DATA 200 - Final Report Draft

Erin Jones **Filipe Santos**

1 Abstract

This project is focused on the development of a specialized heart rate prediction algorithm, unique to a given user, based on raw fitness wearable data contained in a data set sourced originally from the endomondo sports tracking application (Ni et al., 2019). Jianmo Ni and colleagues extracted and leveraged this massive data set in 2019 to build an algorithm which they titled FitRec. The algorithm was a Long Short-Term Memory (LSTM) recurrent neural network, customized for each user, built to predict user heart rates for the purposes of activity recommendations. For reasons discussed in further detail in the introduction2, our team sought to build a light-weight algorithm, comparing the performance of both a basic linear regression model with recursive feature selection and a lasso regression model with polynomial features to the performance seen by Ni and colleagues in predicting heart rate. On average, the basic linear model achieved a median RMSE of 15.529 with 75 percent of users achieving and RMSE below 18.7 over a sample of 100 users. The average RMSE for this model was impacted by artifacts of multicollinearity and outliers, which led us to explore a lasso model with more rigorous feature engineering, outlier handling and normalization. This model experienced an average RMSE of 17, which was higher but it worked much faster and the RMSE was not plagued by the outliers seen in the linear regression model. Overall, both models outperformed Ni and colleagues model on average and were much lighter weight than their team's model.

2 Introduction

Heart rate is considered one of the most important biosignals as be used to measure cardiovascular system functions and can be commonly associated with one's general health status (Solan, 2021)(Bogu and Snyder, 2021). With the advent of non-invasive biosensors embedded within wearable devices such as smart watches, mobile phones, wristbands, and smart scales, obtaining individualized physiological data has become much more accessible to users beyond medical professionals. According to a study conducted by Dhingra and colleagues, just under a third of Americans utilize wearable technology to monitor their health and fitness activities(Dhingra et al., 2023).

Historically, heart rate prediction heavily relied on simple linear regression methods, using a fixed intercept (theta zero), and age as the only predictor variable (theta one), as demonstrated by Tanaka et al (Tanaka et al., 2001). The prediction of heart-rate and detection of anomalies has expanded vastly over the years, especially with the advent of more advanced sensor data providing access to raw inter-beat interval data. For instance Keun Ho Ryu and colleagues (Oyeleye et al., 2022) used the robust Multilevel Monitoring of Activity and Sleep in Healthy (MMASH) people data set, which provides significant amounts of metadata about participants paired with a 24 hour stream of data sourced from two separate research grade sensors providing raw electrodermal activity (EDA) readings, photoplethysmography (PPG) readings, and accelerometer readings. This study achieved an accuracy of less than 2 beats per minute (RMSE) with the deployment of a linear regression model, also providing highly accurate results for a range of other models. Ryu and colleagues achieved an extremely accurate prediction in terms of their reported RMSE when compared to that of Ni, Muhlstein and McAuley (Ni et al., 2019). However, it is crucial to note that the data scraped by Ni and colleagues **WAS NOT raw research grade EDA/PPG**, but rather data made available for scraping by the endomondo. The 'black box' that exists between readings from devices like AppleWatch or FitBit is the source of much research and concern, as it causes a significant loss in granularity

of the raw data being captured and the companies reporting metrics like heart rate to users have few requirements in terms of sharing how they are manipulating the data(Bent et al., 2020). Furthermore, sensor data captured in the real world has a higher tendency towards being noisy and inaccurate when compared to the medical grade devices used by Ryu's team in a laboratory setting (Oyeleye et al., 2022). The large difference between these two data sets has even been evaluated, with a study by Daniel Fuller and his colleagues claiming that heart rate measurements captured by wearable devices is highly variable and accuracy depends on the device, but is much less reliable when not being recorded in a lab setting (Fuller et al., 2020).

That said, due to the proliferation of wearables, it is increasingly powerful to derive insights from data as provided by the wearable device companies, whether that data is pristine or not. The data set used for the linear regression and lasso regression model created for this project was identical to that scraped by Ni and colleagues(Ni et al., 2019). Ni's team scraped this set from the endomondo sports tracker website's data set, which no longer exists and has largely been replaced by Strava. Given the vast differences in data used in these experiments, we sought to build a model and compare its functionality directly with the results of Jianmo Ni and colleagues, who similarly were working to use predictive models to build custom heart rate predictions at the user level.

Specifically, we sought to answer the following two research questions:

1. Is it possible to achieve an RMSE within 5% of the best error metric for average RMSE across users achieved by Ni and colleagues using using the regression models learned in D200 i.e RMSE within the range 17.89 to 19.7232 beats per minute.

2. Is it possible to achieve an RMSE that is equivalent to the best error metric achieved by Ni and colleagues FitRec model without embeddings using regression models learned in D200 i.e. RMSE better than 18.784?

