Predicting absolute strength of powerlifters using regression methods

1. Introduction

1.1 Background and overview

Powerlifting is a strength sport where competitors aim to lift maximal amount of weight in three lifts: squat, bench press and deadlift.(1) As discussed in section 2, this machine learning problem aims to predict powerlifters' strength levels using data from World Championships 2023 scoresheet. Linear and polynomial regressions as well as random forests are used to attain effect of age and bodyweight on total lifted weight. Section 3 discusses feature selection, data splitting as well as the choice of loss function. In sections 4 and 5 we discuss results and conclude our findings.

1.2 Application domain

Results can be used in forecasting powerlifting competition results by predicting the weight competitors will lift. Results are likely not applicable when studying men's class participants(2) or leisure and grassroot level lifters.

2. Problem formulation

2.1 Aim of the project

We want to predict total lifted weight of a powerlifter competing in women's class using their characteristics as predictors, aiming to extract effects on absolute strength.

2.2 Data

Data studied is the score sheet from World Open Classic Powerlifting Championships 2023 arranged by International Powerlifting Federation (IPF). Data is collected from OpenPowerlifting(3) and cleaned data includes 166 data points, each describing a women's class powerlifter who attended the competition.

Original data has 41 columns describing competitors' characteristics (name, bodyweight, sex, age, country, division etc.), all lifts (squat, bench, deadlift, and their total) performed in the competition, and performance indicators (placement, dots, wilks etc.). We are interested in age, bodyweight, and lifts, all of which are continuous variables. Age is measured in years and bodyweight and lifts in kilograms. Weight lifted is calculated as follows ('Best3' indicating the best result out of 3 attempts):

$$\text{Best3SquatKg} + \text{Best3BenchKg} + \text{Best3DeadliftKg} = \text{TotalKg} \quad (4)$$

2.3 Machine learning task

First, we will run a linear regression and polynomial regression, using bodyweight and age as features and total weight lifted as the label. Second, we are constructing random forest to predict the total weight lifted. These are forms of supervised learning, as we are using labelled input and output data.

3. Methods

3.1 Data cleaning

Original data includes total of 367 data points and 41 columns. From raw data we exclude all men's class competitors and those women's class competitors who failed all three attempts of at least one of the lifts (squat, bench, or deadlift), as their total result is not calculated. We are keeping columns describing competitors' background information (Name, Sex, Age,

BodyweightKg, WeightClassKg), lifted total weight (TotalKg) and key performance indicators (Place, Dots and Wilks). Note that competitors compete within their weight class, so column 'Place' can have matching values. After data cleaning we have total of 166 data points for the study. Figure 1 shows an example of the cleaned data.

	Name	Sex	Age	BodyweightKg	WeightClassKg	TotalKg	Place	Dots	Wilks
0	Mara Hames	F	25.0	75.10	76	472.5	17	459.88	448.80
1	Monica de La Torre	F	27.5	51.85	52	390.0	11	476.33	487.27
2	Martina Malzová	F	27.5	51.65	52	372.5	13	456.18	466.80
3	Miriam Amri	F	27.5	56.70	57	432.5	9	497.23	503.95
4	Mayara Soares	F	30.5	46.70	47	327.5	11	431.25	442.49

Figure 1. First 5 rows of cleaned data.

