squatting

- Each series of collections will be one long sequence in the order of standing, squatting then bending.
- Make sure the tag is placed with the words "Texas Instruments" upright (can be read without tilting your head), and the light is facing away from the body. (i.e. the battery is touching the body).
- Place t-tag on the lumbar section, and s-tag on outer thigh (as shown in below image)
- Start the data collection program, and perform each motion between 2-4 seconds continuously, for a total of 5 minutes per action.



Once the data collection has been completed, and that the output has been confirmed to be stored within the AWS database, we could then move onto the data processing procedure.

Data Preprocessing

For the data processing stage, the aim was to set up the data in such a way that we could perform a clean and accurate machine learning prediction in the next step. The order of our data processing was as follows:

1. **Data manipulation in feature extraction** :
   At first, we took the x,y,z gyro and acc columns from TTag and subtracted the respective STag columns from it. In other words, we made a new column called “gyro_x”, which was the

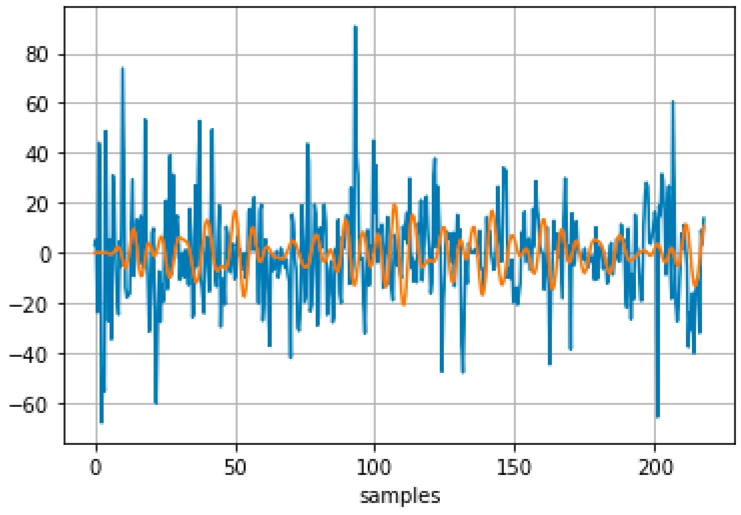
TTag_gyro_x - STag_gyro_x. We proceeded to do this for all x,y,z for both the gyrometer and accelerometer results. This way we could calculate the relative change between the two sensors.

2. **Splitting the dataset into the 3 datasets**: Pose1, pose2, pose3, where they represent standing, squatting and bending over respectively.

3. **Noise-cancelling filter**:

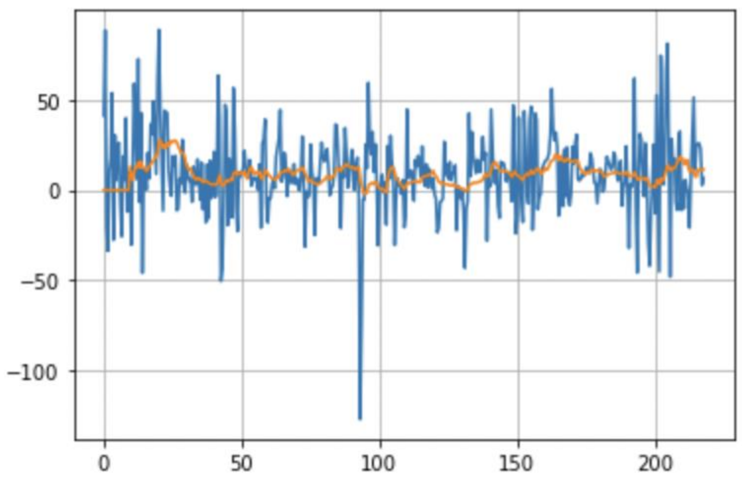
   Initially, we attempted to filter the data using a Butterworth Low Pass Filter in Python. The parameters that were used for filtering were based on another paper, written by Pr W.Hu, with low-cut as 0.5Hz and high-cut at 10Hz, since “important human actions tended to be below 10Hz” [7].

   However, when trying those parameters, since our system doesn’t have nearly as high a sample rate as that in the referenced paper, the result was not ideal for our set of results (as you can see in the Butterworth Low Pass Filter image below).



