MATLAB
Fitness Tracker App
Report

Group 17: Kevin, Tenzin, and Justin

Abstract/Summary

a. We developed an app that tracks both the user's fitness activities and food intake throughout the day, allowing them to create a personalized plan to help reach their goals. Our motivation stemmed from the understanding that fitness and diet are crucial elements in achieving objectives such as weight loss, strength building, or enhancing longevity. The technology stack we employed consisted primarily of MATLAB for the backend and App Designer for the graphical user interface. One of our main challenges was training our models to accurately classify activities. We addressed this by utilizing MATLAB Mobile to collect data on the transition from walking to running on various types of terrain. Ultimately, the project culminated in a functional app that not only presents valuable data regarding fitness and diet but also provides users with a tailored plan to guide them on their journey to success.

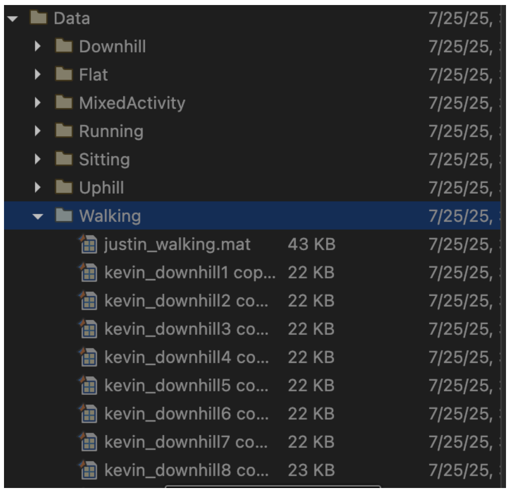
Goals of the Project

b. The first goal for our group is to develop a functional app that mimics a typical fitness tracker available on the App Store. This requires us to determine which technology stacks we'll use and the data to display to users.

c. Another goal is to become skilled in the machine learning process. This involves several key steps: selecting the appropriate data to gather, training a model with that data, evaluating the model's performance, and conducting extensive testing to confirm its reliability and accuracy.

Inputs for data

d. Our project relied on sensor data collected through the MATLAB Mobile app, focusing on acceleration and position measurements. Using this setup, we recorded approximately 100 activity sessions spanning various movement types and environments, including standing, walking, and traversing flat, downhill, and uphill terrain. This diverse dataset served as the foundation for training our models. Developing efficient and accurate classification models requires substantial and varied input data.



```
▾ Data                                    7/25/25,
  ▸ Downhill                              7/25/25,
  ▸ Flat                                  7/25/25,
  ▸ MixedActivity                         7/25/25,
  ▸ Running                               7/25/25,
  ▸ Sitting                               7/25/25,
  ▸ Uphill                                7/25/25,
  ▾ Walking                               7/25/25,
      justin_walking.mat      43 KB       7/25/25,
      kevin_downhill1 cop...  22 KB       7/25/25,
      kevin_downhill2 co...   22 KB       7/25/25,
      kevin_downhill3 co...   22 KB       7/25/25,
      kevin_downhill4 co...   22 KB       7/25/25,
      kevin_downhill5 co...   22 KB       7/25/25,
      kevin_downhill6 co...   22 KB       7/25/25,
      kevin_downhill7 co...   22 KB       7/25/25,
      kevin_downhill8 co...   23 KB       7/25/25,
```

Why we collected

e. Acceleration was used to classify different segments of the workout (as either running, walking, or sitting) and determine the intensity of the exercise.

f. Position was used to calculate the total number of steps taken, the total distance traveled, and to classify different segments of the workout (as uphill, downhill, and flat).

How we collected

g. All members of our group recorded data through the MATLAB Mobile app. Each member recorded several different activities at different intervals of time. They included various forms of movement at different time intervals-for example, a 1-minute run, a 5-minute hike, and a 10-minute walk.

Samples of Data

h. Video demo of how to run the program

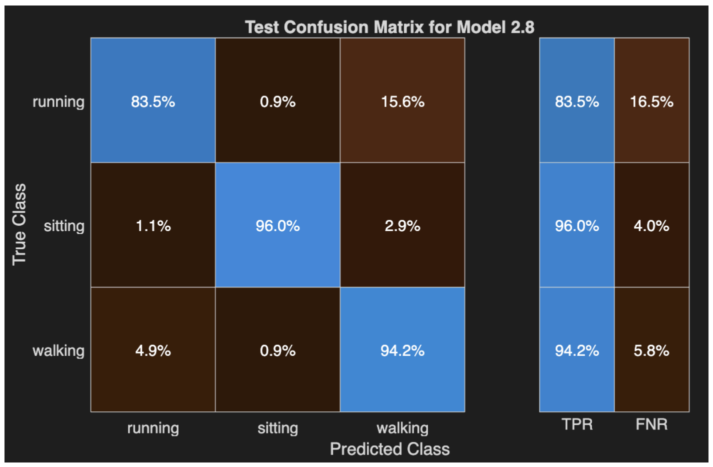
Methodology -why we chose those models, show measurements of how well classification worked (how accurate the testing data was)

i. To classify our data, we utilized MATLAB's Machine Learning tool, specifically the Classification Learner. We trained and validated our models on acceleration datasets using 10-fold cross-validation, while reserving 20% of the data for testing purposes. We will explain how we trained 3 models for our project.

