Understanding offensive football with machine learning

Frederik Rohmann & Lukas Wolf

Data overview

- Data from all matches the Danish Superliga season 22/23
- Data was given to us by NBI postdoc Mathias Heltberg, FCN by proxy
- Positions and speed for all players and ball of football teams during matches
- ‘xG’ data from Opta, ie. **chances**
- Key word: **Sequential data**

Project overview

- Motivation: Why machine learning?

    - Match data has high dimensionality, and may display complex patterns

    - In data collection, distinction between different scenarios is already desired

- Fundamental assumption: Individuals are unimportant. We can understand segments of play by looking at patterns.

- **Classification of play:** Does a situation involve a chance of goal or not? Also, timescale of chances (RNNs – LSTM & GRU)

- **Classification of chances:** Will a chance result in a goal or not? (RNNs – LSTM & GRU)

- **Clustering chances:** Can we distinguish between different types of chances?(Time-Series KMeans)

Classifying play: chance or not?

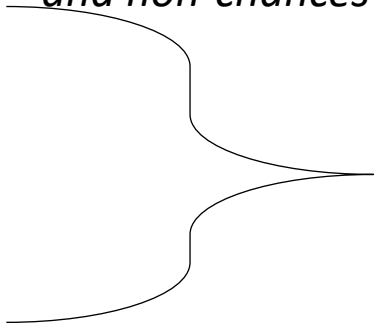
Data preparation and preprocessing

- From entire gasmes to moments

Game 1	
Time step 1	Ball and player information
.	…
.	…

Game 2	
Time step 1	Ball and player information
.	…
.	…

Extract moments around chances and non-chances*



Information about all moments	
Time series #1	Ball and player information at all time steps
.	…
.	…

**Select a time frame around the chances. This can be rather challenges, as the time stamps provided by Opta do not always line up with what we see in the raw data*

Preprocessing

- Cleaning – which time series do we want? **Data with all players present**

- What exactly are the features?

Players are identified by their **names** and **numbers –** But this is not meaningful in our analysis.

We need a way to compare player features **across moments**

Information about all moments, before further preprocessing			
Time series #N	Player #4 info	Player #6 info	Player #15 info
Time series #N+1	Player #1 info	Player #4 info	Player #15 info
…	…		

When looking for patterns in the football data, players do not necessarily have anything to do with one another, even if they share the same number

Information about all moments, before further preprocessing			
Time series #N	Player #4 info	Player #6 info	Player #15 info
Time series #N+1	Player #1 info	Player #4 info	Player #15 info
…	…		

Solution: Calculate average distance between ball and player in time series and rank them

Information about all moments, after preprocessing			
Time series #N	Closest player info	2nd closest player info	3rd closest player info
Time series #N+1	Closest player info	2nd closest player info	2nd closest player info
…	…		

Preprocessing continued – dimensionality reduction

- Data has initially dimensionality **117 –** What could we do?

First filter:	**Remove unimportant non-gameplay information: Player numbers and ball possession. Explicit time.**

$117 \rightarrow 92$

Preprocessing continued – dimensionality reduction

- Data has initially dimensionality **117 –** What could we do?

First filter:	**Remove unimportant non-gameplay information: Player numbers and ball possession. Explicit time.**	$117 \rightarrow 92$
Second filter:	**Remove presumably unimportant gameplay information: z-coordinate, (velocity)**	$92 \rightarrow 70\ (46)$

Preprocessing continued – dimensionality reduction

- Data has initially dimensionality **117 –** What could we do?

First filter:	**Remove unimportant non-gameplay information: Player numbers and ball possession. Explicit time.**	$117 \rightarrow 92$
Second filter:	**Remove presumably unimportant gameplay information: z-coordinate, (velocity)**	$92 \rightarrow 70\ (46)$
Third filter:	**Remove players further away from ball: Riskier – these players could still be important?**	$46 \rightarrow 14$

Preprocessing continued – dimensionality reduction

- Data has initially dimensionality **117 –** What could we do?

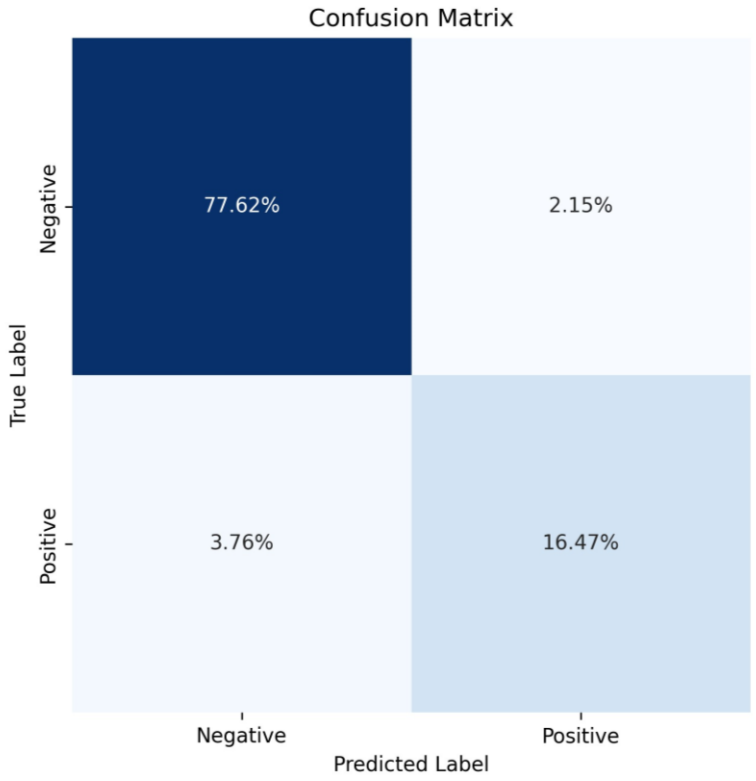
First filter:	**Remove unimportant non-gameplay information: Player numbers and ball possession. Explicit time.**	$117 \rightarrow 92$
Second filter:	**Remove presumably unimportant gameplay information: z-coordinate, (velocity)**	$92 \rightarrow 70\ (46)$
Third filter:	**Remove players further away from ball: Riskier – these players could still be important?**	$46 \rightarrow 14$
Joker:	**Remove ALL players! The dynamics of the ball is good enough.** **Remove ball!**	$14 \rightarrow 2$ Depends on choices in third filter

Machine learning methods

- LSTM or GRU (winner) But VERY similar performance. Binary cross entropy.
- We tried to obtain the best possible model in the following ways:
    - Reduce complexity of the data
    - Scale data properly
    - Hyperparameter optimization
    - Validation
    - Use dropout layers

