Luis Perez

Professor Gill

Machine Learning

8 May 2023

Predicting Total Weight Lifted by Powerlifters using Machine Learning

For context powerlifting is a strength sport consisting of three main lifts done by competitors. Each competitor gets three opportunities on each lift to lift the max amount possible starting with the squat, then the bench press, and finally the deadlift. The goal of competing is to lift as much weight as possible in each of these lifts, and the total weight lifted across all three lifts is used to determine the winner of a competition. The main objective of this project is to create a model that predicts the total weight lifted in kilograms (TotalKg) by powerlifters based on the following features: age, weight class, body weight, sex, equipment, and division. The potential applications of this project include providing a baseline weight goal for a powerlifter to push for during a competition, we can also identify key trends and patterns in powerlifting. The dataset from Openpowerlifting.org contains powerlifting competition data, which includes information about individual lifters, the sex of the lifter, the event they participated in, and which type of equipment they used in their lifts, such as raw or wraps. In our data set, we have other data points that include the lifter's country of origin, federation, meet date, meet country, meet state, and meet name, which we will not be using. Given that our data set has various data points that are not useful we must be cautious in cleaning and processing our data to avoid biases and other challenges. Regarding how this data was prepared we can assume each federation or competition meet has officials who record the data from the competition and publish them to their federations and is then processed by Openpowerlifting.org and made open to the public.

However, we have to keep in mind potential biases found in the data. For one the data contains professional elite-level lifters and amateur novice lifters, this could make it difficult to accurately represent the broader population of powerlifters since there will be a very significant difference in performance between these two classes. Another potential bias that is out of our control is what actually happens in competition. Variables such as, how strict the judges or officials are in enforcing competition rules, like proper lift execution, or depth requirements will impact the total amount lifted by any competitor. Finally, the most difficult potential bias to deal with is the possibility of competitors using performance enhancing drugs, each federation has different testing protocols, and some allow performance enhancing drug usage. Now based on a preliminary data exploration on the entire dataset, we can see that the dataset contains a huge amount of missing values, and duplicate data points, for reference see section 1. In our preliminary data exploration, we also noticed some trends amongst competitors. First, a large majority of lifters are male, typically powerlifting is a male-dominated sport, but it has had a rise in popularity and more women are competing. Most of the male and female lifters are also between the ages of 15 and 45, with the highest being between the ages of 20 and 25, so this is more likely where we will see the strongest athletes. The majority of male lifters are between 75 and 90 kilograms. The majority of female lifters are between 50 and 60 kilograms. This implies that sex is an important feature in the lift totals of a competitor.

We now discuss our data cleaning and preprocessing of the dataset. The first data cleaning technique seen in section 2a, is all about removing irrelevant data such as dropping columns, converting column data to numeric data, and dropping duplicated rows and rows containing null or NaN values in the features we are using, we utilize pandas and numpy libraries all throughout the project. We also filter the dataset into a small subset consisting of only raw

and wraps as equipment. Raw typically means no equipment is used such as knee sleeves, wrist wraps, etc. The reason for this is that during our preliminary data exploration in section 1, we noticed that the box plot “Lift total in KG by Sex”, has hugely significant outliers; this is due to equipped lifting which heavily increases the weight total of a competitor. In the second data cleaning technique seen in section 3a, we follow a simpler approach we create a new data frame by keeping only the columns containing our features, and we drop all null values in the ‘TotalKg’ column, we also drop duplicated data as well, we, however, don’t drop null or NaN values from the feature columns as we will be using imputation to handle this in our data processing. We decided to apply these techniques from what we gathered in our preliminary data exploration, the data contained inconsistencies such as missing data entries and duplicated data. As for our data processing seen in section 2b, we encode categorical columns to numerical values using a label encoder, so features like sex, equipment, and division are encoded. We also scale features such as age, weight class, and age class using sklearn standard scaler. In section 3b we follow a different approach: we utilize one-hot-encoding. As we previously mentioned during our preliminary data exploration we encountered many missing values, our goal in this data transformation in section 3b is to perform imputation on the missing values. Given that we now have our data cleaned and processed we can now do a data analysis on our data. We split our data into training, testing, and validation, we create a subset for both sexes, this is because there is a notable difference in strength between men and women, reference section 4. Each sex is analyzed using descriptive statistics and visualized using Matplotlib and Seaborn, reference sections 5a-5b. Below is a sample of the descriptive statistics

Descriptive statistics of “sex” competitors:

	Age	BodyweightKg

count	float	float
mean	float	float
std	float	float
min	float	float
25%	float	float
50%	float	float
75%	float	float
max	float	float

The first statistic gives us some insights on the amount of data we have, the more data we have the better it is for the models we will be using since we need to have sufficient data. The mean tells us the average age and body weight of a male a female competitor, this will tell us about where most lifters are competing. The standard deviation tells us about the variability of the data. Given this, for the male competitors the 'Age', has a standard deviation of 13.41 years, and for 'BodyweightKg', it is 18.08 kg. For the female competitors the 'Age', has a standard deviation of 12.08 years, and for 'BodyweightKg', it is 11.20 kg. The min and max are the range of the data for both subsets, these statistics can help us understand possible outliers that can affect the performance of the model.