Acceleration: Our most successful model was the K-Nearest Neighbors (KNN) algorithm, which achieved an accuracy of 90% on the testing dataset. The KNN algorithm proved effective due to its ability to cluster data points, facilitating the formation and classification of different types.

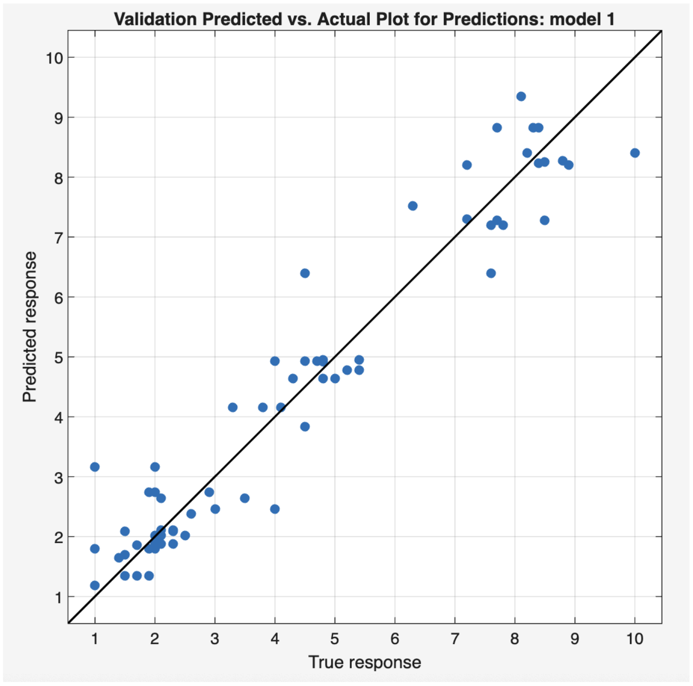
During the training process, a significant discrepancy arises from the model's tendency to misclassify a substantial amount of running data as walking, resulting in an error rate of 15.6%. This issue likely stems from the overlapping acceleration patterns exhibited during both activities. To enhance the model's ability to distinguish between running and walking, it is essential to collect additional data. By doing so, we can reduce the overall error and improve the model's accuracy.

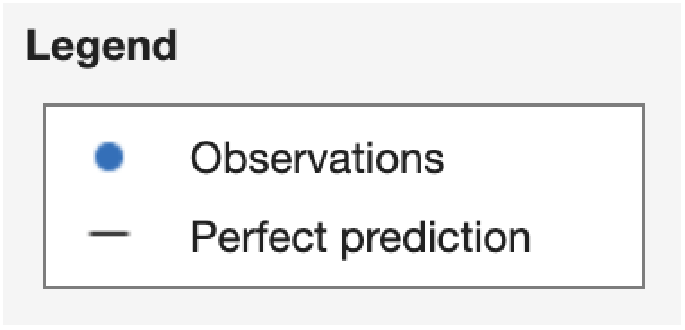
Overall, our training and validation process yielded an accuracy of 90.5% for the KNN model. The final step of the training process involved testing the remaining 20% of the original dataset for validation. This testing yielded an accuracy of 90.3%, demonstrating only a slight drop of 0.2%. This indicates that the model is capable of accurately classifying most activities as either running, walking, or sitting.



Altitude: For the altitude model, we employ the same approach, utilizing a 10-fold validation method that involves setting aside 20% of the data. After training and testing the data, we achieved a 99.2% accuracy. This means our model will accurately identify

Intensity: Our final model was created using MATLAB's Regression Learner toolkit, which was used to determine the intensity (Global). We aimed to develop a consistent, objective-based intensity score, where, given a set of 7 workout features, an intensity value ranging from 1 to 10 would be assigned. We compared several models but found the highest-performing model to be a Fine Tree Regression. Validations: R-Square: 0.93, RMSE: 0.74879, MSE: 0.5723, MAE: 0.57230, MAPE: 18.7%. This data was trained using over 70 labeled workouts with seven predictor variables: Step, Duration, Distance, AvgPace, MaxAccel, and AltGain. These features were selected for their correlation with perceived workout effort. The model was also trained using 6-fold cross-validation, and 10% of the data was used for training to combat underfitting and overfitting. Example results include: a 4.2-mile hike with over 800 ft of elevation gain might result in an 8 or 9, whereas a shorter 0.5-mile jog might score a 4. These values are displayed on the graph for the user's use and in the console for app development debugging.



