

Exploration of Threshold Heart Rate Data on Performance in the 800m Race

Will Dauenhauer

University of Montana

MSBA 2023

Table of Contents:

I. Executive Summary

II. Background

III. Overview

IV. Research Questions

V. Data

VI. Analysis

VII. Code Explanations

VIII. Limitations

IX. Recommendations

X. Conclusion

I. Executive Summary

This paper explores the relationship between desired heart rate during training and performance outcomes in the 800m race, specifically distance covered, speed, and time taken. The study utilizes personal data collected from Strava's API to examine this relationship. The findings suggest that there is a correlation between heart rate and performance outcomes, although the degree of correlation varies. The results suggest that desired heart rate during training can have an impact on performance outcomes, and thus should be considered when designing training programs. Further research is needed, however, to fully understand the relationship between heart rate and performance outcomes, and to identify the most effective training programs for different individuals. Overall, this study highlights the importance of monitoring heart rate during training and its potential impact on performance outcomes.

II. Background

I run cross country and track at the University of Montana and have been recording my runs and other aerobic activities on Strava since 2020. Starting that year, every activity I have recorded on my Garmin Forerunner 35 watch has uploaded metrics to an online platform called Strava. Strava is a social fitness network that provides a range of features for runners, including GPS tracking, route mapping, and activity analysis. The platform allows runners to track their runs, rides, and other workouts, and provides insights into their performance, including distance covered, pace, elevation, and calories burned. Strava provides runners with a comprehensive platform to track, analyze, and improve their performance, but it does not give all the answers to the true capabilities for my analysis which brings us to the Strava API integration. Strava's API (Application Programming Interface) provides runners with a way to access their data and perform custom analysis beyond what is available through Strava's website or mobile app. The API allows developers to create custom applications and tools that can access Strava data, including workout data, segment data, and athlete data. This allows for more detailed analysis of workouts, performance metrics, and trends over time. Figure 1 shows the distribution of running over the years of available data.

Chart, bar chart

Description automatically generated

*Figure 1: Runs by Years*

III. Overview

To get context to the choices made in the analysis, we need to explain some running science. Training around your VO2 max can increase your lactate threshold and improve your times in the 800-meter race. VO2 max is the maximum amount of oxygen that can be utilized during exercise, and lactate threshold is the point at which lactate begins to accumulate in the muscles, leading to fatigue. By increasing your VO2 max, you can delay the onset of lactate accumulation, allowing you to run at a higher intensity for longer periods of time.

One study by Esteve-Lanao et al. (2005) found that training at or near VO2 max was more effective than lower intensity training for improving lactate threshold and running performance in highly trained runners. Another study by Laursen et al. (2002) found that interval training at or near VO2 max significantly increased lactate threshold in well-trained runners.

In the context of the 800-meter race, which is considered an anaerobic event, training around VO2 max can be particularly beneficial. A study by Jones et al. (2006) found that interval training at or above VO2 max improved performance in the 800-meter race. The authors suggest that this type of training improves lactate clearance, allowing athletes to maintain a faster pace for longer periods of time.

What is already known and what are we trying to figure out. Figure 2 below shows the different heart rate zones that Strava has given me over the years based on my performance. Figure 3 relates those zones to the increase of lactate threshold. Training in the “threshold” heart rate zone can lead to improvements in running performance, but the extent of these improvements can vary depending on several factors, including an individual's fitness level, training history, and genetics. Threshold heart rate training is working out at or near the maximum sustainable heart rate for a prolonged period of time. This type of training can help improve aerobic capacity, increase lactate threshold, and improve overall endurance. It is commonly used by runners to improve their speed, distance, and race performance. Research has shown that training at threshold heart rate can lead to improvements in running performance, including increased VO2 max, improved lactate threshold, and increased running economy. The magnitude of these improvements can vary depending on individual factors and the specific training program.

Table

Description automatically generated

*Figure 2: Chart of Heart Rate Zones*

Table

Description automatically generated with medium confidence

*Figure 3: Chart of Lactate Threshold related to Heart Rate Zones*

In this analysis, I wanted to see what effect training in or hitting this threshold has on my overall performance in the 800m race. The “why” in this project is driven by the fact that over the years, my training has been all over the place due to different coaching styles, philosophies on heart rate, training for different distance races, different workout types, switching between cross country and track, and overall consistency (or lack thereof) in training. Can all the training data over the years be able to give me a measure of race fitness?

IV. Research Questions

Given the varied nature of my training over the years, I wanted to explore the impact of using heart rate as a threshold for training intensity on my performance. Specifically, I am interested in how hitting in or over my zone 4 heart rate threshold during training affects my overall performance in terms of distance, pace, and time. With these questions in mind, my specific research questions are:

RQ 1: How does reaching into and past my zone 4 heart rate during training predict performance outcomes in the 800m?

RQ 2: How well does max heart rate serve as a predictor for performance outcomes in the 800m race?