Integral to the 'why' behind our research is the changing landscape of edge computing, which allows for the deployment of 'light' machine learning algorithms 'on-device'(Michael, 2021). While largely speculation at this point, this type of computation and deployment could circumvent privacy issues and concerns that arise from the transmission and storage of biometric and location data to the cloud for processing and the generation of insights. The approach used by Ni and colleagues is novel and reasonably successful, but recurrent neural networks are computationally heavy. As such, this algorithm would likely need to be hosted in the cloud, requiring biometric data to be transmitted and stored on servers external to the user's device. The transmission and storage of this type of biometric data increases the risk of a privacy breach(Forrest, 2022), so if an algorithm is light enough to run on the memory allotted for edge computing and is similarly accurate to a more complex and computationally heavy algorithm, there may be reason to choose the lighter model. Furthermore, the less computational power that is needed, the more energy saved, which can quickly add up to a big carbon footprint, making a simpler model the more sustainable choice(Cho, 2023).

Finally, in our Data200 projects, it has made sense to construct a one-size-fits-all model e.g. for the cook county housing data set. However, as evidenced by the strategies seen in our literature review, heart rate is an extremely sensitive metric where the baseline varies based on each user, thus the most common strategy to use is a custom model for each user (Ni et al., 2019)(Oyeleye et al., 2022). We sought to build two light-weight models and compare our results to those achieved by Ni and colleagues per our research questions2. The first model is a simple linear regression model with limited handling of outliers and robust feature engineering. The second model is a lasso regression model with robust handling of outliers and scaling techniques. The results for both models seem to outperform the results of Ni and colleagues with a sample of 100 users, each with an average of 100 workouts averaging 500 timestamps per workout. However, the overall results weren't promising in the grand scheme of heart rate prediction and to better tackle this problem, more granular data, better sensor hardware, and more must be considered, as discussed further in the discussion section 7.

These models could be leveraged in tandem with an anomaly detection model. Should the heart rate in the test set exceed or drop below the predicted heart rate by 3 standard deviations for more than 2 sampling intervals, an anomaly will be flagged. Our results, per our research questions2, will be compared to the results of Ni and colleagues in

terms of goodness of fit, in addition to being assessed via methods learned in Data 200. A brief exploration of anomaly flagging based on model predictions will be compared in accordance with the American Heart Association's Target Heart Rate chart(Association, 2023) in the discussion section of the paper 7.

3 Description of Data

The data set used for this paper consists of a publicly available NumPy binary file containing a trimmed version of the endomondo data set scraped by Ni and colleagues (Ni et al., 2019), which can be found at the FitRec Project website**??**. The 'raw' data, as originally scraped, contained in JSON format, is also available on the website, however there is an issue with the JSON formatting, preventing it from being opened using standard JSON libraries and the formatting is not modifiable via local machine due to its size (∼9.8GB).

The pre-processed set has already been trimmed from an initial 253,020 workouts / 1,104 users to 167,783 workouts / 1059 users based on users Ni and colleagues deem to be abnormal workout samples e.g. having overly large magnitude, mismatching timestamps, abrupt changes in GPS coordinates(Ni et al., 2019). Jianmo and the team also derived speed and distance from the raw measurements and normalized several of the metrics prior to training the model and a handful of the measurements are already in a scaled format, but the non-normalized metrics exist for certain variables as listed below. The RMSE in their report is based on non-normalized measurements.

Each user has multiple workouts, with each workout representing a time series where the following variables are marked for a given time stamp:

- since_begin - this value is reported a single time for each workout in the pre-processed data set, remains static throughout the workout and is not mentioned in documentation and is thus dropped
- since_last - similarly to since_begin, this value remains static and is not mentioned in documentation and is thus dropped
- tar_derived_speed - non-normalized speed measured in miles-per-hour
- derived_speed - normalized speed
- tar_heart_rate - non-normalized heart rate measured in beats-per-minute
- heart_rate - normalized heart rate
- distance - distance traveled in miles (normalized in some fashion as values are often negative, details not provided on normalization)
- sport - nominal categorical variable representing the sport of the individual
- id - workout id within the context of the entire .npy file
- timestamp - timestamp in unix (i.e. seconds since 1/1/1970)
- altitude - altitude measured in feet
- gender - nominal categorical representing an individual's gender
- time_elapsed - normalized representation of minutes elapsed since start of workout
- longitude - in degrees
- latitude - in degrees
- userId - unique identifier for the user

The data was imported and parsed such that a single dataframe was generated for each user (identified by their userId) that contained all workouts available, as associated with the 'id' (i.e. workout id). The dataframe contains all features listed above as separate columns, each row representing a sample of the user's location and biometric data at a given point in their workout. Sampling frequency is not static and occasionally samples are missed in between rows. The importing and parsing of the original set can be duplicated by using the notebook raw_parse.ipynb. The features included in this dataset are sparse, especially when considering the handful that need to be dropped due to their static nature (e.g. since_begin, since_last, gender, etc).