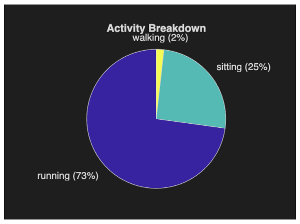
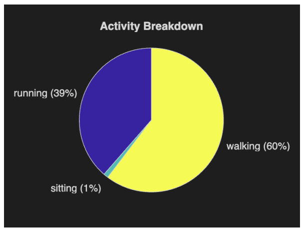
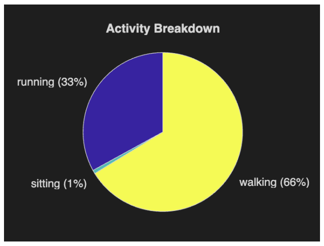
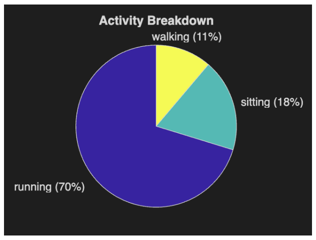
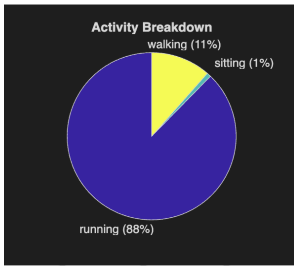
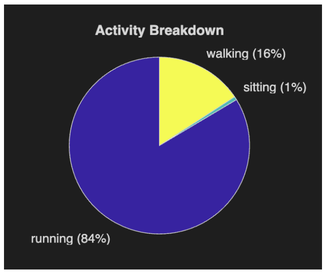


Methods/Results

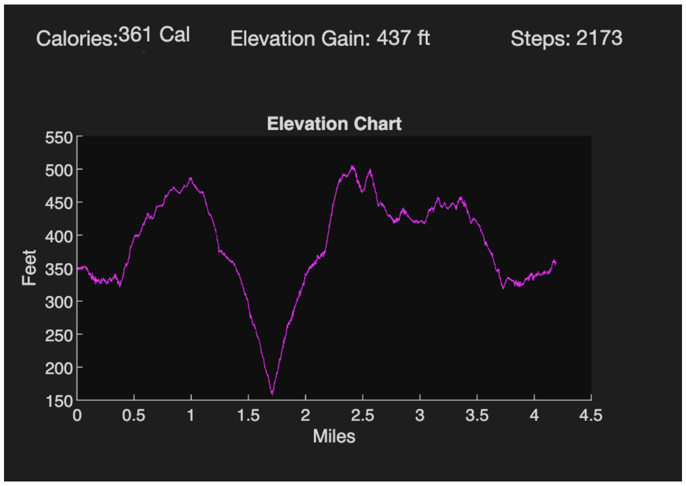
The core aspect of the project involves training a model to distinguish between different segments of a workout and accurately classify their characteristics. We developed three models for this project.

1. Trained Model on Acceleration: The first model focuses on various accelerations to differentiate between running, walking, and sitting. This enables us to inform users of the percentages of their workout associated with each activity. To effectively train the model, it's crucial to understand the data being gathered. Acceleration consists of three components: the x, y, and z directions. We must consider changes in acceleration not only

in the horizontal plane but also in the vertical plane. In our previous tests, we compared the classification results of both the exampleTrainedModel and the team's model, which was specifically trained on running, walking, and sitting on flat terrain. We observed a notable increase in accuracy when considering the output of the workout between the example model given and the first version of our trained model.

Workout	exampleTrainedModel	firstTrainedModel (trained on data collected only on flat terrain)	TrainedModel (trained on data on all terrains: uphill, downhill, and flat)
Hiking (4.2 mi)			
Longrun (5 mi)			

In our initial analysis of the hiking data, most of the walking segments were mistakenly classified as running. This was due to the model not being trained to recognize the differences in acceleration changes along the vertical axis, which are crucial for distinguishing between running and walking. After gathering new data by recording both running and walking activities on uphill and downhill terrains, we retrained our model. This allowed us to achieve a more accurate classification of the segments in our latest model.



