Machine Learning in Fitness and Health

Thor Olesen (TVAO) and Dennis Thinh Tan Nguyen (DTTN)

Superviser: Sami Brandt

CONTENTS

1 INTRODUCTION

Today, most people have a physical goal they want to achieve, such as losing weight, gaining muscle or maintaining their weight while living a healthy life. Common for all of these goals, meal planning and working out is a central part of achieving it. Some people hire a nutritionist or personal trainer to help achieve their goal, but due to expensive prices and personal preferences many people look to digital solutions instead. However, the current landscape of digital solutions seem indifferent to personal preferences and often force people into changing the way they live, which is not realistic for people who wish to live healthy in the long run without compromising their lifestyle. In addition, often people simply don't know how to workout and eat optimally so they instead rely on uninformed or habitual routines.

We are in the era of implementation for artificial intelligence (AI) and machine learning (ML). This is especially proven by the big tech giants such as Amazon, Netflix and Google that all use Machine Learning techniques to provide customized and personal content to improve the customer experience when buying books, watching movies or finding relevant search terms. We believe there is room for such improvement in the landscape of Fitness and Health where Machine Learning has not yet been fully exploited and leveraged to provide personalized, data-driven and automated training or diet plans for ordinary people.

1.1 Research Question

This has resulted in the investigation of answering the following research question:

- Research Question: What data and Machine Learning methods are suitable for optimizing fitness training?

Sub-questions in order to answer the main question:

- Background: What domain knowledge is relevant to optimize training according to current literature?
- Model Design: Which features and machine learning models are suitable for predicting fitness training?
- Model Evaluation: How do these different machine learning methods compare?

2 METHODS

We have used literature studies, in which we read many papers on the topics of powerlifting, strength training and muscle hypertrophy, which helped us gather domain knowledge relevant to the feature engineering in our model design.

We have followed the Data Science Lifecycle in our work, which involves the process of formulating a question or problem, acquiring and cleaning data, conducting exploratory data analysis and using prediction to draw conclusions (24). Namely, we set out to formulate the problem of optimizing strength training as a time-dependent regression problem through a custom growth measure of strength explained in the model design. We acquired training data from three individuals using the Strong app (23) to track their weightlifting workouts by day, repetitions, weights, sets and exercises. The data was preprocessed, which includes cleaning, outlier anomaly detection, reduction, normalization, transformation, feature extraction and selection to obtain training data. The prepared training data was explored using a variety of data visualization techniques.

Feature engineering was performed by using domain knowledge from our literature study and the insights about the features and their correlations from our data exploration to decide on the final set of features.

Finally, we used a variety of Machine Learning methods to model the problem of estimating strength growth over time as a timeseries regression forecasting problem. We experimented with a few different combinations of features, models, hyper parameters, evaluation metrics, time steps and dimensionality reduction. This was done until we obtained a model that is able to generalize well by predicting expected strength growth over time on new unseen test data in the future. All with the aim of showing that it may be feasible to predict optimal training based on some notion of growth in your training data.

3 BACKGROUND

Firstly, a literature study will be performed to figure out what kind of data, features and domain knowledge may be incorporated in our models to predict training plans for individuals. Secondly, we will explore different machine learning models and feature engineering methods to figure out which features and mathematical models best predict training plans for individuals. Finally, we will do a comparison of the used models to make an evaluation and reflection of our results. Notice that we will strive to select existing data from individuals that have already made their training records publicly available to minimize the need of manual data collection.

3.1 The Fitness and Powerlifting domain

General fitness training works towards broad goals of overall health and well-being. Fitness training is undertaken to improve or maintain one's physical fitness and health. This is typically done for competition, growing larger muscles, becoming stronger, increasing endurance or due to concerns over appearance. Fitness training may be supplemented by a healthy die that helps promote calorie burning by boosting metabolism and ensuring a calorie deficit. In this particular project, we strive to figure out how to optimize fitness training based on concrete training data without looking at other hidden (i.e. latent) factors such as diet, sleep, stress, energy, motivation, injuries, sleep, soreness, training protocol and training experience. To do so, we need to agree on a standard metric used to measure the success of optimal training. Namely, one can choose to optimize for many fitness goals such as strength, endurance, weight loss, muscle growth or conditioning.

In our case, **the objective is to optimize for strength in the domain of powerlifting exemplified through the squat powerlift** to ensure the project is scoped and feasible given the limited time frame available to figure out whether it is possible to optimize training. Powerlifting is a strength sport in which three fitness exercises are used to evaluate the overall strength of an individual. The powerlift exercises consist of the squat, deadlift and bench press of which we look at the squat as an indicator of raw strength. This choice boils down to the fact that we need a common reference point for comparison, the exercise with the most workout entries in the available data was the squat, and the squat exercise is a good indicator of raw strength. Namely, the squat relies on the biggest muscle group of the body composed of the glutes, quads and hamstrings. However, it is mainly a good indicator of lower body strength and would have to be complemented by a further analysis of upper body strength in the bench press to give a better impression of overall strength. The exercise can be performed for a range of repetitions, weights and sets, which indicates how many times you can lift a certain weight in an exercise.

3.2 Literature study

According to literature, there are a number of factors affecting strength in the context of the three power lifts. Amongst others, body weight, age, height, gender, training protocol, experience and frequency affect strength (9; 18). Other unobserved (latent, hidden) factors may include average daily calorie intake, activity level, genetic factors, stress, sleep, hunger, motivation and body composition. These are all possible confounding factors, which are not readily available in the data set but may have an influence on the training. Based on the publications we read, the following domain knowledge and indicators were found.

Firstly, training groups demonstrate significant strength decreases after certain weeks of detraining independent of prior training intensity, meaning that the frequency of training seems to affect strength (10). Secondly, aerobic training that precedes strength training leads to a diminished volume of work for up to 8 hours and the impairment appears to be localized to the muscle groups involved in the aerobic training (11). Thirdly, it seems that training groups generally experience strength gain for different training intensities ranging from 6 to 15 RM of a given exercise (12) where **1 RM is the most weight you can lift for one repetition**. However, heavier traing loads (70-90% of 1RM) tend to increase peak force in the body more so than the use of lighter training loads of 40-60% of 1 RM. Generally, a high resistance group seems to improve squats more than a low-resistance group. The results of such study support the use of heavier training loads to increase 1RM strength (13).

Further, the range of motion in resistance training affects strength. According to a survey, lifting through a full range of motion was superior to partial range of motion measures in the development of maximal upper-body strength in women (14). Also, strength and power traits are most likely influenced by the various training protocols used (15). The strength of an individual may be evaluated using a variety of predictions equations that estimate the 1RM metric, which is a good predictor of strength. However, it has been shown that these formulas mainly provide a good estimate of 1RM in the three powerlifts if fewer than 10 RTF were used where RTF stands

for reps to failure (16) and only the deadlift is underestimated (20). The most common formulas used to calculate 1RM (i.e. how much weight you can lift maximally in a single repetition) are the Epley and Brzycki formulas:

$$Epley: 1RM = w(1 + \frac{r}{30}), \text{ assuming } r > 1 \text{ where r = repetitions and w = weight}$$

$$Brzycki: 1RM = w \cdot \frac{36}{37 - r} = \frac{w}{\frac{37}{36} - \frac{1}{36}r} \approx \frac{w}{1.0278 - 0.0278r} \text{ where r = repetitions}$$

The 1RM is either calculated directly using maximal testing where the individual lift the most weight possible for one repetition. However, people often prefer the submaximal estimation, sicne it is safer althrough it may underestimate the actual 1RM. Hence, the above formulas help take any combination of weights and repetitions where it is assumed that the lift was performed until maximal exhaustion and predict the 1RM from this. The above formulas return identical results for 10 repetitions, whereas the formula 1 returns a little higher maximum for fewer reps. These kind of calculations may be used as rought indicators of strength (17). Most importantly, one should stick with either formula to get consistent indications of strength progress for a particular exercise over time. As such, we have chosen to use the first formula for all results.

In addition, the effect of training volume (i.e. amount of repetitions) and intensity (i.e. amount of weight) affect improvements in muscular strength and size. Namely, a high-intensity (3-5 RM), low-volume resistance training program with long rest intervals (3 min) is more advantegous than a moderate intensity, high-volume (10-12 RM) program with short rest intrval (1 min) for stimulating body strength and muscle hypertrophy in resistance-trained men. As such, emphasizing training intensity (i.e. heavier weights) over volume (i.e. increase of repetitions) may provide accelerated muscle growth and strength gains. Put simply, individuals that lift heavier weights with fewer repetitions experience greater strength increase (19).

One may also use **Prilepin's Chart**, which is a table of optimal Olympic weight lifting rep ranges to be used for training that was created by A.S. Prilepin who was a Soviet era sports scientist. The chart was developed by reviewing the training journals of thousands of weightlifting athletes and is meant to portray the optimal number of reps per set, and total rep count (volume) for power training required for the Olympic lifts. It does not address the frequency of workout and training age but only measure the maximal power output, which is different than strength training. Hence, we do not know if it applies to "slower" powerlifts like the deadlift, squat and bench press. However, recent research shows that Prilepin's chart cn be used to make strength gains in the three powerlifts in resistance trained males. As such, it can be used as a strength training tool where increasing strength in powerlifts is important. Finally, the frequency of training affects muscle growth (MPS = muscle protein synthesis), which seems to stagnate 20 hours after a workout according to the figure below on the right. Hence, this may indicate that one can train each muscle group at least twice a weak for maximum muscle growth.

Prilepin's Table			
%	Reps/Set	Range	Optimal
<70	3-6	18-30	24
70-79	3-6	12-24	18
80-89	2-4	10-20	15
90+	1-2	4-10	7

Figure 1. Optimal volume and intensity table

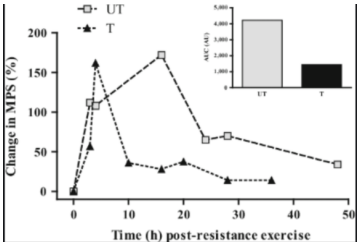


Figure 2. Post-workout muscle protein synthesis

Recall that the overall objective of the project is to determine whether it is feasible to optimize fitness training based on a given growth measure, such as strength, muscle hypertrophy, endurance. The scope has been narrowed down to optimizing strength within powerlifting, since this is the growth metric that has been deemed most tangible, realistic and quantifiable. For example, one would need the training group to measure their body part sizes regularly to detect any change in muscle hypertrophy and we do not have equipment to measure conditioning or endurance. To sum up, strength can be measured directly from the available lifting data and the literature studies should help gain insight into what domain knowledge and features should be considered to optimize powerlifting strength within the domain of fitness and health.

4 MODEL DESIGN: THEORY AND IMPLEMENTATION

4.1 Machine Learning and the Bias Variance Tradeoff

Definition

"ML is the training of a model from data that generalizes a decision against a performance measure"

A machine learning model can be a mathematical representation of a real-world phenomena. The aim is to find a mapping between inputs and outputs (i.e. the program) or a learning algorithm that finds patterns inherent to the input training data such that the input parameters correspond to the target. The output of this is a model, which can make a decision or prediction based on unseen inputs in the future. The model is evaluated using a performance measure based on the actual model predictions and target predictions we want (25). Ultimately, the objective is to find a model that generalizes well to new unseen instances of the phenomena we try to model.

Overfitting and Underfitting

The generalization ability of a model depends on its complexity and is best described through the notions of overfitting and underfitting and the bias-variance dilemma. Namely, a model that has good performance on the training data but poor generalization to other (i.e. test) data is overfitting and too complex. On the other hand, a model that has poor performance on the training data and poor generalization to other (i.e. test) data is underfitting and too simple. Put simply, we need a way to balance between an overfitting model that is too complex and follows the noise of of our training data but misses the patterns that hold across new data and an underfitting, oversimplified model of low complexity that does not capture the underlying pattern inherent in the data.

Bias-Variance Tradeoff

These two modelling opposites above are best described through the two traits, bias and variance. Bias is the degree to which a model oversimplifies the relationships inherent in the data. Variance is the degree to which a model follows a specific data set too closely. Overfitting is a condition wherein the model has low bias, but high variance. Underfitting is a condition wherein the model has high bias, but low variance. Models of either type are unlikely to generalize well, failing to predict values accurately when exposed to data sets outside of the training data set. Thus, the best models strike a balance between overfitting and underfitting, capturing the pattern and not the noise within the data . This is called the bias-variance tradeoff, which is the tension between the prediction error introduced by the bias and the variance that we can control for unlike the error from noise. Recall the generalization (i.e. test or prediction) error is sum of the bias squared, variance and the irreducible error inherent to the noise of data as shown below. Ideally, we want a model with low bias and low bias but these are each other's opposites so we strive for the middle ground. Below is shown the prediction error decomposed into its bias squared, variance and noise error terms. y is the variable we are trying to predict, x is our input data and we assume $y = f(x) + \varepsilon$ where the error term ε is normally distributed with a mean of zero so $\varepsilon \approx N(0, \sigma_e)$. We estimate a model $\hat{f}(x)$ of $y = f(x)$ using a specific model and the expected squared prediction error is the squared difference between the input estimate $\hat{f}(x)$ and our target $y = f(x)$.

$$\text{Mean Squared Error(x)} = E[(y - \hat{f}(x))^2], \text{ where } y = f(x) + \varepsilon$$

$$\text{Bias Variance Decomposition: } (E[\hat{f}(x)] - y)^2 + E[(\hat{f}(x) - E[\hat{f}(x)])^2] + \sigma_e^2$$

$$\text{expected total loss} = (bias)^2 + \text{ variance } + \text{ noise (Bishop, eq. 3.41, p. 149 (2))}$$

Thus, we experiment with different levels of model complexity and choose the complexity level that minimizes the overall error to find a model that generalizes well as shown in the figure below by Scott Fortmann-Roe (7).

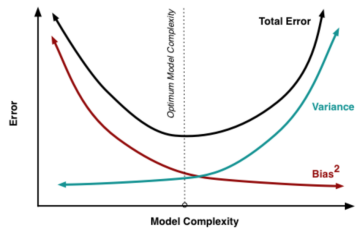


Figure 3. Bias and variance contributing to total error by Scott Fortmann-Roe

4.2 Data Collection: The Strong Weightlifting Data Set

The collected training data is comprised of the following set of features: Date, Workout Name, Exercise Name, Set Order, Weight, Reps, Distance, Seconds, Notes, Workout Notes. The three individuals that logged their training data through the Strong weightlifting. None of them made any workout notes or general notes and the Seconds and Distance features are only used for running and other cardiovascular exercises, which do not focus on strength growth so we disregard those. In the future, one would need to collect substantially more training data to obtain a representative sample of individuals across geography, gender, age, body type, training experience, protocol, frequency, diet and so on. This could be done through fitness facility partnerships, manual collection through a prototype similar to Strong or reaching out to fitness enthusiasts on online training forums.

	Date	Workout Name	Exercise Name	Set Order	Weight	Reps	Distance	Seconds	Notes	Workout Notes
0	2015-10-23 17:06:37	Chest	Incline Bench Press (Barbell)	1	135.0	8	0.0	0	NaN	NaN
1	2015-10-23 17:06:37	Chest	Incline Bench Press (Barbell)	2	135.0	8	0.0	0	NaN	NaN
2	2015-10-23 17:06:37	Chest	Incline Bench Press (Barbell)	3	135.0	5	0.0	0	NaN	NaN
3	2015-10-23 17:06:37	Chest	Incline Bench Press (Barbell)	4	185.0	7	0.0	0	NaN	NaN
4	2015-10-23 17:06:37	Chest	Incline Bench Press (Barbell)	5	230.0	8	0.0	0	NaN	NaN

Figure 4. Raw Strong Weightlifting Workout Data

	Date	Exercise Name	Set Order	Weight	Reps
0	2015-10-23 17:06:37	Incline Bench Press (Barbell)	1	135.0	8
1	2015-10-23 17:06:37	Incline Bench Press (Barbell)	2	135.0	8
2	2015-10-23 17:06:37	Incline Bench Press (Barbell)	3	135.0	5
3	2015-10-23 17:06:37	Incline Bench Press (Barbell)	4	185.0	7
4	2015-10-23 17:06:37	Incline Bench Press (Barbell)	5	230.0	8

Figure 5. Strong Weightlifting Workout Data without Distance, Seconds, Notes and Workout Notes

4.3 Data Preprocessing: Cleaning, Transformation (Outliers, Normalization), Exploration, Selection

Data preprocessing includes data cleaning, data reduction (or feature extraction), data transformation, and feature selection, data normalization, which will be described in this section. We use the Pandas library in Python that provides data structures and data analysis tools for manipulating the weightlifting data in a tabular format.

Data Cleaning (see 1_data_preparation.ipynb)

Data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records in the data where incomplete, incorrect, inaccurate or irrelevant parts of the data are replaced, modified or deleted (26). Firstly, pounds are converted to kg to keep workout entries in a consistent weight metric format:

```
df_workouts['Weight] = df_workouts['Weight] / 2.2046 # kg = lb/2,2046
```

Secondly, we noticed some unusually high weight values across the three powerlifts. For example, one entry said that the individual had squatted 1340 kg for one rep max, which is more than the world record. It is highly unlikely that this individual is able to beat the world record. In such situations, this is probably because the individual mistakenly added another 0 to the entry, since 134 kg seems more reasonable given his past lifts also lie around this value. Normally, we could do data imputation and replace the value by the average weight lifted in the exercise but since it was only very few entries we decided to remove them entirely. Doing data imputation did not seemingly affect our results. However, had there been many of these cases, it would have been wise to do data imputation by replacing such outlier values with average weight values across specific exercises.

```
max_weight_index = kaggle_df['Weight'].idxmax(axis=1)
max_weight_record = kaggle_df.iloc[max_weight_index] # 1340 kg by Kaggle individual
is_beyond_1000_kg = kaggle_df['Weight'] > 1000 # beyond world record
kaggle_df = kaggle_df[ ~ is_beyond_1000_kg]
df_workouts[kaggle_index] = kaggle_df
kaggle_df['Weight'].max() # now around 200 kg
```

Thirdly, we remove any workout running entries, since we do not consider endurance or conditioning but focus on strength in lifting exercises:

```
is_running_exercise = c_df['Exercise Name'].str.contains('Running')
df_no_running = df_workouts[ ~is_running_exercise ] # 862 entries removed
```

Finally, we lower the exercise names, convert string dates to datetime objects and chose to keep exercises separate even though some are very correlated. For example, one may perform variations of the same exercise using different angles (e.g. barbell squat, dumbbell squat, sumo squat). We noticed that the individuals perform somewhat differently on these and deemed it best to look at exercises independently for best results. Intuitively, it makes sense that the volume of work and intensity is affected by the variation of the exercise. On one side, collapsing all variations of the squat would yield more workout entries. However, it would bias the results and the individuals are strongest in the barbell squat, which has the most entries so we use that.

```
df[feature] = df[feature].str.lower()
df['Date'] = pd.to_datetime(df['Date'])
```

To sum up, we have a data set comprised of the features: date, exercise name, weight, reps and set order. Recall from the literature study, that volume (i.e. number of repetitions and sets) and intensity (i.e. amount of weight) and training frequency (obtained from date) are strong indicators of strength.

As such, we will need to augment the original data with some notion of training frequency shown in the feature engineering section where we enrich the data by keeping track of how many days ago an individual last trained based on the date column. Further, we need to somehow measure relative strength over time, which is estimated using the 1RM formula. Finally, to control for any trend and seasonality inherent to the timeseries data (i.e. individuals seem to grow stronger over time as a trend and train differently depending on season as shown in the data exploration section), we focus our attention to how the 1RM estimate develops from day to day through differencing to obtain a stationary time series, which is more stable and easier to model.

Data Transformation (see 1_data_preparation.ipynb)

The raw summary statistics can be found using the Pandas describe() operation and outliers can be identified as the observations that are numerically distant from the rest of the data. This is done using boxplots on all numeric colums (i.e. sets, weights, reps) and outliers are the observations located outside the whiskers (i.e. fences) of the boxplot (e.g. 1.5 interquartile range above upper quartile or below lower quartile).

	Set Order	Weight	Reps
mean	2.450039	34.986926	8.676279
std	1.459510	35.682418	5.503070
min	1.000000	0.000000	0.000000
25%	1.000000	10.732686	5.000000
50%	2.000000	24.000000	8.000000
75%	3.000000	50.000000	10.000000
max	12.000000	1770.000000	75.000000

Figure 6. Summary statistics of all weightlifting data

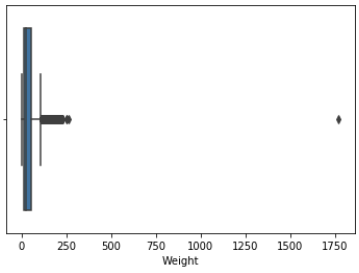


Figure 7. Weight Boxplot

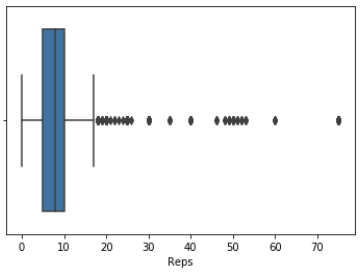


Figure 8. Reps Boxplot

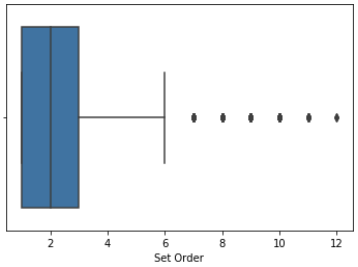


Figure 9. Set Boxplot

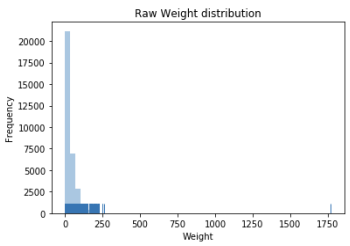


Figure 10. Weight Distribution

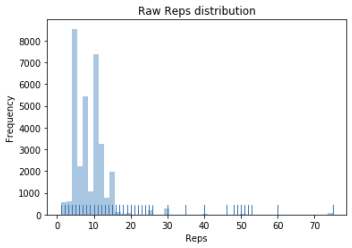


Figure 11. Reps Distribution

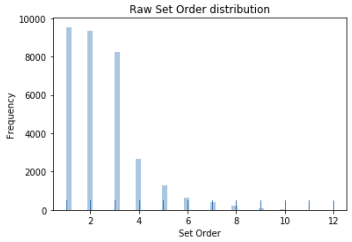


Figure 12. Set Distribution

As can be seen, the distribution of weights is very right skewed due to outliers, the reps distribution is most dense in the range 0 to 20 but also has quite a few outliers in the high rep range although it is only moderately right skewed and the set order only has very few outliers. It seems that all distributions are somewhat right skewed, which indicates there are some times when the individuals have trained with a higher volume (i.e. number of sets and repetitions) and intensity (i.e. amount of weight) than usual.

Outlier Detection and Removal

One quickly notices that the weight distribution has one observation that is absurdly large, which is a squat where over 1750 kg was lifted for one rep. However, this is inconceivable and is most likely a human error caused in the manual entry in the Strong app, since the raw powerlifting world record is below 700 kg. As such, even though it may be considered bad practice to remove outlier observations from a data set due to bias, we deemed this one observation to be absurd. Outliers like these affect most parametric statistics (e.g. means, standard deviations, and correlations) that are highly sensitive to outliers and will mess up our analysis by affecting any regressions we perform or correlations we try to investigate between the numeric features of the data.

Arguably, it is not acceptable to drop an observation just because it is an outlier. Thus, one has to investigate the nature of the outlier before deciding whether to drop it or not, since it could be a legitimate observation. However, it is most likely the case that this particular outlier is a result of human error, meaning it was incorrectly entered so we drop the outlier. Otherwise, it would affect all of our results and assumptions and squich (i.e. flatten) any interesting correlations between the weight feature and reps and sets. Alternatively, one can try a log or square root transformation to pull high numbers down so as to reduce the impact of outliers and their high numeric values. Otherwise, another option is to use models that cope well despite the presence of outliers.

Finally, we noticed that the reps range and set order range was also very high and right skewed. We inspect the data entries that are 1.5 times the interquartile range ($IQR = Q3 - Q1 = q_n(0.75) - q_n(0.25)$) beyond the upper third quartile (75th percentile, Q3). These are the outlier valeus beyond the whisker that denotes the largest observation that falls within 1.5 IQR distance. Surprisingly, we noticed that the female individual did a lot of running sprints where she performed many sets and the high number of repetitions were due to exercises targetted at the abdominal muscles. Thus, we deemed it reasonable to filter out these values that are not related to weightlifting at all.