Beyond having to use the pre-processed data, which has limited information available regarding exact pre-processing steps, the metadata that was scraped for each user is entirely inaccessible due to the same JSON formatting error seen in the raw JSON. This limitation will be discussed in further detail in the discussion section 7, but it limits one's ability to interpret the results.

4 Methodology

Our methodology at a high level is deploying regression models to predict user heart rate at the individualized level i.e. each custom model is trained based on a specific user's historical workout data and predicts results only for that user. This model can then be used to construct Tukey fences for outliers, which can be used to flag anomalies as they relate to a users data (De, 2021).

The results of this research are two separate regression models that can be used to generate custom predictions given a user's workout data. The first model, a basic multiple linear regression model with recursive feature selection, was generated first in tandem with attempts to engineer additional features in light of the limited available features in the original data set. The model and results can be seen in the notebook linear model 1. Using the results from model one, specifically the edge cases where the model was performing particularly well and particularly poorly, visualizations were produced to investigate the model's behavior on both training and test sets to better understand the bias and variance of the data set. This exploration can be seen in eda RFECV model notebook. The learnings from this model were then used to select a second model, features and wrangling methods, which lead to model two - a lasso regression model. The results from this model can be found in linear model 2 notebook, alongside an exploration and visualizations of the results in eda for the lasso model notebook.

We knew from the start that we wanted to employ some sort of automatic feature selection in order to develop a custom model for each user. Due to the unique nature of each user (see figures 1 & 2) in terms of the types of sports they practice, what time of day they practice them, their elevation, age, health etc. it is likely that different features will impact and/or proxy each user's heart rate differently. For instance, across all users, there existed 46 different types of sports

4.0.1 Model 1

Prediction Method- A multiple linear regression was selected for this task, as a strategy supported by much literature seen in the introduction 2 .

- Multiple linear regression optimizes for the root mean squared error and contains no regularization. We opted to start with this model as it is a good baseline for setting up a pipeline and considering various features. Additionally, features were so sparse and showed such lack of correlation (even when considering the Tukey Mosteller bulge diagram) in initial EDA that we opted to give it a shot and see what appeared in the results. This could be considered an exploratory model.

- Linear Regression is also a light model when compared to other regression models, such as a Gradient Boost regressor and the Random Forest regressor which both employ brute force techniques to achieve desired results. If this model performs as well as the deep learning model developed by Ni and collagues, there is a promising chance that something of this nature could be deployed on device to take advantage of privacy benefits on the edge.

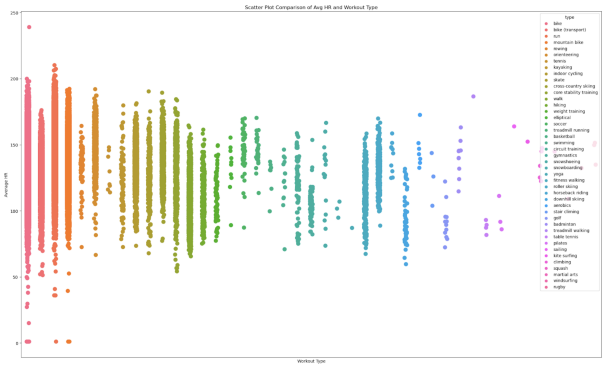

Figure 1: Sports Across Users

Loss Function - The function for RMSE is as follows:

$$RSS = \frac{1}{n}\Sigma_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

Where:

- y_i - the non-normalized heart rate for a user
- $\hat{y}_i$ - the predicted heart rate using the given features (see 4.0.3)
- n = the number of timestamps for a given user i.e. the number of records in the data set

Automated Feature Selection - A recursive feature elimination (RFECV in scikit-learn) is employed with this model to choose the optimal number of features for each user and reduce potential overfitting to the training set. The method recursively eliminates features to determine an optimal

number of features while employing 5-fold time series cross validation 4.0.4 to avoid overfitting. After each recursive iteration, RFECV fits the specified model, ranks the features based on their importance and then removes the least important feature(s). In our case, we chose the hyperparameter of a single feature being eliminated at each recursive step, however this can be set differently. We opted for a single feature given the sparse amount of features available.