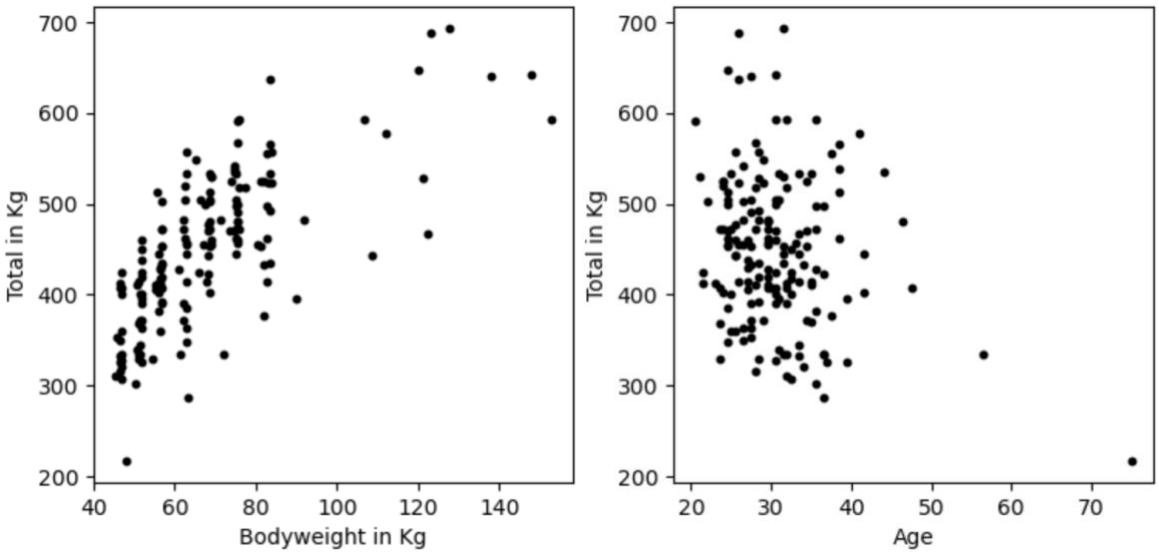
3.2 Model and feature selection

We will run a linear regression and polynomial regression with polynomial degrees 2-5, using bodyweight and age as features and total weight lifted as the label. Bodyweight was chosen as a feature, as it is intuitive that a person's size affects their strength. In powerlifting competition competitors are split to weight classes(4), supporting the link between bodyweight and strength. As all competitors are weighed prior to the event(4), it is also easy to collect ready-to-use data of bodyweight.

Age should also affect strength levels: muscle mass decreases when aging(5), meaning that older lifters should on average be able to lift less. Indeed, mild negative correlation between age and lifted weight was found (see appendix 7.1). Other features were not included, as variables in data were either i) categorical (country, division, weight class) ii) multicollinear (wilks, dots and other performance measures are calculated using weight lifted). Total weight lifted was chosen as label, since it best describes the absolute strength of a powerlifter, summing up results from all different lifts.

Linear maps are used as Figure 2 shows clear positive relationship between bodyweight and total weight lifted, and the relationship seems somewhat linear especially in the lower weight classes. However, plotting age against total result yields negative, but inconclusive relationship (Figure 3). Choosing linear regression as our initial ML model allows us to study the linear relationship and to determine coefficients indicating marginal effects of bodyweight and age on the total weight lifted. However, as the linearity of relationship between chosen features and label is ambiguous, we are also running polynomial regressions of degrees 2 to 5, allowing us to compare different models against each other.

Random forests are chosen as the second model as it provides a tool to implement decision trees on problems with continuous labels, and often yields high accuracy. Random forests construct multiple decision trees and average their results, which also reduces overfitting by single decision trees.(6) We are using maximal tree depth of 3 to keep the computing times reasonable.



Figures 2 and 3. Scatter plot of total lifted weight against bodyweight and age.

3.3 Data splitting and loss function

We are splitting data so that 34 data points (20%) are left out for test set and 132 points (80%) are assigned to training set. This allows us to use most of the scarce data on training the model. Due to the data scarcity, we are using leave-one-out cross validation, where model is trained n=166 times on n-1 data points of the training set and validated on the last left out data point. Leave-one-out cross validation should result in better estimation of the expected loss and prevent bias from poor choice of validation set.(7)

Mean squared error (MSE) is chosen as loss function when training and validating the model. MSE is a standard loss function for both regression and random forest models, and it is computationally cheap due to its convexity and differentiability.(7) When testing the chosen model, we are also calculating mean absolute error (MAE) as it is intuitive and easy to interpret: MAE indicates how many kilograms our prediction of total lifted weight is off on average.(7)