For the outlier detection and removal, we first used the Z-score but quickly realized that the distribution of values in our sample is not Gaussian (i.e. normal) so we can't use the standard deviation of the sample as a cut-off for identifying outliers. Thus, since the distributions are right-skewed we used the interquartile range and got the following results after having removed the outlier observations that were either incorrect or came from running and ab workout entries that are not relevant to the scope of the project. Ideally, one should do the model experimentation before and after the outlier removal to see how it affects our results and assumptions. We did not do this, since the outliers were mostly incorrect entries or entries that have nothing to do with powerlifting - also, they only constituted less than 3% of all the data and only 0.5% of the workout days were removed in total.

	Set Order	Weight	Reps
mean	2.342183	31.803371	8.361074
std	1.227739	27.275165	3.259652
min	1.000000	0.000000	1.000000
25%	1.000000	10.732686	5.000000
50%	2.000000	23.331926	8.000000
75%	3.000000	50.000000	10.000000
max	6.000000	112.000000	17.000000

Figure 13. Summary statistics of all weightlifting data without outliers

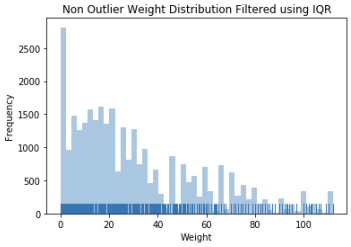


Figure 14. Weight Distribution After Outlier Removal

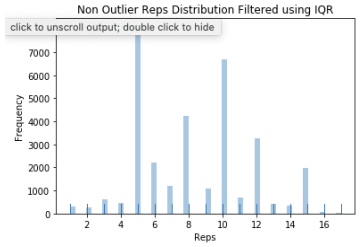


Figure 15. Reps Distribution After Outlier Removal

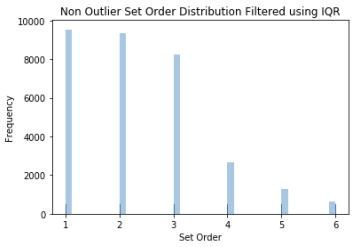


Figure 16. Set Distribution After Outlier Removal

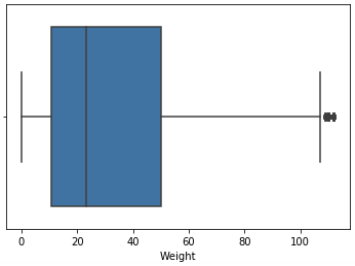


Figure 17. Weight Boxplot After Outlier Removal

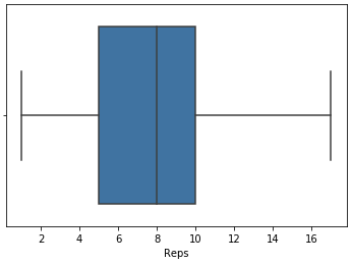


Figure 18. Reps Boxplot After Outlier Removal

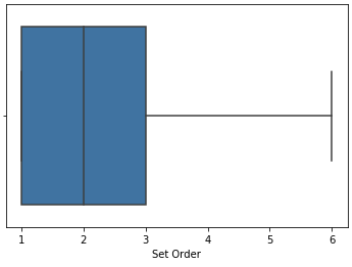


Figure 19. Set Boxplot After Outlier Removal

As can be seen, the weight in kg now ranges between roughly 0 (i.e. body weight exercises) and 112 kg, repetitions range between 1 and 17 and number of sets between 1 and 6. We consulted some of the individuals to ensure that these numbers seemed valid and they all mentioned their 1RM lifts across the three powerlifts, which were all at most around 100 kg. Hence, our previous assumptions seem to somewhat hold and the female individual also approved that the high number of sets and repetitions were due to running and ab workouts. Nonetheless, ideally one should try to do the modelling and run the experiments both with and without these observations to see how they affect results and assumptions and ensue we adhere to the ethics of doing science.

Data Normalization (see 3_model_prediction_and_evaluation.ipynb)

We normalize the three numeric features so their values are between 0 and 1 using a MinMaxScaler from the sklearn preprocessing library. This is done right before the training and test data is fed to the neural network used in the model section.

Data Augmentation (see 1_data_preparation.ipynb)

We augmented the data with a gender and id column that might prove useful to do clustering but realized we don't have enough data. We did not do any data imputation, which could have proven useful when removing outliers by replacing them with average values insteas so as to have a little more data for analysis and experiments.

Data Exploration through visualization: common exercises, feature correlations, trends and seasons (see 2_exploration_data_analysis.ipynb)

We noticed the most commonly performed exercises are big compound exercises where you use multiple muscle groups at the same time such as the back, legs or chest. In particular, the three powerlifts (i.e. squat, bench press and deadlift) show up where we focus on the squat that has the most workout entries:

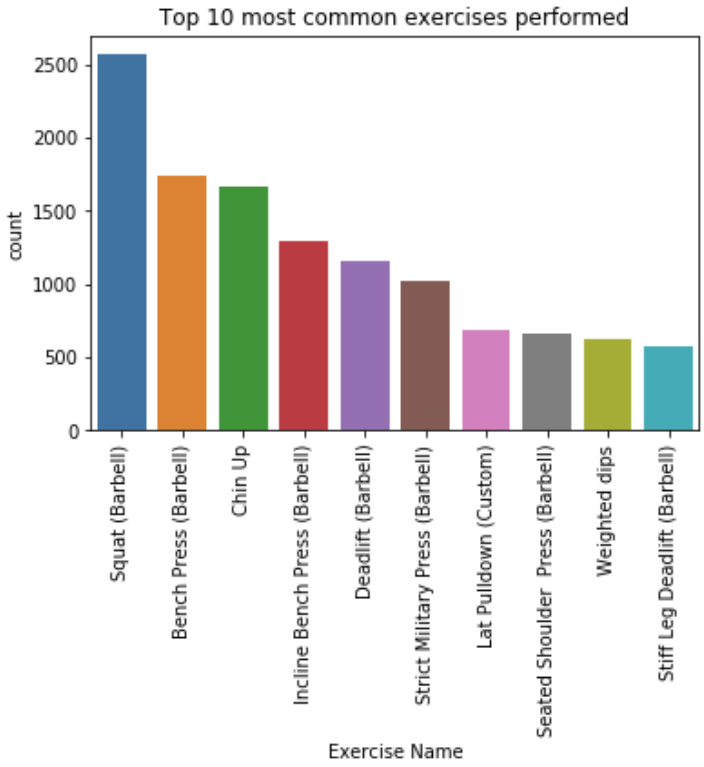


Figure 20. Most common weightlifting exercises

Secondly, we tried to investigate the relationships between the three numeric features (i.e. weight, reps and sets). We noticed that the relationship differs depending on which individual we look at, which may be because they all follow different training protocols, meaning different combinations of sets, reps and weight. For examle, the female did not consistently do weightlifting workouts over time so the correlations and progress became somewhat unreliable (the results are in the Jupyter notebook but have been excluded for the sake of focusing on the individuals that did follow a somewhat consistent weightlifting routine). At first, we used scatter plots to get a rough indication of the correlation between the three numeric features for all three individuals (refer to Jupyter notebook for all three, not all are shown). We found the following interesting correlations for the most observed exercise (i.e. squat):

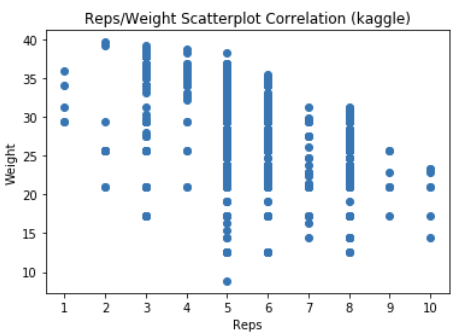


Figure 21. Rep Weight Scatter Kaggle

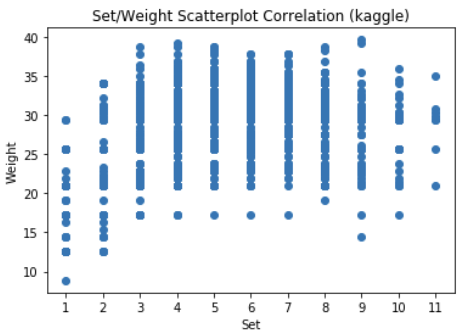


Figure 22. Set Weight Scatter Kaggle

In general, there seems to be a moderately negative correlation between the Reps and Weight variables in which the Weight variable increases positively as the Reps variable decreases negatively. Put simply, the number of repetitions seem to decrease as the weight lifted by the individual increases. Intuitively, this makes sense, since a person should be able to perform less repetitions of an exercise as the amount of work (i.e. the amount he lifts) increases. This was the case of all three individuals, meaning the amount of weight seems to correlate with how many times you can lift the weight in an exercise. Further, one of the individuals seemed to consistently increase the weight as the number of sets increase as shown in the right scatter plot. Again, this most likely makes sense, since people often warm up on the first sets of their exercise and then gradually build up the weight until exhaustion (this is known as pyramid training).

Since there are many entries in the scatter plot that repeat we tried to instead plot the relationship between the average number of reps and the number of sets as shown below. Interestingly, the average number of repetitions performed in the squat as the number of sets increase differs between the individuals. They seem to either build up the number of repetitions as the number of sets increase (as shown on the left) or decrease the number of repetitions as the number of sets increase (as shown on the right). Arguably, this is because they follow a different training protocol where the one on the right maybe increases the weight gradually.

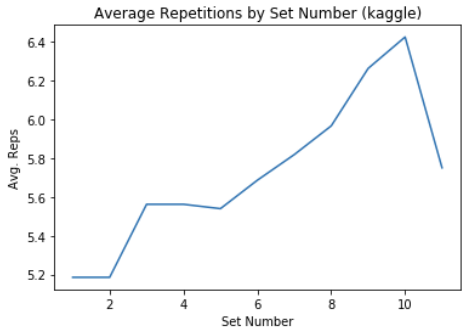


Figure 23. Average reps and Set Scatter Kaggle

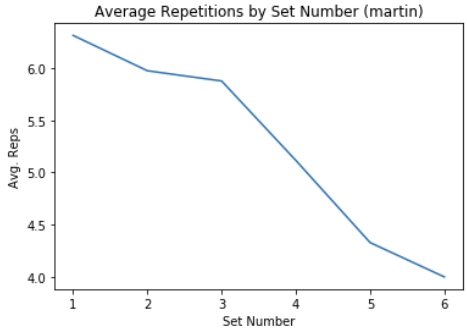


Figure 24. Average reps and Set Scatter Martin

In order to figure this out, we plotted the average weight lifted in the squat as the weight increase as shown below. Interestingly, our hypothesis was not entirely correct, since both individuals seem to increase the weight lifted as the number of sets increase. However, one may notice that the person on the right lifts almost twice as much weight as the one on the left. Hence, it may be the the person on the right is lifting to maximal exhaustion, which explains why the number of repetitions decrease as the weight goes up.

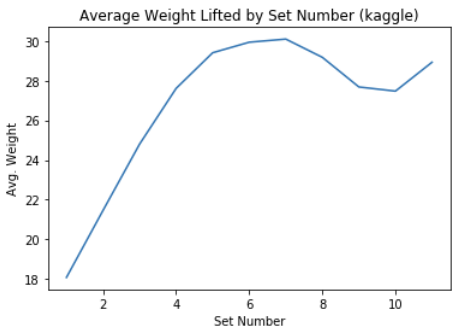


Figure 25. Avg Weight and Set Scatter Kaggle

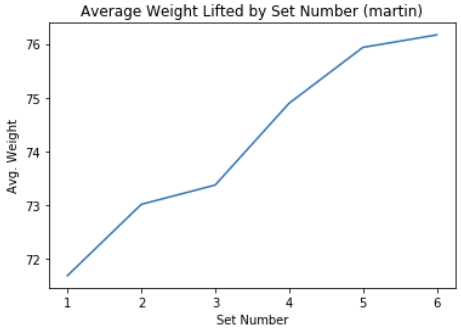
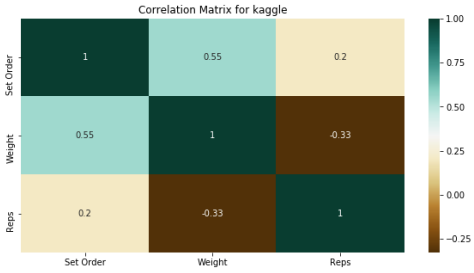


Figure 26. Avg Weight and Set Scatter Martin

The correlations of each individual differs too due to this as shown in the heatmaps below:



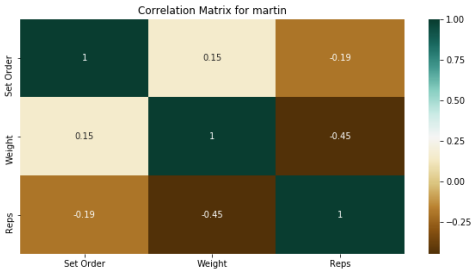


Figure 27. Correlation Heatmap Kaggle **Figure 28.** Correlation Heatmap Martin

As already mentioned, there seems to be a moderately positive relationship between the Weight variable and Set Order variable for the individual on the left side. This means that the weight increases as the number of sets increase. This correlation is not as strong for the individual on the right. Interestingly, they both seem to show a moderately strong correlation between the weight and reps, in which the weight decreases as the number of reps increase. This is further validated by plotting the relationship between the average weight lifted and number of repetitions performed for one of them:

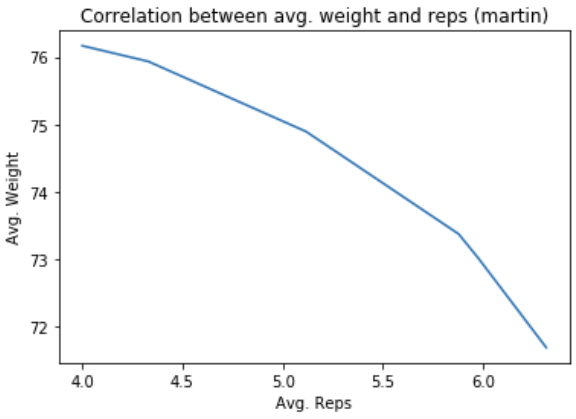
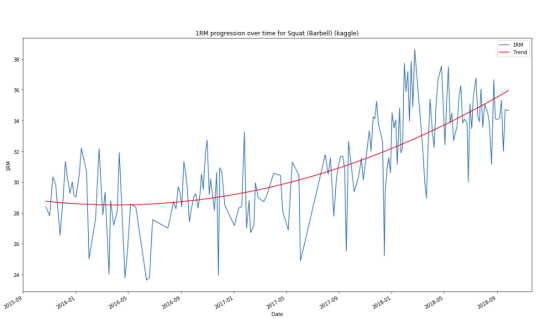


Figure 29. Avg weight and Reps Scatter Martin

Finally, we noticed there is an increasing trend in the strength progress, which we measured by estimating their 1RM (i.e. one-rep-max) for the squat and plotting it as time progresses:



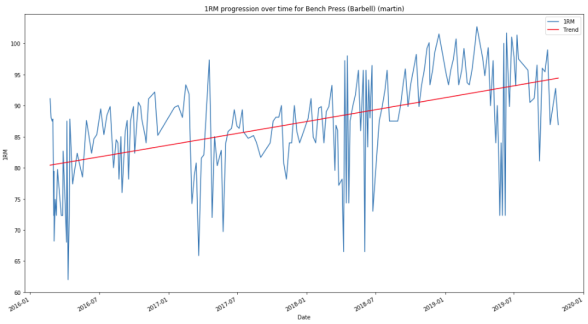


Figure 30. 1RM Squat Progress Kaggle **Figure 31.** 1RM Squat Progress Martin

There also appears to be seasonality in the time series data, since people workout differently depending on the time of the year as shown below. Noticeably, the individuals seem to perform the must volume (i.e. reps and sets) and intensity (i.e. weight) during Summer. On the other hand, this seems to stagnate towards Christmas in December and is at a low in the following months. Obviously, one cannot speculate on the reasons why this happens but there most likely is a shift in the frequency of training depending on the time and seasons of the year. It will be important to control for the increasing trend in strength and seasonal nature of working out throughout the year, since they make it harder to model the progress in strength over time in a stable manner.

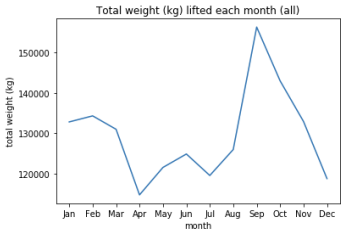


Figure 32. Total weight by month (all)

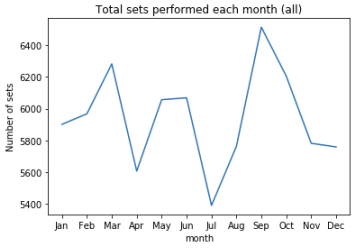


Figure 33. Total sets by month (all)

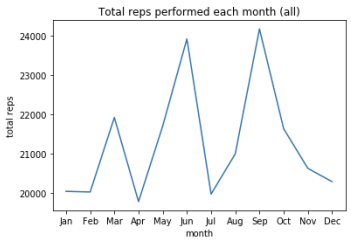


Figure 34. Total reps by month (all)

It should be noted that we also tried to do some clustering based on the individuals and their gender but quickly realized that the findings are somewhat limited, since we only have three individuals, which is not a representative sample of the population in Denmark. However, it would have been interesting to identify individuals based on their gender, training protocol, workout results etc in order to give them meaningful training recommendations based on comparison with other individuals of similar type (i.e. two individuals being closely related in a cluster would most likely benefit from similar workout recommendations). Finally, we used bubble charts to plot the timing of when each individual most often go to the gym during the day but ended up not using the results (see appendix 7.4).

Data Reduction (Feature Extraction)

Feature extraction is a process of dimensionality reduction in which the original data is transformed to a reduced number of variables, which contains the most discriminatory (i.e. saying) information. We do not perform any such data reduction on the raw data, since there are very few feature columns, the data set sizes are small and we don't experience any performance issues. In particular, we only have 1839 unique workout observations spread across three individuals with 583, 657 and 599 workout days respectively. Also, the features we have are all important based on the domain knowledge gathered in the literature study. However, we do perform Principal Component Analysis (PCA) for Dimensionality Reduction on the final prepared data used in our models, since we use a batch of previous workouts to predict strength progress on a current workout (explained later on).

Data Feature Selection

Feature selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested. According to the literature study we made in the background section, the most important variables of strength training seem to be **intensity** (i.e. amount of work required to the activity, which is proportional to the mass of the weights being lifted), **volume** (i.e. number of muscles worked, exercises, sets, and reps during a single workout session) and **frequency** (i.e. how many training sessions are performed per week) (22). Thus, we chose to use weight, set and reps, since there are strong indicators of strength. We also further validated this by the correlations found between the features in the data exploration. Now, we need some feature to represent the training frequency based on the date column feature and some feature that measures strength over time.

4.4 Feature Engineering

Feature engineering is done to create and add new features to the data set in order to add complexity to our models. In our case, we ended up creating three new features from the original data and domain knowledge from the literature study. Firstly, we added a **daysdiff** feature, which is a numeric feature denoting the number of days, since an individual last worked out relative to a particular workout observation. Secondly, we generate 1RM as a metric for strength based on the domain knowledge gathered in the literature study. The way we calculate it is by calculating the 1RM for all sets in the feature **1rm_set** and taking the maximum (i.e. best 1RM) in the feature **max_1rm**, which was usually the heaviest set of a workout exercise across all workouts. This ensured that the 1RM estimate was based on a working set where the individual is maximally exhausted and simultaneously kept the repetitions below 10 by inspection. This was important, since the literature study revealed that the formula we use to estimate 1RM is unstable if we use a combination of weight and repetitions beyond 10.

Differencing (see 3_model_prediction_and_evaluation) for delta$_{1rm}$

Finally, we use differencing to cope with the trend and seasonality inherent in the data as shown in the 1RM progress over time for the squat in the data exploration section. Differencing is a method of transforming time series data so that it eliminates (or reduces) trend and seasonality, which helped make our time series stationary. In other words, differencing helps remove so-called temporal dependency and stabilize the mean of the time series by removing changes in the level of the time series. Differencing is performed by subtracting the previous observation from the current observation, which is done for the 1RM estimate to find the delta (difference) from day to day in the feature **delta_1rm**:

$$\text{difference}(t) = \text{observation}(t) - \text{observation}(t\text{-}1)$$

This results in the following data set:

	Date	Exercise Name	Set Order	Weight	Reps	Gender	Id	1rm_set	max_1rm	days_diff	delta_1rm
22	2015-10-23 22:01:34	Squat (Barbell)	1	61.235598	8	Male	Kaggle	77.565091	180.985213	0	0.0
23	2015-10-23 22:01:34	Squat (Barbell)	2	83.915450	5	Male	Kaggle	97.901358	180.985213	0	0.0
24	2015-10-23 22:01:34	Squat (Barbell)	3	102.059330	7	Male	Kaggle	125.873174	180.985213	0	0.0
25	2015-10-23 22:01:34	Squat (Barbell)	4	124.739182	5	Male	Kaggle	145.529045	180.985213	0	0.0
26	2015-10-23 22:01:34	Squat (Barbell)	5	142.883063	5	Male	Kaggle	166.696906	180.985213	0	0.0

Figure 35. Raw Strong Weightlifting Workout Data

Finally, flattening of the workout sets is performed, since individuals perform multiple sets of each exercise with different weights and repetitions. These workout entries are collapsed into a single workout entry containing all repetitions and weights performed across all sets for a particular exercise. This enables us to batch the data more easily when using a number of workouts in the past to predict the expected strength progress (delta 1rm) on a workout today or a number of days into the future. The delta 1RM feature represents the strength growth difference between today's workout and the workout yesterday. It is the percentage difference between the 1RM I may expect to do today given the workout I plan to do and the 1RM I got yesterday. It becomes the continuous target of our regression problem in our model and the time series of the delta 1RM is stationary unlike the raw 1RM time series that has a positive trend and seasonality so it is easier to model in timeseries forecasting.

	Date	delta_1rm	max_1rm	days_diff	reps_1	weight_1	reps_2	weight_2	reps_3	weight_3	reps_4	weight_4	reps_5	weight_5
0	2015-10-23 22:01:34	0.000000	180.985213	0	8	61.235598	5	83.91545	7	102.059330	5	124.739182	5	142.883063
1	2015-11-01 11:23:25	-0.065574	169.494088	8	8	61.235598	8	83.91545	8	102.059330	8	102.059330	8	111.131271
2	2015-11-08 17:45:26	0.000000	169.494088	7	8	83.915450	8	102.05933	8	124.739182	8	124.739182	8	129.275152

Figure 36. Flattened by sets in Weightlifting Workout Data

4.5 Timeseries Forecasting

Time series forecasting is the use of a model to predict future value(s) of time series based on its current and past values. Interestingly, we realized that we need to use models that work well for time series due to the inherent time component in our data set. We have no experience with time series forecasting but quickly noticed that we could not directly use the tools learned in Machine Learning. For example we tried to do cross validation and split the workout data into training and test data. However, this is problematic, since it only makes sense to use workouts observations in the past to predict strength growth in workout observations of the future.

Also, the strength of a person depends highly on the frequency of their training. Namely, the literature study revealed that training groups that are taken off training quickly diminish their strength. It is also stated in many of the literature papers that a somewhat frequent training routine is required to optimize for strength growth as well as muscle hypertrophy. Thus, the way and frequency in which you workout highly affects your performance and growth in the future. Especially, one might be inclined to think that the most recent workouts before the day of prediction have the highest influence on the strength growth prediction. Intuitively, the condition or strength of a person way back in the past most likely does not say much about the strength of that person today.

To sum up, we decided to represent our research question as a regression problem solved using time series forecasting. In particular, we want to show how one can optimize workouts in a particular domain. For this purpose, we agreed that we need some measure of strength growth in powerlifting, which is the 1RM metric. Now, to model the progression in strength growth over time, we use multivariate time series regression models to maintain the interdependence between the workout variables. Most statistical forecasting methods are based on the assumption that the time series is somewhat stationary, since these are relatively easy to predict. This is because its statistical properties remain the same in the future as they have been in the past. We did this using differencing, which models how the time series of the delta 1RM changes from one day to the next. The time series is shown below and one can observe that the increasing trend is gone, since the level of the time series remain around 0 and the seasonality is gone.