4.0.2 Model 2

Prediction Method- For model 2 we leveraged a Least Absolute and Selection Operator model (Lasso) and ditched the RFECV, as Lasso is much more robust to collinearity, which there is a lot of in our dataset

Loss Function - The function for RMSE is as follows:

$$RSS + Penalty =$$
$$\frac{1}{n}\Sigma_{i=1}^{n}\Big(y_i - \Sigma_{j=1}^{p}X_{ij}\beta_j\Big)^2 + \lambda\Sigma_{j=1}^{p}|\beta_j|$$

Where:

- y_i - the non-normalized heart rate for a user
- $X_{ij}\beta_j$ - the predicted heart rate using the given features (see 4.0.3)
- λ = the L1 penalty
- i = the number of timestamps for a given user i.e. the number of records in the data set
- j = the number of features in the design matrix

Automated Feature Selection - Feature selection is completed by the Lasso Regressor, as coefficients that are unimportant will be assigned near-zero coefficients due to the penalty term λ. This method for feature selection introduces regularization into the model and, as seen in the results section**??**, leads to much less variance in terms of the bias variance trade-off.

Outlier Removal - Due to the results seen in model one, we worked a more robust handling of outliers into the pipeline, removing training data for distance, heart rate, altitude, and speed. Heart rate was handled by removing values below 40 and above 200 per the American Heart Association's guidelines. The other three variables were handled using z-scores, removing values more than 3 standard deviations away from the mean.

Hyper Parameter Tuning - The alpha coefficient for the lasso model is selected based on a 5-fold time series cross validation 4.0.4, similar to that used in model one by the recursive feature eliminator.

4.0.3 Feature Engineering + Data Wrangling

From the given set of variables, we immediately removed since last, heart rate, derived speed, id, gender and userId. Since_last was not available for most users and represented a static value, seemingly untied to heart rate. Derived speed and heart rate were both scaled for research purposes by Jianmo and his colleagues (Ni et al., 2019) and did not have details regarding the reverse of this scaling. Thus both were eliminated in favor of the values left not scaled, such that we could choose our own scaling methods. userId was retained to index various files and dataframes associated, but eliminated as a feature alongside id as both remained static and thus represented a scaled, linear combination bias term in the design matrix. Finally, while gender may impact heart rate at a high level, it remains static with each user and generally does not change with their workout, or their heart rate. Again, its inclusion would cause collinearity with any other static variable, such at the bias term, so it had to be removed.

This is a limitation for our model, when compared to that of Ni and colleagues as their model included an embedding layer that represented gender and sport when the model was being trained. We were able to one hot encode the sport type being performed by the user as this was not static across all workouts and drop the first column to avoid collinearity. What sport a user is doing will significantly impact the expected range of their heart rate (e.g. a higher heart rate when completing strength training than when walking or biking as a commute).

The final feature that was engineered and used in both models was the time elapsed in minutes feature, generated using the unix timestamp column from the raw data. This feature was assumed to relate to a user's heart rate, for which the baseline is likely to increase as the user becomes more fatigued from their workout. 2 This is not a perfect relationship, as a user may take breaks or be performing interval training leading to fluctuations over the duration, however it did not bear the issue of representing the cyclical variable of time as being non-cyclical. Different feature mixes were used

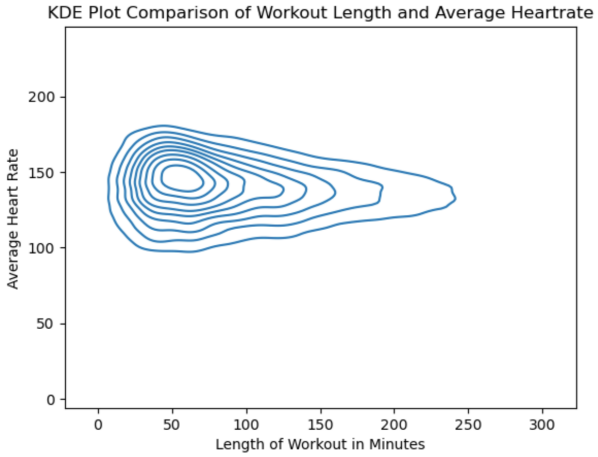

Figure 2: Impact of Workout Length on HR

between the two models, as we gained insight from building the first model and changed our approach for the second.

Model 1 Features

- The unix epoch was converted to a timestamp in the appropriate timezone. This required an external library, timezonefinder, which allowed us to search for a timezone based on a given set of lat/long coordinates. While this suited our purposes and allowed us to engineer a categorical feature for time of day for the workout, it is worth noting that the library is rather large and even when we grouped by workout, only calculating a timezone for each workout, it significantly increased the computation time when running our pre-processing pipeline.
- Haversine - the Haversine point value calculated based on the Haversine formula, which is a way of representing the latitude and longitude features in a single number to proxy an individual's location on earth at the time of workout
- time category - a categorical variable derived from the hour of the day in which the workout occurred, represented by a number. Categories include:
  - 0 - Night (00:00-04:59)
  - 1 - Early Morning (05:00-06:59)
  - 2 - Mid Morning (07:00-08:59)
  - 3 - Midday (09:00-10:59)
  - 4 - Early Afternoon (11:00-13:59)
  - 5 - Late Afternoon (14:00-16:59)
  - 6 - Evening (17:00-23:59)