Butterworth Lowpass Filter

We then tried using a Kalman Filter to remove such noise, however accuracy was not high either. Then we tried using a moving-average filter and found that it produced very clean data. So we proceeded with the moving average filter.



Moving Average Filter

4. It is important to note that, we purposefully **did not normalise the data**. As normalising the data would reduce the amount of useful information for machine learning later on, as the amplitude in signals would also be a vital feature in differentiating between standing and other motions.

5. A **sliding window** was used to collect data/sample each dataset. This was used in order to extract the relevant length and size of data from the time series. A sliding window of 4 seconds with 75% overlap was used. These variables along with the sample rate were then used to calculate the exact number of samples for each window.

Machine Learning

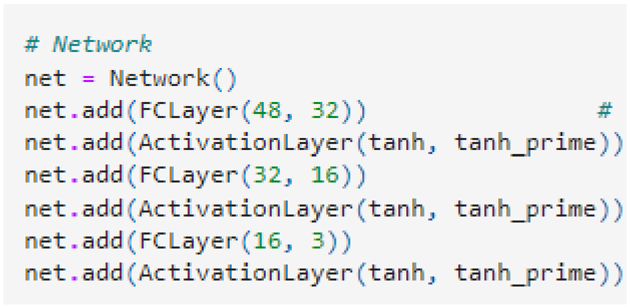
For the machine learning stage, we experimented with several models, with different parameters. The models that we used were: Long-short term memory (LSTM), support vector machine (SVM), Decision tree (DT), K-nearest neighbours (KNN), and a multilayer perceptron.

For the SVM: We tried different kernels. The kernel was found to be best at "rbfl", compared to linear or poly when Butterworth Low Pass filter is used. Whilst the kernel was found to be best at "linear", compared to "rbfl" or poly when moving-average filter is used.

DT: Call tree. DecisionTreeClassifier() of sklearn directly. The default settings were used.

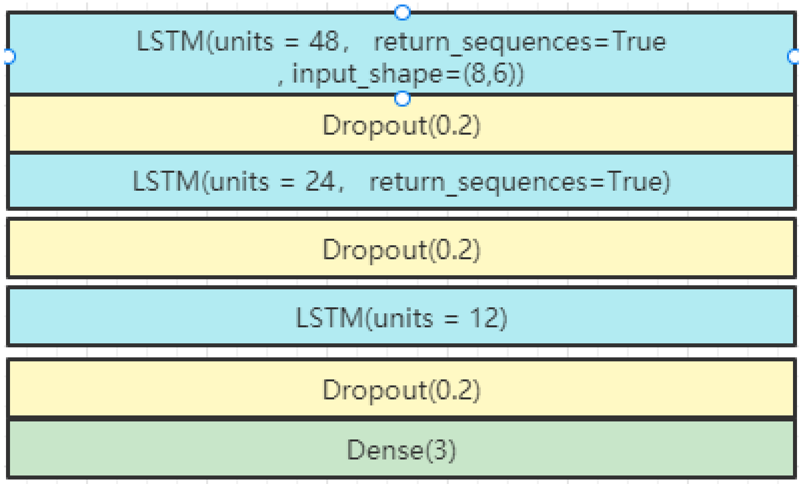
KNN: The KNN model was set to having K neighbours of 3.

Multilayer perceptron (3 layers): The neural network was set to have 1 hidden layer, with 48 nodes of input and 3 layers of output, predicting the 3 poses from the series. A learning rate of 0.01, and 300 epochs was the final setting. The activation function was a tan function.



```
# Network
net = Network()
net.add(FCLayer(48, 32))                    #
net.add(ActivationLayer(tanh, tanh_prime))
net.add(FCLayer(32, 16))
net.add(ActivationLayer(tanh, tanh_prime))
net.add(FCLayer(16, 3))
net.add(ActivationLayer(tanh, tanh_prime))
```

LSTM: It is composed of three LSTM layers. The specific structure is :


LSTM(units = 48,   return_sequences=True
, input_shape=(8,6))
Dropout(0.2)
LSTM(units = 24,   return_sequences=True)
Dropout(0.2)
LSTM(units = 12)
Dropout(0.2)
Dense(3)