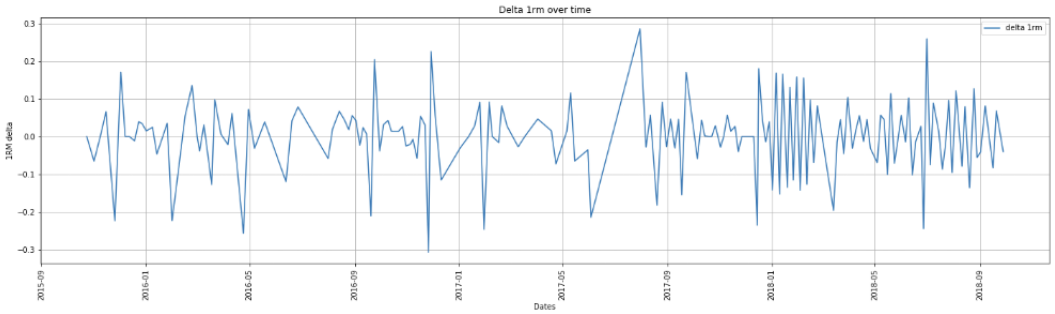


Figure 37. Stationary 1RM Delta Time Series

4.6 Choice of problem and suitable models

Ideally, we would like to predict the combination of sets, repetitions and weight that yields the highest delta 1RM. In other words, finding the workout routine that maximizes the strength growth over time. However, given the time and scope constraints of the project, we have focused our attention to figuring out whether it is even possible to measure and predict strength growth over time. In order to use our knowledge in Machine Learning, we rephrase the time series as a supervised learning problem. We did so by using previous time steps as input variables and use the next time step as the output variable. In other words, we use the delta 1rm value and the input variables at the previous time step(s) to predict the delta 1rm value at the next time-step (sliding window, (32)).

In terms of models, we needed to find a multivariate time-series model that can predict strength growth over time measured by the continuous target variable, delta 1rm target, given the multivariate set of features from feature engineering. For this purpose, we tried to compare a traditional statistical approach such as VAR (vector auto-regressive) models with recent Machine Learning advances in modelling sequences using LSTMs (long shot term memory models). For the neural network, we use the sliding window method where we use prior time steps (i.e. previous workout input and delta 1rm targets) to predict the next time step (i.e. future delta 1rm target).

4.7 Auto Regression

Autoregression is a time series model that uses observations from previous time steps as input to a regression equation to predict the value at the next time step (33). This means that we predict the continuous target value of delta 1rm based on its past value(s), since this may help yield stronger predictions of strength growth if there is in fact a pattern in the relationship between workouts over time, meaning workouts do now simply change randomly over time like in a stochastic process. Our target prediction can be represented mathematically as follows where $\hat{y}$ is our 1rm delta prediction, H is the forecast horizon (i.e. number of steps into the future we predict), p is the lag (i.e. number of days we look back in the past), $y_t...y_1$ are previous 1 rm delta values and X denotes our workout input variables. In this regard, y is called the **endogenous** (i.e. dependent) variable (DV) and X is called the **exogenous** (i.e. independent) variable. This is because we try to model how y (i.e. delta 1rm) changes based on its relationship (i.e. correlation) with previous target values, $y_t...y_1$, and workout input variables, $X_t...X_1$, in the model. The current information denoted Ω_t is then the set of all target and input variables (i.e. y's and X's) to our statistical model or Machine Learning algorithm, f.

$$\hat{y}_{t+H}|t = \text{f(current information)} = f(y_t, y_{t-1}, y_{t-2}, ..., y_{t-p}, X_t, X_{t-1}, X_{t-2}, ..., X_{t-p})$$

The simplest and most naive model would be to forecast tomorrow to be the value observed today:

$$\hat{y}_{t+1} = y_t, \text{ where } H = 1 \text{ and } \Omega_t = \{y_t\} :$$

A univariate time series models only one variable varying over time, which corresponds to only looking at the evolution of delta 1rm strength growth alone:

$$y_{t+1} \leftarrow \{y_t, y_{t-1}, ..., y_{t-p}, X_t, X_{t-1}, ..., X_{t-p}\}$$

The Auto Regressive (AR) and the extended Auto Regressive Integrated Moving Average (ARIMA) model both regress the evolving variable (i.e. y_t or delta 1rm at time step t) on its own lagged (i.e. prior) values (hence the AR part). Arima also specifies that the output variable depends linearly on the current and past values of lagged (i.e. past) errors generated from of a stochastic (i.e. random) process (hence the MA part). Finally, the Integrated (I) part represents the differencing of faw observations to make the time series stationary in ARIMA. They do this using a linear functions of its own timeseries values and error terms:

$$\hat{y} = \mu + \phi_1 y_{t-1} + ... + \phi_p y_{t-p} - \theta_1 e_{t-1} ... - \theta_q e_{t-q}$$

, where μ is a constant, ϕ's are the coefficients of the lagged values of y (AR) and θ's are the coefficients of the lagged errors denoted e (MA).

4.8 Vector Auto Regression (VAR) model

Vector Auto Regression is a generalization of the AR model to multivariate time series, which means it models the interaction between several variable time series. In the real world, it is very likely that the value of one variable in the past affects the value of another variable in the current or future time period. For example, lifting a heavy weight yesterday may negatively influence the weight and reps I perform today, which means the interaction of workout factors may affect strength growth. As such, the output variable (i.e. y or delta 1rm) is not only affected by its own development over time but also by the development of the other input variables of the workout data. Even further, the interaction between the workout input variables seem to mutually affect their development over time as indicated by the data exploration and feature correlations. Thus, we use the Vector Auto Regression to model the delta 1rm strength growth output variable as a time series affected not only by its own past but also by the past of all other variables and their mutual interaction. This allos us to model the dynamics of each of the variable time series and their interdependence.

Vector Auto Regression generalizes the previously shown linear equation to a system of K linear equations where K corresponds to the total number of series. The VAR model assumes that the variables are stationary, meaning their mean and variance do not change over time and the covariance matrix only depends on the time lag (not time itself). One fatal mistake we initially made was assuming that only the output variable time series has to be stationary. Hence, we forgot to check whether the time series of the workout input variables also were stationary, which ultimately may lead to poor results. In the future, one should always remember to check that all variable time series are stationary and otherwise transform the non-stationary series using simple differencing a

number of times until stationary. Differencing (i.e. $y_t - y_{t-p}$) will help stabilise the mean of a time series by removing changes in the level of the time series, thus removing (or reducing) trend and seasonality). Further, one can use the log transformation to help stabilise the variance of a time series:

$$log(y_t) - log(y_{t-p}) = log(\frac{y_t}{y_{t-p}}), \text{ where p is some lag, i.e. number of days back in the past}$$

One needs to remember to inverse transform the predictions in the forecasted series back to the original series:

$$\text{Let } z = log(y_t) - log(y_t - p) = log(\frac{y_t}{y_{t-p}})$$

$$\text{Then, } log(y_t) = z + log(y_t - p)$$

$$y_t = e^{z+log(y_{t-p}} = e^z(y_{t-p})$$

Thus, the forecast is,

$$y_{T+s} = e^z(y_{(t-p)+s}) \text{ where s is the forecast horizon and } s \geq 1$$

The logarithmic transformation helps stabilise the variance of a time series. Differencing help stabilise the mean of a time series by removing changes in the level of a time series, thus eliminating (or reducing) trend and seasonality. The final VAR(p) model describes the evolution of a set of k variables (i.e. endogeneous) variables over a period (t=1, ..., T) as a linear combination of their p lagged (i.e. past) values:

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + e_t$$

, where the observation y_{t-i} (i periods back) is called the i-th lag of y, c is a k-vector of contants (intercepts) and A_i is a $k \times k$ coefficient matrix and e_t is a k-vector of error terms where $E[e_t] = 0$ (i.e. every error term has zero mean), error covariance matrix is positive-semidefinite and there is no correlation between errors across time (34).

Thus, this kind of multivariate time series has more than one time-dependent variable. Each variable depends not only on its past values but also has some dependency on other variables, which is used to make stronger predictions of the future. In our case, the delta 1rm is our output variable that is a linear combination over all features **weight, reps, max_1rm, days_diff, delta_1rm,** as shown in appendix (7). Notice that we make a system of linear equations where there are $K = 26$ equations, since there are at most 11 sets for each pair of weights and reps (22 in total), 3 other features (delta_1rm, max_1rm, days_diff) and 1 constant.

4.9 Reccurent neural networks (RNN)

"Recurrent neural networks, or RNNs, are a family of neural networks for processing sequential data" (3). A recurrent neural network is specialized for processing a sequence of values similar to workouts over time.

What sets aside RNNs from multilayer networks is the need for sharing parameters across different parts of the model. A traditional fully connected feedforward network has separate parameters for each input feature and it does not account for any time-ordering inherent in the data. Inputs are processed independently without any interaction and there is no state to keep the past information. An RNN can retain past information and track and update the state of the world as the network moves forward. It handles variable-length sequential data by having a recurrent hidden state whose activation at each time is dependent on the activation of the previous time as shown next.

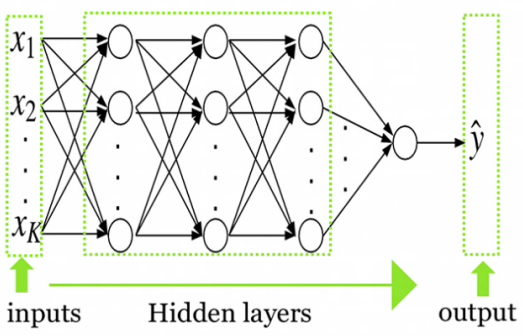


Figure 38. Feed-Forward Network with Single Output by Jeffrey Yau (6)

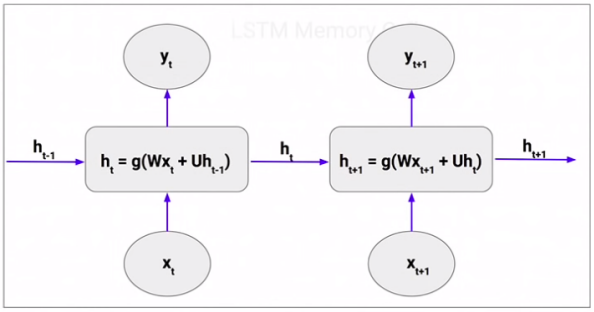


Figure 39. Standard Recurrent Neural Network (RNN) by Jeffrey Yau (6)

As can be seen, the hidden activation (tanh), h_t in an RNN takes not only the current input (y_t, x_t) but also the previous hidden activation, h_{t-1}, as input. Hence, this enables a loopback to the past where the RNN is not only a function of new input but also of the past hidden layer. As can be seen on the right figure, the RNN keeps rolling forward over time, which is why it can handle variable-length sequences unlike the feed-forward network on the left. The problem with RNNs is **the vanishing gradient problem** (or exploding). This problem occurs in neural networks when training using gradient-desccent based method and backpropagation. Recall that the weights are updated proportional to the partial derivative of the error loss function with respect to the current weight in each training iteration. For deep neural networks, we end up multiplying partial derivatives many times. This leads to infinitely small or large partial derivatives, which are used to update the weights. For small partial derivatives, the weight updates become insignificant, rendering those weights useless, since they don't really change.

4.10 Long-Term-Short Term (LSTM) model

Recall that we use non-linear activation functions such as the sigmoid function ($\sigma(x) = \frac{1}{1+e^{-x}}$) in neural networks or the tanh function in recurrent networks to capture non-linear patterns inherent in the data. These common activation functions squeeze the input into a very small output range (e.g. sigmoid maps to [0,1] and tanh to [-1,1]). Consequently, a large region of the input space is mapped to an extremely small range so the gradients become small. This is worsened by the fact that we stack multiple layers of non-linearities. To avoid this, we can use activation functions that do not squash the input. For example, the Rectified Linear Unit (i.e. ReLU) maps x to max(0,x). However, the ReLU is subject to the **dying ReLU** problem where a neuron that gets negative is unlikely to recover. *"LSTMs solve the problem by creating a connection between the forget gate activations and the gradients computation. This connection creates a path for information flow through the foret gate for information the LSTM should not forget"* (36). The LSTM is governed by 6 equations; the hidden state $h(t)$, memory cell $c(t)$, candidate memory cell $\tilde{c(t)}$, output gate $g_O(t)$, input gate $g_I(t)$ and forget gate $g_F(t)$.

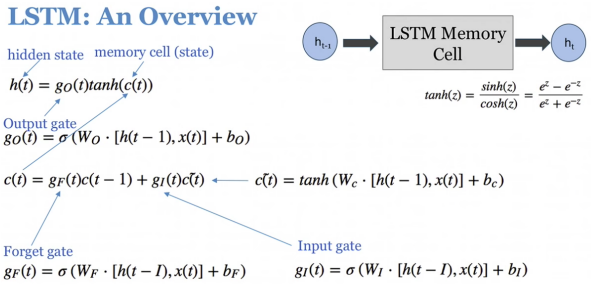


Figure 40. LSTM Memory Cell and equations by Jeffrey Yau (6)

The output gate $g_O(t)$ is a sigmoid used to control past information $h(t-1)$ where an activation of 0 represents no past information flow and 1 represents all past information flow: $g_O(t) = (W_O[h(t-1), x(t) + b_0)$. The forget gate g_F controls how much to forget and the input gate g_I controls how much information to let in. The LSTM is trained using Backpropagation through time (BPTT), which is BP on a network that unfolds over time steps.

5 MODEL EVALUATION

In this section, we will evaluate and compare how well our LSTM and VAR model predict strength growth over time. We will look into the experiments conducted, the results, and how we evaluated the models. In addition, a comparison will be made of the models where we try to derive, which of the two models performed best in predicting and forecasting the delta growth of 1RM.

5.1 How do we evaluate our models?

Before any experiments can be done, it is essential to understand how the models can be correctly evaluated. This is because, when any experiments are performed, the correct evaluation metrics and procedures are required to produce accurate and correct results for measuring the performance of the models, and how well they generalize.

Problem type

The problem type in a machine learning project is one of the crucial factors when selecting evaluation metrics. Recall from the model design that we chose to model the problem as a supervised regression problem. In supervised learning, the task is to create a model by learning a function given training data that consists of pairs of input vectors and correct output targets. The supervised learning algorithm will use the training data to search for patterns in the data that correlates with the output. When the learning algorithm is done, the model is then used to predict new outputs given new unseen input data.(4)

We can either solve classification or regression problems in supervised learning. In our case, we are trying to predict a continuous regression value, namely the 1rm delta growth. Hence, we cannot use a confusion matrix or accuracy metric to evaluate our regression model, since these are based on discrete class labels and not continuous target values. Instead, we want to measure how much the predicted output values differ from the actual outputs using the mean squared error (MSE), root mean squared error (RMSE) or mean absolute error (MAE).

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where N is the number of observations, $\hat{y}$ is the predicted output and y is the actual output.

MSE measures the average squared errors of the predictions by calculating the squared difference between each prediction and the expected target values and then taking the average of the squared errors. The result of the MSE equation is non-negative due to the squared difference between the predicted and actual outputs. The squaring is used to prevent positive and negative values canceling each other out while still taking negative and postive values into account. The quality of the model can then be derived by how close the MSE value is to zero. A high MSE value corresponds to the predictions being very far from the target label, while a low MSE value corresponds to the predictions being very close to the target labels. The MSE is useful when the data does not contain large outlier values, since these magnify the errors.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} = \sqrt{MSE}$$

where N is the number of observations, $\hat{y}$ is the predicted output and y is the actual output.

RMSE shares many properties with MSE, since RMSE is just the square root MSE. It is also sensitive to outliers but the square root ensures that the errors are scaled back so that it can be interpreted in measurement units as a better measure of goodness of fit.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

where N is the number of observations, $\hat{y}$ is the predicted output and y is the actual output.

MAE also calculates the average of the difference between the predicted and target values but takes the absolute of the difference. Thus, MAE is less sensitive to outliers in the data unlike MSE and RMSE. However, if we are interested in the larger variations of the data, then MAE may not be as effective in capturing these variations. After experimentation and hyper parameter tuning, we chose to use MSE but also use the RMSE to better interpret our model results. The MSE produced the best predictions on the test data (see figure 56 and figure 57).

Timeseries Constraints - Train and Test set

To evaluate the performance of our supervised model, we need to test it on unseen data to assess whether our model is overfitted, underfitted, or well generalized. In machine learning, one may usually apply Cross-Validation, which is a technique that can be used for assessing the effectiveness of our model by splitting a portion of the data into test and training subsets. The tests set corresponds to unseen data, and the training set is used when training the model. There are several cross-validation approaches that one may in general use such as *Train/Test split approach* where we randomly split the data into train and test and *k-fold cross validation* where one splits the data randomly into k folds where the model is trained on the k-1 folds and tested on the kth fold.(29).

In this regard, we used *Second Half Prediction* (*figure 41*) method for the Vector Auto regression, where we make one split in the data such that we have a subset of training data that represents current information and a subset of test data that represents the future for prediction - each chronologically correct and respecting the time ordering. Thus, the training data is then used to train the VAR model and the performance of the VAR model is estimated by letting the model predict (i.e. forecast) the values in the test set.

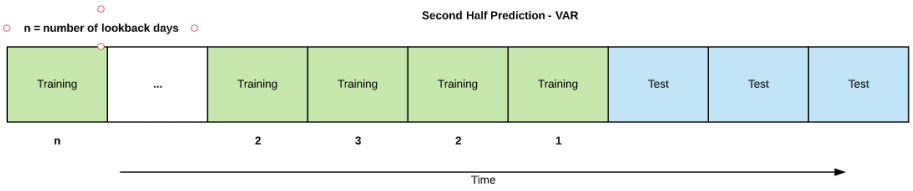


Figure 41. Second half prediction used in VAR models for training and testing

The test set is always the same size and always contains the same latest observations. What we are adjusting is the ***n*** number of days prior to the earliest test observation we wish to use for training (previously the lag p). This allows us to test with a different number of days used for training and then compare how well each VAR model performs based on the number of days used for training before testing. Arguably, due to time constraints, we did not try to use different sections in the data as test set and only used the very latest observations. Thus, our results may be biased towards the latest test observations (in time) in regards to the optimal number of days used in training set. We could include experiments with different subsections of the data as test set where we try different n days for training and then assess how the number of days affects the overall performance of VAR.

For the LSTM we tried to use the *Second Half Prediction* method but since the LSTM uses batches of data to predict the next value, we were unable to apply the same method for LSTM as VAR that takes all the data.

Figure 42. Second half prediction used in LSTM models for training and testing

Instead we still split the data into training and test set similar to *Second Half Prediction* but during training, the LSTM uses mini-batches by batching the training data into t days and predicts the next day in a validation set (see figur 42). Once training is done, the same idea is then applied for testing. That is, the LSTM uses mini-batches of test data to predict the next test observation. Besides the hyperparameters we can adjust, we are interested in assessing the number of t past days included in the mini-batches and how well it affects the performance of the LSTM. Again, similar to the approach with VAR, we only use the same test data and hence our results may be biased towards the latest test observations in time when adjusting the optimal number of days back in the past used in training set.

5.2 Experiments setup

A variety of experiments have been conducted where the main goal is to assess how well the LSTM and VAR models were able to generalize and predict the delta 1rm growth. In this regard, we created three modified data sets based on the workout data with the main focus on the Barbell Squat exercise and its delta 1rm growth.

Heavy Sets Only

The first type of dataset is where we only use the heaviest set during a workout session where any other sets are excluded. The primary features are Weight, Reps, daysdiff, max1RM. The main purpose of this data set is to investigate if the models can predict the delta 1rm growth by only using the heaviest lifts during workout sessions. The hypothesis here is that the heaviest lifts might also have the biggest impact on the growth such that the other sets are irrelevant. On the other hand, it excludes the progression of the workout volume intensity, which may have a considerable impact on the delta 1rm growth.

All Sets Flatten

The second type of dataset is where we use all sets of a workout. All the sets are flattened such that the input vector contains all reps, weights, max1rm, and days diff performed for a given exercise in one row each. The main purpose of this data set is to investigate if the models can predict the delta 1rm growth based on all sets during a workout session. This data set retains all progression during a workout session, which may allow the models to capture the strengt growth of a person through the delta 1rm metric. However, this data contains many more dimensions, since the number of reps and weight dimensions grow proportional to the number of sets and number of days we look back. In Machine Learning, we are concerned with this due to the **curse of dimensionality**, which roughly states that as the dimensionality increases, the volume of the input space increases so fast that the available data becomes increasingly sparse.

PCA reduced All sets flatten

The third and final type of dataset tries to accommodate the potential dimensionality problem of the All Sets flatten data set. We reduce the dimensions with PCA to a set of principal components (PC). Here, we experiment with a different number of PCs and see how well the models can predict the delta 1RM growth given that how we truncate the feature space including the progression from the workout sets. Before the PCs were selected, we calculated the Cumulative variance explained for each PCs. The cumulative variance explains how much of the variability is retained from the original dimensions given that we select a k number of PCs).

For example, looking at figure 43 on the next page, if we use 8 PCs, then these 8 PCs explain 90 percent of the variability, which means that we lose 10 percent of the information explained by the original data. Thus for the experiments, we tried to use 5, 10, 15, 17, and 20 PCs to see how reducing the dimensions affect the models' ability to predict delta 1rm given that PCA compresses the information. Also look at figure 58 in the appendix which shows Cumulative Explained Variance proportional with the number of PCs selected.

PCA Explained Variance Ratio	
Number of PCs	Preseved variance %
1	0.3507
2	0.5823
3	0.7003
4	0.7606
5	0.8124
6	0.8548
7	0.8804
8	0.9019
9	0.9198
10	0.9331
11	0.9443
12	0.9538
13	0.9625
14	0.9691
15	0.9751
16	0.9807
17	0.9855
18	0.9894
19	0.9925
20	0.9951
21	0.9969
22	0.9986
23	0.9996
24	1

Figure 43. Cumulative Explained Variance of kaggle all set flatten dataset.

5.3 Vector Auto Regression Training and Evaluation

Prior to conducting any experiements on vector auto regression models, we set the test set to be a fixed constant - specifically, the test set contains 20 observations which we use to measure how far the VAR model can forecast. What we then adjust is the percentage of look back days (i.e. lag p) we use in the training set. In this regard we experimented with a training size of 10, 20, 25, 50 and 75 percent.

Heavy Sets Only

By observing the results, using 10 percent as training size, the VAR model was not able to predict anything substantial. The predictions fluctuated a lot and did not align with the labels. This is clearly due to the lack of training data and most likely due to the lack of variance or progress inherent to the original workout sets as we only considered the heaviest set. However, as we increased the training size, the early forecasts started to align with the actual test delta 1rm, and the sweet spot seemed to be with a training size of 50 percent of the data (figure 44). . However, we noticed that the model was unable to forecast further than three days ahead as the predictions flatten out. This may very likely be due to anything beyond four days becoming too uncertain for the model to forecast (i.e. predicting the future). See figure 59 for close up results and figure 60 for overview results.

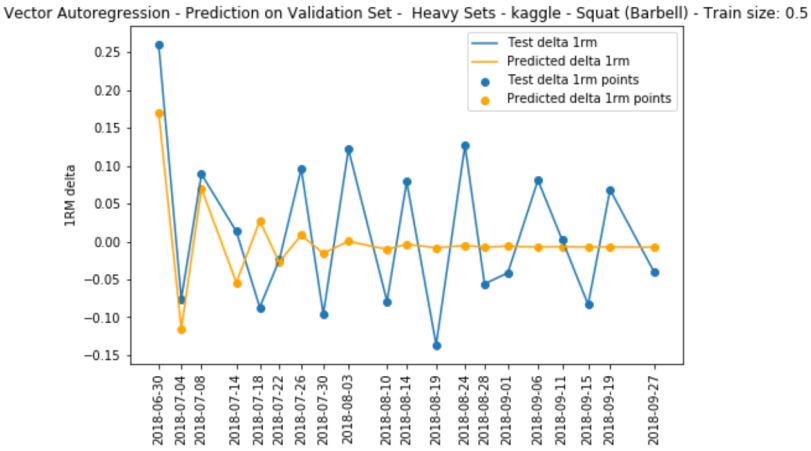


Figure 44. VAR predictions on Heavy sets data set with train size of 50 percent - Test Results

All Sets Flatten

By observing the results using 10 percent as training size, the model did capture the gist of the earlier forecasts of 5 days. As we increase the size of the training set, the model was able to forecast the first five days relatively good when we used a training size of 20 percent (figure 45) compared to the VAR model, which only used heavy set. However, as we increased the size of training set, the predictions started to flatten out. This might be due to the large number of days we look back, and the earlier days may not be current enough such that they become obsolete and hence affects the forecasts. In other words, my strength in the past becomes increasingly irrelevant to my strength today as I go back in time.