The limitations to this which I will discuss later will be on using **max\_heartrate** as our response variable, this is not the best variable as we lack the fluctuation of heart rate throughout the activity. The last thing we must ask is how does the predicted max heart rate help me? Well, I can use the Figure 4’s max heart rates to compare my result against. The logic here is that if my predicted output is over my max heart rates ever recorded, then my time and speed at which to complete the 800m race is not attainable at my current levels. The logic goes the other way as well if it is under. The last thing I want to point out before my analysis is that the model is using data that has never been in an 800m race, I simply never wear my watch during my track races therefore the data does not know what times I have already achieved.

| **Top 10 Max Heart Rates Recorded** |
| --- |
| 218.0 |
| 211.0 |
| 210.0 |
| 209.0 |
| 208.0 |
| 205.0 |
| 204.0 |
| 203.0 |
| 203.0 |
| 203.0 |

*Figure 4: My Top 10 Max Heart Rates Recorded*

V. Data

The type of data that was pulled from Strava including the new variable created called ‘hr\_over\_180’ which returns either a true or false value for if the max heartrate is over 180. This variable will serve as a separator later in the model to get only data points in my training where I hit over 180 BPM’s and got into my Zone 4 or Zone 5 heartrate zone. The data includes 914 observations for just running activities, and the variables recorded include distance, moving time, elapsed time, total elevation gain, achievement count, kudos count, comment count, athlete count, average speed, and other variables such as elevation low, pr count, weekday, moving time in minutes, distance in kilometers, pace, average speed in kilometers per hour, maximum speed in kilometers per hour, elevation, and year.

Next, I cleaned the data (seen in code appendix) to remove variables not used in my analysis as well as clean up runs that were less than a mile and GPS watch errors in speeds that are not attainable while running. The variables removed included 'id', 'achievement\_count', 'kudos\_count', 'comment\_count', 'pr\_count', 'athlete\_count', 'year', and 'weekday'.

Here is the final list of the quantitative variables in the analysis after cleaning with a short descriptive analysis on each variable.

* **distance**: The total distance covered in the activity ranged from 9.8 meters to 23.9 kilometers, with an average of 6.8 kilometers and a standard deviation of 4.3 kilometers.
* **moving\_time** and **elapsed\_time**: The moving time and elapsed time for the activity were similar, with an average of around 31 minutes and a standard deviation of around 19 minutes. The maximum moving time was 1 hour and 16 minutes, while the maximum elapsed time was 2 hours and 13 minutes.
* **total\_elevation\_gain**: The total elevation gain for the activity ranged from 0 to 697 meters, with an average of 61 meters and a standard deviation of 98 meters.
* **average\_speed** and **max\_speed**: The average speed for the activity was 5.9 kilometers per hour, while the maximum speed was 15.7 kilometers per hour. The standard deviation for both measures was relatively low, indicating that most of the activity was completed at a similar pace.
* **average\_cadence**: The average cadence (pedal revolutions per minute) for the activity was 79.5, with a standard deviation of 3.4.
* **average\_heartrate** and **max\_heartrate**: The average heart rate for the activity was 155 beats per minute, with a maximum of 218 beats per minute. The standard deviation for both measures was relatively high, indicating that heart rate varied widely throughout the activity.
* **elev\_high** and **elev\_low**: The highest and lowest elevation points during the activity were 2126.1 meters and 932.3 meters, respectively. The average elevation was 978.9 meters, with a standard deviation of 257.6 meters.
* **moving\_time\_minutes**: The moving time converted to minutes ranged from 0.32 to 129.27, with an average of 31.2 minutes and a standard deviation of 19.3 minutes.
* **distance\_km**: The distance converted to kilometers ranged from 0.01 to 23.87, with an average of 6.8 kilometers and a standard deviation of 4.3 kilometers.
* **pace**: The pace (in minutes per kilometer) ranged from 0.18 to 325 minutes per kilometer, with an average of 13.3 minutes per kilometer and a standard deviation of 2.3 minutes per kilometer.
* **avg\_speed\_kmh** and **max\_speed\_kmh**: The average speed and maximum speed converted to kilometers per hour were 4.99 kilometers per hour and 21.1 kilometers per hour, respectively. The standard deviation for both measures was relatively low, indicating that most of the activity was completed at a similar pace.
* **elev**: This column appears to contain the same data as **elev\_high** and **elev\_low**, so it is not clear what additional information it provides.
* **hr\_over\_180:** A variable called hr\_over\_180 was created and becomes a Boolean variable for if the activity had a heart rate over 180 bpm.

VI. Analysis

Recursive Feature Elimination (RFE) is a method that is used to train a selected model and eliminate the least important features recursively. The Code Explanations section will go further into detail on the results from code as well as the full linear regression process. To perform RFE, it is necessary to have the model already selected which in this case is linear regression.

The linear regression equation can be written as:

**max\_heartrate** = b0 + b1**average\_speed** + b2**max\_speed** + b3**average\_heartrate** + b4**moving\_time\_minutes** + b5**distance\_km** + b6**avg\_speed\_kmh** + b7**max\_speed\_kmh** + b8**hr\_over\_180**

Where:

**max\_heartrate** is the dependent variable

**average\_speed, max\_speed, average\_heartrate, moving\_time\_minutes, distance\_km, avg\_speed\_kmh, max\_speed\_kmh, and hr\_over\_180** are the independent variables

b0, b1, b2, b3, b4, b5, b6, b7, and b8 are the coefficients to be estimated by the linear regression model

The Mean Squared Error (MSE) is calculated using the values a model predicts for a given set of independent variables compared to the actual values associated with those variables in a data set. It measures how much the model is deviating from the true values. A high MSE value indicates that the model is making a large number of errors in its predictions, which can lead to unreliable or inaccurate results. In this case, the calculated MSE was 42.163, indicating that the model may have significant errors in its predictions. In addition to the MSE, a scatter plot can also be used to visually compare the predicted values with the actual values (Figure 5)

Chart, scatter chart

Description automatically generated

Figure 5: Scatterplot of Predicted vs True Values

The line to the scatter plot to represent the ideal case where the predicted values are equal to the true values.

