Aaron Liu
Prof. Ghosh
CSCI 183
12/3/2023

The Problem:

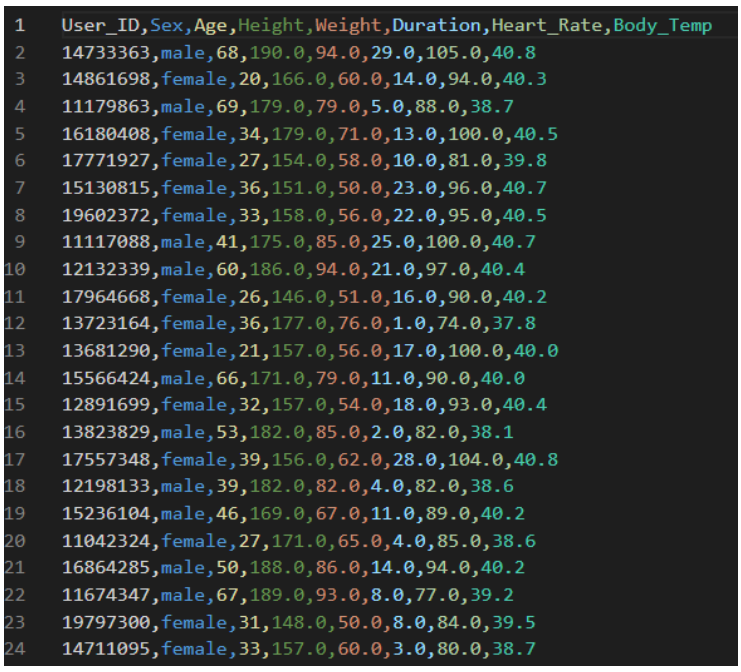
I like to hike to take a break from computer science classes, so I sometimes wonder how I could combine this interest of mine with this major of mine. Utilizing concepts I learned from this class, I decided to implement a more data-driven approach to estimating the various aspects of hiking on trails. Namely, given data on a hiking trail path, I would like to estimate the time, calories burnt, and difficulty of the trail.

Background:

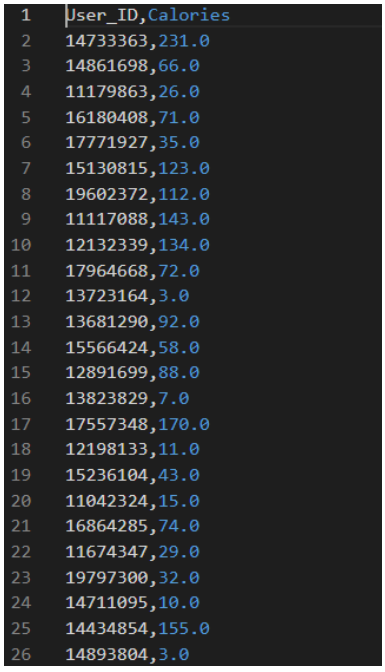
Given the problem, there is only a bit of research done concerning hiking speed. Andrew Wood, William A. Mackaness, and contributors from the University of Edinburgh published *Improved prediction of hiking speeds using a data driven approach* in March of 2023, improving upon preexisting methods such as Naismith's Rule, Langmuir's Rule, and Tobler's Hiking Function, and does so by considering both the walking and hill slopes for a path. To best determine the hill slopes in Digital Elevation Models, a study by Mathew Dunn and Robert J. Hickey in June 1998 measured the most effective methods to measure hill slopes.

I collected two types of data for my project; the first is a .csv collection of .gpx files scraped from hikr.org, which is a special format of .xml files optimized for tasks relating to geographical and navigational services. The collection includes each original .gpx file as a string with partially computed data from using the gpxpy library, with examples such as length, maximum elevation, and trail difficulty (on a 1-6 scale). The generated features were only partially useful, and I needed to parse the .gpx files to extract more useful information about the trails. The second is a dataset on Kaggle.com about the relation between body type, exercise duration, and calories burnt. It is a .csv file containing easily interpreted quantitative and qualitative data for inputting into a machine learning model. Both datasets contain more than