The input data is a three-dimensional array with a shape of (n_sample,8,6). The LSTM layer initial nodes are selected, 96, 64, 48, 32, and the final 48 result is good. Optimized with Adam optimizer with a learning rate of 0.005.For the choice of learning rate, I chose 0.001, 0.005, 0.002, 0.001, 0.0005, and finally 0.005 performed the best. The batch size is 16. Because the data volume is small, in order to prevent overfitting, the epoch is set to 50. When the epoch is 200, overfitting occurs. Compared with the data processed by Butterworth Low Pass filter, the training accuracy of LSTM using the data processed by moving average filter is higher.

Interaction with the front end

We called the pose json data stored locally by the ML script through the method based on the flask library to send it to the front end.

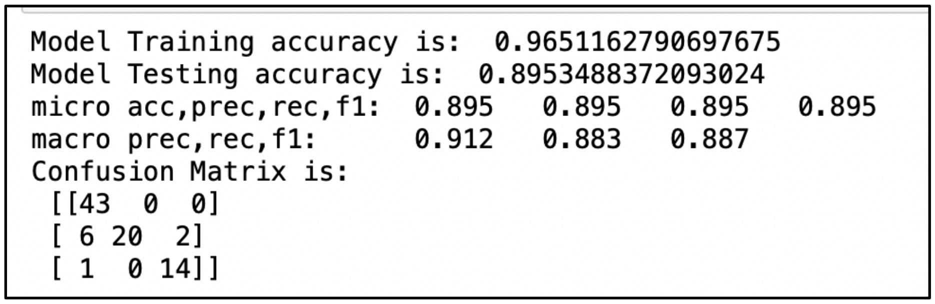
Results:

In terms of results, our project results were split up into ML metrics, live accuracy of the prediction system and the user interface. The machine learning metrics would indicate the results from the machine learning models given the data input. The live accuracy of the prediction system would refer to the prediction results from the whole system. Whilst the user interface will show the UI of our app system.

Machine learning metrics

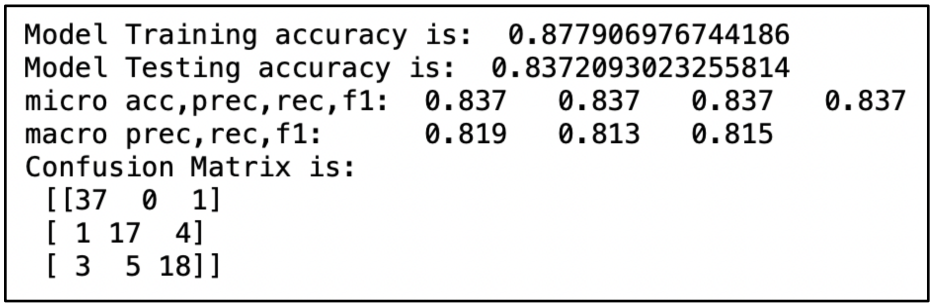
For the machine learning metrics, we present our results from each model one by one. We have presented the model training and testing accuracy as well as the micro and macro accuracy, precision, recall and F1, and we have the confusion matrix:

- SVM (Linear)



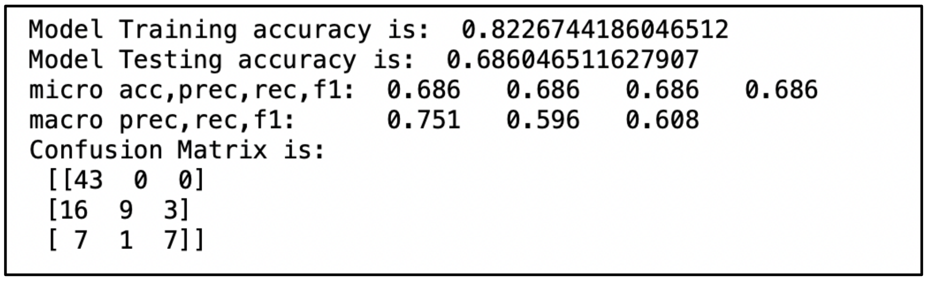
```
Model Training accuracy is:  0.9651162790697675
Model Testing accuracy is:  0.8953488372093024
micro acc,prec,rec,f1:  0.895   0.895   0.895   0.895
macro prec,rec,f1:      0.912   0.883   0.887
Confusion Matrix is:
 [[43  0  0]
 [ 6 20  2]
 [ 1  0 14]]
```