From here we can use our model to predict my heartrate for an 800 time that I have ran before. I entered in speeds that were given to me from a distance calculator to run a 2 minute 800m race which I know I can run easily based on my personal best time of 1:53.3. The code snippet below gives returns us the heartrate.:

**Input 3**

model.predict(

pd.DataFrame(data={

'average\_speed': 6.6,

'max\_speed': 7,

'average\_heartrate': 160,

'moving\_time\_minutes': 2,

'distance\_km': .8,

'avg\_speed\_kmh': 23.76,

'max\_speed\_kmh': 25.2,

'hr\_over\_180': True

},

index=[0]

)

)

**Output 3**

array([183.45568062])

This code is making a prediction using a machine learning model. The model is called using the **model.predict()** function, which takes in a dataframe as its argument.

The dataframe being passed as an argument contains the input features to the model, represented as columns. The columns and their respective values are:

* **average\_speed**: the average speed of the activity, in meters per second
* **max\_speed**: the maximum speed achieved during the activity, in meters per second
* **average\_heartrate**: the average heart rate during the activity, in beats per minute
* **moving\_time\_minutes**: the duration of the activity, in minutes
* **distance\_km**: the distance covered during the activity, in kilometers
* **avg\_speed\_kmh**: the average speed of the activity, in kilometers per hour
* **max\_speed\_kmh**: the maximum speed achieved during the activity, in kilometers per hour
* **hr\_over\_180**: a boolean indicating whether the heart rate was above 180 beats per minute during any part of the activity.

The output of **model.predict()** is the predicted target value based on the input features. The specifics of the target variable and the model are not apparent from the code snippet.

The values for these features are passed to a Pandas DataFrame, which is then passed to the **predict** method of the **model** object. The **predict** method will return the predicted class label for the new data point based on the learned patterns in the model.

The model is able to make an accurate prediction based on my recorded max heart rates being well above the output heartrate showing that my training is at a place to where I can run those speeds for that time. It is interesting to see how the model acts when I put in speeds closer to my personal best, the heart rate result becomes much higher and most of the time over my max heartrate of 219 BPM’s. Even with a moderate MSE values, suggests that the model has some predictive power when it comes to athletic performance and max heart rate. While the MSE values indicate that the models are not perfectly accurate in their predictions, they are still able to provide some valuable insights into the relationship between these variables.

It's important to note that while the model was able to make accurate predictions, this does not necessarily mean that they will always be accurate in their predictions. There are many factors that can impact athletic performance, and it's possible that the models may not be able to account for all of these factors. Additionally, the models were trained on a specific dataset and may not perform as well on new, unseen data.

Overall, while the model may not be perfect, they are still a useful tool for understanding the relationship between max heart rate and athletic performance. By taking into account the limitations of the models and using them in conjunction with other sources of information, such as expert knowledge and personal experience, athletes and coaches can make more informed decisions about training and competition strategies.

To give more context to the areas of the data that we are in the next two figures show basis for the reasoning of my analysis, with Exercise Time vs Max Heart Rate, Distance (meters) vs Max Heartrate. Figure 6 shows the relationship between exercise time and maximum heart rate. This plot allows us to visualize whether there is a correlation between the duration of the exercise and the maximum heart rate that is achieved during that exercise. It can help us identify any trends or patterns that may emerge from the data.

Chart, scatter chart

Description automatically generated

*Figure 6: Exercise Time vs Max Heartrate*

Figure 7 plots the distance covered during each exercise session against the maximum heart rate achieved during that session. This plot can help us identify whether there is a relationship between the distance covered and the maximum heart rate achieved. It can also help us see if there are any outliers in the data, such as unusually high or low maximum heart rates.

Chart, scatter chart

Description automatically generated

*Figure 7: Distance vs Max Heartrate*

Overall, the figures provided give the reader a visual representation of the data and help them understand the basis for the analysis.

The correlation matrix (Figure 8) is a table that shows the correlation coefficient between variables. In this case, the focus is on variables related to heart rate. A correlation coefficient is a statistical measure that indicates the strength and direction of the relationship between two variables. It ranges from -1 to +1, where -1 indicates a perfect negative correlation, 0 indicates no correlation, and +1 indicates a perfect positive correlation.

A picture containing chart

Description automatically generated

*Figure 8: Correlation Matrix of Cleaned and Useful Variables*

The correlation matrix shows the correlation coefficient for each pair of variables in a table format. In this case, the matrix is focused on variables surrounding heart rate. The lighter squares in the matrix indicate stronger correlations between these variables. The correlation between two variables can be positive or negative. A positive correlation means that when one variable increases, the other variable also increases. A negative correlation means that when one variable increases, the other variable decreases.

By focusing on the lighter squares related to variables surrounding heart rate, we can see which variables are strongly correlated with heart rate. This information is useful for understanding how different variables affect heart rate and for developing predictive models that include heart rate as a predictor variable.

VII. Code Explanations

Recursive Feature Elimination (RFE) is a method that is used to train a selected model and eliminate the least important features recursively. To perform RFE, it is necessary to have the models already selected which in this case is linear regression.

It is important to note that RFE can be computationally expensive for high-dimensional datasets.

To implement RFE, a function is defined and used for linear regression. The function trains the selected model and eliminates features based on their importance recursively. For each iteration, the model is trained on the remaining features until only the most important ones are left. In this case, eight features are selected.