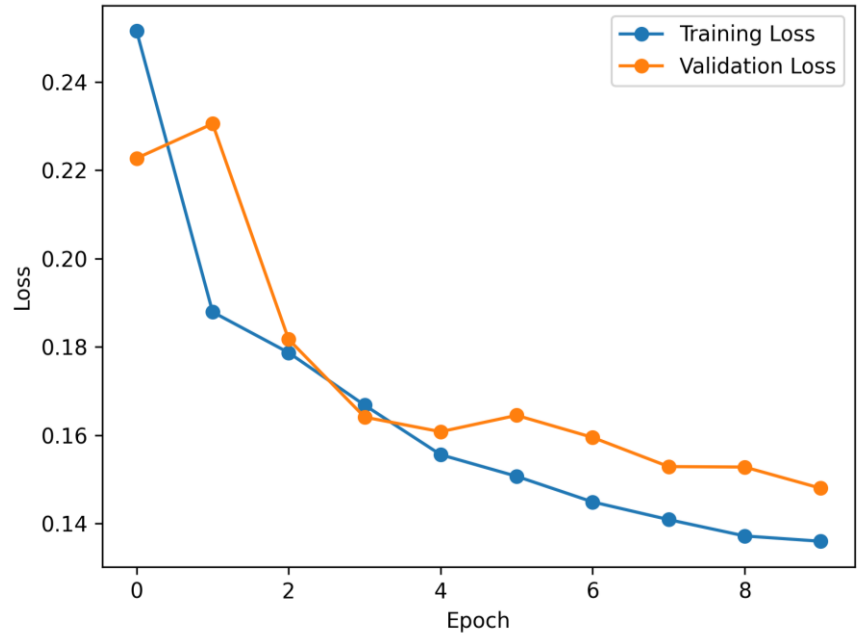
Results – 11 s leading up to chance

Results with all players



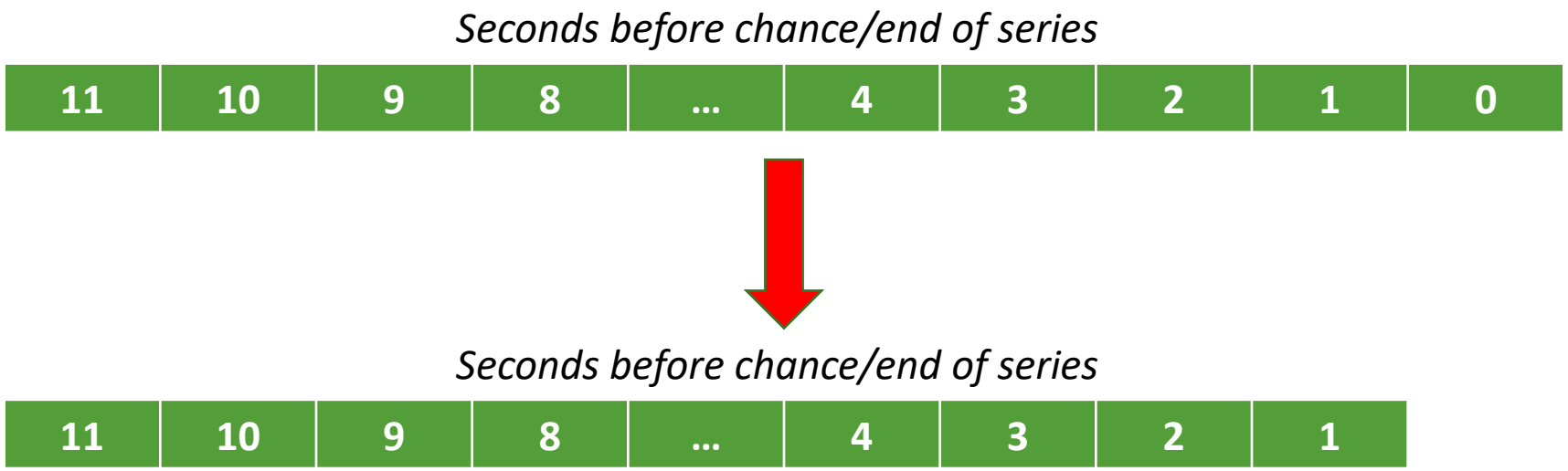
GRU (2 hidden layers, 32, 20 neurons)	
Results on unseen data	
Accuracy	95.24 %
ROC AUC-score	97.74%
F1 Score	87.99%

Epochs: 10, Activation: All sigmoid. Learning rate: 0.1



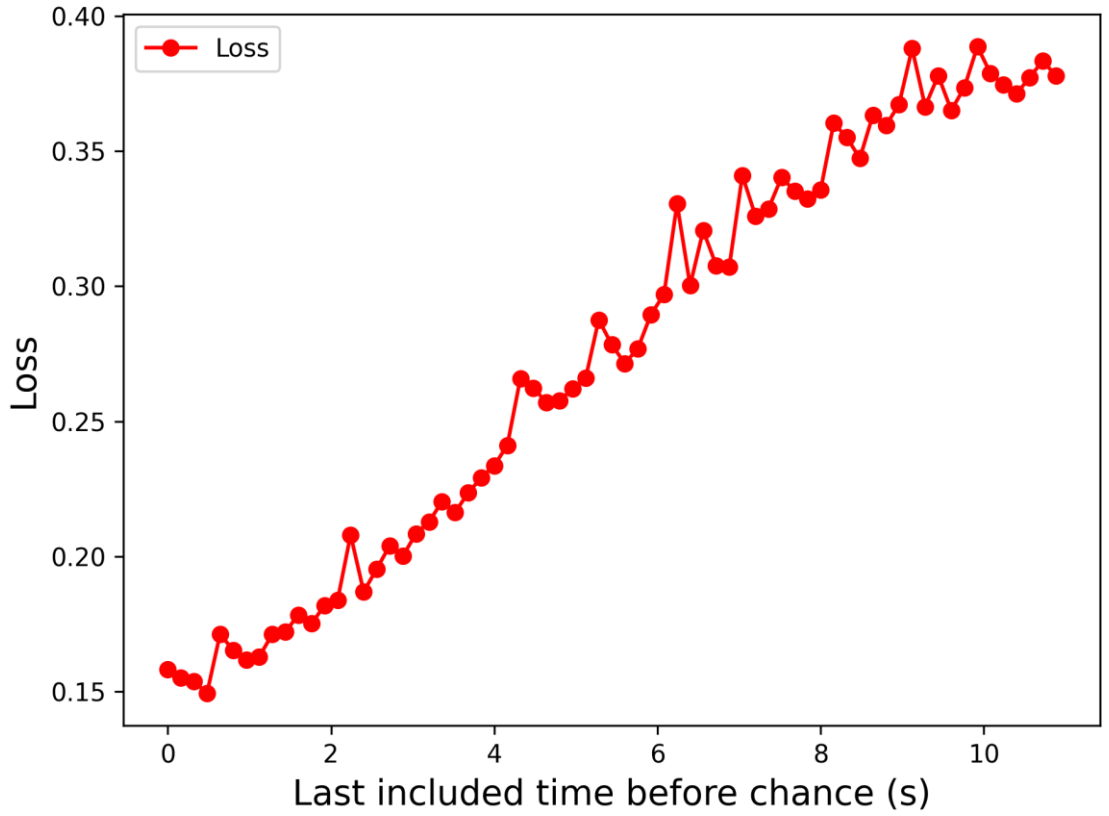
Looking into timescales

- How quickly do we lose predictive power?
- Remove more and more data close to chance