- SVM (rbf)



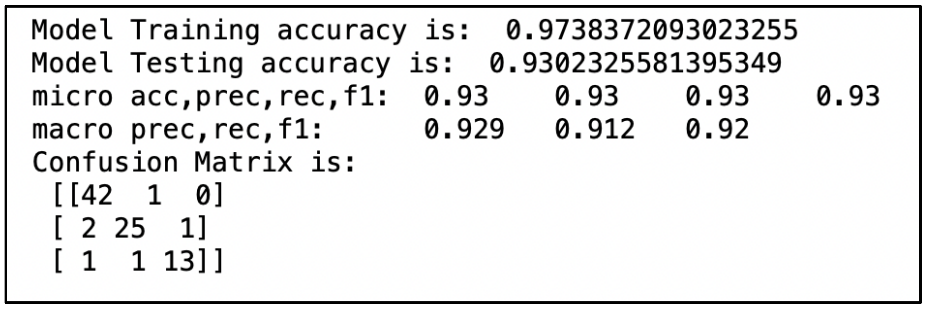
```
Model Training accuracy is:  0.877906976744186
Model Testing accuracy is:  0.8372093023255814
micro acc,prec,rec,f1:  0.837   0.837   0.837   0.837
macro prec,rec,f1:      0.819   0.813   0.815
Confusion Matrix is:
 [[37  0  1]
 [ 1 17  4]
 [ 3  5 18]]
```

- SVM (polynomial)



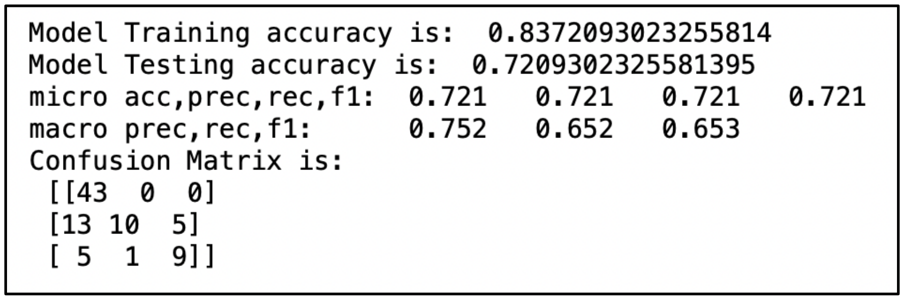
```
Model Training accuracy is:  0.8226744186046512
Model Testing accuracy is:  0.686046511627907
micro acc,prec,rec,f1:  0.686   0.686   0.686   0.686
macro prec,rec,f1:      0.751   0.596   0.608
Confusion Matrix is:
 [[43  0  0]
 [16  9  3]
 [ 7  1  7]]
```

- Decision Tree



```
Model Training accuracy is:  0.9738372093023255
Model Testing accuracy is:  0.9302325581395349
micro acc,prec,rec,f1:  0.93    0.93    0.93    0.93
macro prec,rec,f1:      0.929   0.912   0.92
Confusion Matrix is:
 [[42  1  0]
 [ 2 25  1]
 [ 1  1 13]]
```