In summary, RFE is a powerful feature selection method that can help to improve the performance of machine learning models by eliminating irrelevant or redundant features.Top of Form

Bottom of Form

**Input 1**

**def** get\_best\_rfe\_features(X,y, model):

rfe **=** RFE(model, step**=**0.05)**.**fit(X, y)

selected\_features **=** [i **for** i, j **in** zip(X**.**columns, rfe**.**support\_) **if** j]

**return** selected\_features

y **=** tweak\_runs['max\_heartrate']

X **=** tweak\_runs**.**drop('max\_heartrate',1)

encoded\_y **=** preprocessing**.**LabelEncoder()**.**fit\_transform(y)

model **=** LinearRegression()

linear\_feats **=** get\_best\_rfe\_features(X, encoded\_y, model)

**Output 1**

['average\_speed', 'max\_speed', 'average\_heartrate', 'moving\_time\_minutes', 'distance\_km', 'avg\_speed\_kmh', 'max\_speed\_kmh', 'hr\_over\_180']

This code defines a function called **get\_best\_rfe\_features** that takes in three parameters:

* **X**: A pandas DataFrame containing the input features for the model.
* **y**: A pandas Series containing the response variable to be predicted by the model.
* **model**: A scikit-learn estimator object representing the model to be used for feature selection.

Within the function, the Recursive Feature Elimination (RFE) method is used to recursively eliminate features that are not deemed important by the model. Specifically, the **RFE** function is called with **model** as the estimator object and **step=0.05**, which means that 5% of the features will be eliminated in each iteration. The **fit** method is then called on **rfe** with **X** and **y** as inputs to train the model and select the best features. The **selected\_features** list is then created by zipping the column names in **X** with the Boolean values in **rfe.support\_** (which indicate whether or not each feature was selected), and filtering out the features that were not selected.

After defining the function, the code sets **y** and **X** to the response variable and input features from a pandas DataFrame called **tweak\_runs**, respectively. The **LabelEncoder** from scikit-learn is then used to encode **y** into integer labels, which are necessary for using **LinearRegression**. Finally, **get\_best\_rfe\_features** is called with **X**, **encoded\_y**, and a **LinearRegression** object to select the best features for a linear regression model. The selected features are stored in a list called **linear\_feats**.

In linear regression models, the objective is to predict a target variable y using a set of independent variables x. The model learns the values of the coefficients in the equation of a line by adjusting several lines in the feature map and selecting the line that yields the smallest error between the predicted and actual values of y.

In this specific application of linear regression, only the features selected in the previous step and stored in the variable **linear\_feats** will be used. To evaluate the performance of the algorithm, the dataset will be split into 80% for training and 20% for testing. This enables us to measure how well the model generalizes to unseen data.

After the data split, the training dataset is passed to the fit function of the Linear Regression model, which adjusts the coefficients of the line that best fits the training data. The model is then used to make predictions on the test dataset, and the performance of the model is evaluated by comparing the predicted values with the actual values.

**Input 2**

y = runs['max\_heartrate']

X = runs[linear\_feats]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

model = LinearRegression()

model.fit(X\_train,y\_train)

y\_pred = model.predict(X\_test)

Code chunk 2 is an implementation of linear regression on a dataset using the scikit-learn library in Python.

* The first line of code selects the target variable 'max\_heartrate' from the dataset 'runs' and assigns it to the variable 'y'.
* The second line selects the feature variables from the dataset 'runs' based on the features selected in the previous step (stored in the variable 'linear\_feats') and assigns them to the variable 'X'.
* The third line splits the dataset into training and testing sets using the 'train\_test\_split' function from scikit-learn. The training set will contain 80% of the data and the testing set will contain 20% of the data. The feature and target variables are split into separate training and testing sets.
* The fourth line creates an instance of the LinearRegression model.
* The fifth line fits the model on the training data using the 'fit' method of the LinearRegression object. This step trains the model by finding the best fit line that predicts the target variable based on the selected feature variables.
* The sixth line uses the trained model to predict the target variable 'max\_heartrate' for the testing data set by calling the 'predict' method of the LinearRegression object and passing it the feature variables 'X\_test'. The predicted values are assigned to the variable 'y\_pred'.

VIII. Limitations

While the linear regression model can be useful tools for predicting performance from max heart rate, it has its limitations.

The linear regression model can capture a more continuous range of outcomes, but they may oversimplify the relationship between max heart rate and athletic performance. Regression models assume a linear relationship between predictor and outcome variables, which may not hold in reality. Additionally, regression models may be sensitive to outliers and may not capture non-linear relationships between variables.

Similarly, the mean squared error of the regression model was 42.163, indicating that there was a relatively high amount of error between the predicted and actual values. This could be due to the limitations of assuming a linear relationship between max heart rate and athletic performance, as well as the potential for outliers and non-linear relationships to be present in the data.

IX. Recommendations

The recommendation to create a new variable by binning the average max heart rates in the activity can be a useful way to provide the model with additional information for better prediction of performance. The idea behind binning is to categorize a continuous variable, such as max\_heartrate, into discrete intervals or bins. In this case, we could divide the range of max\_heartrate values into 10 equal sections, and create a new variable that assigns each observation to one of these sections.

By creating this new variable, we are effectively converting a continuous variable into a categorical variable, which can be easier to work with in some cases. In addition, this can provide the model with more information to use in its predictions, as it can now take into account the specific range of max\_heartrate values that an observation falls into.

However, it is important to note that binning also has some limitations. For example, it can lead to loss of information, as we are reducing the precision of the variable by categorizing it into discrete intervals. Additionally, the choice of the number and size of the bins can affect the results of the analysis. If the bins are too small, we may end up with too few observations in each category, which can reduce the power of the analysis. On the other hand, if the bins are too large, we may lose some of the variability in the data and may not capture important patterns.

Therefore, when creating a new variable by binning the max\_heartrate variable, it is important to carefully consider the number and size of the bins, as well as to evaluate the impact of the binning on the model's performance. This can be done through techniques such as cross-validation, which can help us to assess the robustness of the model and ensure that it is not overfitting to the data.