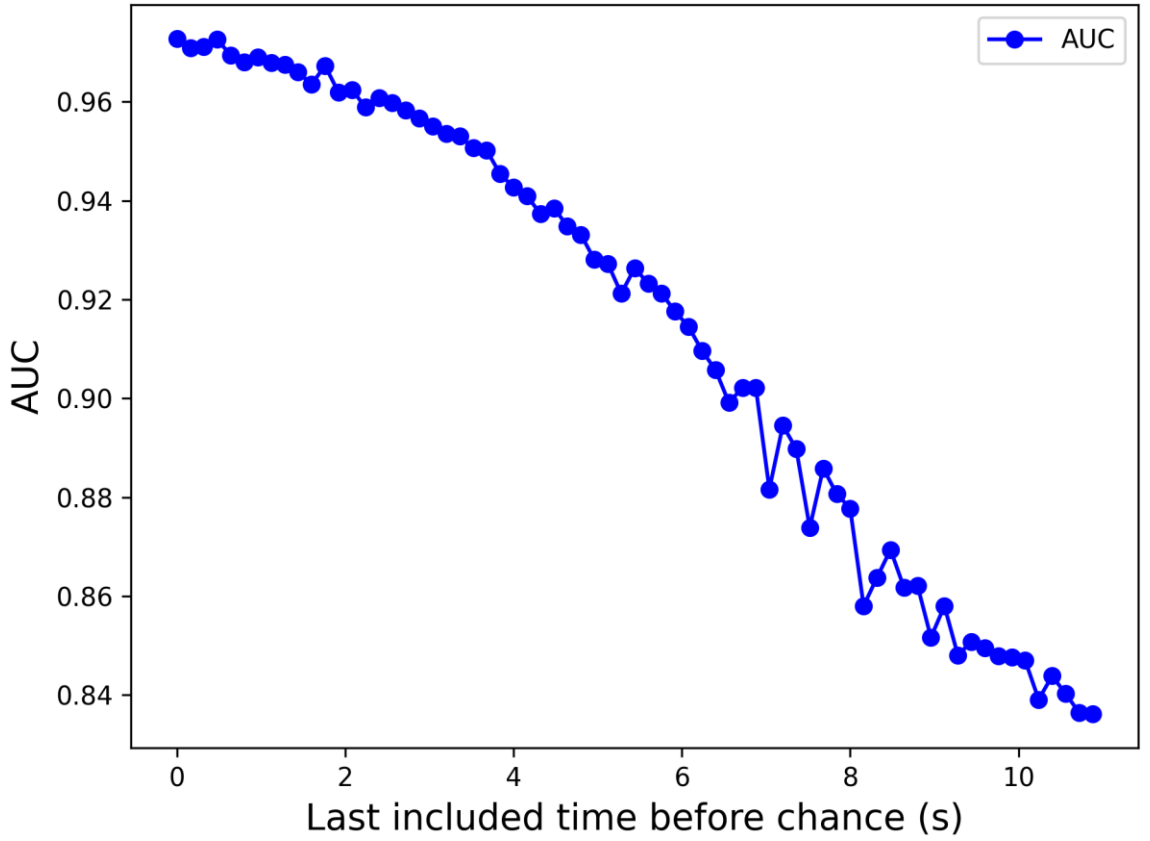
Results – Timescales

Results with all players, no velocity or z-coordinate



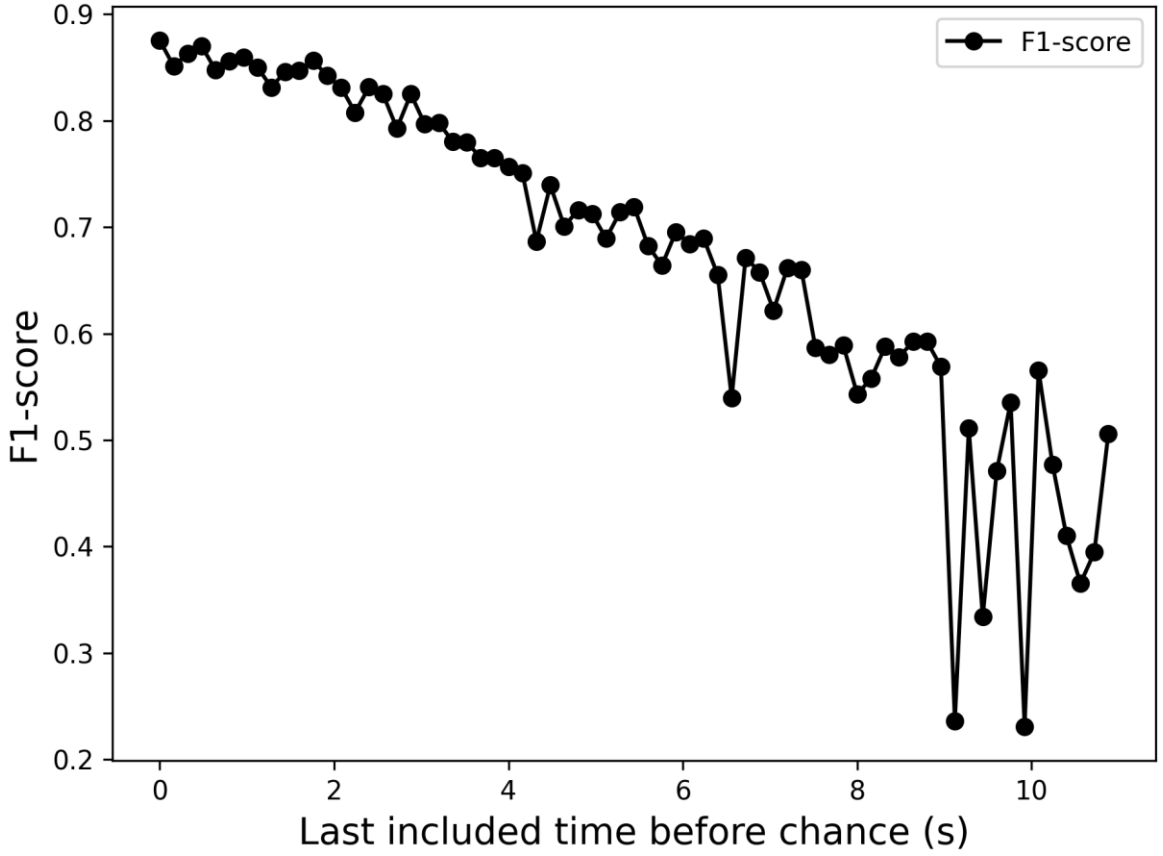
Results – Timescales

Results with all players, no velocity or z-coordinate



Results – Timescales

Results with all players, no velocity or z-coordinate



Classifying chances: goal or not?

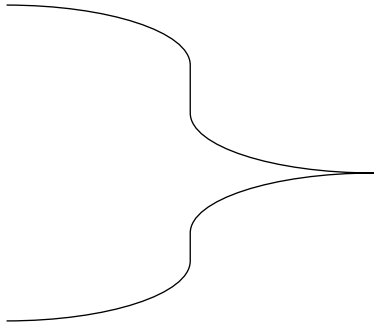
Data preparation and preprocessing

- From entire games to chances

Game 1	
Time step 1	Ball and player information
.	…
.	…

Game 2	
Time step 1	Ball and player information
.	…
.	…

Extract data around chances



Information about all chances	
Time series chance #1	Ball and player information at all time steps
.	…
.	…

Overview

- Challenges:

  -The amount of data is small compared to the complexity- **Overfitting is very hard to avoid.**

  - Imbalanced class distribution: **Goals are rare – this is a property of the problem domain.**