We now discuss model implementation. We implement three machine learning models: Linear Regression, K-Nearest-Neighbor, and Multi-layer Perceptron. Linear regression is a supervised learning algorithm that models a linear relationship between a dependent variable and one or more independent variables. Linear regression is the interpretable model, in our context, however, it may not be the most optimal algorithm to use, but it will serve as a good base model to compare the performance of the other models we will be using. K-Nearest-Neighbor is another

supervised learning algorithm that can be used for regression and classification problems, this algorithm has some flexibility as we can adjust the value of K to enhance algorithm performance. In essence, K-Nearest-Neighbor works by looking at the K-nearest training examples of a particular data point. Based on this it then predicts the output by considering these neighbors around the data point for classification problems. Averaging the values of the neighbors surrounding a data point is used in regression problems. The reason we utilize K-Nearest-Neighbor is the possibility the relationship between the variables in the data is not linear, so relying solely on linear regression would not be optimal. Multi-layer Perceptron is a type of feedforward artificial neural network composed of multiple perceptron models. The perceptrons in multi-layer perceptrons are interconnected and parallel, this consists of input and output layers that are fully connected. The model is trained iteratively by updating the parameters with the partial derivatives of the loss function. This machine learning algorithm can capture more complex non-linear relationships between variables. However, we must acknowledge some strengths and weaknesses of each model. The strengths of linear regression come from the assumption of a linear relationship that our data might have. The weakness comes from the limits of the model as it is not able to capture more complex non-linear relationships. The strength of K-Nearest-Neighbors is that it can capture more advanced relationships between the variables and has an easy implementation. Its weakness comes from the fact that it is computationally expensive as the dataset size increases. Also, it requires careful selection of the hyperparameter K which is crucial for high model performance and can be difficult to find. Multi-layer Perceptron has the strength advantage of capturing complex, non-linear relationships in the dataset and possibly performing better than the previous algorithms. However, its weakness

comes from the careful selection of hyperparameters, and the challenge of determining how exactly the model made its predictions.

We now discuss training and fine-tuning the models. We use mean absolute error, mean squared error, and R-squared to measure model performance. MSE tells us about the average absolute difference between actual and predicted values, a lower MAE means better model performance. MSE tells us the average squared distance between the actual and predicted values, a lower MSE means better model performance, this metric is also sensitive to outliers. R-squared tells us the proportion of variance in the dependent variable that is predictable from the independent variables, the higher the R-squared value the better the model performance. In sections 6, 7a-7b we notice that the alternative data cleaning and processing consistently outperforms the original data cleaning and processing for all three models in terms of mean absolute error, mean squared error, and R-squared. This suggests the imputation and one-hot-encoding, and min-max scaling were more suitable for this dataset. Among the three models, multi-layer perceptron achieved the best performance on the validation set, with the lowest mean absolute error, error and mean squared error, and the highest R-squared value being respectively MAE: 129.89, MSE: 24898.97, R-squared: 0.3975. This tells us this model most likely does not require fine-tuning and is better suited to this dataset; however, it does come with a trade this model took significantly longer to train on the alternative data cleaning and processing dataset. We now discuss section 8 where we look at the training performance and validation performance of each model to discern which models are overfit or underfit. Linear regression on the original data cleaning and processing had similar performance on the training and validation with relatively high MAE and MSE, indicating that the model did not overfit, but we had a low R-squared value, perhaps showing signs of underfitting. The alternative data cleaning and

processing improved in performance, with a lower MAE, MSE, and a higher R-squared value, however, it still shows signs of underfitting. K-Nearest Neighbors on the original data cleaning and processing had better performance than linear regression but had a weaker performance on the validation set which could mean it was overfitting since the model is performing better on the training data than on the validation data. The performance of the model also improved with the alternative data cleaning and processing in both the training and validation, but the R-squared value for the validation was still lower than the training. The model might also have slight overfitting since there is a noticeable difference between the training and validation MAE and MSE. Multi-layer Perceptron on the original data cleaning and processing had a worse performance as compared to the alternative data cleaning and processing. This model seems to be underfitting since both the training and validation set performances have relatively high MAE and MSE values and a low R-squared value. The alternative data cleaning and processing approach had training and validation set performances with a lower MAE and MSE and a higher R-squared value. Overall K-Nearest Neighbors has an even higher R-squared value and is likely the best-performing model among all six techniques. To address overfitting in K-Nearest Neighbors, we can try increasing the number of neighbors k. One thing we could have done to improve multi-layer perceptron model performance is to add more layers or nodes, or adjust activation functions, however, due to the time it took to train the model we opted out of this since it was computationally expensive.

