SmartGrip: Low-Impact Grip Strength Analysis

DGMD S-14 Final Project

Abstract

Grip strength is a known biomarker for aging adults, and presents as a benchmark for therapies aimed at regaining or building fine motor skills [1]. However, measurable tracking of baselines and progress using sound metrics are primarily only available in clinical settings. The goal of this project is to provide a low-impact option to measure grip strength for at-home therapies using sensors to analyze squeeze strength and duration. This was done by utilizing machine learning to differentiate between squeeze types and durations across several types of grip strength exercises. The primary model resulted in an accuracy of 85% in determining squeeze types using a two-class averaged perceptron algorithm.

1. Introduction

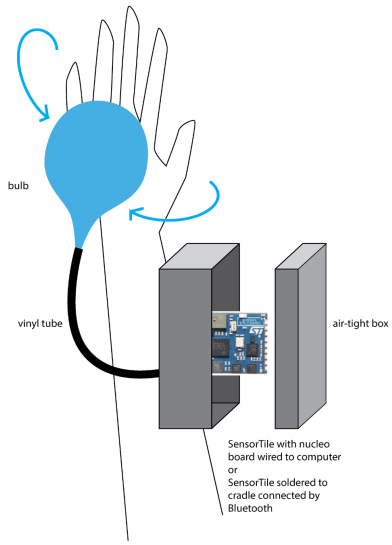
Project Motivation

Therapeutic applications are of particular interest to group members. Jeremiah's young daughter Harper suffers from severe cerebral palsy due an anoxic brain injury in a near drowning accident. Grip therapy exercises have been an integral part of her recovery, as grip strength in conjunction with fine motor skills allow for activities such as play and self-feeding. The application of a device like this, especially if gamified for children, could have far reaching implications for Harper and others in similar therapy settings. CB Lillback has personal experience with strengthening grip via sports such as rock climbing and aerial silks to compensate for mild neurological difficulties with balance. Based on this and on our research, we believe that a SensorTile-equipped device could support people with a variety of conditions.

Prototype Design and Implementation

The pressure sensor on the SensorTile is used to differentiate between different grip strength exercises [2]. A bulb attached to a piece of tubing will run to an airtight box containing the SensorTile. As the user squeezes the bulb, we monitor the change in air pressure and length of the squeeze. The box containing the SensorTile is wired to a computer to obtain data. An application that runs in a computer terminal will prompt the user to perform certain exercises. The user will be asked to perform different pinches and squeezes, and to hold the action for a short(1s) or long(5s) amount of time. The app provides the user with real time feedback on whether they completed the exercise correctly or not.

Prototype.mp4



Pinch Example

Squeeze Example

1.1. Data Collection, Description & Visualization

Data Collection

To collect our test data, we connected the SensorTile to an android phone with the ST BLE sensor app. For each gesture, we ran a 3-minute test. The data for these tests were saved in CSV files and used for model development, and statistical analysis. Each gesture involved manipulating the bulb with a squeeze or pinch. For each grip gesture, we measured the change in pressure for each instance and repeated this calculation over all of the 3-minute tests.

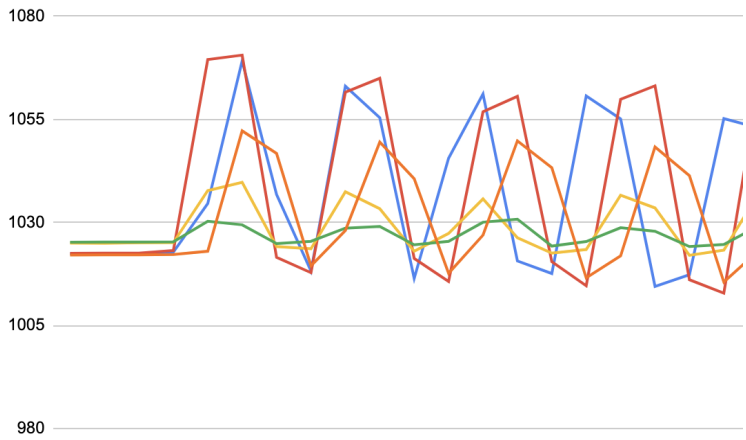
We researched current therapeutic grip exercises, which then inspired the gestures that we chose to test. Our different tests were:

- Fist squeeze
- Two finger pinch
- 4 finger pinch
- Index and middle
- Thumb and pinky

Each test was performed on the dominant and non-dominant hand. These gestures were performed in two different duration periods. One set of tests was taken while each gesture was alternately held for one second and released for one second. The second set of tests was taken while each gesture was alternately held for 5 seconds and released for one second.