Model 2 Features

Model two, designed after an investigation of the results of model 1, stripped many of the features developed due to collinearity and the amount of time they added to the run time of model training. The one hot encoded sports category and the

- Model 2 includes a polynomial feature function, which leverages scikit-learn's PolynomialFeatures module to generate polynomial features from distance, altitude, time elapsed, and speed up to degree three.
- Additionally model 2 introduces a MinMaxScaler and custom scaler to scale the design matrix numerical variables and target variable respectively prior to plugging them into the lasso model. This was added after investigating the distribution of those users for whom the model performed poorly

4.0.4 The Pipeline

A pipeline for data processing was built to clean, add features, and scale the data as needed while ensuring no data leaks exist. A similar pipeline framework was used for both models, as we leveraged the modular pipeline framework and data.pipe presented in Assignment 2 Part 2. The mechanics of this pipeline can be viewed in the process_data_pipe and process_data_final functions part of both the model_1 and model_2 notebooks. Pandas data pipe feature is employed to allow multiple feature engineering functions to be implemented and then, depending on whether the data fed into the pipeline belongs to the test set or the training, the data is split into the design matrix X with selected and engineered features and a series containing the actual y values.

Train Test Split + Cross Validation on Time Series

Initially, a set of 500 users is selected randomly from the set of 1000 users, however the random seed was set to 42 to ensure our results can be duplicated. In the end, the timeout settings of Jupyter notebooks prevented us from running the models on more than 100 users each, as the application required significant amounts of babysitting to ensure the kernel was not killed before the model finished running. This limitation is discussed in further detail in the discussion section of the paper.

Each user's time series is then split into a training and a test set using a 75/25 split and scikit-learn's train_test_split function. This is done without shuffling the data, at retaining the order for time series data is of significant importance(Hyndman and Athanasopoulos, 2014). Should all of the data points be shuffled, you are potentially using future data to predict past values. For the initial train-test split, we simply set the shuffle value to False so the first 75% of the data set is the test set and the last 25% of the data.

The model is run is trained on the first 75% of data and then predictions are generated for both the training and test set, producing a training and validation RMSE for each user. More details regarding the results are included in the results section. 5

Cross-validation is a little bit messier when dealing with time series, so in both cases where it is employed (for automated hyper-parameter tuning and recursive feature elimination) we used a special time series split to build the folds. Fortunately, scikit-learn offers TimeSeriesSplit() which allows for the simple implementation of time series cross-validaton best practices as described in Hyndman's best practices (Hyndman and Athanasopoulos, 2014). Specifically, the procedure implemented automatically by the scikit-learn method is, "known as "evaluation on a rolling forecasting origin" because the "origin" at which the forecast is based rolls forward in time," at each fold, five times in total.

Anomaly Detection Function

Once a set of predictions and actual results exists, the results can be fed into the anomaly detection function. This function is set to flag an anomaly if the actual result exceeds the outer bounds of the predicted results set in two or more timestamps consecutively. As mentioned above, it is an implementation of Tukey fences for they detection of outliers.

4.0.5 Handling Missing Values

The number of missing values were limited. For the first model, the linear regression model, they were dropped during data processing. For the second model, the lasso model, they were filled using the backfill method. The assumption behind this choice is that heart rate does not vary significantly from second to second and as such, using past data to fill the missing values would be an appropriate choice.

5 Summary of Results

RMSE was used as the loss function for evaluating the goodness of fit for both models as it is what was used by Ni and colleagues and allowed for our comparison with their results (Ni et al., 2019). Both model one and model two were run on the same sample of 100 users using a random.seed of 42 to ensure the same sample was selected both times.

Overall, both models were able to out perform The results of Ni and

Model 1 Results

Instead, we wished to see how the model performed without any feature manipulation beyond the addition of features mentioned about

The results for the Linear Regression model as run on 100 are as follows:

- The global mean across training set users for RMSE is 14.931908
- The global mean across test set users for RMSE is 14 quadrillion (due to outlier 4)

The distribution of the RMSE from the training set for the RFECV model is displayed in the following chart