4. Results

4.1 Linear and polynomial regression

Regressions of different polynomial degrees yield mean squared errors presented in Figure 4. Linear regression gives intercept of 348.3 and coefficients 2.9 (bodyweight) and -3.2 (age), indicating that 1 additional kilogram of bodyweight increases total result by 2.9 kilograms and additional year of age decreases weight lifted by 3.2 kilograms (appendix 7.2). We see that training error of linear regression (=polynomial regression of degree 1) is ≈2900 and error is reduced by adding polynomial degrees. However, validation error is at its lowest when using polynomial regression of degree 2, and rapidly increases when adding degrees. This indicates overfitting in the training data. Thus, polynomial regression of degree 2 is the best of regression models studied.

Loss of linear and polynomial regressions:

polynomial degree	training error	validation error
1	2888.5	3041.6
2	2608.9	2845.0
3	2500.2	5103.2
4	2352.9	4014.9
5	2083.8	5057309.6

Figure 4. Training and validation errors (MSE) in regressions.

4.2 Random forests
Random forests yields training error (MSE) of 1984.2, which is significantly smaller than errors obtained from regressions. However, k-fold cross-validation error (MSE) is 3183.1, which is slightly higher than of linear regression and 2nd degree polynomial regression. Thus, random forests fails to provide a better model and our final chosen method is polynomial regression of degree 2.

4.3 Testing the model
As discussed in section 3.3, 20 percent of data is randomly chosen and left out for testing the model. This is performed by using Scikit learn's(8) train_test_split-function. We are testing the 2nd degree polynomial regression model by comparing predicted label values against actual label values in the test set. This gives us MSE of 2994.1 and MAE of 40.6, both of which are our test errors. See appendix 7.3 for summary of model. Our final model is as follows (X_{bi} denoting bodyweight and X_{ai} denoting age of competitor i):

$$\hat{y}_i = 259.37 + 6.03 \times X_{bi} - 6.00 \times X_{b}^{2}{}_{i} - 0.03 \times X_{ai} + 0.04 \times X_{a}^{2}{}_{i} + 0.01 \times X_{bi} \times X_{ai}$$

5. Conclusion
5.1 Summary and interpretation of results
As discussed, 2nd degree polynomial regression was chosen to predict the strength levels of women's class participants using their age and bodyweight as features. Model yields mean absolute test error of 40.6, which indicates an average misprediction of ≈41 kilograms on the total weight lifted. Mean of label variable is 449 kilograms, meaning a misprediction of around one tenth. Training and validation errors are close to each other (2609 and 2845, respectively) indicating that there is no significant overfitting.

5.2 How to develop the ML method
To attain a better estimate, more datapoints and feature variables should be introduced in the model. More datapoints could be attained easily by collecting and merging data from other competitions. This would reduce variance and diminish effect of outliers. However, increasing number of feature variables is trickier, as it is unlikely to find utilizable public data of all or most individual competitors. Some possible feature variables could include training background and antidoping testing history of competitors.

Also, random forests regression results indicate a possible overfitting, as the validation error is a lot higher than testing error. This problem could be addressed by reducing the number of trees or removing branches of decision trees (pruning). As random forests is sensitive to variance, overfitting could also be significantly reduced by including more data points in the study.(9)

5.3 Result optimality
Results seem reasonable when considered the scarce amount of data and small number of possible feature variables. However, model would fail to predict the outcome of a competition, as it gives only a rough estimate and does not take individual differences, such as genetics or training background, into account. It is important to note that no model can perfectly predict outcome of a powerlifting competition, as random events such as injuries, competitor's mood, or even pure luck, can affect the results on a big day. After all, the unpredictability is what makes sports competitions exciting.