Statistical Analysis

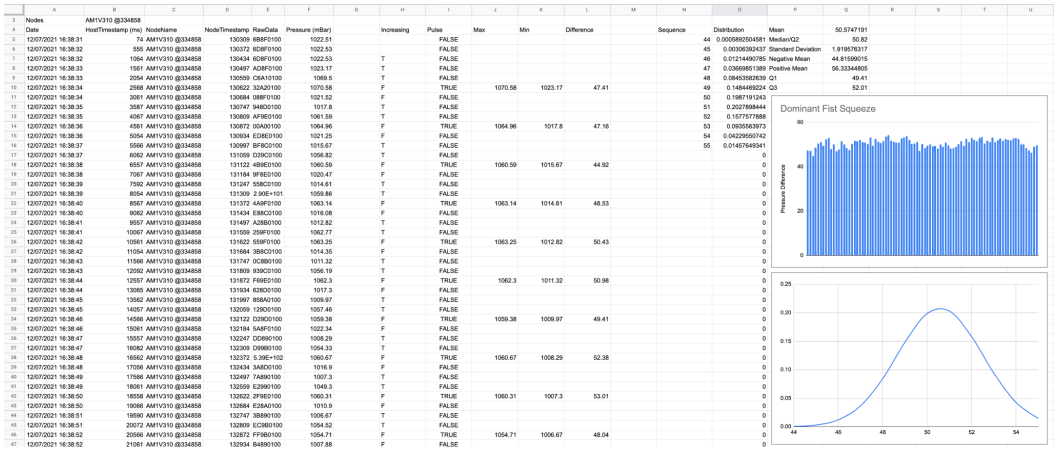
After we obtained data from all of our 3-minute tests, we were able to compare the pressure readings across all the different gestures. The first value that we looked at was the change in pressure during each gesture. If you think of the pressure reading for a single gesture as a pulse, the amplitude is the value that we want.

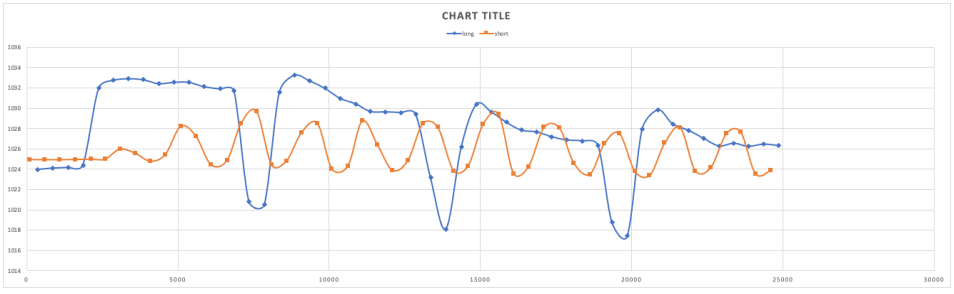


We used Google Sheets to analyze the data and pick out where there were any large changes in pressure. We were then able to single out the minimum and maximum during each rising edge of the pulse. Having all the data displayed in a spreadsheet helped to visually see where each pulse was being recognized.

We then calculated the difference between the minimum and maximum and compared all the gestures. Very early in this process it was clear that some of the gestures had very similar results. Because of this, we decided to focus only on the fist squeeze and the two finger pinch. We also did not see much of a difference between the dominant and non dominant hand, and decided to focus solely on the performance of the dominant hand for further analysis.

After we had all the amplitudes calculated for our decided gestures, we performed a statistical analysis. We calculated the mean, standard deviation, and quartiles, as well as displayed the distribution. This analysis was performed for both short and long gestures. For short gestures, we only had to consider a short pulse with one amplitude calculation. For long gestures, the amplitude calculation was taken twice. Once during the rising edge of the pulse, and again during the falling edge. For the long gestures, there was some loss of pressure over the 5 second duration, and this was normal. To account for this, the same statistical analysis was performed for this drop in pressure as well.