We now discuss the results of the project as seen in sections 9-10. We have the following metric with the right being the model performance on the first data cleaning and processing method, and on the left the alternative.

Model name: linear regression

Model name: linear regression

Total training time: 0.078 seconds

Memory size: 48 bytes

Mean Absolute Error: 148.56

Mean Squared Error: 30980.34

R-squared: 0.25

Total training time: 0.10 seconds

Memory size: 48 bytes

Mean Absolute Error: 140.78

Mean Squared Error: 27644.24

R-squared: 0.33

Model name: k-nearest-neighbor

Total training time: 0.46 seconds

Memory size: 48 bytes

Mean Absolute Error: 127.85

Mean Squared Error: 27358.32

R-squared: 0.34

Model name: k-nearest-neighbor

Total training time: 0.51 seconds

Memory size: 48 bytes

Mean Absolute Error: 119.71

Mean Squared Error: 24122.29

R-squared: 0.42

Model name: multi-layer perceptron

Model name: multi-layer perceptron

Total training time: 306.41 seconds	Total training time: 1124.63 seconds
Memory size: 48 bytes	Memory size: 48 bytes
Mean Absolute Error: 135.05	Mean Absolute Error: 129.39
Mean Squared Error: 27910.39	Mean Squared Error: 24773.67
R-squared: 0.32	R-squared: 0.4

Based on these metrics we can see that the best-performing model was the K-Nearest Neighbor trained on alternative data cleaning and processing. It had the highest R-squared value, the lowest MAE, and the second lowest MSE among all the models. Linear regression performed worse and had a lower R-squared value with higher error rates meaning it was not the best model for this particular problem. Multi-layer perceptron models on the other hand had relatively high R-squared values with lower error rates but were more computationally expensive as it took longer to train these models. The reason K-Nearest Neighbor performed the best is that it is able to capture complex relationships between features and the target variable when the relationships in the data are not linear, which is most probable for this dataset. Changing the hyperparameter k improved model performance.

We now finish off with a discussion and conclusion. This project is aimed to predict the total weight lifted in kilograms during a powerlifting competition based on features such as age, weight class, body weight, sex, equipment, and division. We performed an exploratory data analysis that allowed us to make decisions on how to clean and process the raw data. We then

cleaned and preprocessed the data in two distinct methods highlighting the importance of cleaned and processed data. We split the data into training, testing, and validation. We analyzed the male and female subsets of the training data which provided us with some insight on the type of competitors in our dataset. We implemented three distinct machine learning models and found the best-performing model to be the K-Nearest Neighbor algorithm trained on the alternative data cleaning and processing approach. This model can be used to provide a baseline weight goal for powerlifters to aim for during a competition, with further work it can help us identify key trends and patterns in powerlifting. Moving forward with research in this topic and dataset, we can take into account the biases and challenges we discussed including if a competitor was tested for performance enhancing drugs, as this could significantly impact the data. One thing we could do is to keep only competitors from federations in the dataset with stringent performance enhancing drug testing. As we gathered from the data cleaning and processing section, the method for which we clean and process data has a huge impact on the model's performance, moving forward we must apply more cleaning and processing techniques as the quality of the data is one of the most crucial components in model performance. We also must experiment with other models and improve upon the work done with K-Nearest-Neighbor and apply more appropriate feature engineering work to the project.

Works Cited

Abba, Ihechikara Vincent. “KNN Algorithm – K-Nearest Neighbors Classifiers and Model Example.” *FreeCodeCamp.org*, FreeCodeCamp.org, 25 Jan. 2023, https://www.freecodecamp.org/news/k-nearest-neighbors-algorithm-classifiers-and-model-example/.

Ajitesh KumarI have been recently working in the area of Data analytics including Data Science and Machine Learning / Deep Learning. I am also passionate about different technologies including programming languages such as Java/JEE. “Sklearn Neural Network Example - Mlpregressor.” *Data Analytics*, 2 May 2023, https://vitalflux.com/sklearn-neural-network-regression-example-mlpregressor/.

McCullum, Nick. “How to Build and Train Linear and Logistic Regression ML Models in Python.” *FreeCodeCamp.org*, FreeCodeCamp.org, 30 June 2020, https://www.freecodecamp.org/news/how-to-build-and-train-linear-and-logistic-regression-ml-models-in-python/.

Navlani, Avinash. “Multi-Layer Perceptron Using Python.” *Medium*, Medium, 13 Dec. 2022, https://avinashnavlani.medium.com/multi-layer-perceptron-using-python-68315c00ec8b.