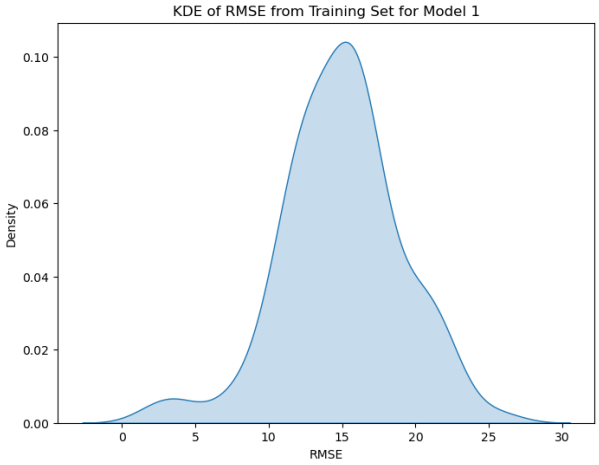

Figure 3: Model 1 Training Set RMSE Distribution

Based on these results, our model outperforms Ni and team's custom-per-user model on average, but not all the time. The figure below**??** indicates that while the largest number of user results outperformed Ni's model, a significant number also performed on par or worse than Ni's model Taking a closer look at the results, however, reveals some unique findings with regards to how the feature selection is performing and an interesting fact about

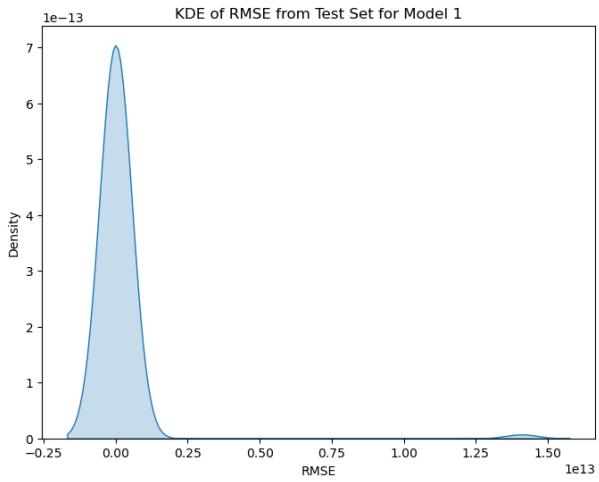

Figure 4: Model 1 Test Set RMSE Distribution

this data. We decided to take a closer look at the users for which the model performed the best and the worst to see if any patterns could be identified from within the results.

To analyze and understand the results generated by the model, the first step was to look into the actual and predicted heart rate values. The following chart shows the distribution of the actual heart rate for the users with the best and worst performing models (lowest and highest RMSE, respectively).

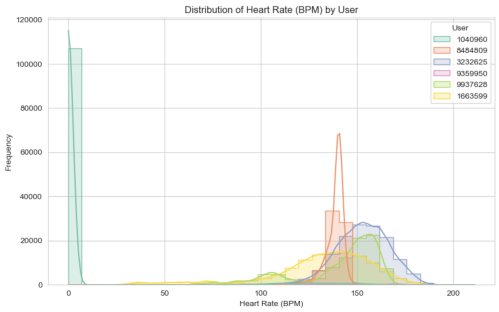

Figure 5: Heart rate distribution by user

The main insight from the chart above is the apparent data issue with user id = '1040960': The heart rate distribution is very close to zero. As deeply analyzed in the attached notebook, this behavior appears to be the consequence of malfunctioning sensors or a problematic data collection process. Nevertheless, it is fundamental to acknowledge that the results associated with the top performer user with the lowest RMSE (0.071) result from poor-quality target data. Therefore, such results do not imply a good predictive model.

Similarly, when looking into the bottom performer (user id = '9937628'), the RMSE value

User	RMSE	
1040960	0.071	['c
9359950	3.581	
8484809	5.574	
1663599	36.845	['sport_bik
3232625	44.002	['distan
9937628	1.4e16	['tar_derived_speed', 'distance', 'altitude', '

Table 1: Closer Look @ Selected Features

obtained looks significantly wrong. After performing some exploratory analysis, as presented in the attached notebook, we could conclude that the reason for such behavior is that running the RFE model for that particular user removed the 'OHE' variable related to the 'sport' type. Furthermore, when defining the betas for each feature, the model defined a high weight for these 'sports' dummy variables. Since the dummy variables were automatically eliminated in this particular case, some predicted Y's ended up with an absurdly low and negative value, as presented in the following summary statistics table.

	array_data
count	2.341900e+04
mean	-2.911145e+12
std	1.378419e+13
min	-6.817611e+13
25%	1.320625e+02
50%	1.426641e+02
75%	1.495391e+02
max	4.924375e+02

Figure 6: Summary statistics of the predicted Y for user '9937628'

RFECV on initial tests and a correlation matrix for each user's augmented design matrix is shown below: This was a red flag as it did not line up with initial results of the correlation. The models performing the worst all are solely using the one hot encoded representations of the sports being done. Given that these columns nearly add up to the bias column (drop='first' parameter was used), in essence, the model is overly simplistic as it regresses to the mean i.e. a null/baseline model. This actually says a lot about the data for these users – the features are too few and have very little predictive power for the given user based on the selected training set. It also says something about the heart rate for these users - the range of values is likely small enough that the probability an actual value falls within 20-30 beats per minute from the mean of the training set is high.