Primary Model Data Preparation

Our main concerns about our data were atmospheric/barometric pressure (mBar), time interval, and data cleaning.

Atmospheric Pressure

As evidenced in the last column in the figure below, all pressures are over 1000mBars, meaning that the data was collected in a place where the atmospheric pressure was about 1000mBars. While building the primary model, we assumed an atmospheric pressure of about 1000mBars (when the read is 1070mBars, it means a squeeze pressure of 70mBars). However, for the application, we first removed the atmospheric pressure. This way, the application can work in any location.

DATA SHOWING ATMOSPHERIC PRESSURE
OF ABOUT 1000mBARs FROM WHERE DATA WAS COLLECTED

A	B	C	D	E	F
		dominant_long_fist			
Log start on	**2021-07-13 17:02:46**				
Feature	Pressure				
Nodes	AM1V310 @334858				
Date	HostTimestamp (ms)	NodeName	NodeTimestamp	RawData	Pressure (mBar)
13/07/2021 17:02:47.011	95	AM1V310 @334858	158749	2B900100	1024.43
13/07/2021 17:02:47.507	591	AM1V310 @334858	158812	29900100	1024.41
13/07/2021 17:02:48.017	1101	AM1V310 @334858	158874	38900100	1024.56
13/07/2021 17:02:48.514	1598	AM1V310 @334858	158937	3B900100	1024.59
13/07/2021 17:02:49.009	2093	AM1V310 @334858	158999	43900100	1024.67
13/07/2021 17:02:49.516	2600	AM1V310 @334858	159062	9B910100	1028.11
13/07/2021 17:02:50.014	3098	AM1V310 @334858	159124	80A50100	1079.04

For the model training

We assumed a 1000 atmospheric pressure

1070 read -> meant squeeze pressure of 70

For the application

Remove atmospheric pressure

Model can work on any geographic location

Time Interval

Initially, we tried using the average from 5 second intervals of movements. But, sometimes, it gave us overlapping averages (which would not result in a good model). For this reason, we then tried 10 seconds intervals of movements. This gave us different values for long and short movements (there is almost an overlapping, but not quite):

- Short movements averages at around 35, but the averages ranged from about 28 to 40
- Long movements averages at around 45, but the averages ranged from about 40 to 55

Data Cleaning

Data cleaning was not really necessary because first we chose to collect only data that would be used to build the model, and second, since there is always some pressure happening between the squeezes, eventual absence of pressure was not the case.

Cleaning Data (not really necessary)

1. Data collected pressure (mBar) → there was always some pressure
2. Data was already collected with the necessary data

dominant_long_fist

Log start on	2021-07-13 17:02:46				
Feature	Pressure				
Nodes	AM1V310 @334858				
Date	HostTimestamp (ms)	NodeName	NodeTimestamp	RawData	Pressure (mBar)
13/07/2021 17:02:47.011	95	AM1V310 @334858	158749	2B900100	1024.43
13/07/2021 17:02:47.507	591	AM1V310 @334858	158812	29900100	1024.41
13/07/2021 17:02:48.017	1101	AM1V310 @334858	158874	38900100	1024.56
13/07/2021 17:02:48.514	1598	AM1V310 @334858	158937	3B900100	1024.59
13/07/2021 17:02:49.009	2093	AM1V310 @334858	158999	43900100	1024.67
13/07/2021 17:02:49.516	2600	AM1V310 @334858	159062	9B910100	1028.11
13/07/2021 17:02:50.014	3098	AM1V310 @334858	159124	80A50100	1079.04
13/07/2021 17:02:50.508	3592	AM1V310 @334858	159187	8CA40100	1076.6
13/07/2021 17:02:51.034	4118	AM1V310 @334858	159249	68A30100	1073.68
13/07/2021 17:02:51.514	4598	AM1V310 @334858	159312	97A20100	1071.59

2. Related Work

Medical Internet of Things

The medical Internet of Things (IoT) includes wearables like the Apple Watch that catch potential health problems as they arise. One example is detection of falls. Manual grip strength is important in the prevention of falls [3].

Smart Ball for Serious Games and Therapy

An IoT-enabled squeeze grip interface, the Squegg, counts and times squeezes for simple “serious games”[4] that use grip as a mouse click within their app [5].