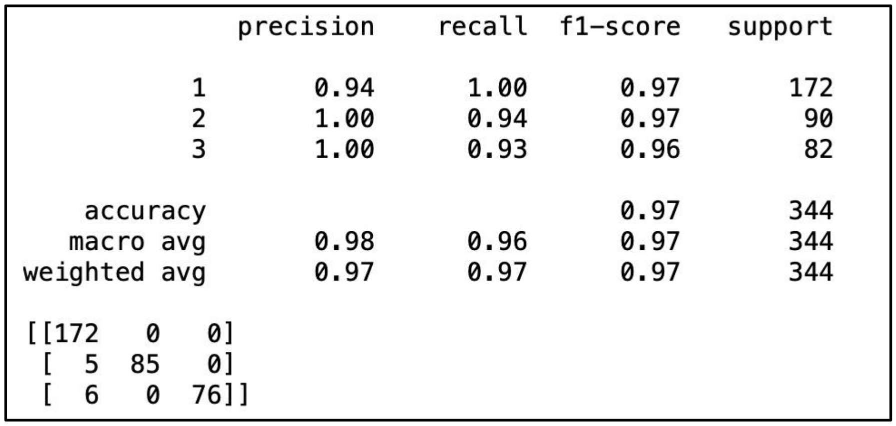
- KNN



```
Model Training accuracy is:  0.8372093023255814
Model Testing accuracy is:  0.7209302325581395
micro acc,prec,rec,f1:  0.721   0.721   0.721   0.721
macro prec,rec,f1:      0.752   0.652   0.653
Confusion Matrix is:
 [[43  0  0]
 [13 10  5]
 [ 5  1  9]]
```

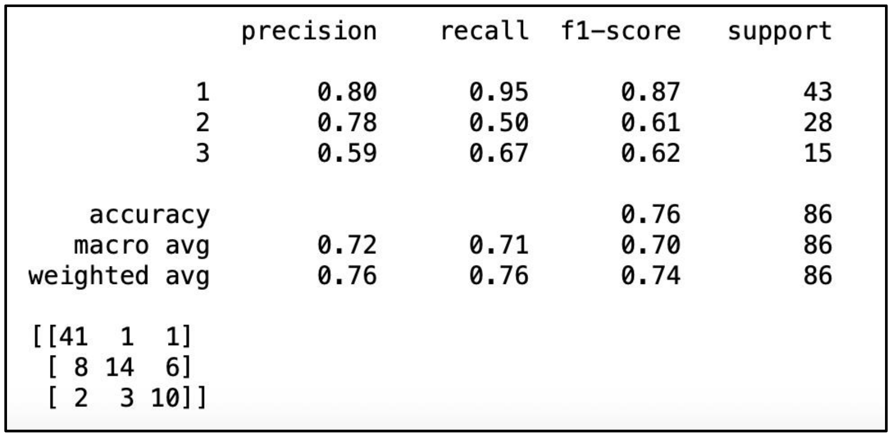
- Multilayer perceptron

- *Training*



```
              precision    recall  f1-score   support

           1       0.94      1.00      0.97       172
           2       1.00      0.94      0.97        90
           3       1.00      0.93      0.96        82

    accuracy                           0.97       344
   macro avg       0.98      0.96      0.97       344
weighted avg       0.97      0.97      0.97       344

[[172   0   0]
 [  5  85   0]
 [  6   0  76]]
```

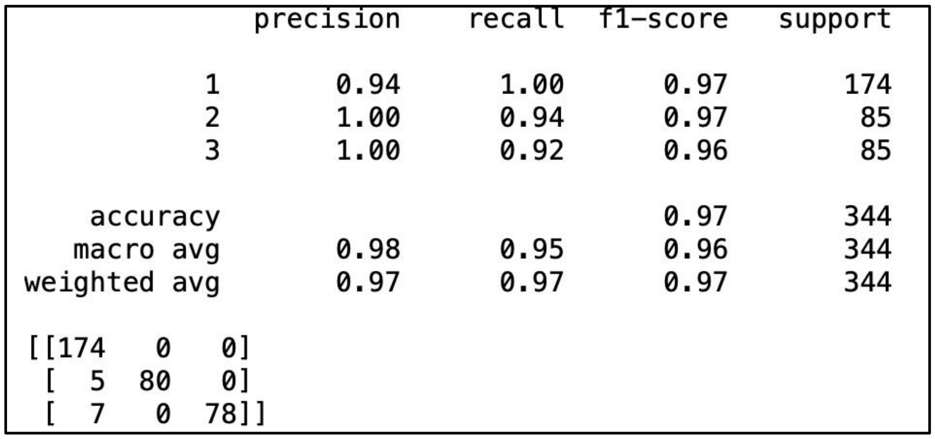
- *Testing*



```
              precision    recall  f1-score   support

           1       0.80      0.95      0.87        43
           2       0.78      0.50      0.61        28
           3       0.59      0.67      0.62        15

    accuracy                           0.76        86
   macro avg       0.72      0.71      0.70        86
weighted avg       0.76      0.76      0.74        86

[[41  1  1]
 [ 8 14  6]
 [ 2  3 10]]
```