Another pattern that we noticed is that the first two delta rm test observations deviated quite a lot compared to the others. We were unable to closely predict these two target observations in our experiments. One reason for this may be due to their large deviations, and therefore, the model may not be fully able to model them. See figure 61 for close up results and figure 62 sets overview for overview results.

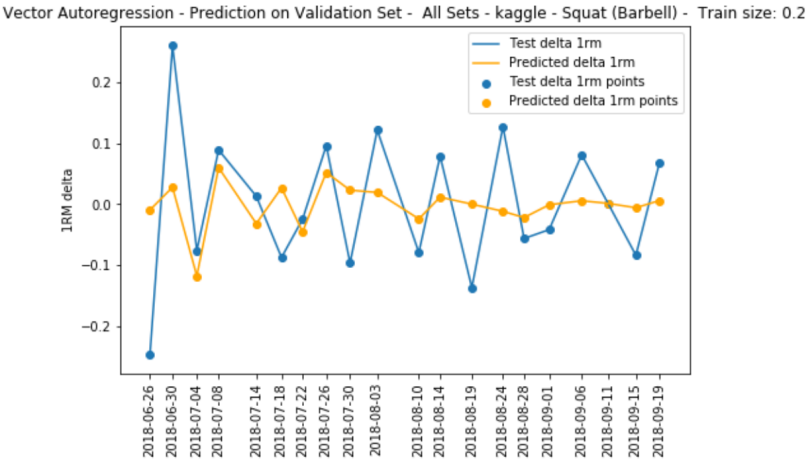


Figure 45. VAR predictions on all sets data set with train size of 20 percent - Test Results

PCA reduced All sets flatten

One may observe that using 5 PCs components, the VAR model was unable to predict the labels closely despite increasing the training set size. This makes sense as we only use 5 PCs, which explains 81% of the variability. Thus we lose 19% of the information from the original dimension space and hence the poor results. As we increase the number of PCs, the predictions fit better to the test delta 1rm. This makes sense since we then also increase the Cumulative Explained Variance and hence increase the amount of information we retain from the original dimension space. Hence, a VAR model that was trained on a data set with a high number of PCs was better able to predict delta 1rm than a VAR model trained with a low number of PCs. The observed sweet spot was a VAR model trained on 17 PCs with a training set size of 25 percent (figure 46). Finally, we got extremely incorrect predictions when we used a training set size of 10 % using 15, 17, and 20 PCs. This is most likely because the training size is very small and the resulting model is underfitted, which results in very poor predictions. See figures from 64 - 68 for close up results and figures from 69-73 overview for overview results.

5.4 Best VAR Model

Given all the experiments conducted for the Vector Autoregression models, this section will wrap up the results. We tried to evaluate the models by calculating the RMSE for each model as well as looking at the actual predictions on the graphs to see if the calculated RMSE reflected the VAR models' ability to predict delta 1RM.

Based on figure 74, which shows the RMSE for all models generated from the experiments, we can see the worst-performing models are the under fitted PCA models with a training size of 10% and PCs 15, 17 and 20. The highest RMSE calculated was 35622, which is an absurdly high number, but if we look at the predictions from figures 66 - 68, this does indeed make sense as the predictions are very extreme. Arguably, this is because we try to look far into the future - 20 days with a minimal training set.

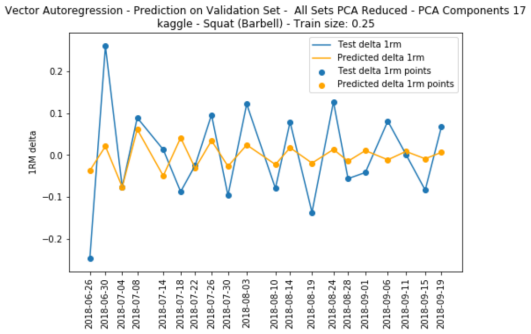


Figure 46. VAR predictions on PCA set with 17 PCs and train size of 25 percent - Test Results

Scoping future horizon down to 3 days (figure 63) yielded an RMSE of 1.5962, which, compared to the other models, is still very poor. On the other hand, the best VAR model was produced for the heavy sets dataset with a training size of 50%. The RMSE was 0.0588, which is the lowest RMSE calculated for all models. By observing the test results on figure 59 we see that the first 3 predictions do closely align with the test labels. However, the predictions did oscillate a lot in the initial experiments with a small training set. Thus, increasing the training size just seemed to damper the oscillating range, which resulted in the predictions aligning closer to the actual delta 1rm. Maybe the model did capture the delta growth or it was just because of the oscillation being reduced.

Nonetheless, by observing how the VAR models performed with respect to the All Sets data sets and PCA data sets, then the All sets models performed better than the models that used the PCA sets. The reason is most likely due to PCA truncating information and therefore not being able to fully capture the delta 1rm growth compared to the models that used all the dimensions. Finally, looking at figure , which showcases the top-performing VAR models for each respective datasets, then we have the heavy sets models on top, but the All sets model with a training size of 75%, are in 2nd place with an RMSE differenc 0.009673 RMSE.

Top VAR

Models	Test Size	RMSE
Heavy Sets	0.5	0.058786
All Sets	0.75	0.068459
PCA - 17	0.5	0.102519
PCA - 15	0.5	0.103383
PCA - 20	0.25	0.103901
PCA - 10	0.25	0.106535
PCA - 5	0.5	0.108018

Figure 47. Top VAR models based on RMSE

The results are based on the RMSE but looking at the predictions graphs of all models, the All Sets model seems to be the best performing model with a training size of 20 percent. Within the same category, the All Sets model with a 75% training size t is the worst model as the predictions do not align with the test targets (see figure 48).

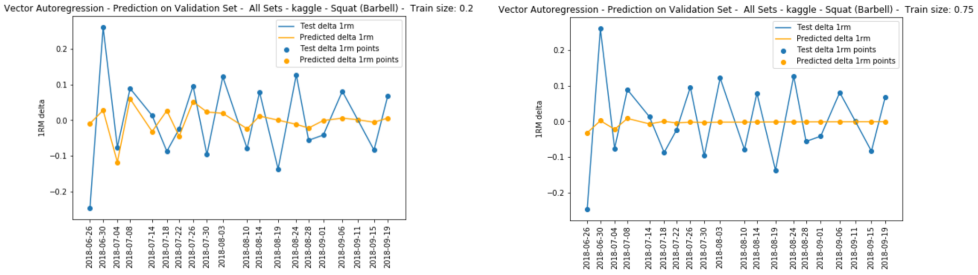


Figure 48. VAR training size 20 percent vs. 75 percent

5.5 LSTM Training and Evaluation

Prior to conducting any experiments on LSTM models, we split the data into two 80% training and 20% test sets. The test contained 35 observations out of 172 workout observations. Both sets were chronologically correct (i.e. respect time ordering). During the experiments, we tested the LSTM models with three mini-batch sizes corresponding to the number of days we look back in order to predict the next day. In this regard, we tested with 1, 3, and 5 days. However, due to time constraints, we did not perform this during the PCA tests as we only tested the different combinations of principal components. Finally, it must be stated that due to the nature of the mini-batches, we do not predict the first n test observations as they would be part of the first mini-batch. For example, assuming n is 3, then the first three observations are not predicted as the LSTM needs a mini-batch of 3 observations. Therefore, the test will only predict test observation 4 and up. Arguably this means that we are not predicting all the test observations, which makes it more difficult to compare the LSTM models. Finally, due to how our LSTM slides a window from observation to observation, then our LSTM only forecasts the next days delta 1rm and not multiple days ahead as we tried in our VAR model. Hence we must consider this factor when we later compare the LSTM and VAR models.

Another critical factor was how we selected the models based on the loss development through the epochs. We strived to minimize the loss function for our tests and only selected the models which loss was still below the training loss. Otherwise, if we selected a model whose test loss was higher than the training loss, that would imply that the model was starting to overfit to the training data and lose the ability to generalize well to new unseen test data.

Heavy Sets Only

For the heavy sets data set experiments, we were unable to train an LSTM that was able to predict delta 1rm despite adjusting the mini-batch size and the other hyper parameters. All experiments yielded the same poor results where the delta 1rm predictions more or less linear at around delta 1rm 0.01 on the test set (see figure 78). However, by observing the loss over time, we can see that training loss is decreasing while the test loss stays mostly linear, roughly around 0.0100, with a small initial decrease (see figure 77]). We first assumed that it was related to our code, but since we used the same LSTM code for the other experiments that yielded much better results, then it was not clear why the LSTM performed poorly on the heavy sets data set. Our primary assumptions are either due to the small dimension space of the heaviest data set or due to the missing set progression from the original data that makes the LSTM unable to predict the delta 1rm growth.

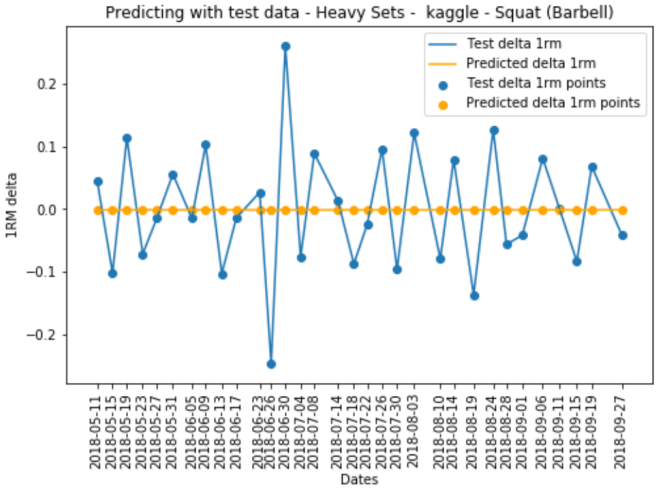


Figure 49. LSTM test predictions on Heavy Sets - mini-batch = 3

All Sets Flatten - Manual Hyperparameter Tuning

For the all sets dataset, we did multiple experiments. In this experiment, we manually tuned the hyper parameters (**number of memory cells, loss function, optimizer, dropout rate, batch size, epochs**) (combined with manual testing of future horizon and prior lag), based on the loss, as well as how the predictions aligned with the test targets and came up with the hyper parameters depicted in figure 75.

From the results we see that using a mini-batch size of 3 produced predictions that are best aligned with the test labels compared to mini-batch size 1 and 5. Likewise, when we consult the RMSE table in figure 76, then the LSTM model that used a mini-batch of size 3 also has the lowest RMSE value. Hence, looking back 3 days yielded the best predictions amongst the three mini-batch sizes. Mini-batch size 1 is the worst performing model of the three models, based on its RMSE of 0.08298. Arguably this is due to the lack of days we look back into, and therefore the model is unable to predict the next day given only 1 day to look back. Likewise, a mini-batch of size 5 has an RMSE 0.0828, which is very close to the RMSE of mini-batch 1. This is further supported as both tests visualizations closely resemble each other (figure 81). One possible reason why a mini-batch of size 5 does not work well compared to a mini-batch of size 3 may be due to the extended number of days we look back. Our assumption is due to the fact that looking further back than 3 days may not provide us with current enough data, thus rendering it obsolete. Consequently, this may affect the models' performance. Finally, one should note that the models for mini-batch 1 and 5 do not perform terrible in predicting the test targets, since they do manage to relatively capture the delta 1rm growth of the test set. However, they do not perform as good as the model with mini-batch size 3.

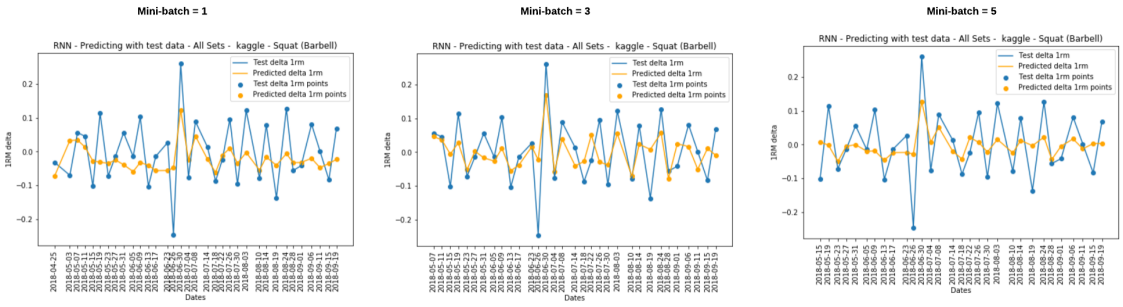


Figure 50. LSTM manual tuned hyperparameters - predictions on all sets data set - Test Results

All Sets Flatten - Grid Search Hyperparameter Tuning

Initially, we manually tuned the hyper parameters of the All Sets dataset. However, we wanted to see whether we could do hyperparameter tuing using grid random search to improve the performance of our LSTM. Random Search (RS) is a hyperparameter tuning method that evaluates random samples of hyper parameters. It has a probability of 95 percent of finding a combination of parameters within the 5 percent optima using only 60 iterations(**??**). This is very important, since the LSTM does have many hyper parameters to tune, and using a standard grid search that performs an exhaustive search, is not computationally feasible for any of the machines used in this project. The hyperparameters found by random search can be seen on figure 51). Observing the results, the best performing model used a mini-batch size of 3. Interestingly, the model trained from RS hyperparameters performed better than the model trained with our manually tuned hyper parameters, which was expected from RS. However, the performance difference between the two models is small. The RMSE of the RS Model is 0.72021, while the RMSE for the manually tuned model is 0.73596. Thus the difference is 0.01575 RMSE. Also, when we inspect the test results (figure 89) there are not huge differences between the predictions (see figure GridVsManual). Despite that, tuning the hyper parameters for our LSTM yielded positive results and it only took us ten minutes, whereas our traditional grid search stalled completely. Notice, we do not use Cross-Validation (CV) in the grid search to respect the time ordering inherent to the data.

```
#Random Search - 60 iteration
NUM_LSTM_CELL = 50
LSTM_DROPOUT_RATE = 0.2
LSTM_INPUT_BATCH_SIZE = 1
VALIDATION_SPLIT = 0.1
LOSS_FUNC = 'mse'
OPTIMIZER = 'rmsprop'
EPOCH = 10
TRAIN_SPLIT = 0.8
```

Figure 51. Random Search results for LSTM Hyperparameters

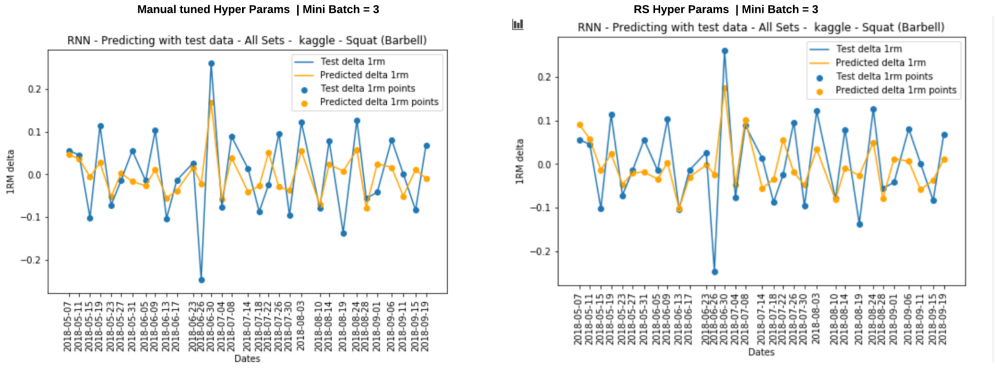


Figure 52. LSTM Random Search hyper parameters vs Manual Parameters

PCA reduced All sets flatten

Likewise, in the VAR experiments, we will try to assess how well our LSTM can predict the delta 1rm growth on a dataset whose dimensions have been reduced with PCA. As stated earlier, due to time constraints, we will compare the LSTM models based on the number of principal components. Thus we set the mini-batch size to 3 as the best LSTM model in our prior experiments on the All sets dataset used a mini-batch size of 3. Hence It would be interesting to see whether applying PCA would produce a better model.

The number of PCs we tested was likewise the same as in the VAR experiment. Specifically, 5, 10, 15, 17, and 20. By observing the results, the worst-performing model used 5 PCs. Again this makes sense as we lose information as we decrease the number of PCs, which affects the models' ability to generalize. The best LSTM model produced was a model that was trained with a dataset containing 17 PCs (53). See figure 86 for close up results and figure 87 sets overview for overview results.

In general, the models trained on the PCA reduced dataset was still able to capture some notion of the delta 1rm growth and was not far off compared with the models trained on the original feature space. However, the LSTM models trained on the PCA reduced dataset did not beat the best model from the previous experiments.

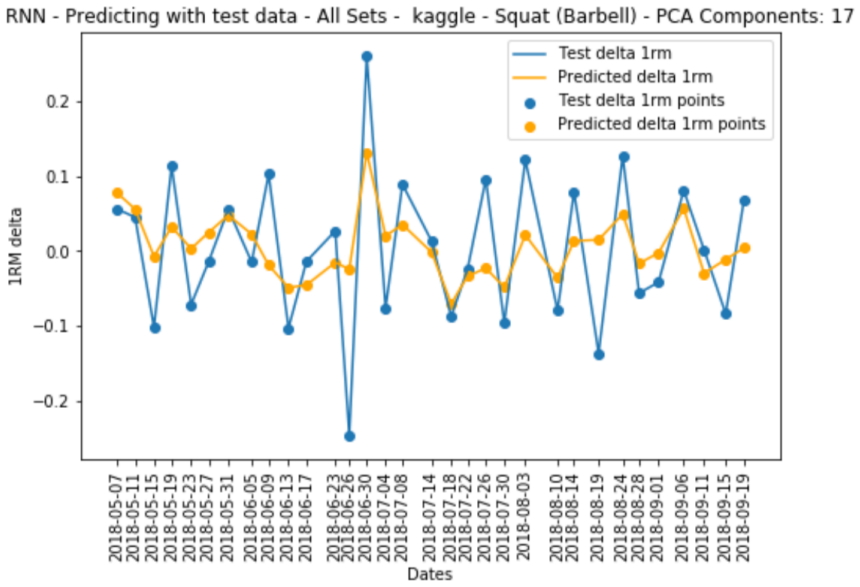


Figure 53. LSTM Test results with 17 PC

5.6 Best LSTM Model

Based on all observations, the best LSTM model based on RMSE, is the model trained with Random Searched Hyper parameters with a mini-batch size of 3. This model has the smallest overall RMSE of all LSTM models whereas the worst performing model was LSTM model trained with dataset of 5 PCs. (Figure **??**). By visually inspecting the predictions of delta 1rm for each model, we can see that the above statements are reflected as well.

LSTM - ALL SETS - Different lookback range		
Hyperparams set	**Look back days**	**RMSE**
Manual Params - 1	1	0.082980
Manual Params - 1	3	0.073596
Manual Params - 1	5	0.082820
Random Search Params - 2	1	0.073465
Random Search Params - 2	3	0.072021
Random Search Params - 2	5	0.089600

LSTM - PCA - ALL SETS	Look back = 3	
Hyperparams set	**PCA Components**	**RMSE**
Manual Params - 4	5	0.095159
Manual Params - 4	10	0.095114
Manual Params - 4	17	0.078741
Manual Params - 4	15	0.083804
Manual Params - 4	20	0.083635
Manual Params - 3 - Pre reducing epochs	17	0.109869

Figure 54. LSTM RMSE Table for the models

NB: the LSTM graphs are wrongfully labelled RNN

5.7 LSTM and VAR Model Comparison

The following observations have been made when comparing the VAR and LSTM model for multivariate time series forecasting in our experiments. Firstly, training the models on a PCA reduced dataset yielded poorer results than using the dataset with the original feature space retained, which is most likely due to loss of information. On the heaviest dataset, the LSTM model was unable to predict the delta 1rm growth, whereas the VAR model was able to do that. This might be due to the lack of progression between workout sets being important but we were surprised that a simpler VAR model managed to somewhat capture the pattern of delta 1rm growth. In terms of prediction, we were unable to directly compare how the two models predict the data because the VAR model was used to forecast the strength growth N days into the future, whereas the LSTM model used the sliding window method on mini-batches to forecast the strength growth on the next day in the test set.

However, we are able to compare how well each of the two models predict future delta 1rm strength growth. Using the all sets data set, both models were able to predict the delta 1rm growth in the near future. Yet, the quality of their predictions depended on how far we look back (i.e. the lag). Interestingly, if we look too far back, then both the LSTM and VAR model was unable to fully capture the delta 1rm strength growth. Again, this is likely due to how current the past data used is, since workout observations way back in the past become a somewhat obsolete explanation of your strength today or in the future. In some way, either trying to predict the strength growth today using workouts too far back or trying to predict strength growth way into the future both seem like challenging if not impossible endeavors given the amount of uncertainty inherent in the future. Nonetheless, using more recent workout observations yielded best results. This does indeed make sense as old workout observations do not seem to be relevant to your delta 1rm strength growth. Put into perspective, using the past 3 workout observations to predict the delta 1rm strength growth today seem reasonable, since that corresponds to using the last week of workouts if the person went to the gym three times a week.

Going back to the model comparison, we noticed that the LSTM was way more prone to overfitting, since it is a more complex model than the VAR model. This also showed in the training time where the VAR model was always substantially faster. Surprisingly, the VAR model achieved the lowest RMSE over the LSTM, which suggest that simpler statistical models may suffice over using very complex models like the LSTM neural network that are hard to interpret. Further, the LSTM required a substantial amount of additional preprocessing, since we needed to build our own batching methods to preserve the time ordering that many libraries violate and the hyperparameter tuning also took a lot of time. This was way simpler with the VAR model that only requires us to transform the non-stationary time series of our workout variables. In that regard, the LSTM proved more flexible as it required no transformation of the time series whatsoever.

According to **Occam's razor**, we should pick the simplest (i.e. VAR) of the two models given that they perform equally well based on the RMSE. However, by visual inspection, we notice that the LSTM is seemingly better able to pick up on the pattern inherent to the delta 1rm strength growth as explained through the correlation and evolvment of our workout observations over time.

Top All Sets

Type	Models	RMSE
VAR	All Sets - Train 75 %	0.068459
LSTM	Random Search Params 2 - look back 3	0.072021
LSTM	Manual Params - 1 All Sets	0.073596
LSTM	Manual Params - 4 - PCA 17 - All Sets	0.078741
VAR	PCA - 17 - Train 50 %	0.102519

Figure 55. All Models - RMSE Table for All sets Dataset

6 CONCLUSION

In this research project, we wanted to investigate what data and Machine Learning methods that are suitable for optimizing fitness training. We scoped the project to the domain of powerlifting where we wanted to determine whether it is possible to predict strength growth over time in the Squat powerlift. In order to do this, we first collected relevant domain knowledge from literature studies that explain what features and data is relevant to optimizing strength. We learned that three of the most important features for predicting strength growth are volume (i.e. sets and repetitions performed), intensity (i.e. weight lifted) and training training. Further, the literature studies revealed that heavier training loads tend to increase strength more than lighter training loads. Finally, we found the 1RM (i.e. one-rep max) estimate to be a valuable measure of strength measured by how much you can lift maximally in a single repetition or through estimation from a combination of a weight and number of repetitions. These were the most important findings used in the design of our models although we found a substantial amount of useful domain knowledge. In future work, one could experiment with Bayesian models to encode the domain knowledge (e.g. Prilepin's Chart) through priors given that we have very little data to build from.