Model 2 Results When analyzing the LASSO

Linear Regression results, the first impressions were already significantly different from the RFECV. This model performed significantly better than the Linear model due to its more robus handling of outliers and scaling of features. This can be seen in the distribution of result below.

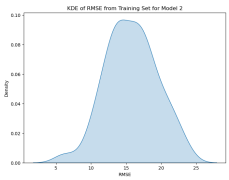

Figure 7: RMSE Distribution for Model 2 Training Set

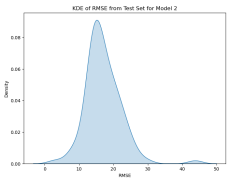

Figure 8: RMSE Distribution for Model 2 Testing Set

The outlier still existed, however the average RMSE for the training set was 15.754358 and the for the test set was 17.001074 which beat the performance of Ni's model. Furthermore the amount of variance that was seen in model one is reduced significantly by the second model which performs much better on the training set

This can easily spotted in the following table with the RMSE of the top and bottom performance users:

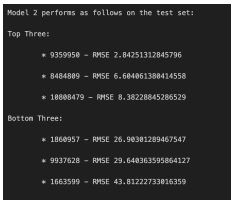

Figure 9: RMSE Results for the Top and Bottom Performers

Because of the learning process from our first model, we implemented an outliers cleaning function that allowed us to ensure that only possible heart rate numbers would be considered, with a clear cut on values below 40 bpm and above 200 bpm. The following image illustrates the results of the actual heart rate values for the top and bottom performance users.

Similar behavior was observed when analyzing the predicted values for the same users. There are

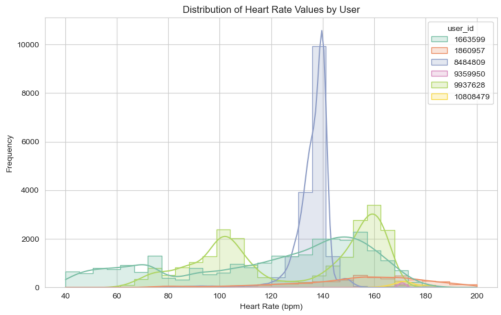

Figure 10: Heart rate distribution by user (Lasso)

no significant outliers in terms of the predicted value of heart rate, as we had identified in the RFE model. The findings from this initial visual inspection truthfully aligned with the results obtained in terms of the accuracy of this model, as displayed in the following image.

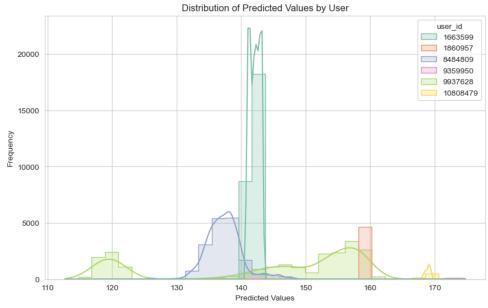

Figure 11: Heart rate Prediction distribution by user (Lasso)

6 Discussion

Overall, we were able to wrangle and build a pipeline that is customized over a vast amount of time series data. Furthermore, both models out performed Ni and colleagues models which indicates good amounts of possibility for creating a lightwight on device anomaly detection algorithm

6.0.1 Limitations

A list of limitations seen in our research is as follows:

- We did not have access to enough computational power to produce results outside of our 100 randomly selected users
- Due to time limitations we were unable to complete significant amounts of bootstrapping or nested cross validation
- In general, the recursive feature selection algorithm may eliminate too many features on

such a sparse data set, especially if too few linear relationships exist in the data. Several limitations and areas for future work are discussed below. It is important to note that the implementation of an algorithm like this should be marketed cautiously if deployed - an anomaly detector is built to detect anomalies, not to diagnose medical issues. Each flagged anomaly should be taken with a grain of salt.

•

6.0.2 Areas of Future Research

Anomalies Compared to Target Heart Rate Charts

The three figures below show initial results of the anomaly detection algorithm on sets of 3 test results as produced by the anomaly detection notebook included in the folder. These were produced for fun on a set of early results and do not reflect any of our final model results, but they do a fair job of illustrating our original intent for these models - to predict heart rate and then flag anomalies. We are including this in future work as an area for possible future works

The purple line represents the actual heart rate. The blue line represents predicted results. The gray shaded area around the predicted line shows the buffer bound by the Tukey fences. Figure 4 does a good job flagging an anomalous result at the upper end of the heart rate spectrum12. It also shows an issue with the RFECV, where it has obviously eliminated all features besides the one hot encoded variable for sport and by chance most of the heart rates fall in the same range (i.e. the individual is relatively regular). Figures 5 and 6 are similarly unresponsive to the heart rate, which again would indicate that the model is really lucking out in its predictions, given the variability in the data set. These two also show a slew of anomalies flagged at the lower end which suggest that the training data did not contain many instances where the heart rate was low. This could be solved with more normalization/regularization of variables added to the pipeline. **Additional Data and Metadata**

We explored the prospect of adding weather data as if an individual is exercising outdoors, humidity and outdoor air temperature can impact heart rate, but OpenWeather only recently provided us with educational access to their historical API and we have yet to engineer any features from this data. We hope to have this data added in time for the final report. Furthermore having metadata could be very useful to further improving the results.