12000 samples. Figure 1-3 provides snapshots of the raw datasets.



```
User_ID,Sex,Age,Height,Weight,Duration,Heart_Rate,Body_Temp
14733363,male,68,190.0,94.0,29.0,105.0,40.8
14861698,female,20,166.0,60.0,14.0,94.0,40.3
11179863,male,69,179.0,79.0,5.0,88.0,38.7
16180408,female,34,179.0,71.0,13.0,100.0,40.5
17771927,female,27,154.0,58.0,10.0,81.0,39.8
15130815,female,36,151.0,50.0,23.0,96.0,40.7
19602372,female,33,158.0,56.0,22.0,95.0,40.5
11117088,male,41,175.0,85.0,25.0,100.0,40.7
12132339,male,60,186.0,94.0,21.0,97.0,40.4
17964668,female,26,146.0,51.0,16.0,90.0,40.2
13723164,female,36,177.0,76.0,1.0,74.0,37.8
13681290,female,21,157.0,56.0,17.0,100.0,40.0
15566424,male,66,171.0,79.0,11.0,90.0,40.0
12891699,female,32,157.0,54.0,18.0,93.0,40.4
13823829,male,53,182.0,85.0,2.0,82.0,38.1
17557348,female,39,156.0,62.0,28.0,104.0,40.8
12198133,male,39,182.0,82.0,4.0,82.0,38.6
15236104,male,46,169.0,67.0,11.0,89.0,40.2
11042324,female,27,171.0,65.0,4.0,85.0,38.6
16864285,male,50,188.0,86.0,14.0,94.0,40.2
11674347,male,67,189.0,93.0,8.0,77.0,39.2
19797300,female,31,148.0,50.0,8.0,84.0,39.5
14711095,female,33,157.0,60.0,3.0,80.0,38.7
```



```
User_ID,Calories
14733363,231.0
14861698,66.0
11179863,26.0
16180408,71.0
17771927,35.0
15130815,123.0
19602372,112.0
11117088,143.0
12132339,134.0
17964668,72.0
13723164,3.0
13681290,92.0
15566424,58.0
12891699,88.0
13823829,7.0
17557348,170.0
12198133,11.0
15236104,43.0
11042324,15.0
16864285,74.0
11674347,29.0
19797300,32.0
14711095,10.0
14434854,155.0
14893804,3.0
```

(Fig.1-3) Snapshots of the raw data

Design:

The problem is divided into two parts: time and calories, and difficulty, which require two separate machine learning algorithms.

The learning task of the first algorithm requires exercise duration and body type to predict the amount of calories burnt. The data is already pretty clean which makes it mostly ready for training a machine learning model. To connect the model with my problem statement, I simply need to substitute the exercise duration with the time it takes to complete a hiking trail. To find the time, I refer to the final findings in the paper by Andrew Wood et al.; they provided an elegant exponential equation to find the average walking speed over a certain distance by the walking slope and hill slope. I would need to extract the waypoints that connect to a complete hiking trail, then query the walking slope, hill slope, and segment length between each waypoint to find the time it takes to walk through each segment. Adding the time for all of the segments returns the estimated duration of the hike.

The learning task of the second algorithm requires features extracted out of the trails to return a discrete but numerical difficulty rating between 1-6. The collection of .gpx files already includes some data extracted out of the .gpx files, and as a result of scraping from hikr.org, it also includes the difficulty ratings for each trail, which I intend to predict. I can use these features and labels to build a model.

The features and models I tried and ultimately implemented will be covered in the next section.

Implementation:

In addition to training the model, there was also a lot more code needed to process the gpx files. I organized the project into a main folder with different subfolders, for the model, processing scripts, included data, and a directory for user input data.

Time and Calories:

The first thing I decided to do was to find a way to extract information from .gpx files. I first converted the .csv file into a dataframe for access to the .gpx files, which were represented as strings in one of the columns. I then converted the strings to element trees using the lxml library. Since .gpx files for paths always have a tree for track segments (trkseg, for recording paths), I searched for elements with the trkseg tag, which returns the tree storing pairs of latitude and longitude points to represent points in a path. I then extracted the lat/long pairs as tuples into a 2D array for further analysis. I used Google's Elevation API, which returns elevation based on latitude and longitude to find the difference in elevation between consecutive lat/long pairs, as well as the elevation of points surrounding the starting pair. The criteria for surrounding points are explained in the diagram below (Fig.4). The data for each consecutive lat/long pair is then fed into an equation determined by Andrew Wood et al's studies to determine the average speed for each segment, and by calculating the distance between each consecutive lat/long pair we can determine the time it takes to complete the hike (in hours), which will be fed into the machine learning model as an input during testing after converting it to minutes.