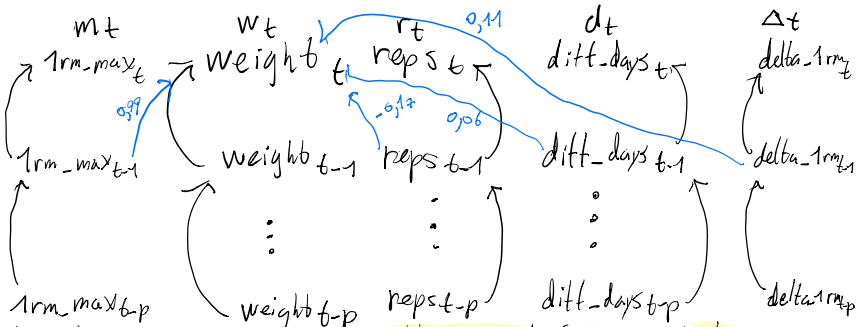
Secondly, we figured out how to model strength growth as a supervised multivariate time series regression problem. We realized that we could not use many traditional Machine Learning methods that we are familiar with such as cross validation and ordinary hyper parameter tuning that both violated the time-ordering inherent in the workout observations. Instead, we realized that we had to model strength growth over time, which time series forecasting is suitable for when wanting to predict future observations based on current and past observations. In our case, the strength growth is a continuous metric, which led to a regression problem. Finally, since we have multiple features in the workout data that correlate together along with the strength growth, we chose to only consider multivariate models. For this purpose, a traditional approach in statistics was compared to a more recent and advanced Machine Learning approach. The statistical model was the Vector Auto Regressive model that models future workout strength growth as a linear combination of previous workout variables and strength growth. The Machine Learning method was the Long-Term-Short-Term neural network, which is a specialized Recurrent Neural Network that is able to model sequential (here a time series) data and retain information about past observations to predict future observations.

In conclusion, we figured out that it is possible to measure and predict strength growth through the 1RM metric as a time series regression problem. Surprisingly, the VAR model achieved the best lowest error compared to the LSTM model despite being simpler, faster to train and easier to interpret. The LSTM required a substantial amount of additional preprocessing by batching the workout data, since we had to reframe the time series forecasting problem into that of a supervised learning problem, which required us to batch the time series workout data using the sliding window method. However, by visual inspection the LSTM does seem to better pick up on the complex pattern in the development of strength growth over time.

In future work, we would like to experiment with other measures of growth in the fitness industry such as muscly hypertrophy, weight loss, conditioning, endurance or power. Further, we would like to investigate how to find the optimal combination of sets, weights and repetitions that led to the maximum growth in strength. Finally, we would like to test out the theory in practice through our own weightlifting app that enables you to track your workouts and receive recommendations on how to lift for strength. We have realized that we have very little data and need to spend more time on data collection in order to obtain a representative sample of workout observations from the population in Denmark. Ultimately, with the aim to generalize how individuals should workout based on their similarity to other individuals of the population.

7 APPENDIX

7.1 Var 1 Equations



VAR (p=1): NB: w and r variables included for max #sets

$$w_t = c_{11} \cdot w_{t-1} + c_{12} \cdot r_{t-1} + c_{13} \cdot m_t + c_{14} \cdot d_t + c_{15} \cdot \Delta_t + \varepsilon_{w,t}$$

$$r_t = c_{21} \cdot w_{t-1} + c_{22} \cdot r_{t-1} + c_{23} \cdot m_t + c_{24} \cdot d_t + c_{25} \cdot \Delta_t + \varepsilon_{r,t}$$

$$m_t = c_{31} \cdot w_{t-1} + c_{32} \cdot r_{t-1} + c_{33} \cdot m_t + c_{34} \cdot d_t + c_{35} \cdot \Delta_t + \varepsilon_{m,t}$$

$$d_t = c_{41} \cdot w_{t-1} + c_{42} \cdot r_{t-1} + c_{43} \cdot m_t + c_{44} \cdot d_t + c_{45} \cdot \Delta_t + \varepsilon_{d,t}$$

$$\Delta_t = c_{51} \cdot w_{t-1} + c_{52} \cdot r_{t-1} + c_{53} \cdot m_t + c_{54} \cdot d_t + c_{55} \cdot \Delta_t + \varepsilon_{\Delta,t}$$

$k=5$ + 10 sets $\cdot (w_t, r_t) = 25$ in code

$$\begin{bmatrix} w_t \\ r_t \\ m_t \\ d_t \\ \Delta_t \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} \end{bmatrix} \cdot \begin{bmatrix} w_{t-1} \\ r_{t-1} \\ m_{t-1} \\ d_{t-1} \\ \Delta_{t-1} \end{bmatrix} + \begin{bmatrix} \varepsilon_{w,t} \\ \varepsilon_{r,t} \\ \varepsilon_{m,t} \\ \varepsilon_{d,t} \\ \varepsilon_{\Delta,t} \end{bmatrix}$$

$\vec{y}_t$: $k \times 1 = 5 \times 1$ $\quad C$: $k \times k = 5 \times 5$ $\quad \vec{y}_{t-1}$: $k \times 1 = 5 \times 1$ $\quad \vec{\varepsilon}_{y,t}$: $k \times 1 = 5 \times 1$

In the final model, we have a weight and reps variable for all sets

$$\vec{y}_t = C \vec{y}_{t-1} + \vec{\varepsilon}_{y,t} \quad \Leftrightarrow$$

$k \times 1$ $\quad k \times k$ $\quad k \times 1$ $\quad k \times 1$

25 vars + 1 constant

$26 = 22 + 3 + 1$

25 vars + 1 constant: (11 sets x weight, rep) + (delta_1rm, days_diff_max_rm) + c

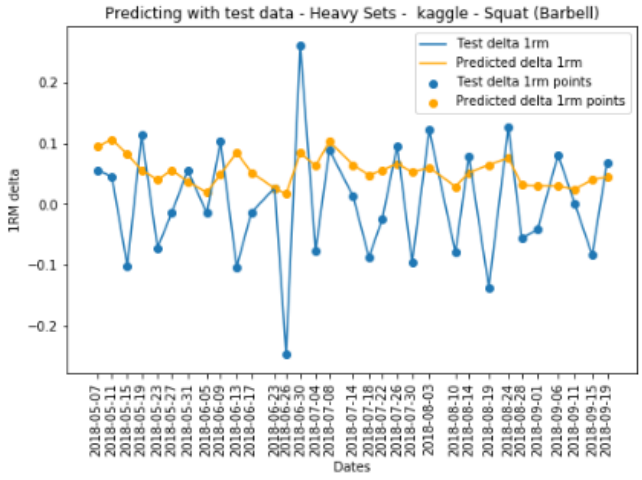


Figure 56. Delta 1RM prediction on test data with MAE as loss function

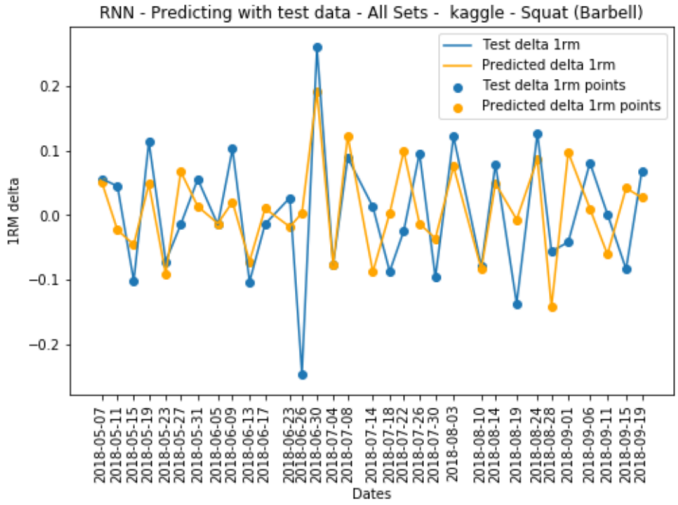


Figure 57. Delta 1RM prediction on test data with MAE as loss function

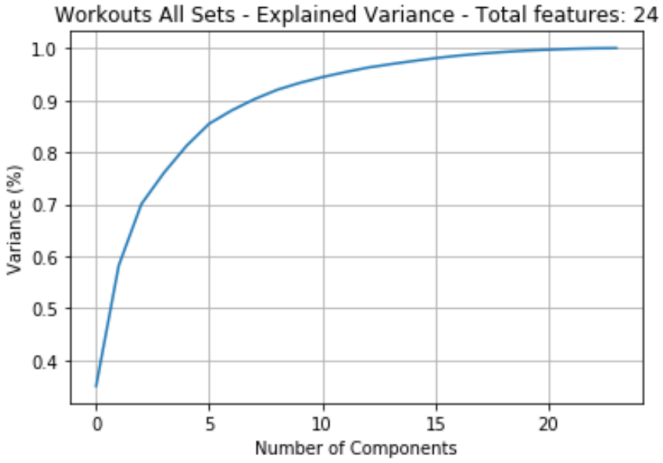


Figure 58. Cumulative Explained Variance proportional with the number of PCs selected.

7.2 Vector Auto Regression - Results

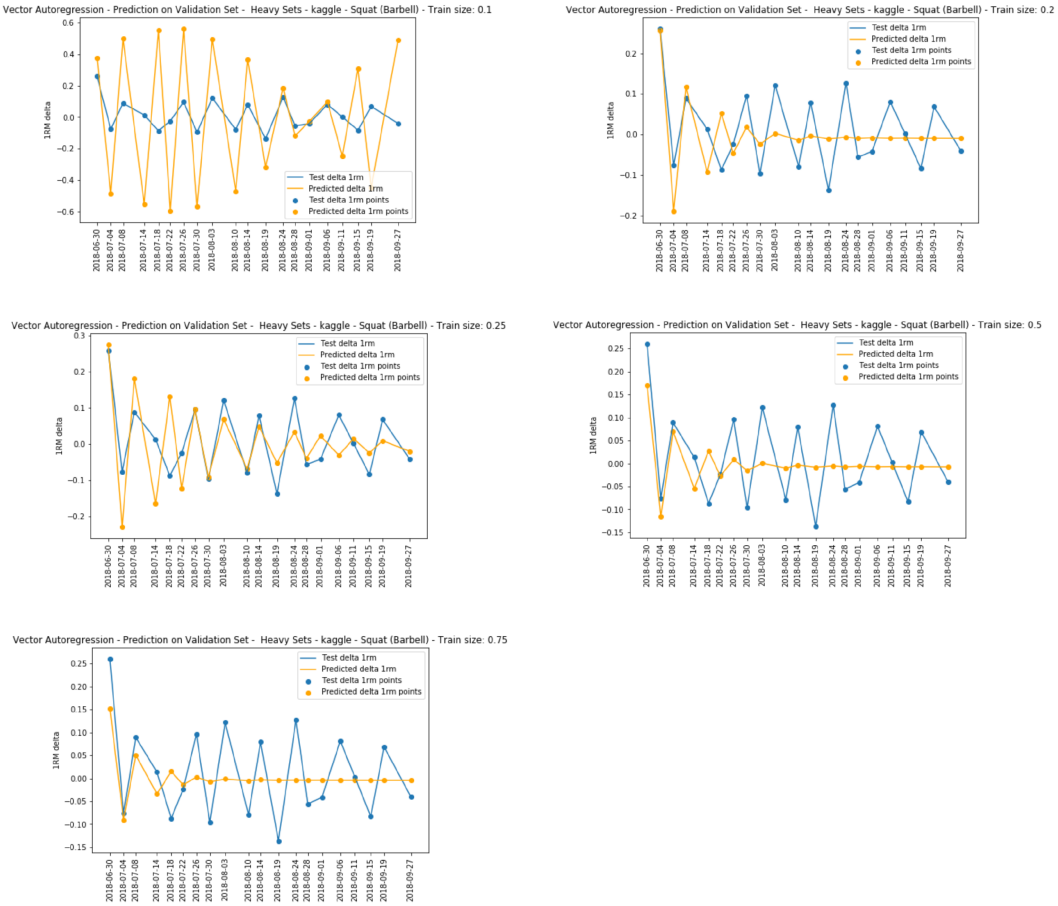


Figure 59. VAR predictions on heavy sets data set - Test Results

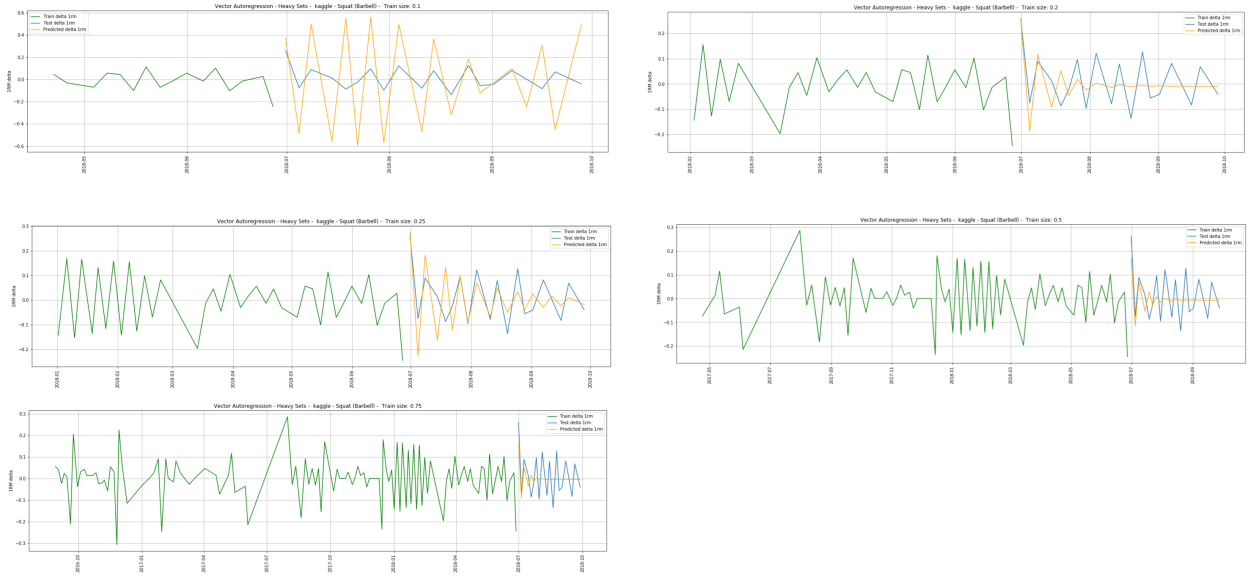


Figure 60. VAR predictions on heavy sets data set - Overview Results

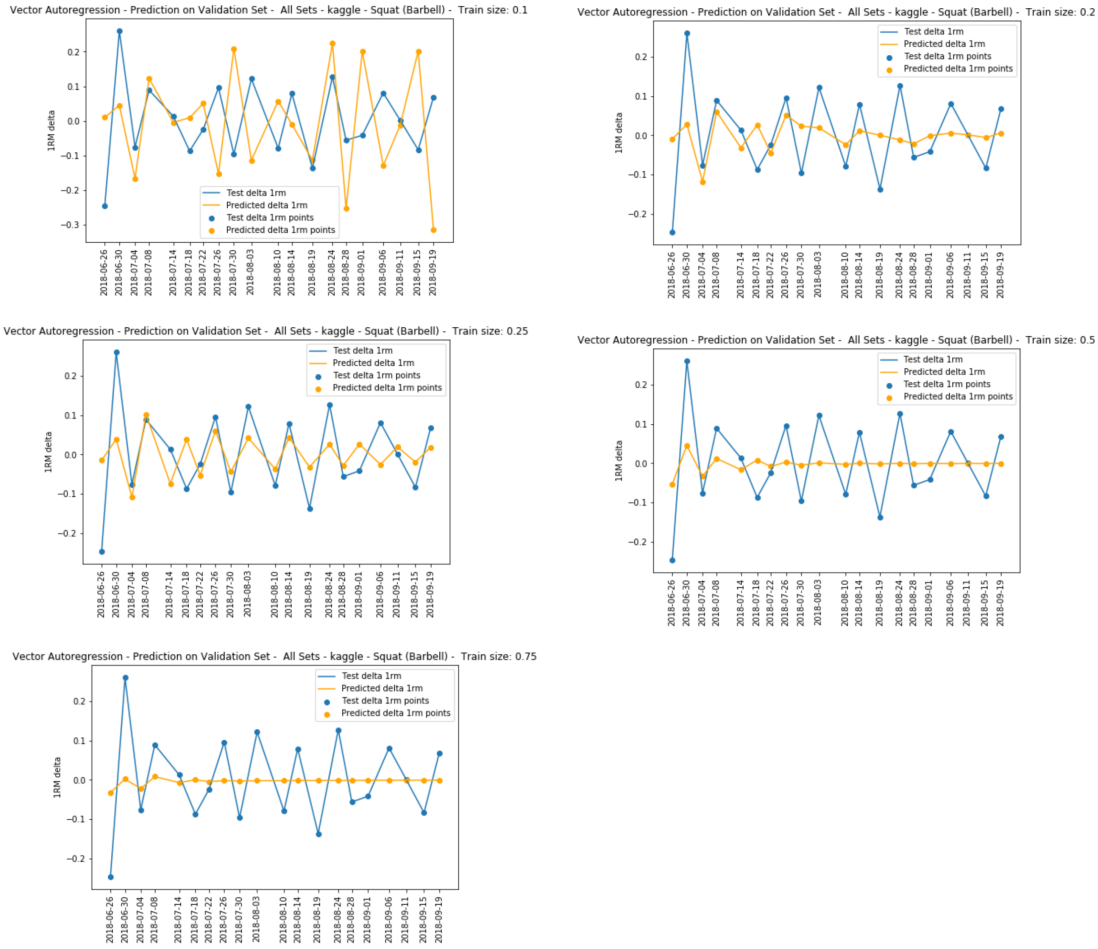


Figure 61. VAR predictions on all sets data set - Test Results

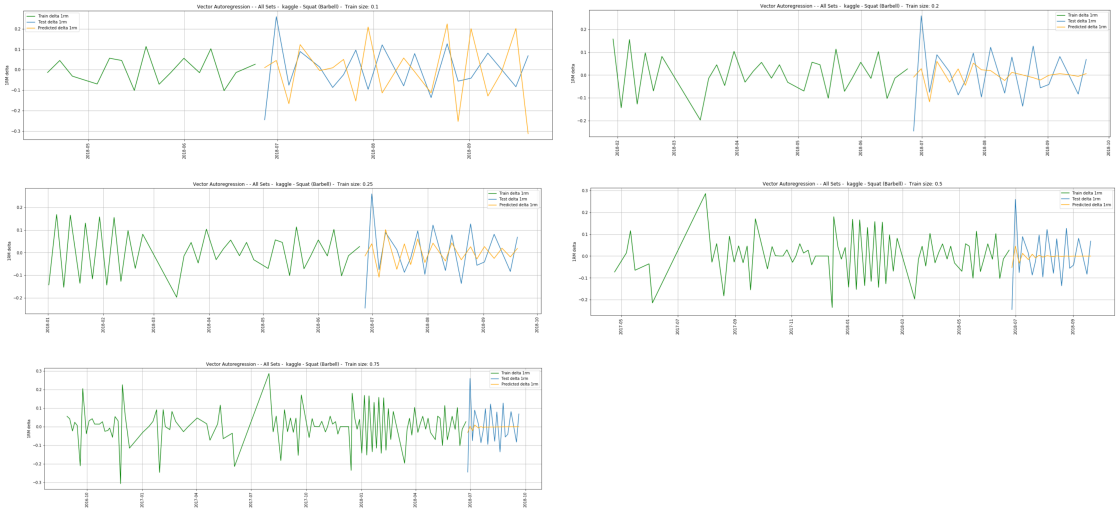


Figure 62. VAR predictions on all sets data set - Overview Results

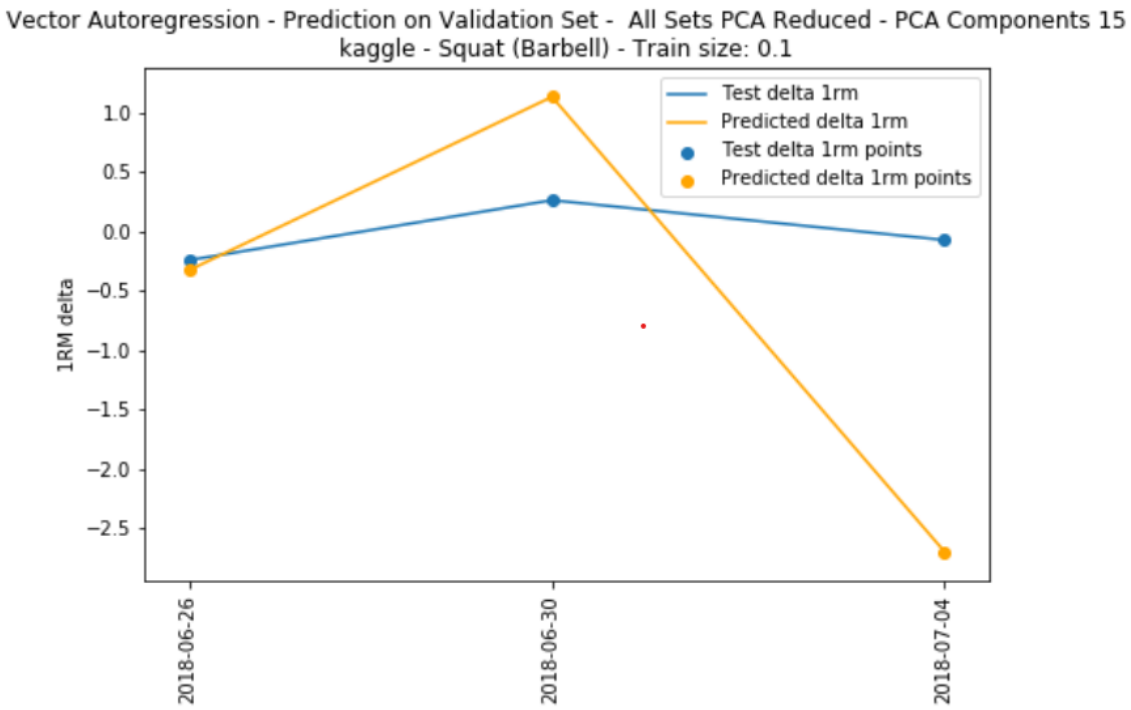


Figure 63. VAR predictions on all sets data set - Test Results - Scoped to 3 days

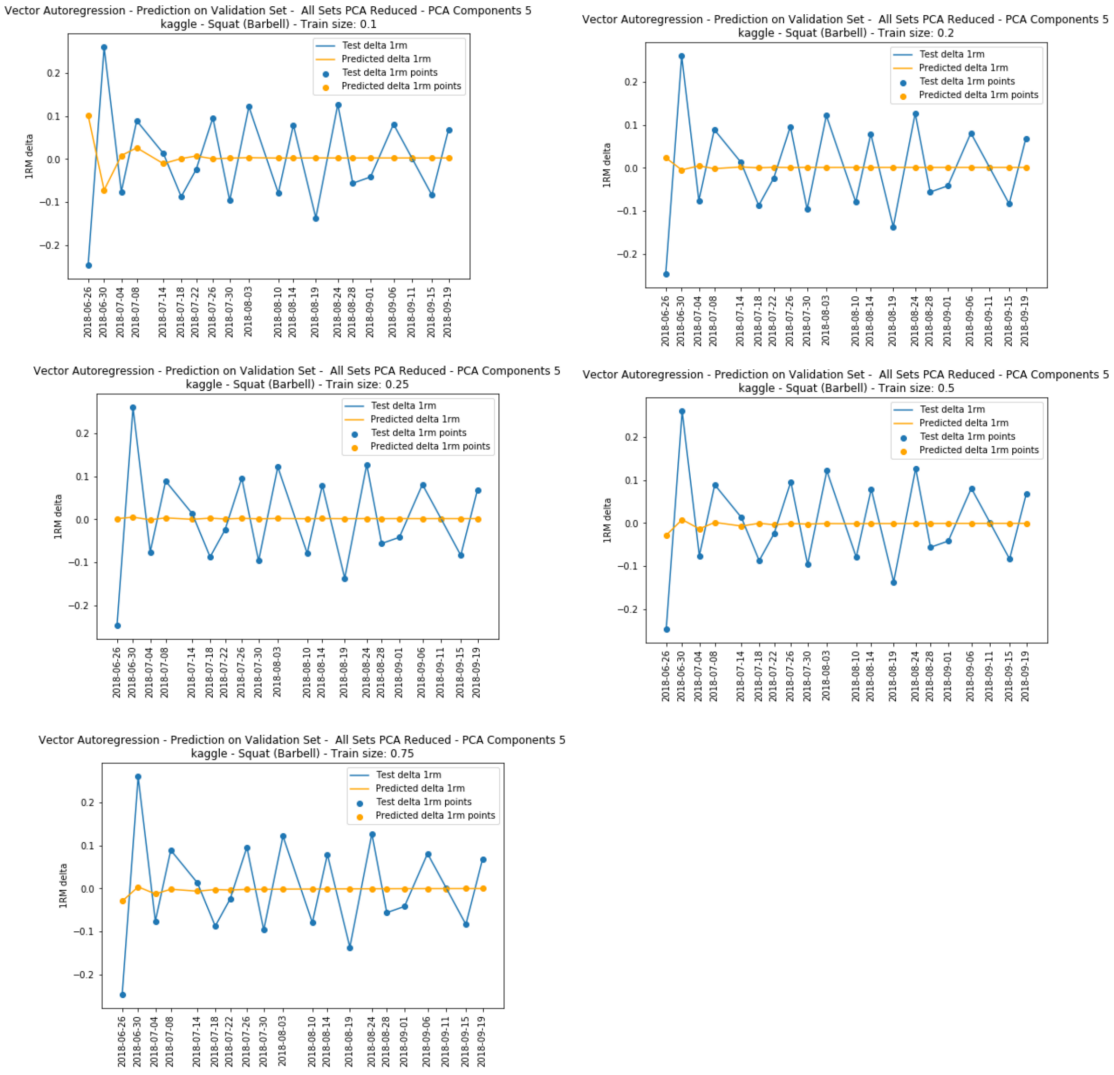


Figure 64. VAR predictions on PCA reduced all sets dataset with 5 PCs - Test Results