3. Team Organization

3.1. Team Members & Roles

Ashleigh Kenworthy - Programming (terminal app), data acquisition, prototype
Cara-Beth Lillback - Ideation, video editing, and technical writing
Jeremiah Candelaria - Programming (model training), model validation
Isabely Ohata Ataide - Programming (model training), data preparation, model validation

4. Software & Developing Tools

4.1. Software

- STM32CubeIDE
- ST BLE Sensor app on Android
- Jupyter/Google Colab Notebooks
- Azure Machine Learning
- Edge Impulse
- Google Drive
- GitHub

4.2. Laptop/Desktop Setup

- MacBook Air M1 macOS Big Sur version 11.4
- Anker USB-C Hub A8331

4.3. Sensor/Hardware

- STMicroelectronics SensorTile
- STMicroelectronics Nucleo-64 Board
- Android phone with Edge Impulse
- Ait-tight container
- Vinyl tubing
- Bulb Aspirator

5. List of Milestones

Meeting Schedule: Weekdays 2:00-2:30 EST

- **Week 1** 6/20: Procure equipment, set up environments
- **Week 2** 6/27: Begin sensor tutorials, brainstorm projects, form team
- **Week 3** 7/04: Create proposal, build prototype, research
- **Week 4** 7/11: Proposal, create test plan, gather data
- **Week 5** 7/18: Program development, model training
- **Week 6** 7/25: Write report, film presentation
- **Week 7** 8/01: Presentations

6. Results & Discussion

Implementation of Primary Model

Machine Learning in the Cloud

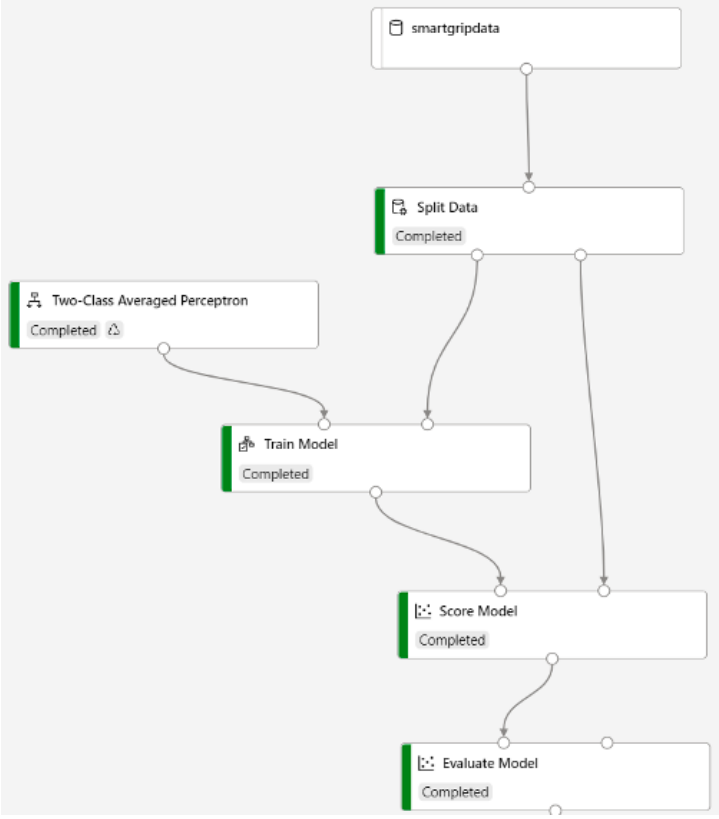
First, we compared the concepts of a model hosted in the cloud versus a model hosted in a firmware or application. When a model is in the cloud, the application is sending the data from the sensor to the cloud and getting a response back. Below there is a comparison between a model hosted in the cloud versus a model hosted in a firmware or application.

Model Hosted in the Cloud	Model Hosted in Firmware or Application
Once the model is updated, every user will have the same version	Models needs to be updated in each user's firmware or application (users can have different versions)
Model can be updated anytime	Model usually updates when most users are not active (overnight)
Model can be updated as often as needed	Model is usually updated less often (inconvenient process)
Need internet for most updated version But when there's no internet can leverage a copy of the model hosted on the firmware or application software	Doesn't require constant internet connection (simpler device)
Cloud providers (AWS, Azure, GCP) deliver scalability – future of SmartGrip	Cheaper

Primary Model Training on Azure Machine Learning

Azure Machine Learning is a cloud-based environment to train, deploy, automate, manage, and track Machine Learning models. It was the tool used to develop our primary model [6].