The data for training the model is already clean and organized, so with minimal processing, we can begin to analyze the data. I put the data into a dataframe, and generated the correlation between the features and the label. Since sex is not a numerical data type, it was omitted. User I.D., body temperature, and heart rate are also omitted because they won't be obtainable during real-world use without additional equipment (ex. smartwatches). Here it is below (Fig. 5). Clearly, duration has a great correlation with calories burned. We move forward with the visual analysis of the rest of the features. For age, height, and weight, I did a scatter plot for each using matploblib, with the size of each dot being the amount of calories burned (Fig. 6-8). For a given duration, there is no noticeable difference in calories burned as the sample's age, height, or weight changes. Therefore, they will not contribute to the learning curve and will be omitted. The remaining features, then, are duration and potentially sex. There seems to be a difference in the amount of calories burned between the sexes, as shown in the labeled scatterplot. (Fig. 9) Upon running linear regression for the entire data grouped by sexes, the decision functions seem to be different, suggesting that it might be worthwhile to include sex as a feature in the final model. I also noticed that linear regression did not capture the slight upward curve in the data, so I experimented with a polynomial regression as well, with one session with separate decision functions grouped by sex and one without (Fig. 10). The mean absolute errors (MAE) of the functions are shown below; I chose to use MAE because the data did not have many outliers on visual inspection, and the results are easily interpretable. (Fig. 11) Since the loss of the polynomial regressions separated by sex is the lowest and it also produced different decision functions between the sexes, I conclude that data A. more closely follows the trend of a

curved function than a linear one, and B. sex impacts the amount of calories burned. As such, I one-hot encoded the sex feature to convert it into numerical data for learning. To clarify the choice of machine learning algorithm, I trained both a linear and a polynomial regression model (features: sex, duration, labels: calories) 30 times, using sklearn's train_test_split with a test size of 0.3 and taking note of each model's MAE as it predicted the test data. I then grouped the models based on their algorithm type and averaged the MAE of each group. The average MAE of the linear regression models was over 2 calories more than that of the polynomial regression models- 13.53 vs. 11.33 calories. Therefore, we choose polynomial regression as our algorithm of choice. The final model used the entire dataset, and when tested at a random 30% of the data, it returned an MAE of 11.25 calories. The result label is also in units of calories