Data after sorting	
Chances:	3492
Goals:	327

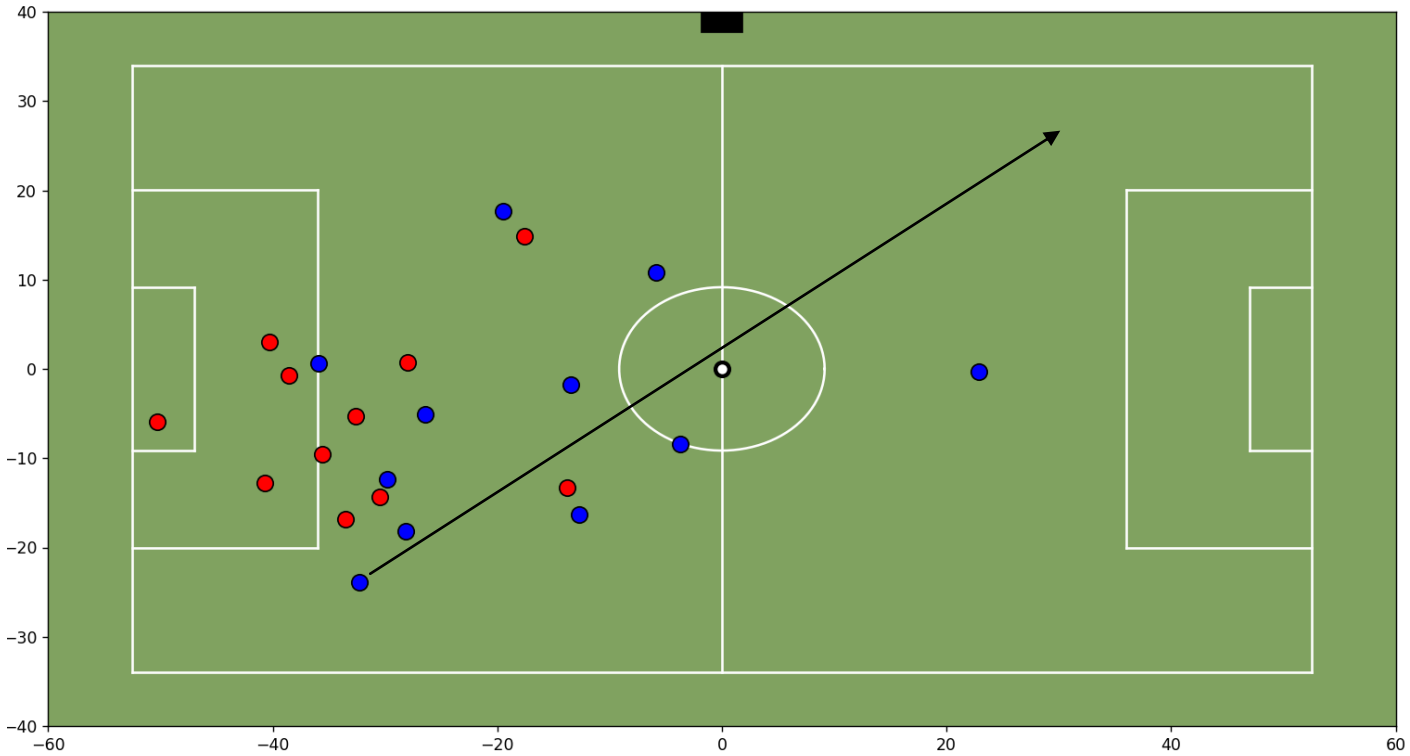
Data after typical train/test split	
Chances, Training	2793
Goals, Training	274
Chances, Test:	699
Goals, Test:	53

- We tried to obtain the best possible model in the following ways:

  - Reduce complexity of the data

  - K-fold cross validation

Reducing complexity



Football pitch image credit: Mathias Heltberg

- Include fewer players
- Exclude dead balls
- ”Reflect” data, so all chances happen on right side of the pitch

“Best” results

GRU (1 hidden layer, 20 neurons)	
Best result on unseen data	
Accuracy	90.97 %
ROC AUC-score	60.54%
F1 Score	**0.0%**

Epochs: 5, Activation: All sigmoid. Learning rate: 0.07

Conclusion: Based on our data, we are not able to decide which chances result in goals

Clustering chances

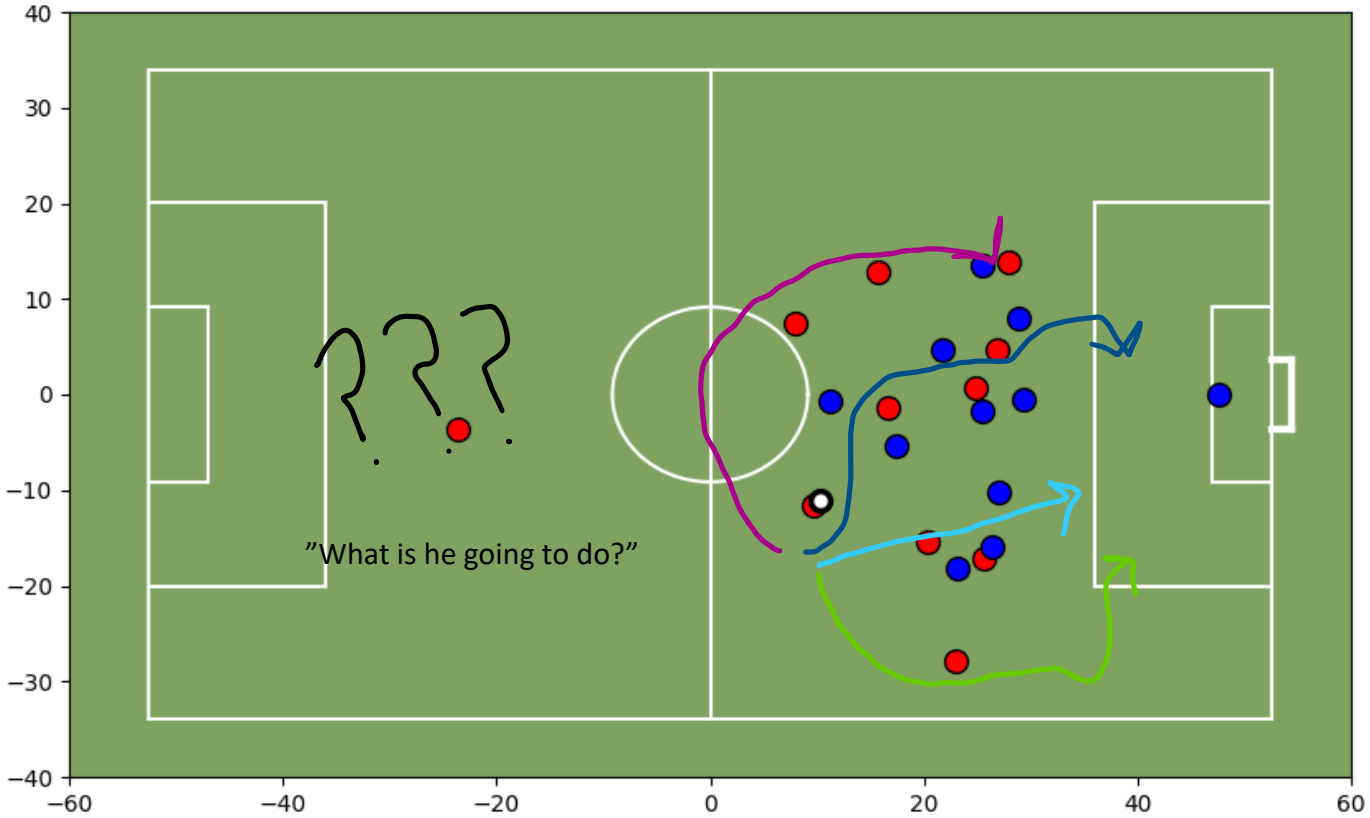
Data selection and preprocessing

- Reduce complexity: Fewer players, field symmetry.
- Choice of complexity:
  - Include dead balls? **YES!**
  - Include sequences with Red Cards? **MAYBE?**
- Easiest feature(s) to understand in time
  - **xy-position**
  - Speed? (Implicit)
  - z-position (Zero)

Overview:

Goal: Categorize attack strategies:

- How many categories?
- How effective are they?