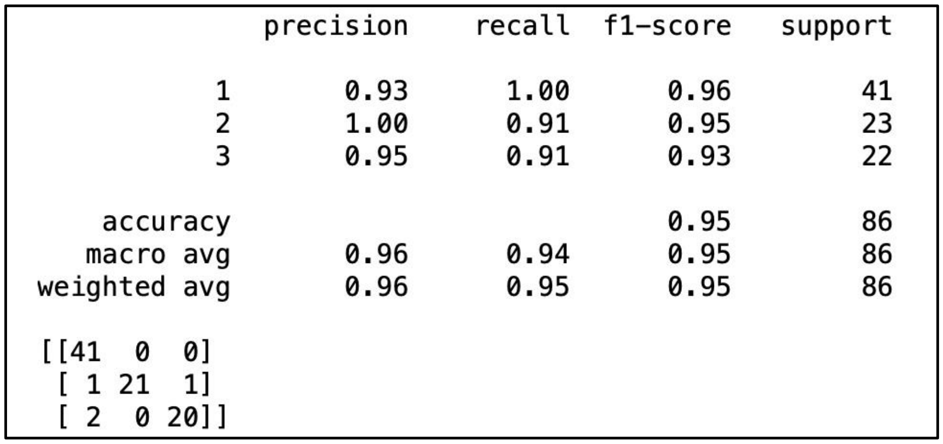
LSTM

- *Training*



```
              precision    recall  f1-score   support

           1       0.94      1.00      0.97       174
           2       1.00      0.94      0.97        85
           3       1.00      0.92      0.96        85

    accuracy                           0.97       344
   macro avg       0.98      0.95      0.96       344
weighted avg       0.97      0.97      0.97       344

[[174   0   0]
 [  5  80   0]
 [  7   0  78]]
```

- *Testing*



```
              precision    recall  f1-score   support

           1       0.93      1.00      0.96        41
           2       1.00      0.91      0.95        23
           3       0.95      0.91      0.93        22

    accuracy                           0.95        86
   macro avg       0.96      0.94      0.95        86
weighted avg       0.96      0.95      0.95        86

[[41  0  0]
 [ 1 21  1]
 [ 2  0 20]]
```

Summary:

ML Method	Training Accuracy	Testing Accuracy
SVM (linear)	**96.5**	**89.5**
SVM (rbf)	87.8	83.7
SVM (poly)	82.2	68.6
Decision Tree	**97.4**	**93.0**
KNN (N = 3)	83.7	72.1
Multilayer perceptron neural network	97.0	76.0
LSTM	**97.0**	**95.0**

As we can see above, the top three models were the LSTM, Decision Tree, and the SVM (linear) model. The LSTM has a testing prediction of 95%, training prediction of 97%.

Live accuracy of the prediction system

We have tested our live sensor prediction system by making corresponding actions and manually identifying the accuracy of actions.

Given the testing of our system, the overall live prediction accuracy can be calculated from the following equation (TP + TN)/(TP + TN + FN + FP).

Live Prediction Accuracy = (TP + TN)/(TP + TN + FN + FP) = 13/15 = 86.7 %

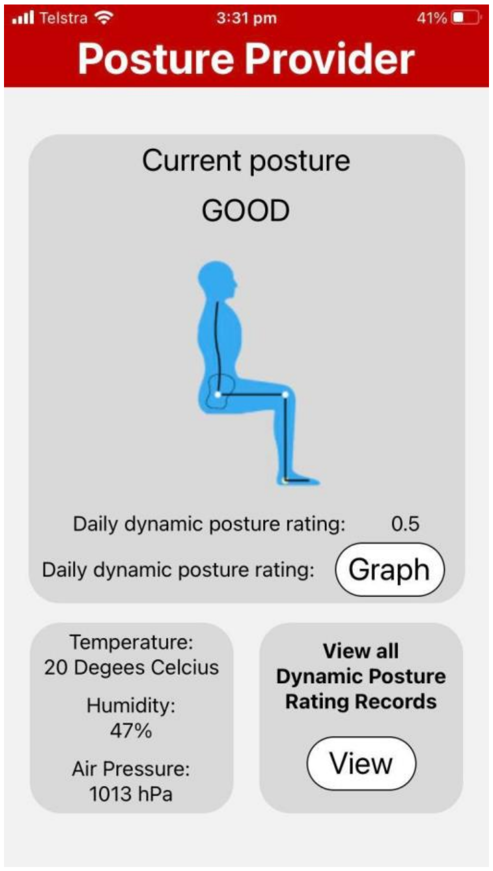
Which gives us an accuracy of 86.7%. The details of our final results of our live sensor prediction is as shown below.