Pipeline: This is the pipeline for the primary model that will be explained in detail.



Full data Set (Random Data Split): from the full data set, which had 100 averages (50 longs and 50 shorts), we randomly split the data in 60/40, being 60 for the training data and 40 for test data.

FROM FULL DATA SET --> RANDOM DATA SPLIT

100 averages (50 longs and 50 shorts)
Training Data / Test Data
60 / 40

LONG

smartgripdata

10spressuresum	classification
50.855085	long
49.854985	long
46.90469	long
55.155515	long
59.855985	long
40.0042	long
45.154515	long
44.80448	long
43.80438	long
47.30473	long

SHORT

smartgripdata

10spressuresum	classification
31.69792448	short
34.53865368	short
31.47247693	short
33.14078875	short
34.53865368	short
36.47741238	short
36.20687533	short
33.45641531	short
31.60774546	short
30.97649234	short

Algorithm: two-class averaged perceptron algorithm

The algorithm used is the “two-class averaged perceptron”. This algorithm is a very simple version of a neural network (suited to learning linearly separate patterns). The inputs are classified into possible outputs based on a linear function. Inputs are then combined with a set of weights derived from the feature vector, hence the name “perceptron”.

It is also important to mention that, initially, we attempted several models, including logistic regression and a multi-class algorithm for short, long and wrong movements, but the results were not satisfactory. Therefore, we chose to use the two-class averaged perceptron for this project, considering we had a short amount of time to conclude the project. However, in the future, a multi-class algorithm will be a better option for updating the model.

Primary Model Test on Azure Machine Learning

Test Set Score - Scoring

The test set score is composed of pressure, the correct classification, the predicted classification and the certainty level. As we can see in line 5, when the pressure was about 37.8, this was a short movement, but the model labeled it as long, with a probability of around 46%. Overall, the model performed correctly in most cases.

SCORING - Test Set Scores

10spressuresum	classification	Scored Labels	Scored Probabilities
44.121101	long	long	0.263759
46.222106	long	long	0.210782
40.005769	short	long	0.389064
31.607745	short	short	0.6732
37.788462	short	long	0.464735
34.269802	short	short	0.586756
42.748633	long	long	0.302659
45.480769	long	long	0.228537
37.10371	short	long	0.488608
29.497212	short	short	0.734529
38.30393	short	long	0.446863

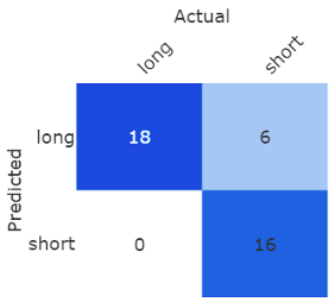
Evaluation - Accuracy

The accuracy of the primary model is 85%. The comparison between predicted versus actual is satisfactory, with all the long movements predicted correctly and about 2/3 of the short movements predicted correctly.

EVALUATION FROM TRAINED MODEL WITH TEST SET

Accuracy of 85%

Accuracy	0.85
Precision	1
Recall	0.727
F1 Score	0.842
AUC	1



Primary Model Deployment on Azure Machine Learning

The model was deployed in a docker container inside Azure. The deployed model is wrapped in a REST API, this way all devices and applications that have the key to the REST API can call the model through the web. Since our model is in the cloud, it is called by our application to classify the squeezes.

Primary Model Use on Azure Machine Learning

When our model is called, it sends a JSON response. Here we can see that in the first response, when the movement is very easy to identify (for example, a very clear short movement of 25 mBars), the certainty level is high (about 83%). However, when the movement is in a grey area (for example, a pressure of 40mBars), even though the model predicted the correct movement, the certainty level is smaller (about 39%).