Football pitch image credit: Mathias Heltberg

Step 1: PCA

- Reduce multi-dimensional data to 1D
- Time-sequence, one dimension ”locked”*

4 different ”reduced features”:

- Ball
- XGPlayer
- TeamA
- OppTeam

*For easy temporal understanding

Step 1.5: PCA (Continued)

x- and y-position chosen for PCA.

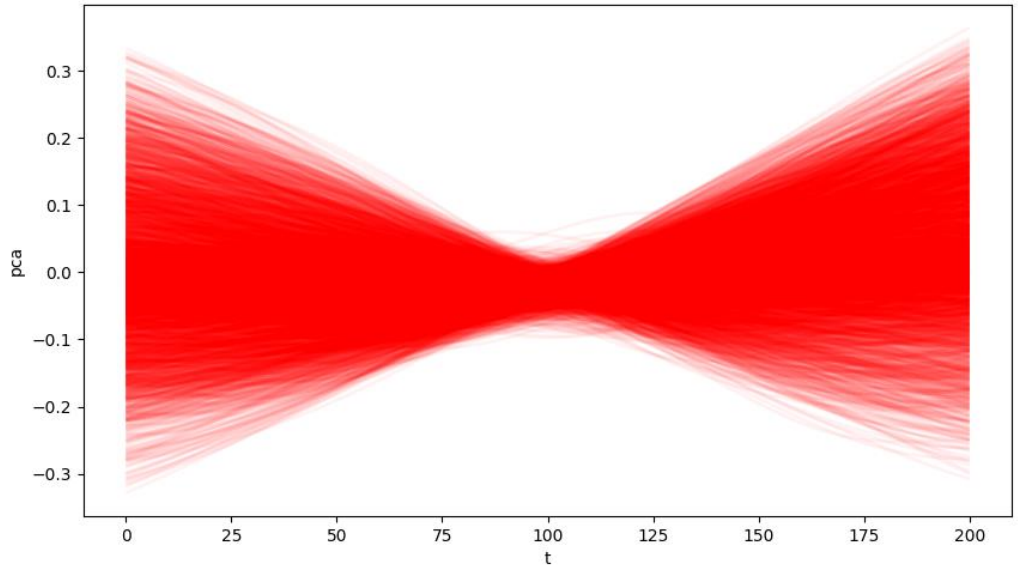
Challenge:

5x TeamA & 5x TeamOpp = Consequence for PCA?

How well described is the new dimension?

Step 1.5.5: PCA Results

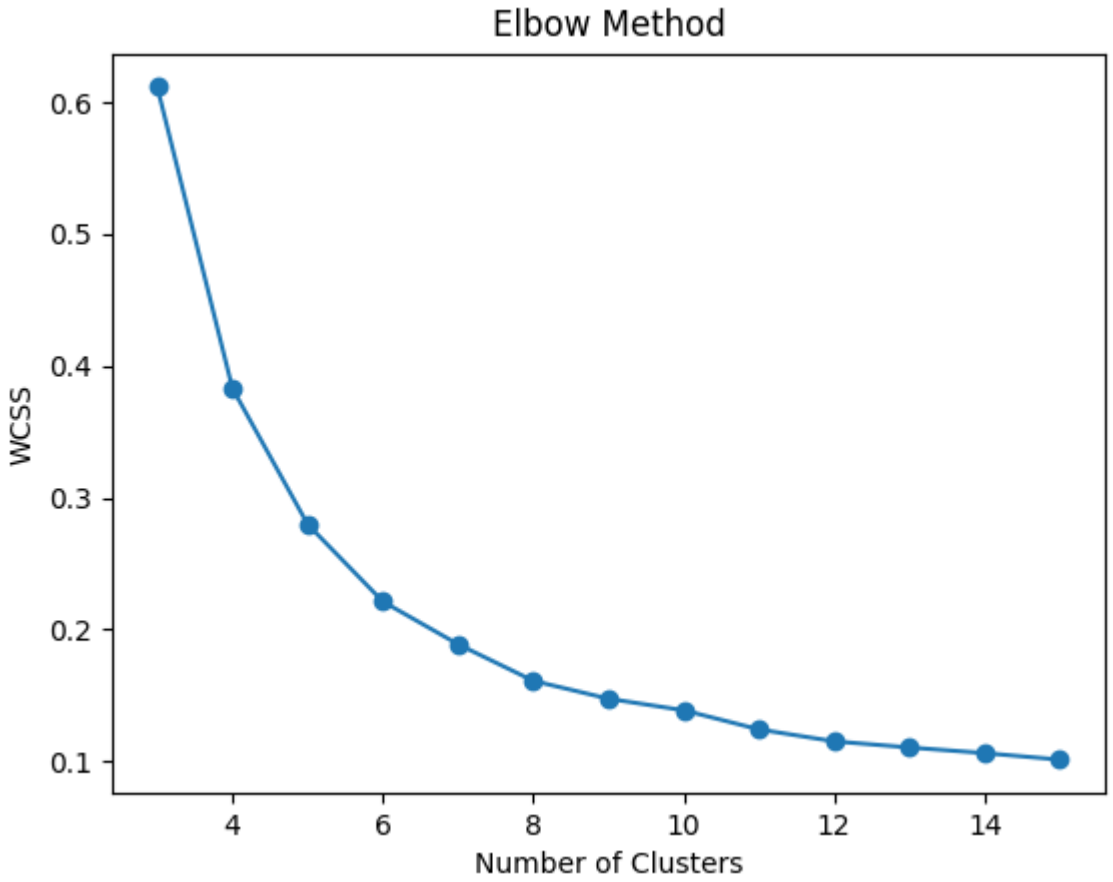
Hyper-feature:	Explained Variance Ratio (Explained)
Ball	94.15%
XGPlayer	99.32%
TeamA	95.22%
TeamOpp	98.97%



Example plot: Distribution of time-series PCA (XGPlayer)