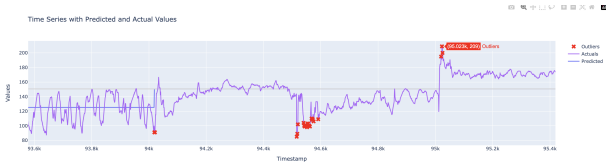

Figure 12: Correct Flagging of an Anomaly

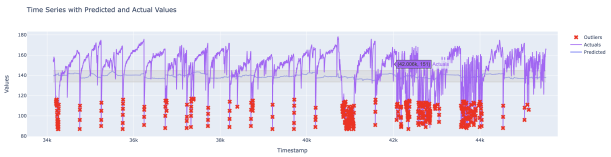

Figure 13: Issue with Model Not Following Variability of HR

A significant limitation of this data set was the lack of access to meta data on each user, which may have included key demographic details such as age, that may have allowed us to better assess bias in the results. This is echoed in the lack of access to the raw data set. It may be interesting to deploy these models in the future on a different wearable fitness tracker data set.

Implementation of Models On Cloud

Running these results and bootstrapping on the cloud could have produced a more thorough comparison and evaluation of the model on all 100,000 users in the dataset but was not possible due to computational cost and increased effort on computational efficiency could have been implemented with more time to better understand the feasibility of implementing this model on an edge computing device. This would have also allowed for the production of metadata.

7 References

References

American Heart Association. 2023. Target heart rates chart.

Brinnae Bent, Benjamin A. Goldstein, Warren A. Kibbe, and Jessilyn P. Dunn. 2020. Investigating sources of inaccuracy in wearable optical heart rate sensors. *npj Digital Medicine*, 3(1):18.

Gireesh K. Bogu and Michael P. Snyder. 2021. Deep learning-based detection of covid-19 using wearables data. *medRxiv*.

Renee Cho. 2023. Ai's growing carbon footprint.

Moloy De. 2021. Tukey fences for outliers.

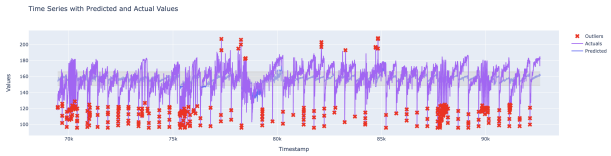

Figure 14: Not Accurately Predicting Lower HR

Lovedeep S. Dhingra, Arya Aminorroaya, Evangelos K. Oikonomou, Arash Aghajani Nargesi, Francis Perry Wilson, Harlan M. Krumholz, and Rohan Khera. 2023. Use of Wearable Devices in Individuals With or at Risk for Cardiovascular Disease in the US, 2019 to 2020. *JAMA Network Open*, 6(6):e2316634–e2316634.

Fi Forrest. 2022. Wearables: managing complexities in data privacy and consent in healthcare tech.

Daniel Fuller, Emily Colwell, Jonathan Low, Kassia Orychock, Melissa Ann Tobin, Bo Simango, Richard Buote, Desiree Van Heerden, Hui Luan, Kimberley Cullen, Logan Slade, and Nathan G A Taylor. 2020. Reliability and validity of commercially available wearable devices for measuring steps, energy expenditure, and heart rate: Systematic review. *JMIR Mhealth Uhealth*, 8(9):e18694.

R.J. Hyndman and G. Athanasopoulos. 2014. *Forecasting: Principles and Practice*. OTexts.

J. Michael. 2021. Security and privacy for edge artificial intelligence. *IEEE Security Privacy*, 19(04):4–7.

Jianmo Ni, Larry Muhlstein, and Julian McAuley. 2019. Modeling heart rate and activity data for personalized fitness recommendation. In *The World Wide Web Conference*, WWW '19, page 1343–1353, New York, NY, USA. Association for Computing Machinery.

Matthew Oyeleye, Tianhua Chen, Sofya Titarenko, and Grigoris Antoniou. 2022. A predictive analysis of heart rates using machine learning techniques. *International Journal of Environmental Research and Public Health*, 19(4).

Matthew Solan. 2021. Your resting heart rate can reflect your current and future health.

Hirofumi Tanaka, Kevin D Monahan, and Douglas R Seals. 2001. Age-predicted maximal heart rate revisited. *Journal of the American College of Cardiology*, 37(1):153–156.