X. Conclusion

Overall, we have highlighted the limitations and successes of a linear regression model predicting max heart and stresses the importance of considering other factors that may impact performance. The recommendation to create a new variable by binning the average heart rates in the activity is seen as a useful approach to provide the model with more information and improve performance prediction. By dividing the range of max\_heartrate values into 10 equal sections, the model can gain a more detailed understanding of how different ranges of max heart rates impact athletic performance. At the end of the day, this is a tool for personal use. I have a an intimate relationship with this dataset because I have curated it over the years. The choices made in the model were done to provide information to myself. Overall, this approach provides myself with a tool to see what times I can run, the only things I can realistically control in my race is how fast I go out and how long I can hold on. As a runner I don’t think of what my average heart rate needs to be or what my average speed should be at. This was an interesting dive into blending the runner with his data to get a better idea of everything that I have done in the last couple years.

Sources:

Silva, L. de B. (2022, December 30). *Data Analysis with Strava*. Medium. Retrieved March 26, 2023, from https://levelup.gitconnected.com/data-analysis-with-strava-7251327698e3

*The strava API*. Strava Developers. (n.d.). Retrieved March 26, 2023, from https://developers.strava.com/

*What you need to know about threshold training*. Runner's World. (2022, December 31). Retrieved March 26, 2023, from <https://www.runnersworld.com/advanced/a20807282/threshold-training/>

Esteve-Lanao, J., San Juan, A. F., Earnest, C. P., Foster, C., & Lucia, A. (2005). How do endurance runners actually train? Relationship with competition performance. Medicine and Science in Sports and Exercise, 37(3), 496–504.<https://doi.org/10.1249/01.MSS.0000155393.78761.03>

Jones, A. M., Doust, J. H., & Marsh, M. A. (2006). Sprint performance may be enhanced by increasing oxidative capacity. Journal of Sports Sciences, 24(10), 1021–1031.<https://doi.org/10.1080/02640410500457109>

Laursen, P. B., Blanchard, M. A., & Jenkins, D. G. (2002). Acute high-intensity interval training improves Tvent and peak power output in highly trained males. Canadian Journal of Applied Physiology, 27(4), 336–347.<https://doi.org/10.1139/h02-023>

*Will dauenhauer – track and field results & statistics*. TFRRS. (n.d.). Retrieved March 26, 2023, from https://www.tfrrs.org/athletes/6876302/Montana/Will\_Dauenhauer.html

Descriptive Appendix:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **distance** | **moving\_time** | **elapsed\_time** | **total\_elevation\_gain** | **average\_speed** | **max\_speed** | **average\_cadence** | **average\_heartrate** | **max\_heartrate** | **elev\_high** | **elev\_low** | **moving\_time\_minutes** | **distance\_km** | **pace** | **avg\_speed\_kmh** | **max\_speed\_kmh** | **elev** |
| **count** | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 | 914.0 |
| **mean** | 6797.030415754920 | 1874.4168490153200 | 2034.9792122538300 | 61.21083150984680 | 3.690333698030640 | 5.855002188183810 | 79.47100113540060 | 154.35582730542500 | 171.99441530268300 | 978.9473741794310 | 932.3180525164110 | 31.240371991247300 | 6.797144420131290 | 4.987381347868200 | 13.28382932166300 | 21.078074398249500 | 46.62932166301970 |
| **std** | 4280.375117836230 | 1158.5967104852000 | 1253.955303065690 | 98.3789080746621 | 0.6494957714172040 | 1.7697768177544700 | 3.3552252083383000 | 11.429453688894800 | 14.043918077041800 | 284.1541242133320 | 257.58063727914800 | 19.310013975397200 | 4.280419113142810 | 10.625443900994600 | 2.3345315281766000 | 6.3711766930827400 | 89.57637525309000 |
| **min** | 9.8 | 19.0 | 19.0 | 0.0 | 0.05 | 0.0 | 53.5 | 117.9 | 128.0 | 0.0 | 0.0 | 0.32 | 0.01 | 2.4761904761904800 | 0.18 | 0.0 | 0.0 |
| **25%** | 3207.925 | 885.25 | 922.25 | 9.1 | 3.37325 | 4.7115 | 78.4 | 148.525 | 165.0 | 981.0 | 972.2 | 14.755000000000000 | 3.21 | 4.2964908654563800 | 12.1425 | 16.965 | 5.399999999999980 |
| **50%** | 6473.700000000000 | 1742.5 | 1885.5 | 25.9 | 3.664 | 5.4 | 79.2 | 153.75 | 171.0 | 989.2 | 976.1 | 29.04 | 6.47 | 4.550104190132910 | 13.185000000000000 | 19.44 | 12.550000000000000 |
| **75%** | 9667.275 | 2604.0 | 2801.25 | 63.875 | 3.87825 | 6.6 | 80.1 | 161.075 | 181.0 | 1016.35 | 978.5 | 43.4 | 9.67 | 4.940462269738340 | 13.967500000000000 | 23.76 | 26.37500000000000 |
| **max** | 23872.3 | 7756.0 | 8027.0 | 697.1 | 6.887 | 15.7 | 101.5 | 193.6 | 218.0 | 2126.1 | 1845.3 | 129.27 | 23.87 | 325.0 | 24.23 | 56.52 | 618.6000000000000 |

Code Appendix:

**Creating Token**

**import** os

**import** json

**import** requests

response **=** requests**.**post(

url**=**'https://www.strava.com/oauth/token',

data**=**{

'client\_id': client\_id,

'client\_secret': client\_secret,

'code': strava\_code,

'grant\_type': 'authorization\_code'

}

)

strava\_tokens **=** response**.**json()