Step 2: Find N_Clusters

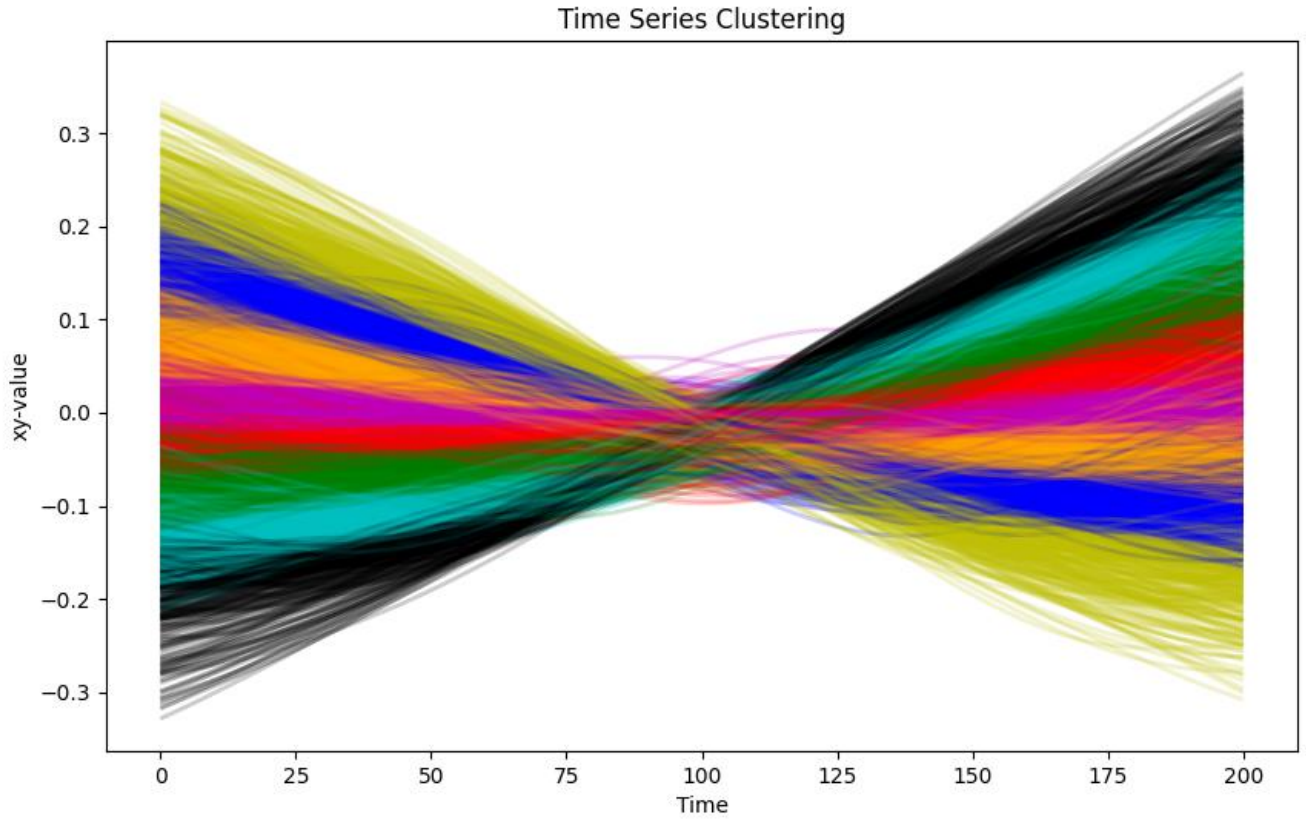
- Use Elbow method to find N_clusters:



Example plot: Elbow method of TeamA

Challenge:
- With no ”elbow”, what choice of N_clusters?
- What metric? Euclidean? DTW?

Step 3: Plot clusters

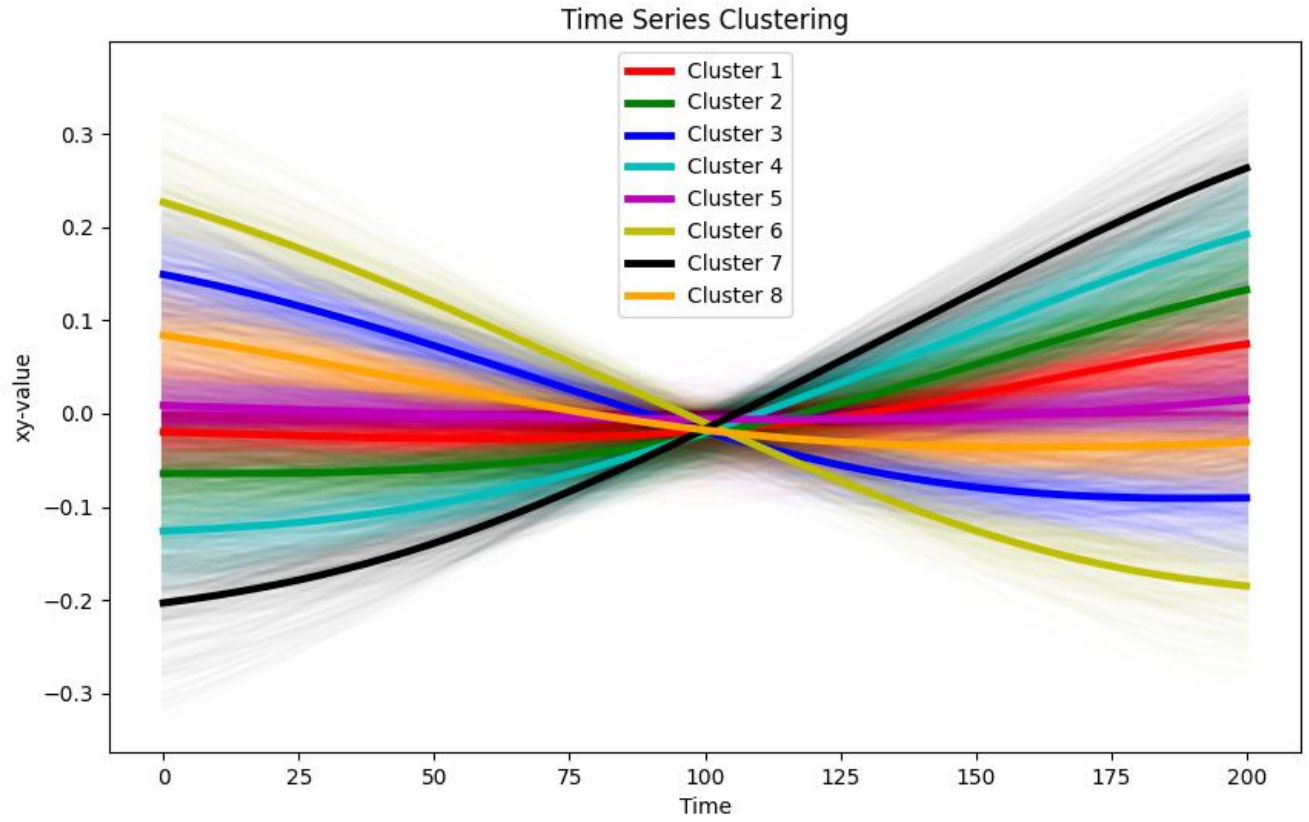


Example plot: Clustering of PCA for XGPlayer

It would appear that there are systematic strategies for XGPlayer.
The Silhoutte score for each ”reduced-Feature” is:

Hyper-Feature	Silhoutte Score
Ball	0.22
XGPlayer	0.34
TeamA	0.39
TeamOpp	0.40

Step 3.5: Plot clusters with mean



The Mean of each cluster, allows us to generalize the strategy/movement of XGPlayer.

Example plot: Clustering of PCA, as well as mean of PCA, for XGPlayer

Step 4: Prob. of goal

Take the mean of goals scored from all time-series in each cluster:

XGPlayer:

Cluster #	Prob. of goal (Mean of N_goals)
1	8.64%
2	8.18%
3	8.82%
4	10.75%
5	11.66%
6	10.80%
7	**18.46% <- WINNER! (-ish)**
8	7.84%

Important note: Approx. 10% of ALL time-series end in goals

Conclusion

- We were able to conclude if a section of gameplay included a chance or not with high precision. Looking further back in time from the chance showed steady drop-off in f1-score
- We were able to observe drop-off in model performance, when model was given less information
- We were not able to predict whether a chance would result in a goal. Reason: Too little data, and suspicion of too much ‘randomness’
- We were able to make reasonable distinct clusters, and find that some of these included more goals than others

Contributions

- Frederik and Lukas made equal contributions

Special acknowledgement and disclosure: *Big thanks to Postdoc Mathias Heltberg for supplying data, scripts converting data to .pkl files and scripts for reading raw data. Also, images of football pitch were made by Mathias Heltberg, as explicitly stated in presentation.*