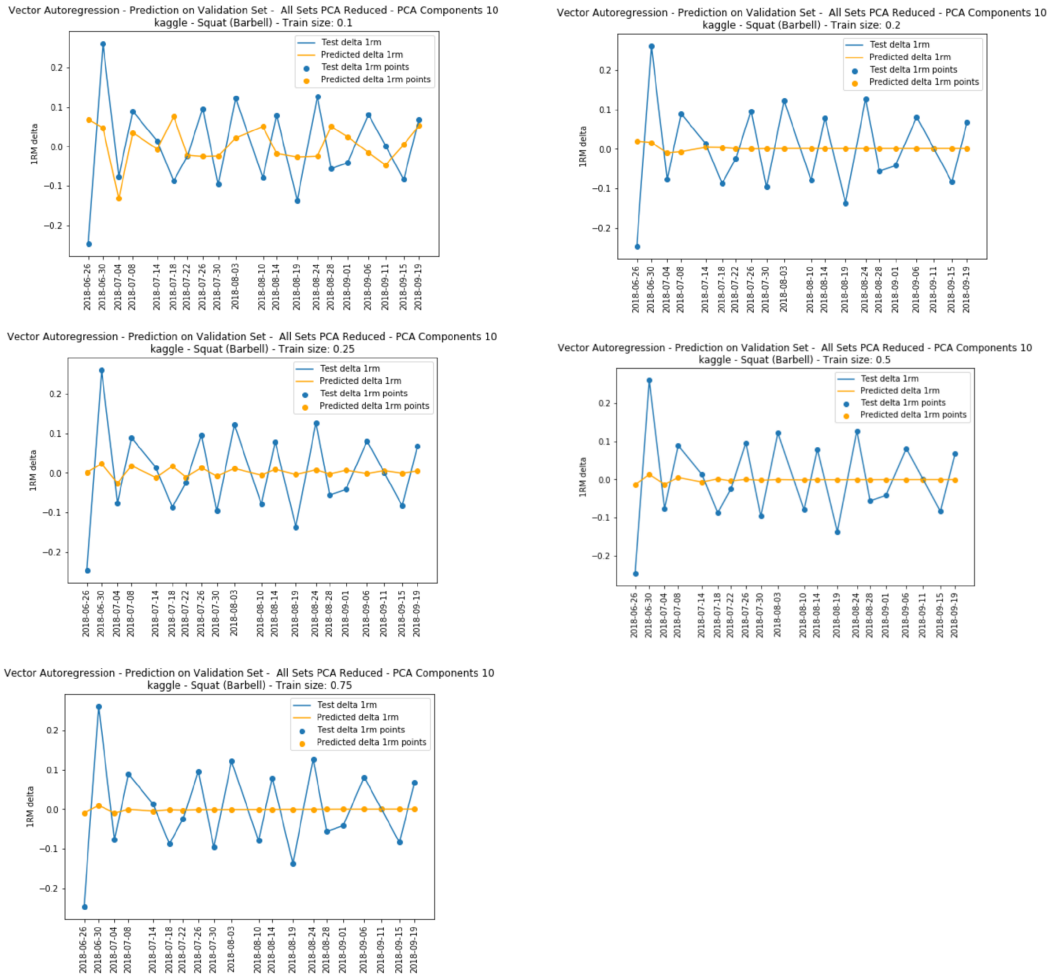


Figure 65. VAR predictions on PCA reduced all sets dataset with 10 PCs - Test Results

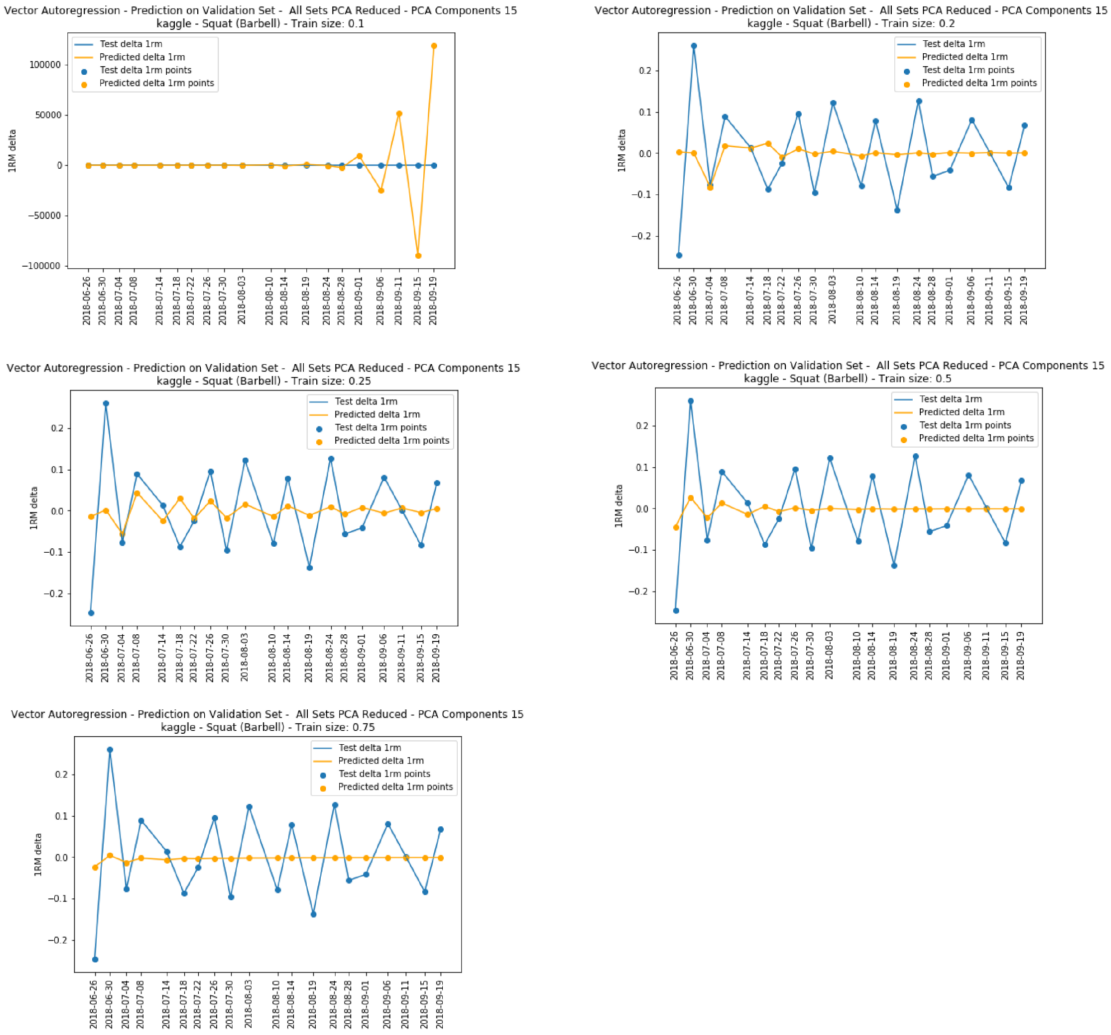


Figure 66. VAR predictions on PCA reduced all sets dataset with 15 PCs - Test Results

Figure 67. VAR predictions on PCA reduced all sets dataset with 17 PCs - Test Results

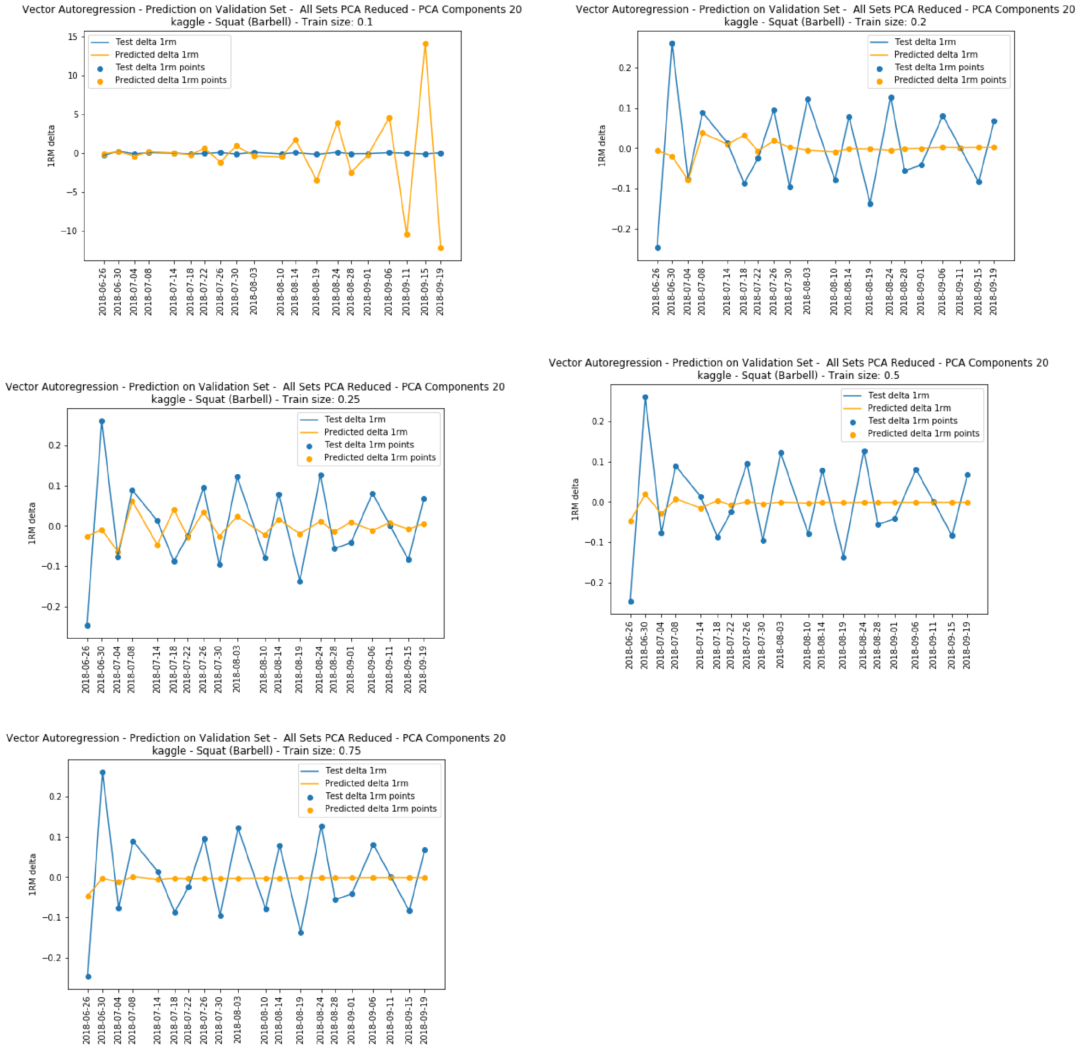


Figure 68. VAR predictions on PCA reduced all sets dataset with 20 PCs - Test Results

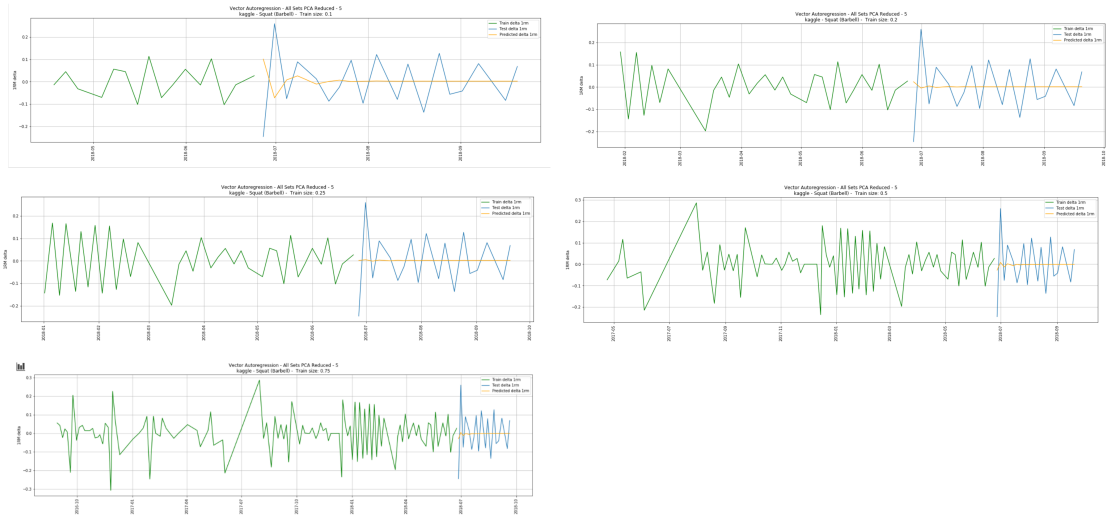


Figure 69. VAR predictions on PCA reduced all sets dataset with 5 PCs - Overview Results

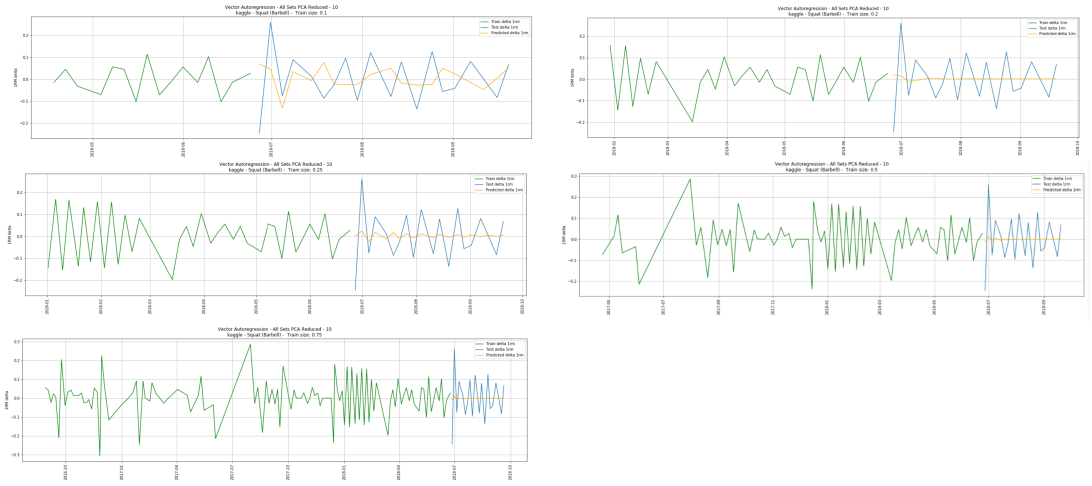


Figure 70. VAR predictions on PCA reduced all sets dataset with 10 PCs - Overview Results

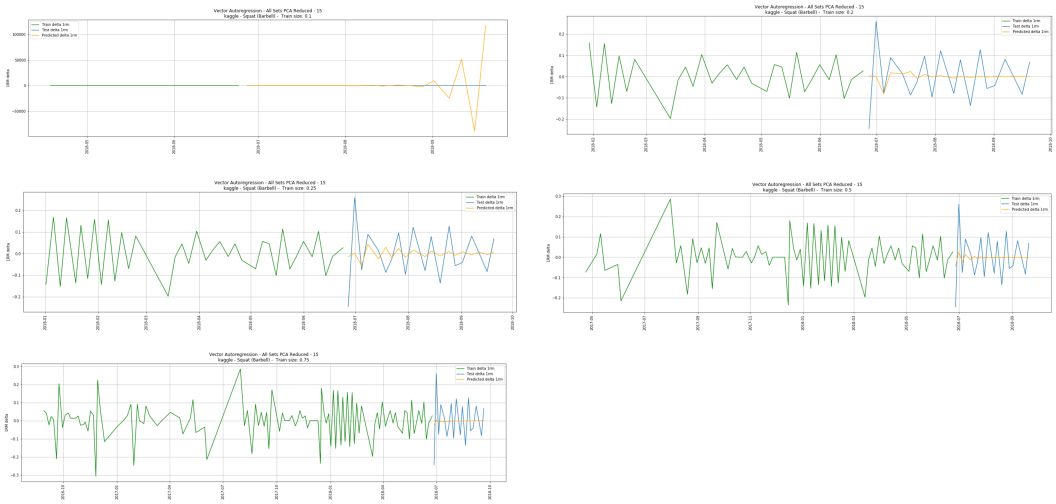


Figure 71. VAR predictions on PCA reduced all sets dataset with 15 PCs - Overview Results

Figure 72. VAR predictions on PCA reduced all sets dataset with 17 PCs - Overview Results

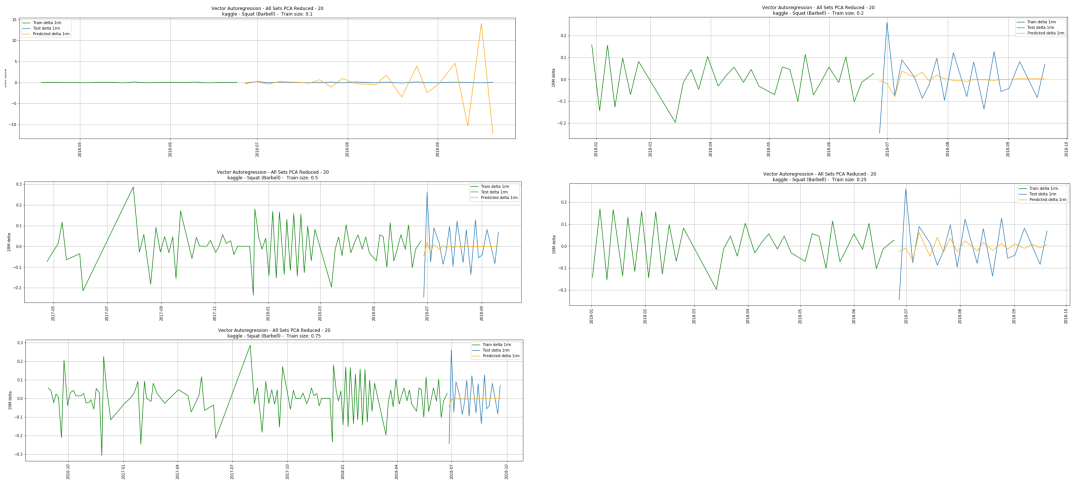


Figure 73. VAR predictions on PCA reduced all sets dataset with 20 PCs - Overview Results

Vector Auto Regression	Forecast length = 20		
Workout Set	Training set size %	RMSE	
All Sets	0.10	0.143392	Wost all sets
All Sets	0.20	0.080006	
All Sets	0.25	0.103072	
All Sets	0.50	0.066876	
All Sets	0.75	0.068459	Best All sets
Heavy Sets	0.10	0.087174	
Heavy Sets	0.20	0.075014	
Heavy Sets	0.25	0.116418	Worst Heavy sets
Heavy Sets	0.50	0.058786	Best Heavy sets and VAR
Heavy Sets	0.75	0.075287	
PCA - 5	0.10	0.132919	Worst PCA 5
PCA - 5	0.20	0.115856	
PCA - 5	0.25	0.111822	
PCA - 5	0.50	0.108018	Best PCA 5
PCA - 5	0.75	0.108550	
PCA - 10	0.10	0.123054	Worst PCA 10
PCA - 10	0.20	0.112932	
PCA - 10	0.25	0.106535	Best PCA 10
PCA - 10	0.50	0.108752	
PCA - 10	0.75	0.109978	
PCA - 15	0.10	35622.800000	WORST ALL
PCA - 15	0.20	0.110340	
PCA - 15	0.25	0.104933	
PCA - 15	0.50	0.103383	Best PCA 15
PCA - 15	0.75	0.109109	
PCA - 17	0.10	126.400000	Worst PCA 17
PCA - 17	0.20	0.108936	
PCA - 17	0.25	0.098866	
PCA - 17	0.50	0.102519	Best PCA 17 and all PCA
PCA - 17	0.75	0.106072	
PCA - 20	0.10	5.100000	Worst PCA 20
PCA - 20	0.20	0.112514	
PCA - 20	0.25	0.103901	Best PCA 20
PCA - 20	0.50	0.104000	
PCA - 20	0.75	0.107646	

Figure 74. VAR RMSE results for all experiments

7.3 LSTM Results

```
# HYPER PARAMS FALLBACK
NUM_LSTM_CELL = 30
LSTM_DROPOUT_RATE = 0.2
LSTM_INPUT_BATCH_SIZE = 1
VALIDATION_SPLIT = 0.1
LOSS_FUNC = 'mse'
OPTIMIZER = 'adam' # ADAM IS BEST AFTER MANUAL EXPERIENCES
EPOCH = 15 # OVERFITS AFTER 15
TRAIN_SPLIT = 0.8
```

Figure 75. LSTM manual tuned hyperparameters

LSTM - ALL SETS - Different lookback range		
Hyperparams set	Look back days	RMSE
Manual Params - 1	1	0.082980
Manual Params - 1	3	0.073596
Manual Params - 1	5	0.082820
Grid Search Params - 2	1	0.107679
Grid Search Params - 2	3	0.094798
Grid Search Params - 2	5	0.097202