When model is called -> model sends a JSON response


```
{
  "Results": {
    "WebServiceOutput0": [
      {
        "10spressuresum": 25,
        "classification": "-",
        "Scored Labels": "short",
        "Scored Probabilities": 0.8384010659394607
      }
    ]
  }
}
```


When movement is clear (very low pressure)

Model has a high accuracy rate


```
{
  "Results": {
    "WebServiceOutput0": [
      {
        "10spressuresum": 40,
        "classification": "-",
        "Scored Labels": "long",
        "Scored Probabilities": 0.3892554138557037
      }
    ]
  }
}
```


When movement is in grey area

Model has a very low accuracy rate

Rest API in Jupyter Notebook

We used Jupyter notebooks to loop through a rest API that calls the Azure machine learning model. In the SensorTile main program, there is an “AzureSensing” function that calculates an average pressure over 10 seconds and prints the value. The function then waits 5 seconds and repeats the 10 second average. The Jupyter notebook uses python code to read this average value as serial data and sends it to the Azure Machine Learning model.

Terminal Application

The app uses the ALLMEMS1 project for its base code. The reason for this being that we can easily read the sensor tile data through the ST BLE Sensor application on an Android phone.

A “PressureSensing” function was created in the SensorTile main program to handle the processing of data from the user. This function works as a state machine. It commands the user to perform different grip exercises and provides real time feedback in a terminal window.

There are 4 different parts to the app’s exercise routine.

First the app will prompt the user to perform 10 short 1 second fist squeezes and then 10 short two finger pinches. The app will provide feedback on whether the user applied the right amount of pressure. The feedback can read “perfect pressure,” “pressure low,” “pressure too low,” “pressure high,” and “pressure too high.”

```
1664            // Print z-score
1665            ALLMEMS1_PRINTF("Z-Score=%ld\r\n",zscore);
1666            *HighPress=0;
1667
1668            // If z-score is  within +- 3 standard deviations, pressure is perfect
1669            if ((zscore >= -3) && (zscore <= 3)) {
1670                ALLMEMS1_PRINTF("Perfect Pressure!\r\n");
1671            }
1672
1673            // If z-score is within -10 to -3 standard deviations, pressure is low
1674            else if ((zscore < -3) && (zscore >= -10)) {
1675                ALLMEMS1_PRINTF("Good, Pressure Low\r\n");
1676            }
1677
1678            // If z-score is within +10 to +3 standard deviations, pressure is high
1679            else if ((zscore > 3) && (zscore <= 10)) {
1680                ALLMEMS1_PRINTF("Good, Pressure High\r\n");
1681            }
1682
1683            // If z-score is less than -10 standard deviations, pressure is too low
1684            else if (zscore < -10) {
1685                ALLMEMS1_PRINTF("Pressure Too Low\r\n");
1686            }
1687
1688            // If z-score is greater than +10 standard deviations, pressure is too high
1689            else if (zscore > 10) {
1690                ALLMEMS1_PRINTF("Pressure Too High\r\n");
1691            }
```

Then the app will move onto the long 5 second gestures. 10 fist squeezes first, and then 10 two finger pinches next. The feedback for these exercises are the same as previously stated, except the app will also monitor the duration of the gesture and look to see that pressure is being maintained for the whole 5 second duration. If the pressure drops more than 10 millibars, the app recognizes the grip as being released and will respond with “grip too short.” If the pressure drops between ~4 and 10 millibars, the grip pressure is not being maintained and the app will respond with “pressure not maintained.”

```
1693            // If pressure drops rapidly, grip is released. If timer is less than 5 seconds, show error.
1694            if ((*ExType > 1) && (*TimeCount < 5000) && (PressRead < *PrevPress - 1000)) {
1695                ALLMEMS1_PRINTF("Grip Too Short! Please Hold For 5 Seconds.\r\n");
1696            }
1697
1698            // If pressure is not held during long squeeze, show error
1699            if ((*ExType == 2) && (*TimeCount < 5000) && (PressRead > *PrevPress - 1000)) {
1700                ALLMEMS1_PRINTF("Squeeze Pressure Not Maintained.\r\n");
1701            }
1702
1703            // If pressure is not held during long pinch, show error
1704            if ((*ExType == 3) && (*TimeCount < 5000) && (PressRead > *PrevPress - 1000)) {
1705                ALLMEMS1_PRINTF("Pinch Pressure Not Maintained.\r\n");
1706            }
```

The pressure sensing function retains information for one gesture at a time. Along with monitoring the current pressure, it also retains a high and low pressure value, the index of the current pinch or squeeze, the type of exercise that was prompted to the user, and a timer.