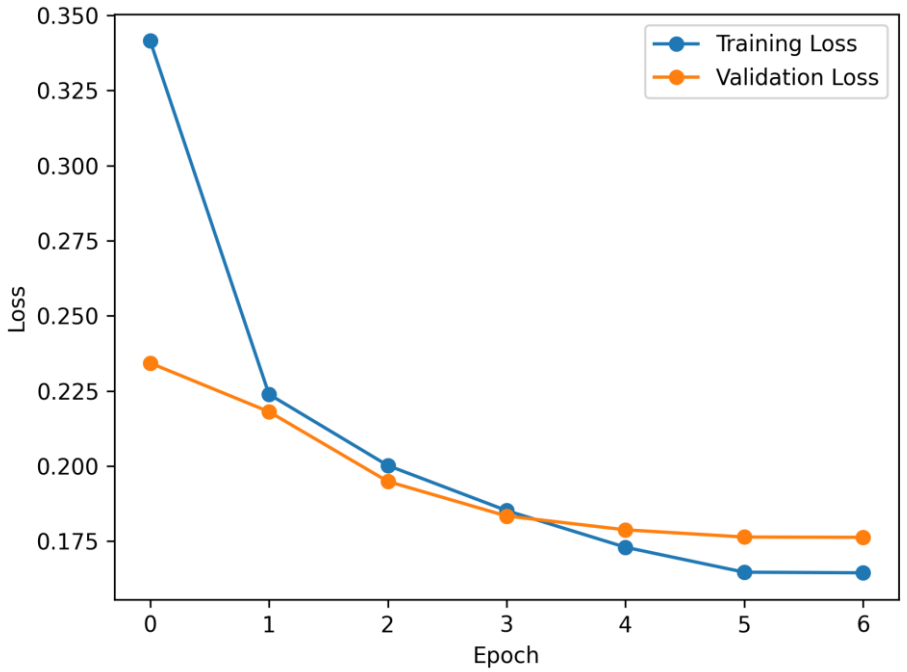
All machine learning, data extraction and data manipulation explained in this presentation as well as in the Appendix was done by Frederik and Lukas. For the purposes of this project, we start out with just the complete data from matches and XGdata, not any work done on the data.

APPENDIX

Appendix I: Chance or not

Comments on RNN architecture

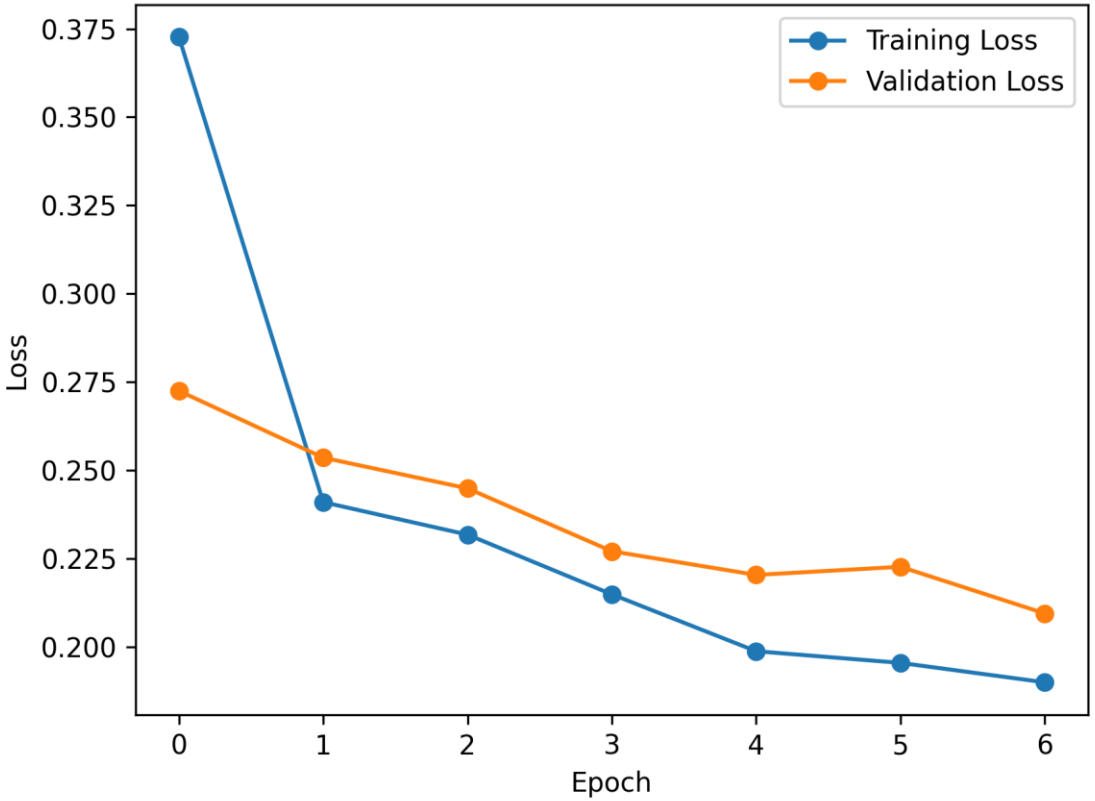
- The models generally require relatively low complexity, but Example shown below for 2 hidden layers (20,16), 7 epochs, shows higher loss, but still very good results. Here shown for all players. Performance was good enough so that in time scale analysis, we used this architecture with dropout to avoid overtraining on data with less information.
- Training time is basically identical, so choice comes down to the slight improvement in choice
- HP optimization with grid search, as more advanced methods we not really needed. We adjusted by trial and error



GRU (2 hidden layers, 20, 16 neurons)	
Results on unseen data	
Accuracy	94.34 %
ROC AUC-score	96.26%
F1 Score	85.99%

Results – 11 s leading up to chance

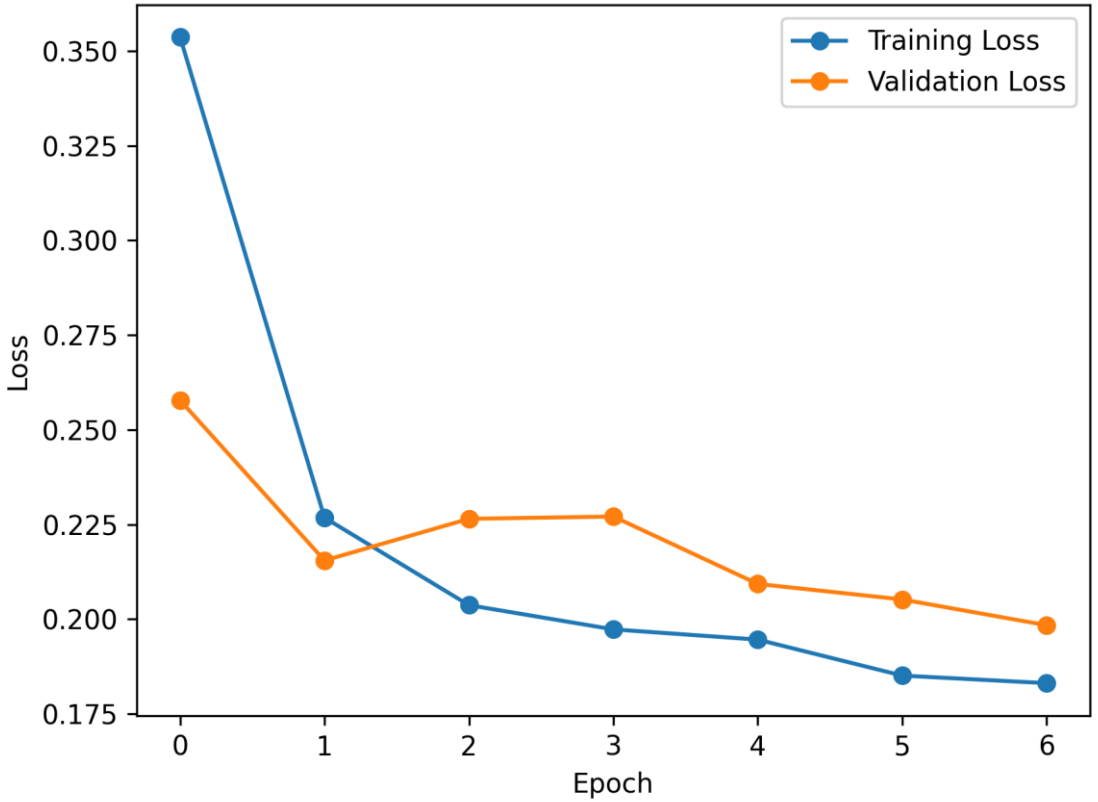
Results with only ball



GRU (2 hidden layers, 20, 16 neurons)	
Results on unseen data	
Accuracy	92.69 %
ROC AUC-score	96.37%
F1 Score	82.15%