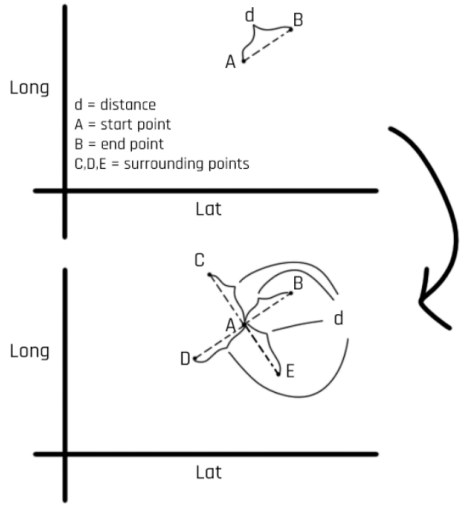


Fig. 4

Age	0.154395
Height	0.017537
Weight	0.035481
Duration	0.955421
Calories	1.000000

Fig. 5

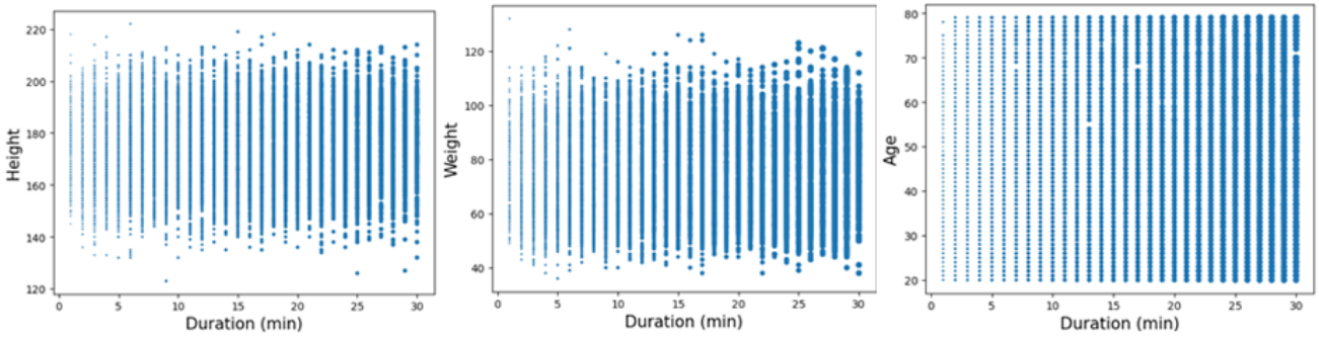


Fig. 6-8

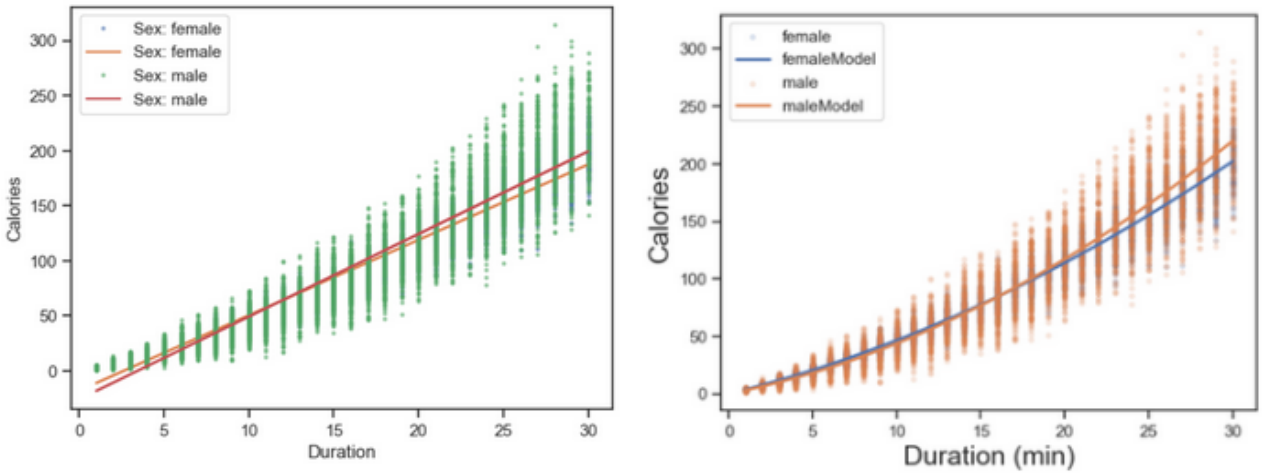


Fig. 9-10

	Male	*Female*
Linear Regression(separated)	*17.200*	*9.729*
Polynomial Regression(separated)	*14.747*	*7.652*
Polynomial Regression	*14.841*	*7.923*

Fig. 11

Difficulty:

The collection of .gpx files also gave us ready-to-use features for machine learning. The first thing I did was to remove several features that were clearly unimportant- I.D, user, gpx string, url, bounds, and name were for identification, while start and end time, moving time, max_speed, and length_3d either relied on a single person's hiking pace and preferences or was a duplicate of another feature. This leaves us with length_2d, uphill, downhill, max_elevation, min_elevation, and difficulty. Since the difficulty label format consists of a number between 1-6 and a description, and I am only concerned about the number, I used regex to replace the information in the column to consist of only the number. In addition, I removed any samples with invalid values in any of the columns and removed outlier data points by calculating the z-score of each sample's attributes against the distribution of features in the data using numpy, and removing any samples that have a z-score of over 3 in any of their attributes. Then, using Seaborn, I generated a scatter matrix for the features and labels, shown below. (Fig. 12) In these scatter matrixes, the tone of a point is defined by its difficulty rating, and one can clearly distinguish a gradient-like effect. This is indicative that these features can potentially all contribute toward training the machine learning data. For the final model, I decided a KNN

model is a good choice since the difficulty of a hike is defined non-linearly- two hikes can be the same difficulty, but wildly differ in length, elevation, altitude, etc. On the other hand, the model can determine what difficulty a hike is by referring to the difficulty of the hikes most similar to it. I also defined the difficulty rating to be a numerical label to better bridge difficulty gaps between similar hikes should they arise. I also chose MAE to be the loss function for analysis since it is most interpretable. To find the MAE, I used the entire dataset to train the KNN model, and gave it a random 20% of the dataset to predict and return the MAE, repeating 10 times to average the returned MAE, which was around 0.53 difficulty units. The weighted function reduces the error by about 17%, as compared to unweighted KNN, which averages at around 0.64 difficulty units. For prediction, I implemented a function to give the higher difficulties more weight, as it is better to predict higher than lower for hiking safety. There is no optimizing the function, the final machine learning model will be the same as this. The output label will be a number between 1 and 6, with 1 being the least difficult to 6 being the most difficult.