For the functionality of the app, we used the mean, standard deviation, Q1, and Q3 measurements from the statistical analysis. When the user performs an exercise, the app registers it by looking for a pressure increase greater than 5 millibars. The lowest pressure before this increase is saved as the minimum value. The app continues to look for increasingly higher values. Once the pressure starts dropping again, the app saves the maximum value.

For a short exercise, the app then immediately calculates a z-score to determine how close or far the user is from the recorded mean. To calculate the z-score we first find the difference between the maximum and minimum that we just read, subtract the mean, and then divide by the standard deviation. A positive z-score shows that the user applied a pressure higher than the mean, and a negative z-score shows that the user applied a pressure lower than the mean. For a long exercise, the z-score is calculated when the exercise is completed.

```
1628            // Compute z-score for short squeeze
1629            // mean = 5057
1630            // standard deviation = 192
1631            if (*ExType == 0) {
1632                zscore = (*HighPress - *LowPress - 5057)/192;
1633                ALLMEMS1_PRINTF("Looking for Short Squeeze... ");
1634                *SqueezeID += 1;
1635            }
1636
1637            // Compute z-score for short pinch
1638            // mean = 3149
1639            // standard deviation = 97
1640            else if (*ExType == 1) {
1641                zscore = (*HighPress - *LowPress - 3149)/97;
1642                ALLMEMS1_PRINTF("Looking for Short Pinch... ");
1643                *PinchID += 1;
1644            }
1645
1646            // Compute z-score for long squeeze
1647            // mean = 5214
1648            // standard deviation = 315
1649            else if (*ExType == 2) {
1650                zscore = (*HighPress - *LowPress - 5214)/315;
1651                ALLMEMS1_PRINTF("Looking for Long Squeeze... ");
1652                *SqueezeID += 1;
1653            }
1654
1655            // Compute z-score for long pinch
1656            // mean = 3320
1657            // standard deviation = 111
1658            else if (*ExType == 3) {
1659                zscore = (*HighPress - *LowPress - 3320)/111;
1660                ALLMEMS1_PRINTF("Looking for Long Pinch... ");
1661                *PinchID += 1;
1662            }
1663
```

Based on this score, the app will provide appropriate feedback to the user. A perfect pressure is between 3 standard deviations plus or minus from the mean. A good high or low pressure is between 3 and 10 standard deviations plus or minus from the mean. A pressure that is too high or too low is outside 10 standard deviations plus or minus from the mean.

For a long exercise we also have to look at the time the gesture is being held and if the pressure is being maintained. If the exercise is held for less than 10 seconds or if the pressure drops too low, the app will stop monitoring the pressure and it will then provide the user with an error. During the 5 second exercises, the pressure will drop slightly, and this is normal. We have the mean and quartiles of this pressure drop. The app will only send the error if the pressure drops outside of the Q3 value.

Once all the exercises are done, the app will loop back to the beginning.

```
Press Detected! Change in Pressure=5739 Looking for Short Squeeze... Z-Score=3
Perfect Pressure!
Short Squeeze test #4
Short Squeeze test #4
Short Squeeze test #4
Short Squeeze test #4
Short Squeeze test #4
Short Squeeze test #4
Short Squeeze test #4
Press Detected! Change in Pressure=5467 Looking for Short Squeeze... Z-Score=2
Perfect Pressure!
Short Squeeze test #5
Short Squeeze test #5
Short Squeeze test #5
Short Squeeze test #5
Short Squeeze test #5
Short Squeeze test #5
Short Squeeze test #5
Press Detected! Change in Pressure=4890 Looking for Short Squeeze... Z-Score=0
Perfect Pressure!
Short Squeeze test #6
Short Squeeze test #6
Short Squeeze test #6
Short Squeeze test #6
Short Squeeze test #6
Short Squeeze test #6
Press Detected! Change in Pressure=5492 Looking for Short Squeeze... Z-Score=2
Perfect Pressure!
Short Squeeze test #7
Short Squeeze test #7
Short Squeeze test #7
```

Secondary Model

In this section we describe a secondary approach we explored to train a logistic regression model to classify long and short squeezes.