Figure 76. LSTM All Sets RMSE table

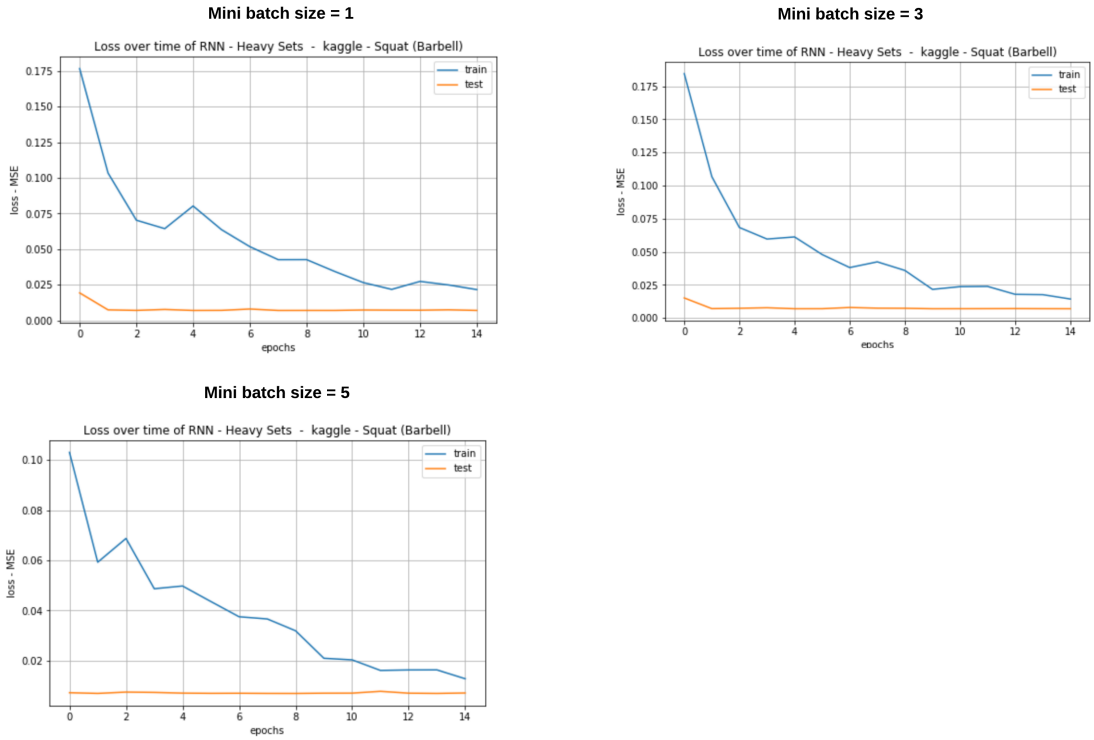


Figure 77. LSTM Heavy Sets - loss

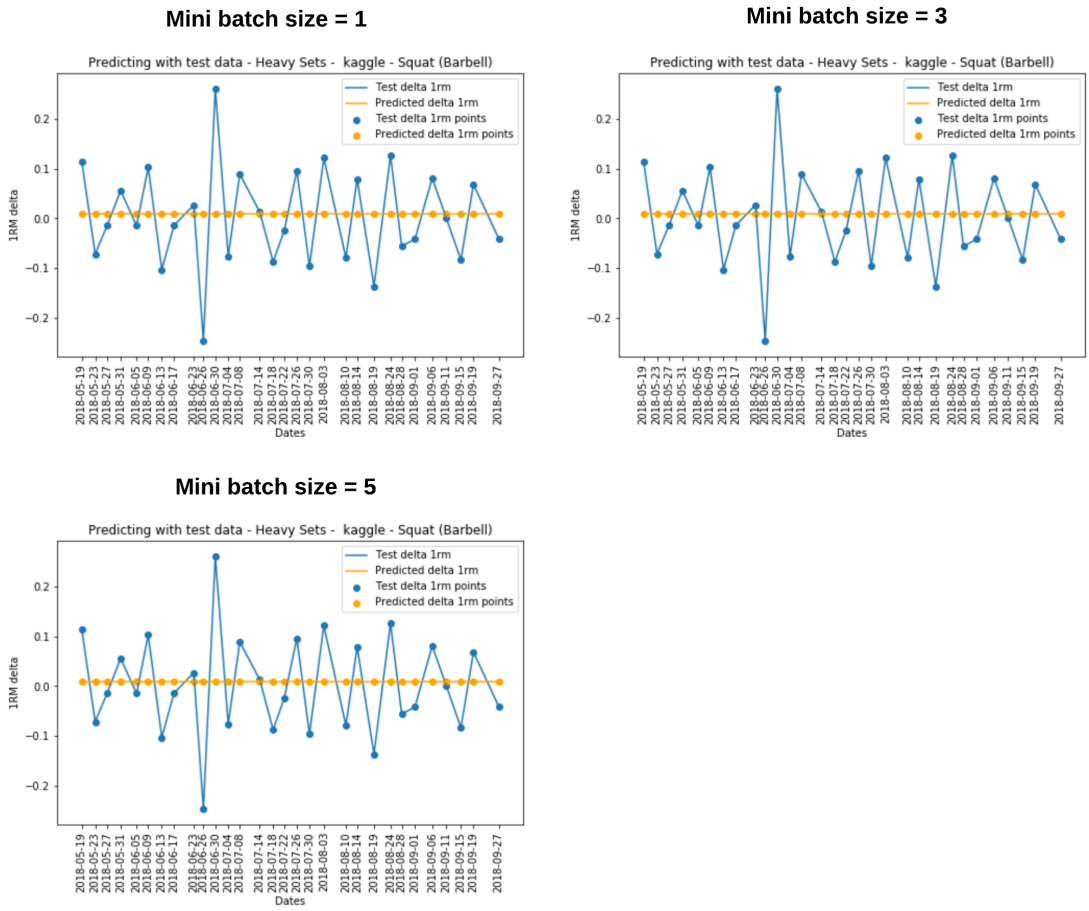
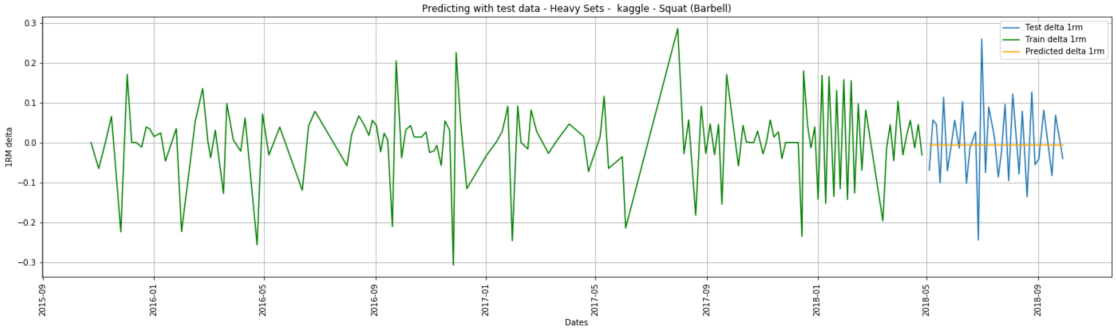
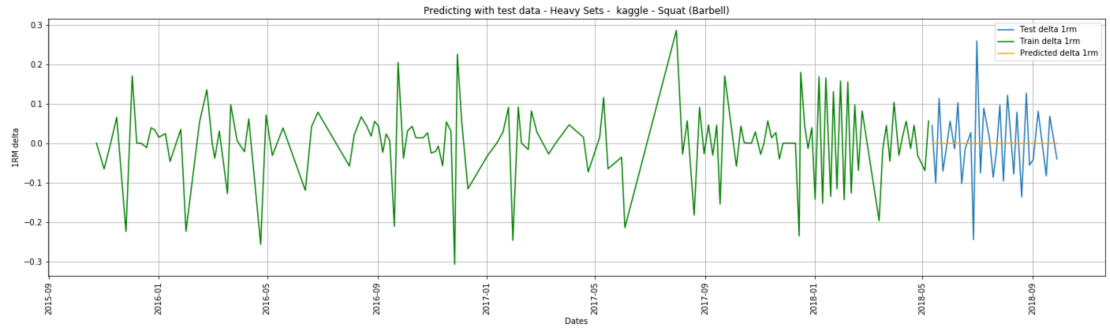


Figure 78. LSTM Heavy Sets - Test Results

Mini batch size = 1



Mini batch size = 3



Mini batch size = 5

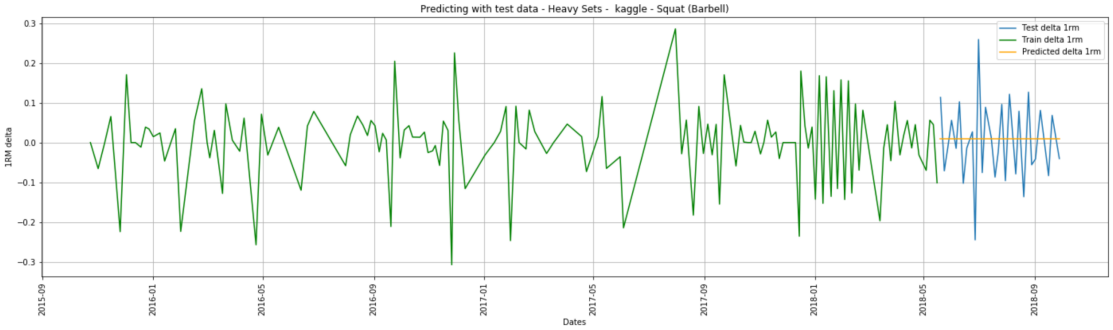


Figure 79. LSTM Heavy Sets - Overview Results

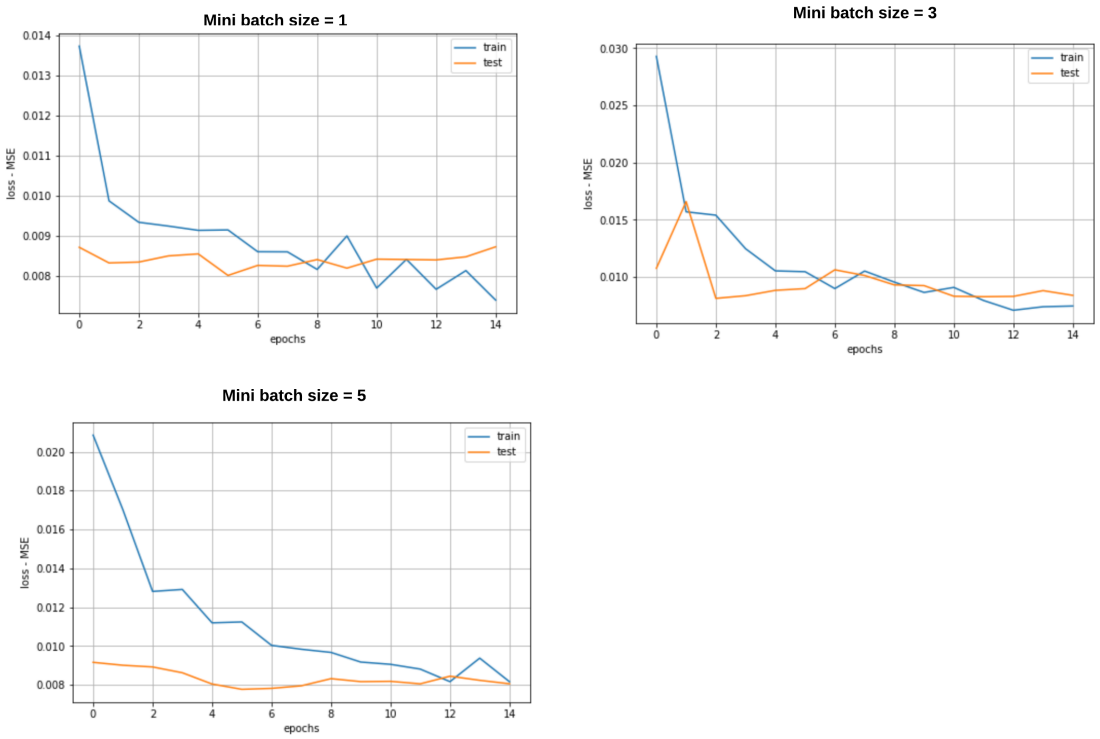


Figure 80. LSTM manual tuned hyperparameters - loss - All sets Datasets

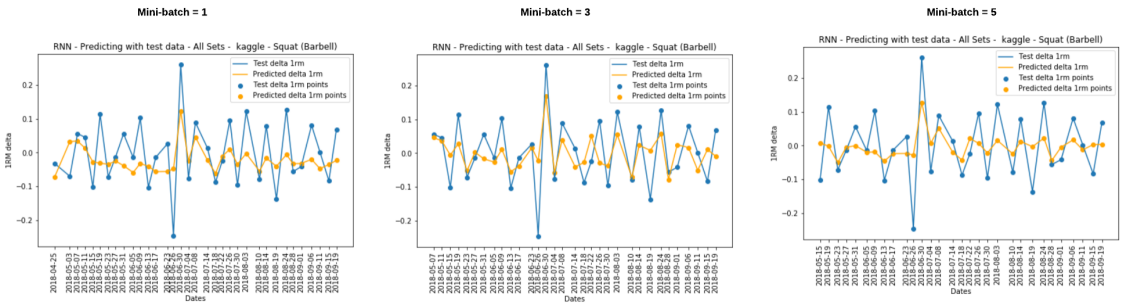
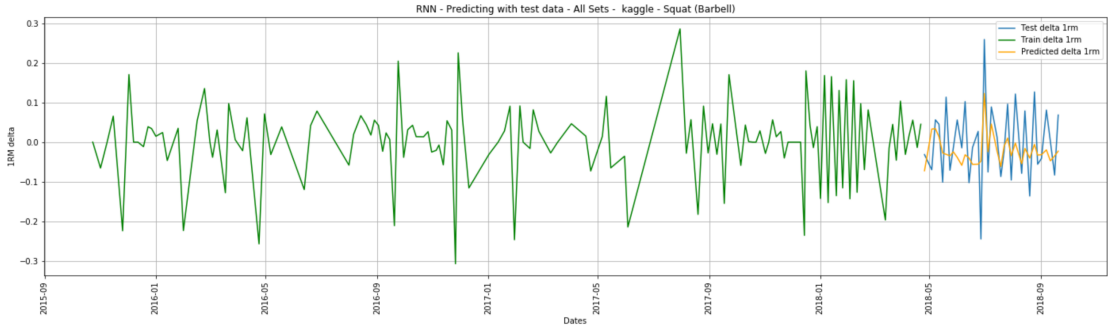
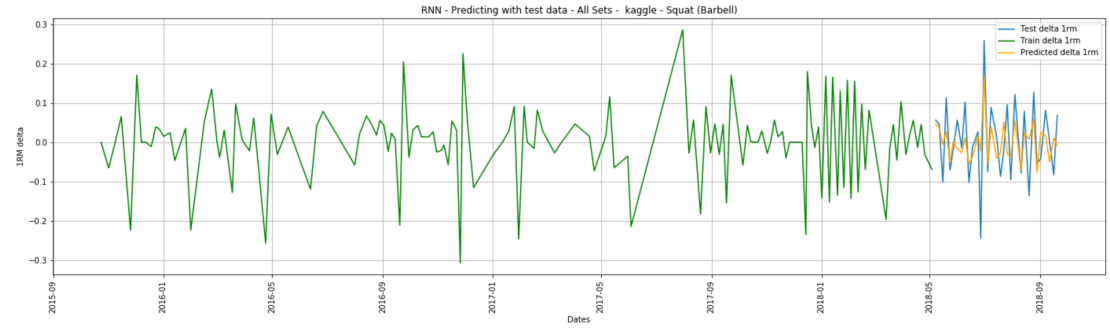


Figure 81. LSTM manual tuned hyperparameters - predictions on all sets data set - Test Results

Mini batch size = 1



Mini batch size = 3



Mini batch size = 5

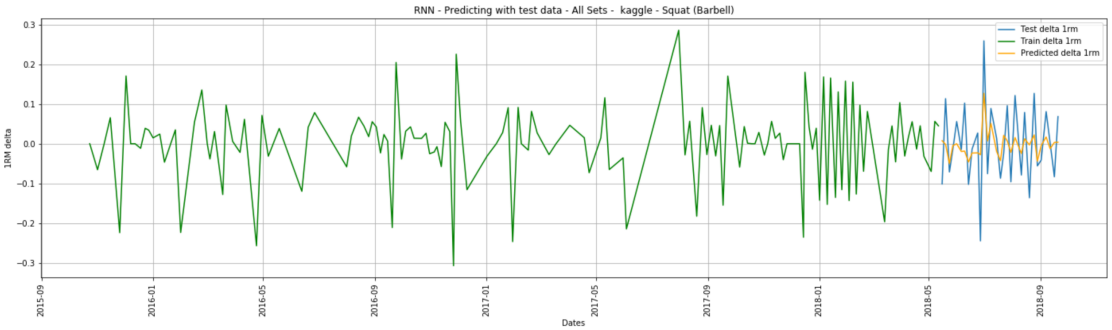
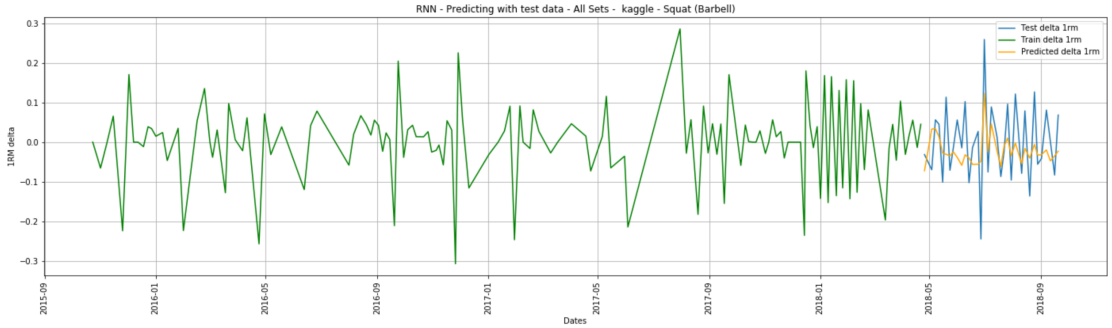
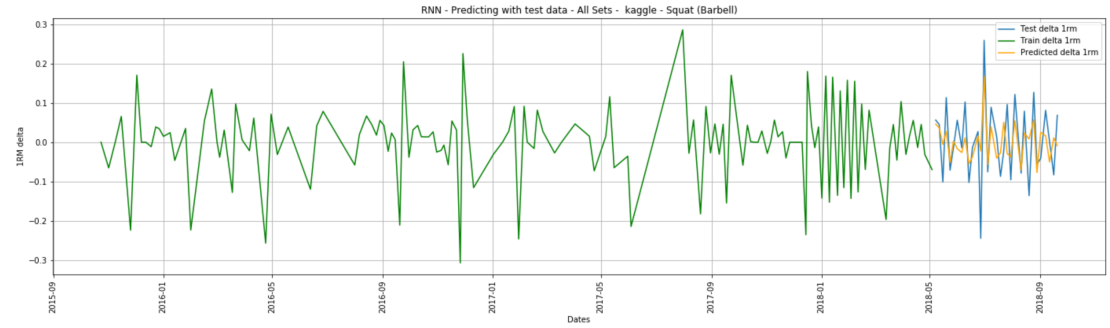


Figure 82. LSTM manual tuned hyperparameters - predictions on all sets data set - Overview Results

Mini batch size = 1



Mini batch size = 3



Mini batch size = 5

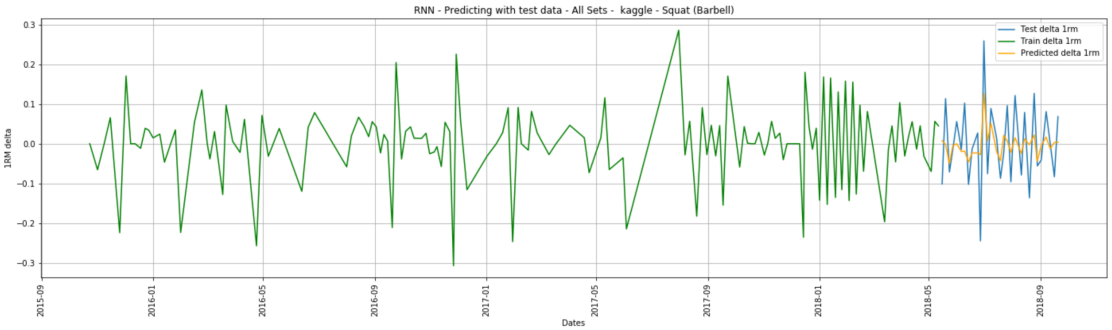


Figure 83. LSTM manual tuned hyperparameters - predictions on all sets data set - Overview Results