	Stand	Squatting	Bending	Recognition Rate
Stand	5	0	0	***100%***
Squatting	0	4	1	***80%***
Bending	1	0	4	***80%***
Overall				***86.7%***

From the data analysis, it can be seen that there may be errors in the recognition of action switching by the model. For example, when a person switches from squatting to standing, part of the time it may overlap with the posture features of bending, resulting in error information.

User Interface

We have made our project available to all by developing an IOS and Android application that can be used alongside the device. This user interface displays data collected by the devices in a user friendly way, displaying current daily dynamic posture rating as well as the current temperature, humidity and air pressure environmental readings. We designed a user interface for our project to make the data easy to understand and available to projects target users, workers who are at risk of work-related musculoskeletal disorder, such as construction or office workers.



Posture Provider

Current posture

GOOD

Daily dynamic posture rating: 0.5

Daily dynamic posture rating: Graph

Temperature:
20 Degees Celcius

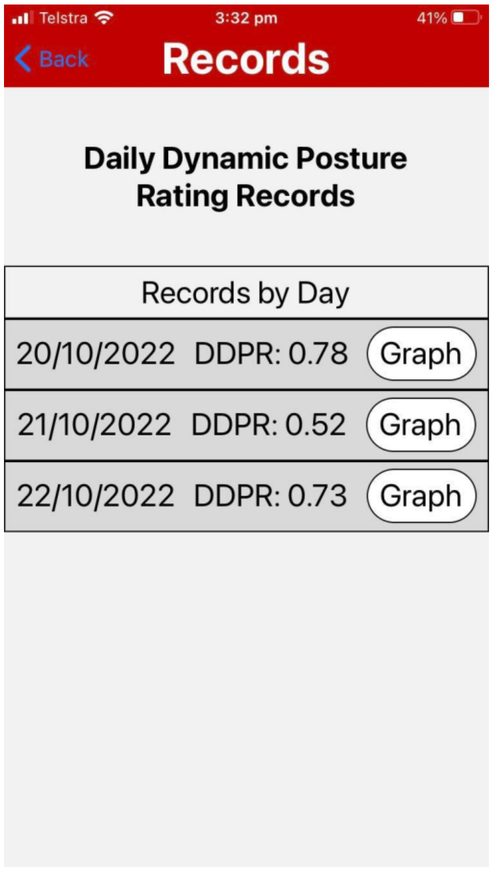
Humidity:
47%

Air Pressure:
1013 hPa

View all Dynamic Posture Rating Records

View

The application also shows records all previous days posture ratings. Each record contains the daily rating over time graphs and raw average per hour data so users are able to review their average posture over the course of the day and the specific times their posture could have been improved. This will allow users to be conscious of their posture during these times and work on improving it.


Telstra 3:32 pm 41%

Back **Records**

Daily Dynamic Posture Rating Records

Records by Day	
20/10/2022 DDPR: 0.78	Graph
21/10/2022 DDPR: 0.52	Graph
22/10/2022 DDPR: 0.73	Graph

This user interface has made our project production usable as when used alongside the devices, it allows the everyday worker to both monitor and improve their posture, leading to a future of better working posture and less work-related musculoskeletal disorder.

Discussion:

1. Overall, from the results, it made a lot of sense that the LSTM performed the best, due to the nature of the neural network model, outperforming others such as SVM, DT. However, it was a surprise to see the multilayer perceptron model perform so poorly, and that the DT outperformed other models such as the SVM. In general, the micro and macro metrics such as precision, recall and F1 all made sense and nothing was out of place for all the model results. However, the overall results from the machine learning models were of high satisfaction.