2. Trained Model on Altitude: One of our goals was to accurately calculate the total elevation gain, only considering the ascent when

the user is moving uphill. While a traditional iterative approach could analyze each segment and sum any positive slopes, the variability of our recorded data, characterized by natural fluctuations in elevation, introduces noise that complicates this calculation. To address this, rather than relying on a conventional algorithm to identify when the user is ascending, we employed machine learning techniques to enhance

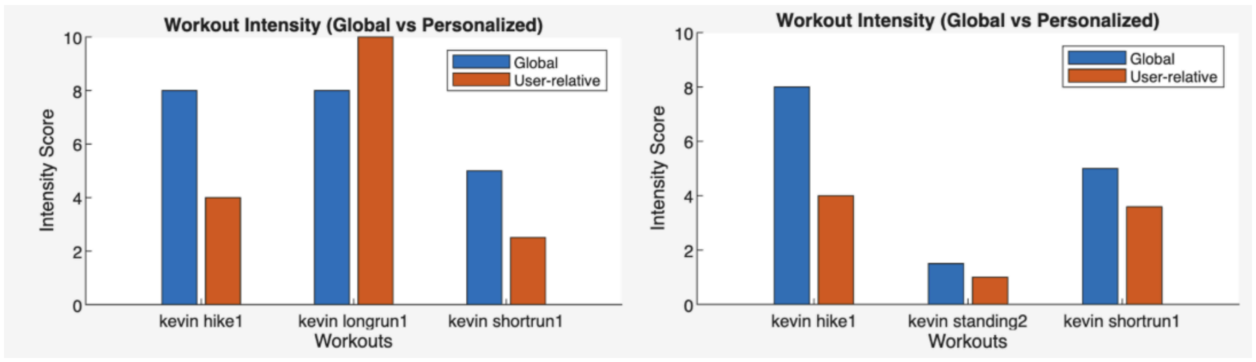
decision-making and learn from the collected data. We incorporated a diverse dataset that included recordings from uphill, downhill, and flat terrain, encompassing a range of activities, including running, walking, and standing. This trained model enabled our team to achieve precise measurements and visual representations of elevation gain.

3. In addition to the global intensity, our app also provides users with a personalized relative intensity score that adjusts to each user's workout history. This value is created by a custom function that computes the z-score for a new workout using the mean and standard deviation of previous sessions (across the seven workout features). Calculations are found as:
    - Given that the user has no workout history, a baseline of 4 is automatically assigned.
    - Z-scores are calculated for each feature, with only positive deviations from the user average being considered (no penalty for easier workouts)
    - Each feature is weighted (with number of steps and duration given more influence), and a non-linear scaling is applied to map the final value on a scale of 1-10.

   Whereas the Global Intensity value for each workout is based on the Regression model, the Relative Intensity value is dynamically calculated based on user-specific data. This ensures that short but intense workouts can still be rated as highly challenging for beginners, while the same session may score low for a seasoned athlete.

    - A workout that once resulted in a relative intensity value of 9 may later drop to a 7 or 6 if the user improves.
    - On the other hand, a plateau in relative intensity value may indicate consistent training intensity.

   Combining a Global Regression Model with a personalized relative intensity function enables a comprehensive performance tracking metric, allowing users to both benchmark their progress and understand the effort level of each workout in context.



To help user better interpret how their workouts are evaluated-both objectively and personally, our app used Global and User-Relative Intensity scores side by side.

In the Workout Intensity Graphs above, we can see our Left graph having a more active history. This can be seen in the first two challenging workouts, “kevin hike1” and “kevin longrun1” both with a high global intensity score (~8). Because of this activity history, the relative score for the third workout “kevin shortrun1” is significantly lower (~3), since it displays a drop in effort compared to the baseline.

Whereas the right graph, where the user has also completed “kevin hike1” but also had a less strenuous workout session, “kevin standing2”. Because of this activity history, the relative score for the third workout “kevin shortrun1” is higher than the left (more activity history) graph. This right graph’s third workout has a relative intensity score of ~4, since it was more intense compared to the previous norm.

This example highlights the key feature of our relative intensity algorithm: it dynamically adjusts based on the user’s recent training history. The same workout can be “easier” or “harder” depending on what the user has been doing–a principle grounded in real-world training load and recovery concepts.

Conclusion

In conclusion, our team successfully developed a functional fitness tracking app that integrates both workout and nutrition monitoring while leveraging machine learning to deliver personalized insights. Through the use of MATLAB’s Classification Learner and Regression Learner, we created accurate models capable of classifying user activities, estimating elevation gain, and generating both global and personalized intensity scores. By collecting and analyzing real-world sensor data, we overcame challenges such as distinguishing between similar movement patterns and built models that achieved high accuracy, including 90% for activity classification and 99% for altitude detection. This project not only provided us with valuable experience in machine learning and app development but also resulted in a working tool that can help users better understand their fitness patterns and make informed decisions toward achieving their health and performance goals.