Secondary Model Data Preparation & Cleaning

	pressure	short_grip
0	1024.43	0
1	1024.41	0
2	1024.56	0
3	1024.59	0
4	1024.67	0
...	...	...
737	1006.71	1
738	1007.64	1
739	1008.29	1
740	1008.94	1
741	1009.42	1

742 rows × 2 columns

For this secondary approach, we were concerned with the pressure and grip length (short vs. long). In this attempt, we chose to read in 371 raw data points from the long and short squeezes, and did not account for atmospheric pressure. We instead chose to preprocess and scale the data using the StandardScaler module from scikit-learn. We stacked the data in a single table and labeled each pressure as either short grip (1), or long grip (0).

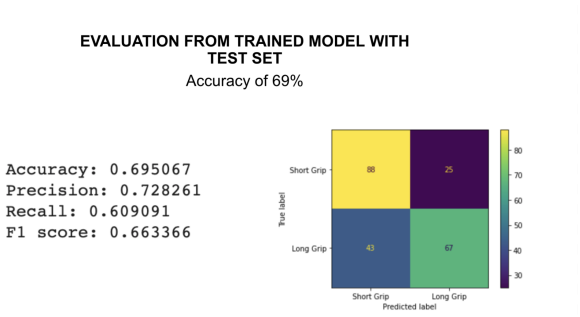
Secondary Model Training

Next, we shuffled and re-indexed the data, splitting it to train and test, using a 70/30 ratio, and utilizing a random state of 43. As expected, our baseline accuracy after initial training was 50%:

```
baseline_accuracy = max(y_train.value_counts(normalize=True))
baseline_accuracy
```

```
0.5009633911368016
```

Finally, we trained the data using scikit_learn's LogisticRegression, ultimately receiving an accuracy of 69%. We could have further cleaned the data and tuned this model, but given a shortage of time and the low accuracy results of the regression model, we chose the two-class averaged perceptron algorithm as our primary model to complete the project.



7. Conclusions

The goal of this project was to utilize machine learning to differentiate between different squeeze types and durations across several grip strength exercises. Our team was successful in that endeavor. Given the short window of time to complete this project, we are satisfied with the results achieved using the two-class averaged perceptron algorithm. However, given more time, we would have liked to explore other models and expand features to differentiate between even more gestures and squeezes. Considering the ideas and next steps of the future work section, it would be possible to use this project as the seedling for a startup or a capstone project. Our final conclusion is that it would be possible to bring to market a low-impact product to measure grip strength for at-home therapies using sensors to analyze squeeze strength and duration.

8. Future Work

With regard to future work and further application, these are several next steps to explore. One option would be to explore different grip pressure standards based on age, gender, and disability. We could continue to utilize the cloud, periodically uploading the new features from the data collected to retrain the model over time. We could also make several modifications to our project, including the prototype, model, and app. To modify the prototype, we could add a glove containing sensors in the fingers to be able to correctly identify different grips and squeezes performed. We could use that data to further tune the model, or even explore different models, such as a support vector classifier. Finally, we could gamify our current application terminal. Creating a smartphone application, we can add games, such as racing, or navigating a maze, all using grip pressure. This would allow at-home therapies for building grip strength to be fun and engaging.

9. References

[1] Bohannon, Richard. Grip Strength: An Indispensable Biomarker For Older Adults. October 2019, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6778477/

[2] SensorTile, from STMicroelectronics. https://www.st.com/en/evaluation-tools/steval-stlkt01v1.html

[3] Arvandi, M. Strasser, B. Volaklis, K. [...]. Mediator Effect of Balance Problems on Association Between Grip Strength and Falls in Older Adults: Results From the KORA-Age Study. March, 2008. https://journals.sagepub.com/doi/10.1177/2333721418760122

[4] Lunardini, F. Borghese, N. Piccini, L. [...]. Validity and usability of a smart ball–driven serious game to monitor grip strength in independent elderlies. January, 2020. https://journals.sagepub.com/doi/10.1177/1460458219895381

[5] Squegg smart grip trainer. https://mysquegg.com/

[6] Azure Machine Learning official documentation by Microsoft. https://docs.microsoft.com/en-us/azure/machine-learning/