6. References

1. Wikipedia. Powerlifting. Available at: https://en.wikipedia.org/wiki/Powerlifting, accessed 6.10.2023.
2. Bartolomei S et al. A Comparison between Male and Female Athletes in Relative Strength and Power Performances. J Funct Morphol Kinesiol. 2021 Feb 9;6(1):17. doi: 10.3390/jfmk6010017. PMID: 33572280; PMCID: PMC7930971. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7930971/
3. OpenPowerlifting dataset. Available at: https://www.openpowerlifting.org, accessed 19.9.2023.
4. IPF Technical Rules Book. (1/2023). Available at: https://www.powerlifting.sport/rules/codes/info/technical-rules
5. Volpi E, Nazemi R, Fujita S. Muscle tissue changes with aging. Curr Opin Clin Nutr Metab Care. 2004 Jul;7(4):405-10. doi: 10.1097/01.mco.0000134362.76653.b2. PMID: 15192443; PMCID: PMC2804956. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2804956/
6. Besheti, N. (2.3.2022). Random Forest Regression. Towards Data Science. Available at: https://towardsdatascience.com/random-forest-regression-5f605132d19d, accessed 6.10.2023.
7. A. Jung,“Machine Learning: The Basics,” Springer, Singapore, 2022
8. Scikit-learn: Machine Learning in Python. Available at: https://scikit-learn.org/stable/, accessed 22.9.2023.
9. Saturn Cloud. (6.7.2023). How to Solve Overfitting in Random Forest of Python sklearn. Available at: https://saturncloud.io/blog/how-to-solve-overfitting-in-random-forest-of-python-sklearn/, accessed 4.10.2023.

7. Appendix

7.1 Correlation table of features and label.

	Age	BodyweightKg	TotalKg
Age	1.00	-0.03	-0.24
BodyweightKg	-0.03	1.00	0.72
TotalKg	-0.24	0.72	1.00

7.2 Linear regression

```
Linear regression intercept:  348.27037323806826
Linear regression coefficient for bodyweight:  2.944024731872602
Linear regression coefficient for age:  -3.2321048814890205
```

7.3 Summary of 2nd degree polynomial regression model.

```
Intercept: 259.37
Coefficients:

 6.0266
-5.9972
-0.0248
 0.0403
 0.0058

Test error (MAE): 40.64
Test error (MSE): 2994.11

Mean of TotalKg: 448.64
```