**with** open('strava\_tokens.json', 'w', encoding**=**'utf-8') **as** outfile:

json**.**dump(strava\_tokens, outfile)

print(strava\_tokens)

**Getting Activities**

**import** requests

**import** pandas **as** pd

**import** os

**import** json

**import** glob

**import** time

**import** requests

**import** pandas **as** pd

**from** dotenv **import** load\_dotenv

load\_dotenv()

**def** main():

url **=** "https://www.strava.com/api/v3/activities"

access\_token **=** get\_credentials()

page **=** 1

print('Getting data from Strava')

**while** **True**:

response **=** get\_data(url, access\_token, 200, page)

**if** 'message' **in** response**.**columns:

**raise** Exception('Authorization Error')

**if** response**.**empty:

**break**

save\_csv(response, f'data/strava\_activities\_page\_{page}.csv')

page **+=** 1

merge\_files('data/', 'result/strava\_all\_activities.csv')

print('Done Successfully')

**def** get\_credentials():

**with** open('strava\_tokens.json', encoding**=**'utf-8') **as** json\_file:

strava\_tokens **=** json**.**load(json\_file)

**if** 'expires\_at' **not** **in** strava\_tokens**.**keys() **or** strava\_tokens['expires\_at'] **<** time**.**time():

strava\_tokens **=** refresh\_credentials(strava\_tokens)

**return** strava\_tokens['access\_token']

**def** refresh\_credentials(strava\_tokens):

response **=** requests**.**post(

url**=**'https://www.strava.com/oauth/token',

data**=**{

'client\_id': int(os**.**environ**.**get('client\_id')),

'client\_secret': os**.**environ**.**get('client\_secret'),

'grant\_type': 'refresh\_token',

'refresh\_token': strava\_tokens['refresh\_token']

}

)

strava\_tokens **=** response**.**json()

**with** open('strava\_tokens.json', 'w', encoding**=**'utf-8') **as** outfile:

json**.**dump(strava\_tokens, outfile)

**with** open('strava\_tokens.json', encoding**=**'utf-8') **as** check:

data **=** json**.**load(check)

**return** data

**def** get\_data(url, access\_token, numb\_item\_page, page):

print(f'Getting data from page {page}')

response **=** requests**.**get(

f'{url}?access\_token={access\_token}&per\_page={numb\_item\_page}&page={page}'

)

response **=** response**.**json()

dataframe **=** pd**.**json\_normalize(response)

**return** dataframe

**def** save\_csv(dataframe, filename):

print(f'Saving {filename}')

dataframe**.**to\_csv(filename)

**def** merge\_files(path, filename):

print('Merging files')

csv\_files **=** [pd**.**read\_csv(\_file)

**for** \_file **in** glob**.**glob(os**.**path**.**join(path, "\*.csv"))]

final\_df **=** csv\_files**.**pop(len(csv\_files)**-**1)

final\_df **=** final\_df**.**append(csv\_files)

save\_csv(final\_df, filename)

**if** \_\_name\_\_ **==** '\_\_main\_\_':

main()

**Strava API Analysis:**

**import** subprocess

**import** calendar

**from** geopy.geocoders **import** Nominatim

**import** pandas **as** pd

**import** numpy **as** np

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

**import** matplotlib.dates **as** mdates

*# machine learning*

**from** sklearn **import** preprocessing

**from** sklearn **import** metrics

**from** sklearn.feature\_selection **import** chi2

**from** sklearn.feature\_selection **import** SelectKBest

**from** sklearn.feature\_selection **import** RFE

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.linear\_model **import** SGDClassifier

**from** sklearn.metrics **import** accuracy\_score

**from** sklearn.metrics **import** confusion\_matrix, ConfusionMatrixDisplay

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.pipeline **import** make\_pipeline

**from** sklearn.preprocessing **import** StandardScaler

df **=** pd**.**read\_csv('result/strava\_all\_activities.csv')

print('Dataframe Shape:', df**.**shape)

df**.**head()

Dataframe Shape: (1043, 60)

Out[1]:

5 rows × 60 columns

In [2]:

df2 **=** pd**.**read\_csv('MeetData2.csv')

df2**.**head(5)

Out[2]:

|  | **Meet Name** | **Event** | **start\_date** | **Result** | **Place** | **PR** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 2023 Big Sky Indoor Track & Field Championships | 800 | 2023-02-24 | 01:57.7 | 22nd (P) | 0 |
| **1** | Bobcat Performance | 800 | 2023-02-10 | 01:57.4 | 4th (F) | 0 |
| **2** | 2023 WSU Open | 800 | 2023-02-03 | 01:56.2 | 17th (F) | 0 |
| **3** | 2023 Lauren McCluskey Memorial | 800 | 2023-01-20 | 01:58.3 | 12th (F) | 0 |
| **4** | 2023 Spokane Indoor Challenge | 800 | 2023-01-13 | 01:55.8 | 6th (F) | 0 |

In [3]:

null\_df **=** [[col, df[col]**.**isnull()**.**sum()] **for** col **in** df**.**columns]

print('Null Data:', df**.**isnull()**.**sum()**.**sum())

list(filter(**lambda** x: x[1]**>**0, null\_df))

Null Data: 7888

Out[3]:

[['workout\_type', 974],

['location\_city', 1043],

['location\_state', 1043],

['location\_country', 104],

['gear\_id', 186],

['average\_cadence', 116],

['average\_heartrate', 114],

['max\_heartrate', 114],

['elev\_high', 57],

['elev\_low', 57],

['upload\_id', 8],

['upload\_id\_str', 8],

['external\_id', 8],

['suffer\_score', 114],

['map.summary\_polyline', 66],

['average\_watts', 948],

['kilojoules', 948],

['device\_watts', 939],

['average\_temp', 1041]]