2. It was expected to see that the results from the live prediction were a little lower than that of the machine learning results. This is because the live data prediction may have been affected by factors such

as the bluetooth input of the sensortag. But overall, the results of the live prediction was still of good satisfaction.

3. A small amount of training information may lead to overfitting of the model, which will reduce the accuracy when it is applicable to other people. We need to improve the amount of data to obtain a more accurate model.

4. Due to the delay of data transmission and the waiting time for data upload to the database, the background will not get the current data, but will get the data after 1s to 2s, which may lead to the delay of prediction information. Specifically, the predicted action may be the action of the device wearer a few seconds ago. However, in the case of a long time (for example, it lasts for more than 20s), the background will accurately distinguish the action the user is in. Our research goal is to prevent undesirable actions that are kept for a long time, so the error in this aspect is within an acceptable range, but we will continue to optimize this aspect in the future if possible.

5. The number of sensors may be few. In the preliminary test, we will find that the model will be confused when predicting squatting and bending. When we distinguish between standing and sitting, standing and bending, the effect is better. This may be that when the wearer performs these actions, several frames of actions are repeated, So we may need to add sensors at other locations to obtain more features to distinguish between these two actions.

6. The final testing of the live accuracy prediction could also be better with more testing samples, however due to time and resource constraint, we could only afford 15 samples in testing.

Conclusion:

We believe our project has achieved our original goal of detecting, monitoring and protecting workers against WMSD. If workers regularly wear our device and monitor the collected data, they will be able to track their overall dynamic posture, improve posture when needed, be aware of when they use incorrect posture and track their safety using environmental data.

Future works:

Since we are all constantly sitting in front of the computer, we will often experience back-related pain, neck-related pain, carpal tunnel syndrome (CTS), and perform repetitive typing motions [8,9]. In the future, we can expand this to the modern day office worker too to solve the occupational diseases of modern office workers. This would be our next goal of branching out.

References:

1. Ye, G., Xiang, Q., Yang, L., Yang, J., Xia, N., Liu, Y., & He, T. (2022). Safety Stressors and Construction Workers' Safety Performance: The Mediating Role of Ego Depletion and Self-Efficacy. Frontiers in Psychology, 12, 818955.

2. Liu, Y., Ye, G., Xiang, Q., Yang, J., Goh, Y. M., & Gan, L. (2023). Antecedents of construction workers' safety cognition: A systematic review. Safety Science, 157, 105923.

3- Anagha, R., & Xavier, A. S. (2020). A review on ergonomic risk factors causing musculoskeletal disorders among construction workers. International Journal of Engineering Research, 9, 1234-1236.

4- Health and Safety Statistics. (n.d.). Retrieved October 9, 2022, from https://www.hse.gov.uk/statistics/

5- Valero, E., Sivanathan, A., Bosché, F., & Abdel-Wahab, M. (2016). Musculoskeletal disorders in construction: A review and a novel system for activity tracking with body area network. Applied ergonomics, 54, 120-130.

6- ThePoorEngineer, "ThePoorEngineer," ThePoorEngineer, 28 July 2018. [Online]. Available: https://thepoorengineer.com/en/calibrating-the-magnetometer/#ErrorModel. [Accessed 19 Nov 2022].

7 - Lin, Q., Peng, S., Wu, Y., Liu, J., Hu, W., Hassan, M., ... & Wang, C. H. (2020, April). E-jacket: Posture detection with loose-fitting garment using a novel strain sensor. In 2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN) (pp. 49-60). IEEE.

8 - Derakhshanrad, N., Yekaninejad, M. S., Mehrdad, R., & Saberi, H. (2021). Neck pain associated with smartphone overuse: cross-sectional report of a cohort study among office workers. European Spine Journal, 30(2), 461-467.

9 - Malińska, M., Bugajska, J., & Bartuzi, P. (2021). Occupational and non-occupational risk factors for neck and lower back pain among computer workers: A cross-sectional study. International Journal of Occupational Safety and Ergonomics, 27(4), 1108-1115.