7.4. Code

October 8, 2023

[ ]:
```
#import numpy, pandas and plotting tools
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from numpy import mean, correlate, cov

# Import regression tools & CV tools
%config Completer.use_jedi = False
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import KFold, train_test_split, cross_val_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression

#change maximum number of rows and columns
pd.options.display.max_rows = 9999
pd.options.display.max_columns = 9999

#import data
df = pd.read_csv('ipf_2023.csv')
df.info()

#clean data
df = df[df['Sex'] == 'F']
df = df[df['Best3SquatKg'] > 0]
df = df[df['Best3BenchKg'] > 0]
df = df[df['Best3DeadliftKg'] > 0]
df = df[df['MeetName'] == 'World Classic Powerlifting Championships']

df = df[['Name','Sex', 'Age', 'AgeClass', 'BodyweightKg','WeightClassKg',␣
 ↪'TotalKg', 'Place','Dots','Wilks']]
print(df.head(5))

#sanity checking the variables under study
MaxValues = df.max()
```

```
print(MaxValues)
MinValues = df.min()
print(MinValues)

#scatter plot (bodyweight)
plt.figure(figsize=(8, 6))
plt.scatter(df['BodyweightKg'], df['TotalKg'], color='black', s=10)
plt.xlabel('Bodyweight in Kg')
plt.ylabel('Total in Kg')

#scatter plot (age)
plt.figure(figsize=(8, 6))
plt.scatter(df['Age'], df['TotalKg'], color='black', s=10)
plt.xlabel('Age')
plt.ylabel('Total in Kg')

#correlation
pd.options.display.float_format = '{:.2f}'.format
print(df.corr(method='pearson', min_periods=1, numeric_only=True))

#setting features and labels
features = []
labels = []
X = df[['BodyweightKg', 'Age']]
y = df['TotalKg']


feature = X.to_numpy()
features.append(feature)
label = y.to_numpy()
labels.append(label)

#split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,␣
 ↪random_state=0)

print(X_test.shape)
print(X_train.shape)

#collect testing and validation errors
stats = []

#run linear regression with n-fold cv
cv = KFold(n_splits=len(X_train), shuffle=True, random_state = 1)

model = LinearRegression().fit(X_train,y_train)
```

```
scores = cross_val_score(model, X_train, y_train, cv=cv,␣
 ↪scoring='neg_mean_squared_error')
y_train_pred = model.predict(X_train)

scores = -scores
mean_error = np.mean(scores)

train_error = mean_squared_error(y_train, y_train_pred)

stats.append([1, train_error, mean_error])

print()
print("Linear regression intercept: ", model.intercept_)
print("Linear regression coefficient for bodyweight: ", model.coef_[0])
print("Linear regression coefficient for age: ", model.coef_[1])
print()

#run polynomial regression with n-fold cv
#Reference: CS-C3240 Machine Learning D (Autumn 2023), Assignment 1 Student␣
 ↪task A1.9
degrees = [2,3,4,5]
cv_poly = KFold(n_splits=len(X_train), shuffle=True, random_state = 1)

for i in range(len(degrees)):
    poly = PolynomialFeatures(degree=degrees[i])
    X_poly = poly.fit_transform(X_train)
    poly_model = LinearRegression().fit(X_poly,y_train)

    y_poly_train_pred = poly_model.predict(X_poly)
    poly_train_error = mean_squared_error(y_poly_train_pred, y_train)

    scores = cross_val_score(poly_model, X_poly, y_train, cv=cv_poly,␣
 ↪scoring='neg_mean_squared_error')
    scores = -scores
    mean_error = np.mean(scores)

    stats.append([degrees[i], poly_train_error, mean_error])

#print stats
table = pd.DataFrame(stats, columns = ['polynomial degree', 'training error',␣
 ↪'validation error'])
pd.options.display.float_format = '{:.1f}'.format
print(" Loss of linear and polynomial regressions:")
print(table.to_string(index=False))

#random forests
cv = KFold(n_splits=len(X_train), shuffle=True, random_state = 1)
```

```
model = RandomForestRegressor(max_depth=3, random_state=0)
model.fit(X_train, y_train)

scores = cross_val_score(model, X_train, y_train, cv=cv,␣
 ↪scoring='neg_mean_squared_error')
y_train_pred = model.predict(X_train)

scores = -scores
mean_error = np.mean(scores)

forest_train_error = mean_squared_error(y_train, y_train_pred)

print("Training error (MSE): ", forest_train_error)
print("K-fold cross validation error (MSE): ", mean_error)

#test model
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X_train)
poly_model = LinearRegression().fit(X_poly,y_train)

X_poly_test = poly.fit_transform(X_test)
y_poly_pred = poly_model.predict(X_poly_test)

test_mae_error = mean_absolute_error(y_test, y_poly_pred)
test_mse_error = mean_squared_error(y_test, y_poly_pred)

#print stats
pd.options.display.float_format = '{:.4f}'.format

intercept = poly_model.intercept_
coefficients = poly_model.coef_[1:]
coefs = pd.DataFrame(coefficients)

mean_total = mean(df['TotalKg'])

print("Intercept: {:.2f}".format(intercept))
print("Coefficients:")
print()
print(coefs.to_string(index=False, header=False))
print()

print("Test error (MAE): {:.2f}".format(test_mae_error))
print("Test error (MSE): {:.2f}".format(test_mse_error))
print()

print('Mean of TotalKg: {:.2f}'.format(mean_total))
```