In [4]:

selected\_columns **=** ['distance', 'moving\_time', 'elapsed\_time',

'total\_elevation\_gain', 'type','sport\_type', 'id', 'start\_date',

'start\_date\_local','location\_country', 'achievement\_count', 'kudos\_count',

'comment\_count','athlete\_count', 'start\_latlng',

'end\_latlng', 'average\_speed', 'max\_speed', 'average\_cadence',

'average\_heartrate', 'max\_heartrate', 'elev\_high','elev\_low',

'upload\_id', 'external\_id', 'pr\_count', 'map.summary\_polyline']

df **=** df[selected\_columns]

In [5]:

df['start\_date\_local'] **=** pd**.**to\_datetime(df['start\_date\_local'], errors**=**'coerce')

df **=** df**.**sort\_values(by**=**'start\_date\_local')

df['weekday'] **=** df['start\_date\_local']**.**map(**lambda** x: x**.**weekday)

df['start\_time'] **=** df['start\_date\_local']**.**dt**.**time

df['start\_time'] **=** df['start\_time']**.**astype(str)

df['start\_date'] **=** df['start\_date\_local']**.**dt**.**date

df **=** df**.**drop('start\_date\_local', 1)

df**.**head()

/Users/willd/anaconda3/lib/python3.7/site-packages/ipykernel\_launcher.py:9: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only

if \_\_name\_\_ == "\_\_main\_\_":

Out[5]:

5 rows × 28 columns

In [6]:

df **=** df**.**drop(df[(df**.**distance **<** 1) **&** (df**.**type **==** 'Run')]**.**index)

df **=** df**.**drop(df[(df**.**distance **<** 1) **&** (df**.**type **==** 'Ride')]**.**index)

df **=** df**.**drop(df[df**.**average\_speed **>** 30]**.**index)

df **=** df**.**reset\_index(drop**=True**)

df['elev\_high'] **=** df['elev\_high']**.**fillna(value**=**0)

df['elev\_low'] **=** df['elev\_low']**.**fillna(value**=**0)

df['upload\_id'] **=** df['upload\_id']**.**fillna(value**=**'unknown')

df['external\_id'] **=** df['external\_id']**.**fillna(value**=**'unknown')

df['map.summary\_polyline'] **=** df['map.summary\_polyline']**.**fillna(value**=**'unknown')

df['average\_cadence'] **=** df['average\_cadence']**.**fillna(value**=**df['average\_cadence']**.**mean())

df['average\_heartrate'] **=** df['average\_heartrate']**.**fillna(value**=**df['average\_heartrate']**.**mean())

df['max\_heartrate'] **=** df['max\_heartrate']**.**fillna(value**=**df['max\_heartrate']**.**mean())

df['moving\_time\_minutes'] **=** round(df['moving\_time']**/**60, 2)

df['distance\_km'] **=** round(df['distance'] **/** 1000, 2)

df['pace'] **=** df['moving\_time\_minutes'] **/** df['distance\_km']

df['avg\_speed\_kmh'] **=** round(60**/**df['pace'], 2)

df['max\_speed\_kmh'] **=** round(df['max\_speed']**\***3.6, 2)

df['elev'] **=** df['elev\_high'] **-** df['elev\_low']

df['year']**=** df['start\_date']**.**map(**lambda** x: x**.**year)

In [7]:

df**.**head()

Out[7]:

5 rows × 35 columns

In [8]:

**def** get\_city\_state\_from\_value(value):

value **=** value**.**replace('[','')**.**replace(']','')**.**split(',')

**if** value **!=** ['']:

location **=** geolocator**.**reverse(', '**.**join(value))

result **=** f'{location[0]**.**split(",")[1]}, {location[0]**.**split(",")[1]}'

**else**:

result **=** 'unknown'

**return** result

In [9]:

geolocator **=** Nominatim(user\_agent**=**"strava\_exploration\_data")

df['location'] **=** df['start\_latlng']**.**map(get\_city\_state\_from\_value)

In [10]:

df**.**head()

Out[10]:

5 rows × 36 columns

In [20]:

df['hr\_over\_180'] **=** np**.**where(df['max\_heartrate']**>=**180, **True**, **False**)

df**.**head()

Out[20]:

5 rows × 37 columns

In [21]:

fig **=** sns**.**catplot(x**=**'year', hue**=**'type', data**=**df, kind**=**'count')

fig**.**fig**.**suptitle('Exercises by Years')

fig**.**set\_xlabels('Year')

fig**.**set\_ylabels('Effortments')

fig

Out[21]:

<seaborn.axisgrid.FacetGrid at 0x7fcb0207c410>

**Chart, bar chart

Description automatically generated**

In [22]:

runs **=** df**.**loc[df['type'] **==** 'Run']

sns**.**regplot(x**=**'moving\_time\_minutes', y **=** 'max\_heartrate', data**=**runs)**.**set\_title("Exercise Time vs Max Heartrate")

Out[22]:

Text(0.5, 1.0, 'Exercise Time vs Max Heartrate')