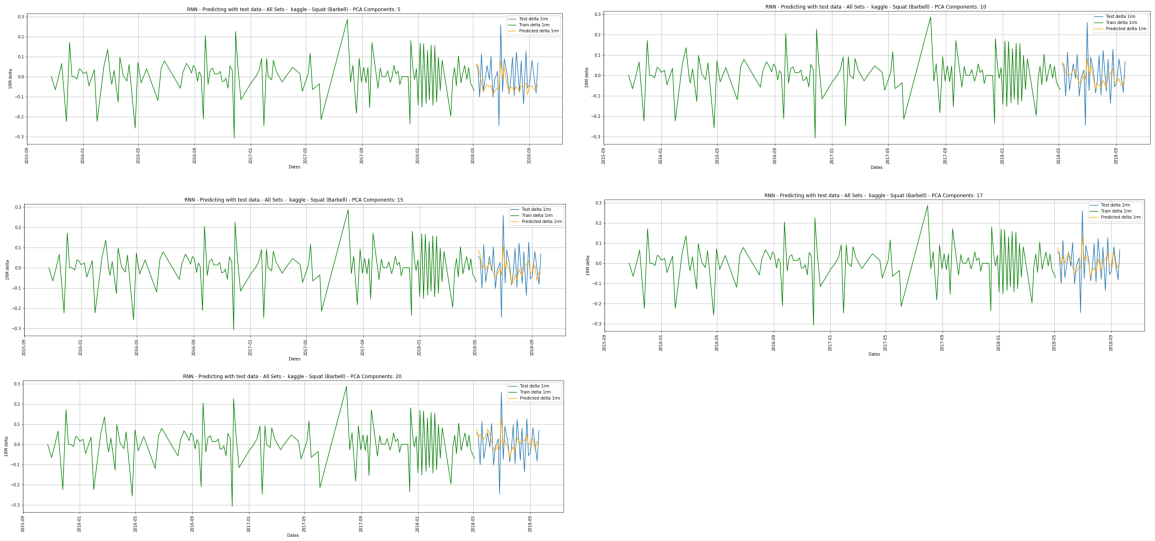


Figure 84. LSTM - predictions on PCA data set - Overview Results

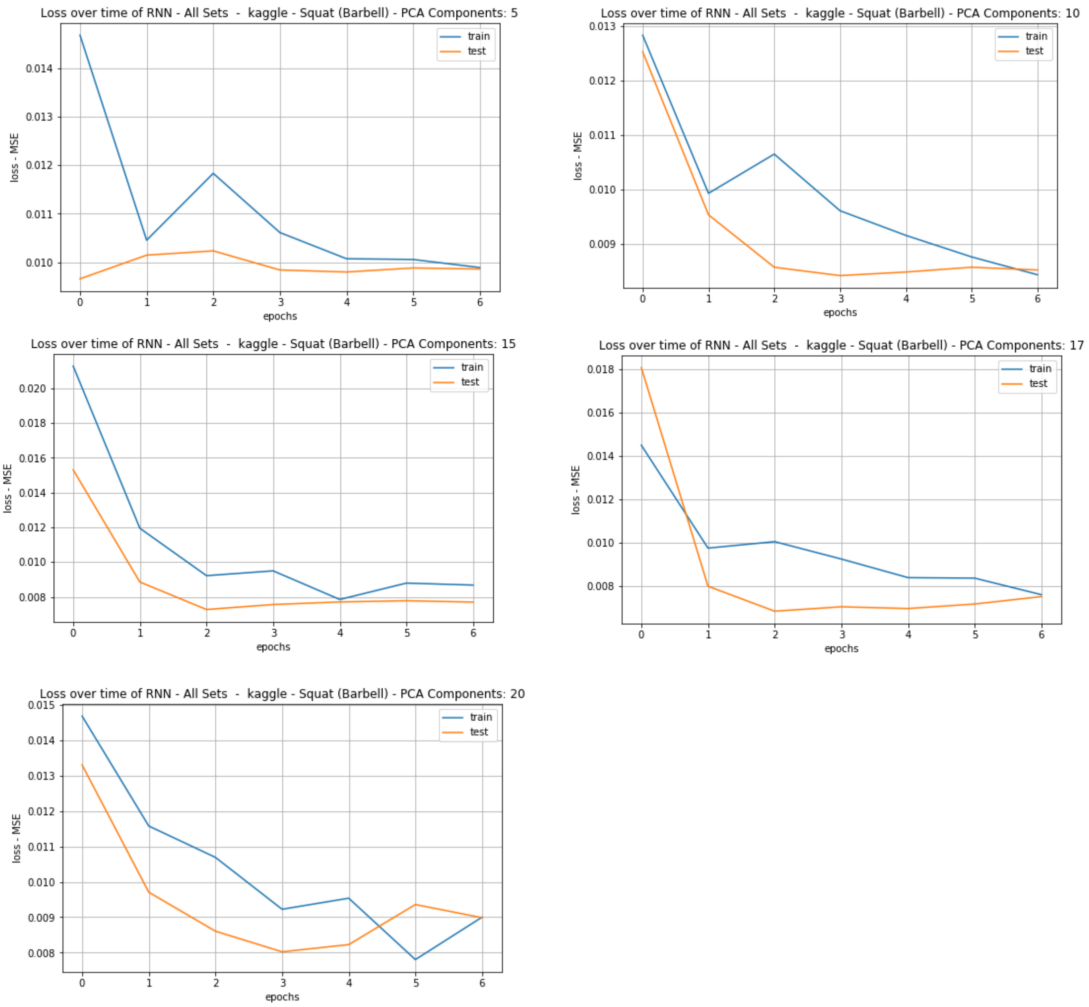


Figure 85. LSTM - predictions on PCA data set - loss over time

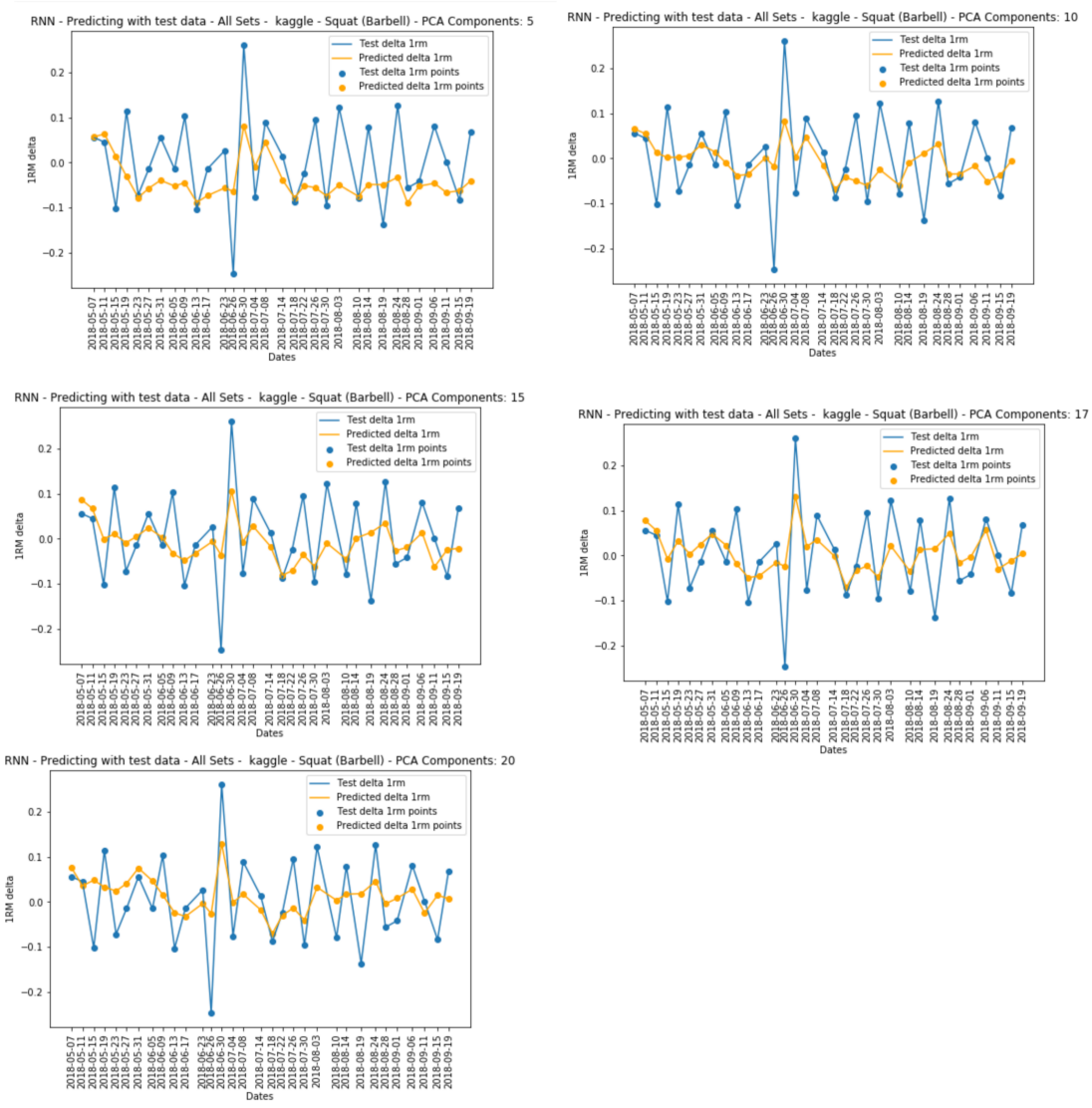


Figure 86. LSTM - predictions on PCA data set - Test Results

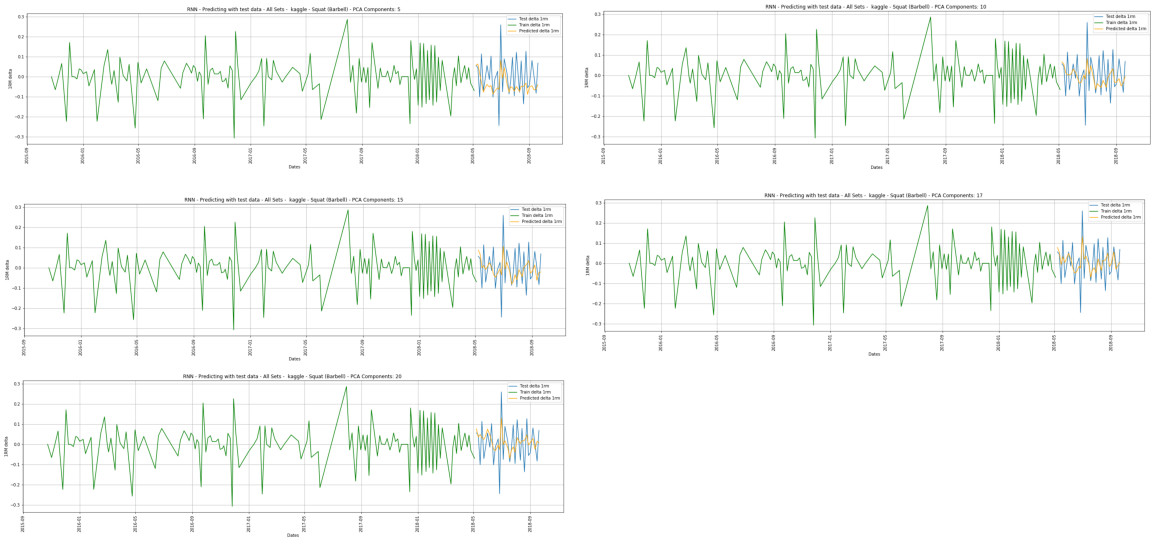


Figure 87. LSTM - predictions on PCA data set - Overview Results

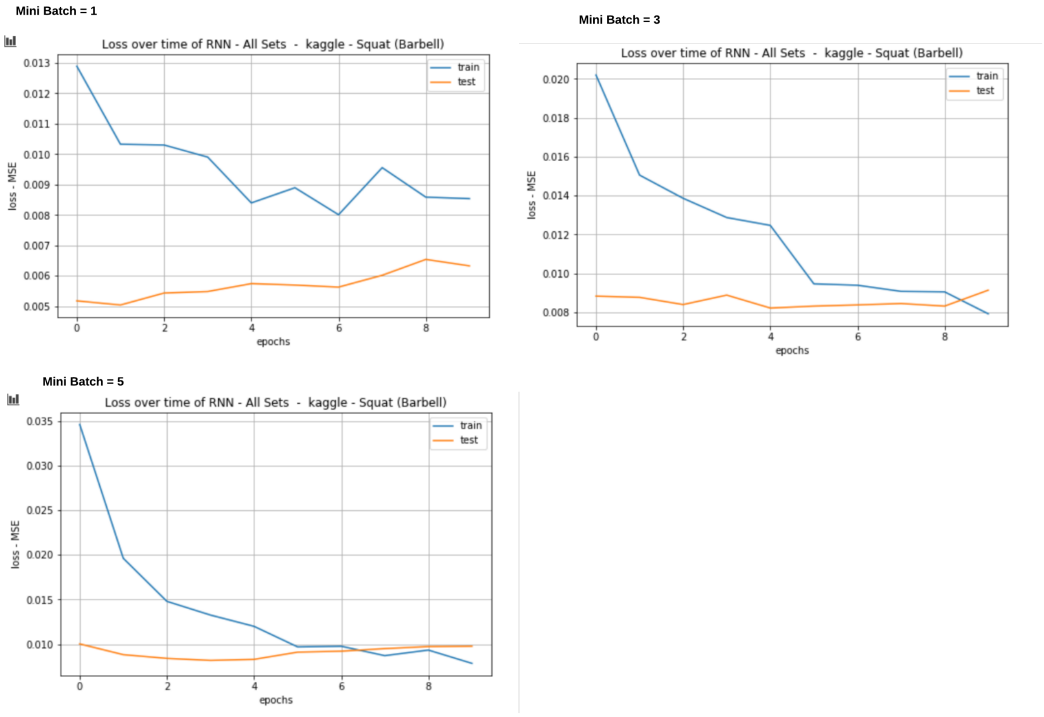


Figure 88. LSTM - predictions on RC hyper parameters - loss over time

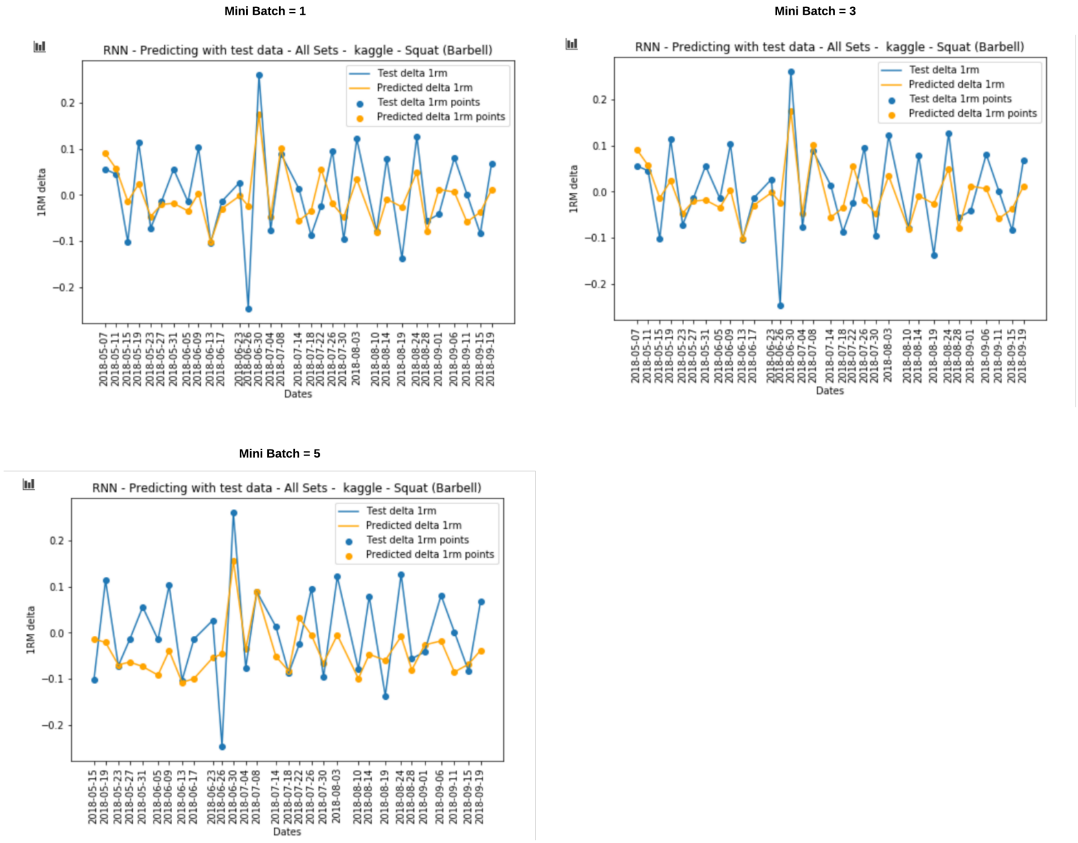
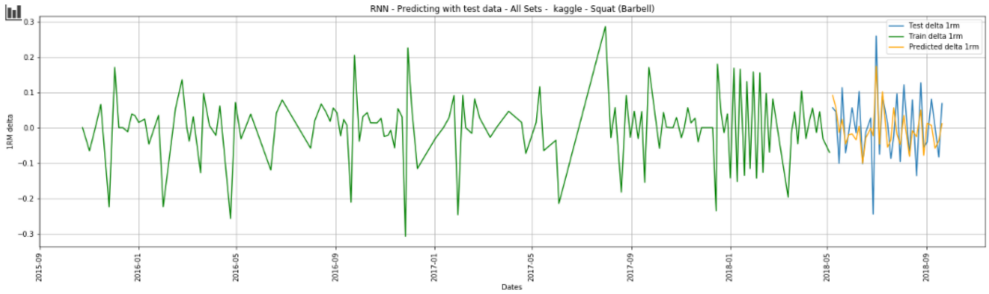
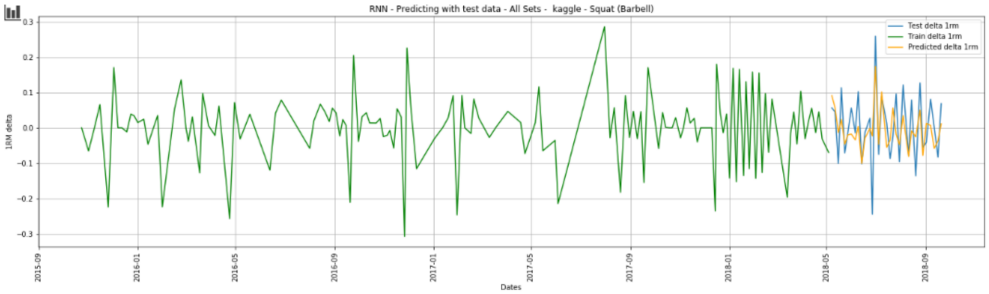


Figure 89. LSTM - predictions on RC hyper parameters - Test Results

Mini Batch = 1



Mini Batch = 3



Mini Batch = 5

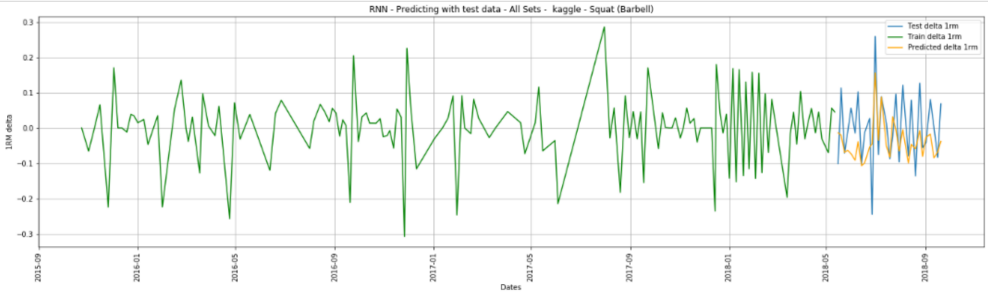


Figure 90. LSTM - predictions on RC hyper parameters - Overview Results

7.4 Gym Timing of Individuals

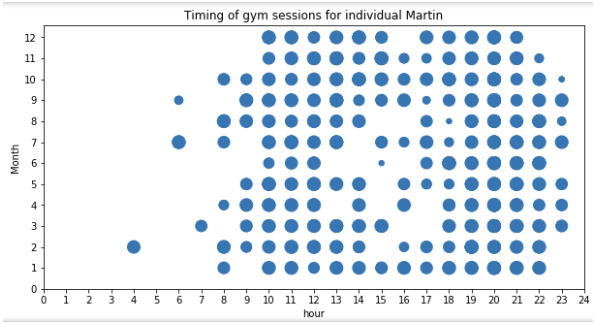


Figure 91. Timing of gym sessions for Martin

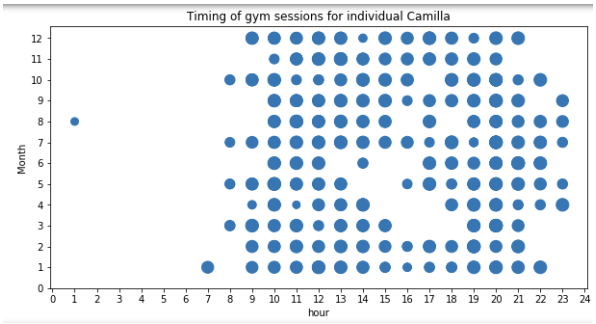


Figure 92. Timing of gym sessions for Camilla

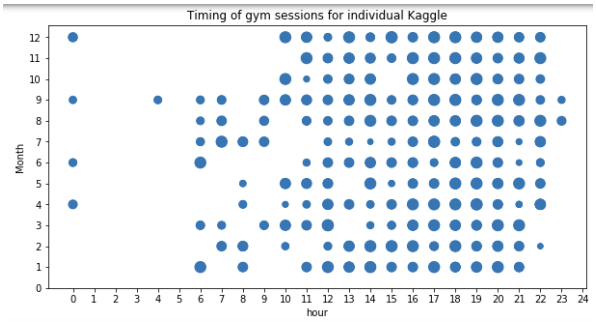


Figure 93. Timing of gym sessions for Kaggle

7.5 Scatter plot of weight and reps by gender

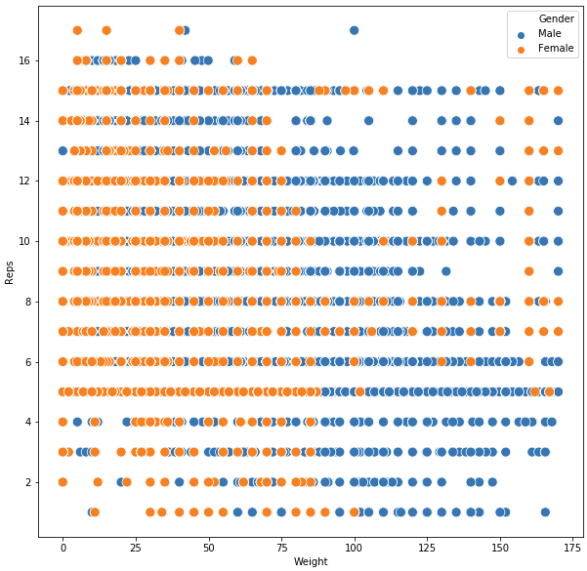


Figure 94. Scatter plot of reps and weight by genders

7.6 PCA Clustering

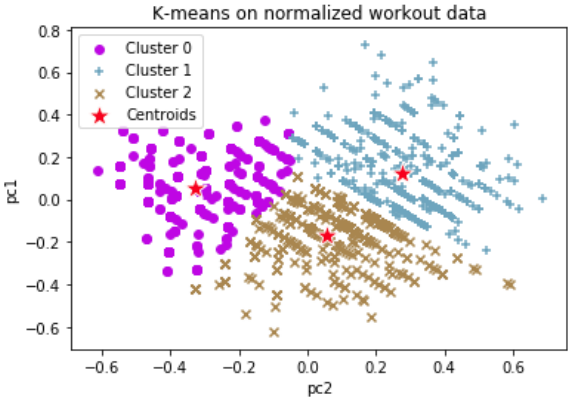


Figure 95. Clustering of three individuals based on PCA of workout data

REFERENCES

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[2] Cristopher M. Bishop *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville *Deep Learning* The MIT Press, Cambridge, Massachusetts London, England, 2016.

[4] Etham Alpaydin *Introduction to Machine Learning - Third Edition*. ISBN 978-0-262-02818-9. page 5.

[5] Etham Alpaydin *Introduction to Machine Learning - Third Edition*. ISBN 978-0-262-02818-9. page 69.

[6] Time Series Forecasting Using Recurrent Neural Network
https://www.youtube.com/watch?v=i40Road82Not=1184s

[7] Understanding the Bias-Variance Tradeoff
`http://scott.fortmann-roe.com/docs/BiasVariance.html`

[8] Knuth: Computers and Typesetting,
`http://www-cs-faculty.stanford.edu/~uno/abcde.html`

[9] Relationships of Body Dimensions to Strength Performance in Novice Adolescent Male Powerlifters, 1993,
https://journals.humankinetics.com/view/journals/pes/5/4/article-p347.xml

[10] Detraining in the Older Adult: Effects of Prior Training Intensity on Strength Retention, 2007
https://www.researchgate.net/publication/6153231_Detraining_in_the_Older_Adult_Effects_of_Prior_Training_Intensity_on_Strength_

[11] Effects of Aerobic Exercise on Strength Performance Following Various Periods of Recovery, 2003,
https://www.researchgate.net/publication/6727936_Effects_of_Aerobic_Exercise_on_Strength_Performance_Following_Various_Perio

[12] The Effect of Resistance-Training Intensity on Strength-Gain Response in the Older Adult, 2004,
https://www.researchgate.net/publication/8153729_The_Effect_of_Resistance-Training_Intensity_on_Strength-Gain_Response_in_the_Older_Adult

[13] The Effects of Varying Resistance-Training Loads on Intermediate– and High–Velocity-Specific Adaptations, 2001,
https://www.researchgate.net/publication/11643870_The_Effects_of_Varying_Resistance-Training_Loads_on_Intermediate-_and_High-Velocity-Specific_Adaptations

[14] Influence of Range of Motion in Resistance Training in Women: Early Phase Adaptations, 2005,
https://www.researchgate.net/publication/7839010_Influence_of_Range_of_Motion_in_Resistance_Training_in_Women_Early_Phase_

[15] A Comparison of Strength and Power Characteristics Between Power Lifters, Olympic Lifters, and Sprinters, 1999,
https://journals.lww.com/nsca-jscr/Abstract/1999/02000/A_Comparison_of_Strength_and_Power_Characteristics.11.aspx

[16] Accuracy of Prediction Equations for Determining One Repetition Maximum Bench Press in Women Before and After Resistance Training, 2008,
https://www.researchgate.net/publication/23181491_Accuracy_of_Prediction_Equations_for_Determining_One_Repetition_Maximun

[17] One-repetition maximum
https://en.wikipedia.org/wiki/One-repetition_maximum

[18] Relationships of Body Dimensions to Strength Performance in Novice Adolescent Male Powerlifters, 1993,
https://journals.humankinetics.com/view/journals/pes/5/4/article-p347.xml

[19] The effect of training volume and intensity on improvements in muscular strength and size, 2015,
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4562558/

[20] The accuracy of prediction equations for estimating 1-rm performance in the Bench Press, Squat, and Deadlift ,1997,
https://journals.lww.com/nsca-jscr/Abstract/1997/11000/The_Accuracy_of_Prediction_Equations_for.1.aspx

[21] The effectiveness of Prilepin's chart for powerlifting strength improvements, 2016,
https://www.researchgate.net/publication/304540647_The_effectiveness_of_Prilepin%27s_chart_for_powerlifting_strength_improvem

[22] Strength training
https://en.wikipedia.org/wiki/Strength_training

[23] Strong app
https://www.strong.app/

[24] The Data Science Lifecycle
https://www.textbook.ds100.org/ch/01/lifecycle_intro.html

[25] What is Machine Learning?
https://machinelearningmastery.com/what-is-machine-learning/

[26] Data cleansing
https://en.wikipedia.org/wiki/Data_cleansing
[27] The 5 Classification Evaluation metrics every Data Scientist must know
https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226
[28] How to select the Right Evaluation Metric for Machine Learning Models: Part 1 Regression Metrics
https://towardsdatascience.com/how-to-select-the-right-evaluation-metric-for-machine-learning-models-part-1-regrression-metrics-3606e25beae0
[29] Why and how to cross validate a model.
https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f
[30] Time series nested cross validation
https://towardsdatascience.com/time-series-nested-cross-validation-76adba623eb9
[31] Total Variance Explained - IBM
https://www.ibm.com/support/knowledgecenter/SSLVMB_26.0.0/statistics_casestudies_project_ddita/spss/tutorials/fac_cars_tve.htm
[32] Time Series Forecasting as Supervised Learning
https://machinelearningmastery.com/time-series-forecasting-supervised-learning/
[33] Autoregression Models for Time Series Forecasting
https://machinelearningmastery.com/autoregression-models-time-series-forecasting-python/
[34] Vector autoregression
https://en.wikipedia.org/wiki/Vector_autoregression
[35] Hyperparameter-tuning
https://www.oreilly.com/ideas/evaluating-machine-learning-models/page/5/hyperparameter-tuning
[36] How LSTM networks solve the problem of vanishing gradients
https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577