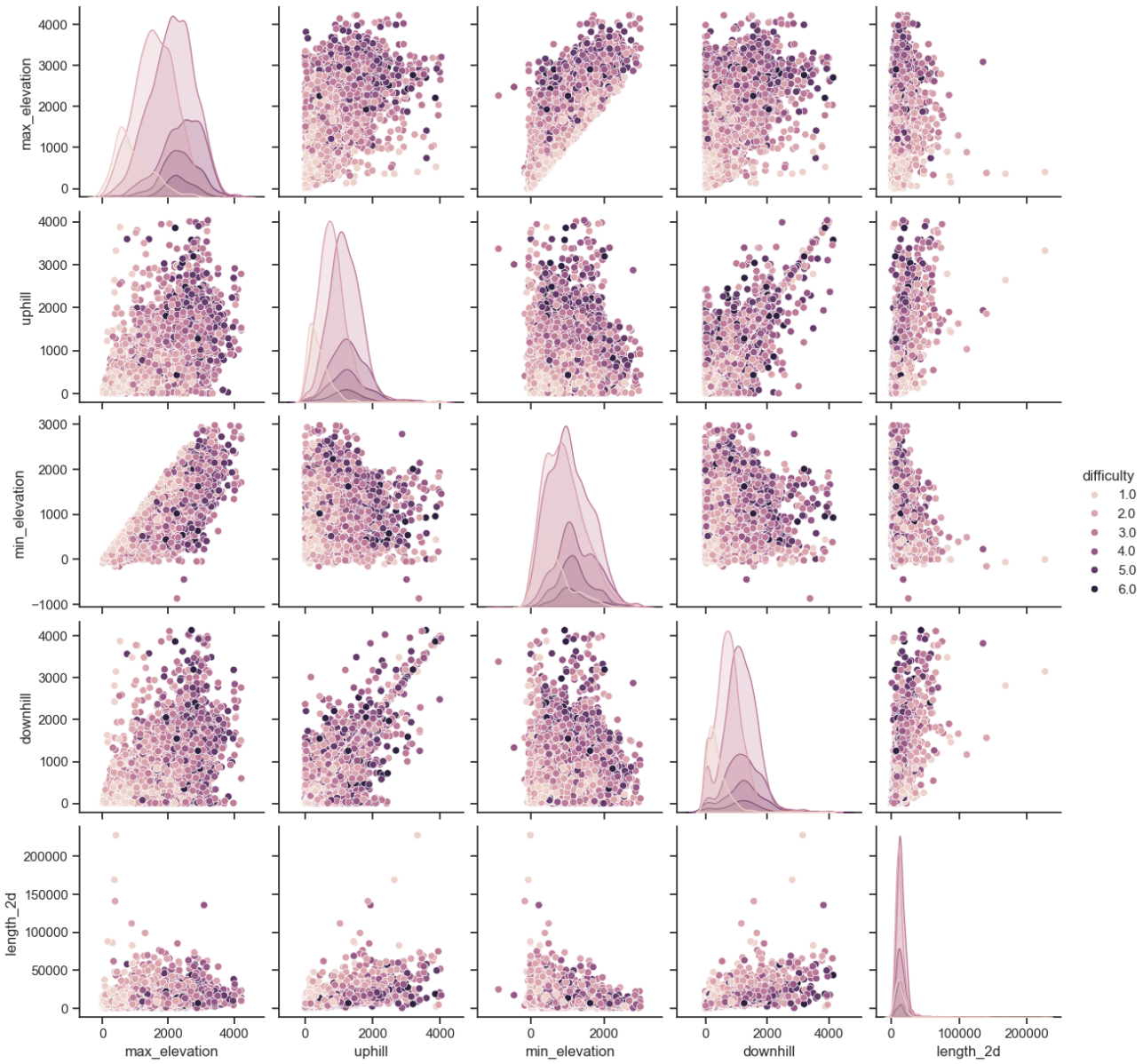


Fig. 12

Results:

Through visualizing the chosen data, first through custom parsing, then by using scatterplots, matrixes, and regression algorithms on graphs, I distinguished patterns in data that could be captured by machine learning models. In the end, I sufficiently trained two machine learning models to estimate the time, calories, and difficulty of a hike given a .gpx file as input. I have included, in the notebook named model, a tab for the reader to input a .gpx file and enter your sex to test the model's outputs.