**Chart, scatter chart

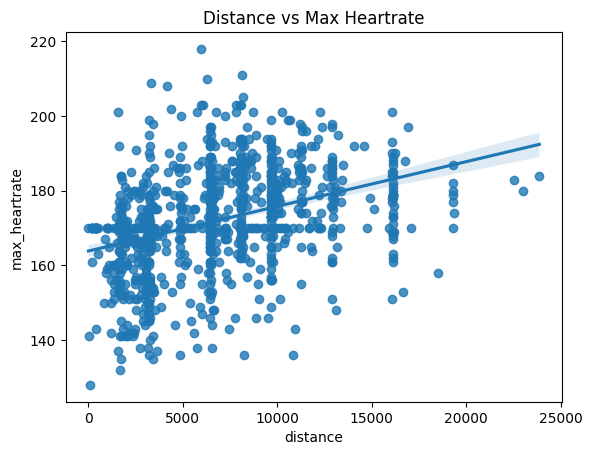
Description automatically generated**

In [23]:

sns**.**regplot(x**=**'distance', y **=** 'max\_heartrate', data**=**runs)**.**set\_title("Distance vs Max Heartrate")

Out[23]:

Text(0.5, 1.0, 'Distance vs Max Heartrate')

****

In [37]:

sns**.**regplot(x**=**'avg\_speed\_kmh', y **=** 'max\_heartrate', data**=**runs)**.**set\_title("Average Speed vs Max Heartrate")

Out[37]:

Text(0.5, 1.0, 'Average Speed vs Max Heartrate')

**Chart, scatter chart

Description automatically generated**

In [24]:

runs**.**groupby('weekday')**.**mean()['max\_heartrate']**.**plot**.**bar()

Out[24]:

<AxesSubplot:xlabel='weekday'>

**Chart, bar chart

Description automatically generated**

In [25]:

corr **=** runs**.**corr()

plt**.**figure(figsize **=** (12,8))

sns**.**heatmap(corr, fmt**=**".2f");

plt**.**title('Correlation between dataset variables')

plt**.**show()

**Chart

Description automatically generated**

In [26]:

df**.**describe()

Out[26]:

8 rows × 25 columns

In [28]:

runs **=** runs**.**sample(frac**=**1)**.**reset\_index(drop**=True**)

categorical\_cols **=** [col **for** col **in** runs**.**columns **if** runs[col]**.**dtypes **==** 'O']

useless\_vars **=** ['id', 'achievement\_count', 'kudos\_count', 'comment\_count', 'pr\_count','athlete\_count','year','weekday']

tweak\_runs **=** runs**.**drop(categorical\_cols**+**useless\_vars, axis**=**1)

tweak\_runs

Out[28]:

914 rows × 18 columns

In [33]:

corr **=** tweak\_runs**.**corr()

plt**.**figure(figsize **=** (12,8))

sns**.**heatmap(corr, fmt**=**".2f");

plt**.**title('Correlation between cleaned dataset variables')

plt**.**show()

**A picture containing chart

Description automatically generated**

In [30]:

max\_hr **=** df['max\_heartrate']**.**nlargest(10)

print(max\_hr)

570 218.0

629 211.0

861 210.0

157 209.0

669 208.0

647 205.0

510 204.0

229 203.0

591 203.0

701 203.0

Name: max\_heartrate, dtype: float64

In [47]:

tweak\_runs**.**describe()**.**to\_csv('tweak\_runs\_summary.csv')

In [32]:

fig **=** sns**.**catplot(x**=**'year', data**=**runs, kind**=**'count')

fig**.**fig**.**suptitle('Runs by Years')

fig**.**set\_xlabels('Year')

fig**.**set\_ylabels('Effortments')

fig

Out[32]:

<seaborn.axisgrid.FacetGrid at 0x7fcb03d384d0>

**Chart, bar chart

Description automatically generated**

In [38]:

**def** get\_best\_rfe\_features(X,y, model):

rfe **=** RFE(model, step**=**0.05)**.**fit(X, y)

selected\_features **=** [i **for** i, j **in** zip(X**.**columns, rfe**.**support\_) **if** j]

**return** selected\_features

In [39]:

y **=** tweak\_runs['max\_heartrate']

X **=** tweak\_runs**.**drop('max\_heartrate',1)

encoded\_y **=** preprocessing**.**LabelEncoder()**.**fit\_transform(y)

model **=** LinearRegression()

linear\_feats **=** get\_best\_rfe\_features(X, encoded\_y, model)

print(linear\_feats)

['average\_speed', 'max\_speed', 'average\_heartrate', 'moving\_time\_minutes', 'distance\_km', 'avg\_speed\_kmh', 'max\_speed\_kmh', 'hr\_over\_180']

/Users/willd/anaconda3/lib/python3.7/site-packages/ipykernel\_launcher.py:2: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only

In [40]:

y **=** runs['max\_heartrate']

X **=** runs[linear\_feats]

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.2)

In [41]:

model **=** LinearRegression()

model**.**fit(X\_train,y\_train)

y\_pred **=** model**.**predict(X\_test)

In [42]:

print('MSE:', metrics**.**mean\_squared\_error(y\_test, y\_pred))

plt**.**figure(figsize**=**(10,10))

plt**.**scatter(y\_test, y\_pred, c**=**'crimson')

p1 **=** max(max(y\_pred), max(y\_test))

p2 **=** min(min(y\_pred), min(y\_test))

plt**.**plot([p1, p2], [p1, p2], 'b-')

plt**.**xlabel('True Values (BPM)', fontsize**=**15)

plt**.**ylabel('Predictions (BPM)', fontsize**=**15)

plt**.**axis('equal')

plt**.**show()

MSE: 45.19387221399379

**Chart, scatter chart

Description automatically generated**

In [43]:

model**.**predict(

pd**.**DataFrame(data**=**{

'average\_speed': 6.6,

'max\_speed': 7,

'average\_heartrate': 160,

'moving\_time\_minutes': 2,

'distance\_km': .8,

'avg\_speed\_kmh': 23.76,

'max\_speed\_kmh': 25.2,

'hr\_over\_180': **True**

},

index**=**[0]

)

)

Out[43]:

array([183.45568062])