Results – 11 s leading up to chance

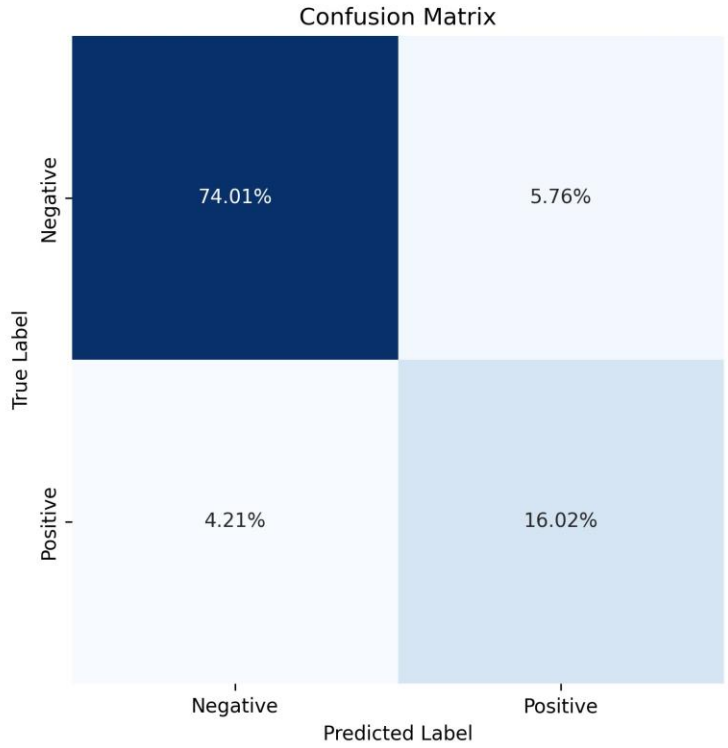
Results with only players



GRU (2 hidden layers, 20, 16 neurons)	
Results on unseen data	
Accuracy	93.44 %
ROC AUC-score	96.56%
F1 Score	84.27%

Conclusion: This problem is solved with high accuracy, and it responds very well to dimensionality reduction.

Showcase of results for different feature selection



Example of confusion matrix. 1 player only, lowest complexity of the problem.

```
Accuracy Score: 95.24%
F1 Score: 87.99%
ROC AUC Score: 97.74%
Accuracy Score: 95.24%
```

Only filter 1 applied Epochs:7, activation: sigmoid, lr: 0.1

```
Accuracy Score: 95.29%
F1 Score: 87.66%
ROC AUC Score: 97.67%
Accuracy Score: 95.29%
```

No velocity or z Epochs:7, activation: sigmoid, lr: 0.1

```
Accuracy Score: 94.44%
F1 Score: 86.83%
ROC AUC Score: 97.06%
```

Only closest players. Epochs: 7, activation: sigmoid, lr: 0.1

```
Accuracy Score: 93.44%
F1 Score: 84.27%
ROC AUC Score: 96.56%
```

Only ball. Epochs: 6, activation: sigmoid, lr: 0.1

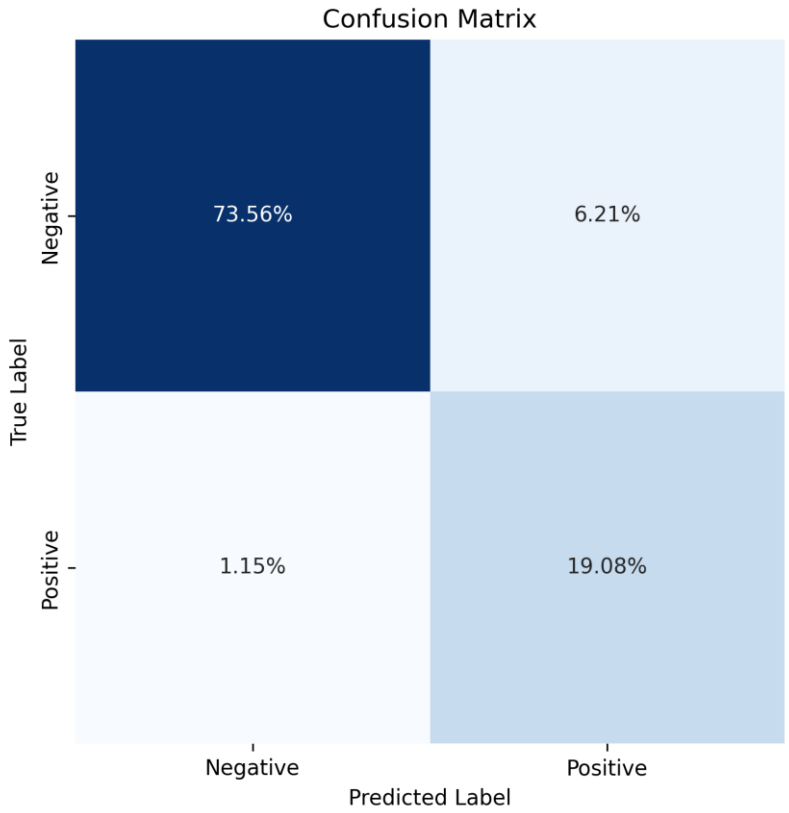
```
Accuracy Score: 89.78%
F1 Score: 75.71%
ROC AUC Score: 93.95%
```

Only players: Epochs: 5, activation: sigmoid, lr: 0.1

```
F1 Score: 76.28%
ROC AUC Score: 93.95%
Accuracy Score: 90.04%
```

Only 1 player!

- Slightly worse LGBM example - Loss: 0.1830



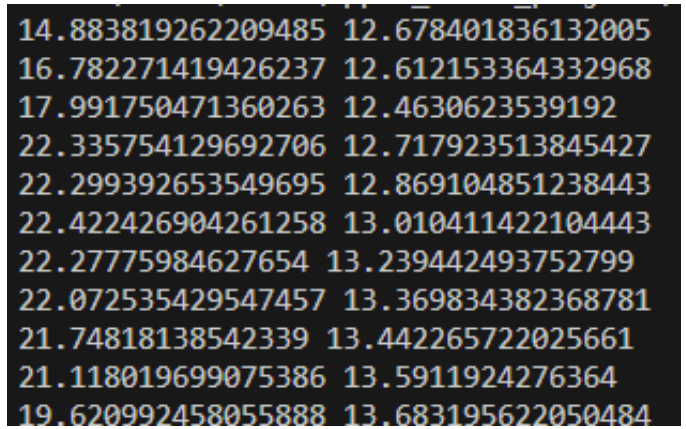

```
F1 Score: 83.83%
ROC AUC Score: 96.57%
Accuracy Score: 92.64%
```

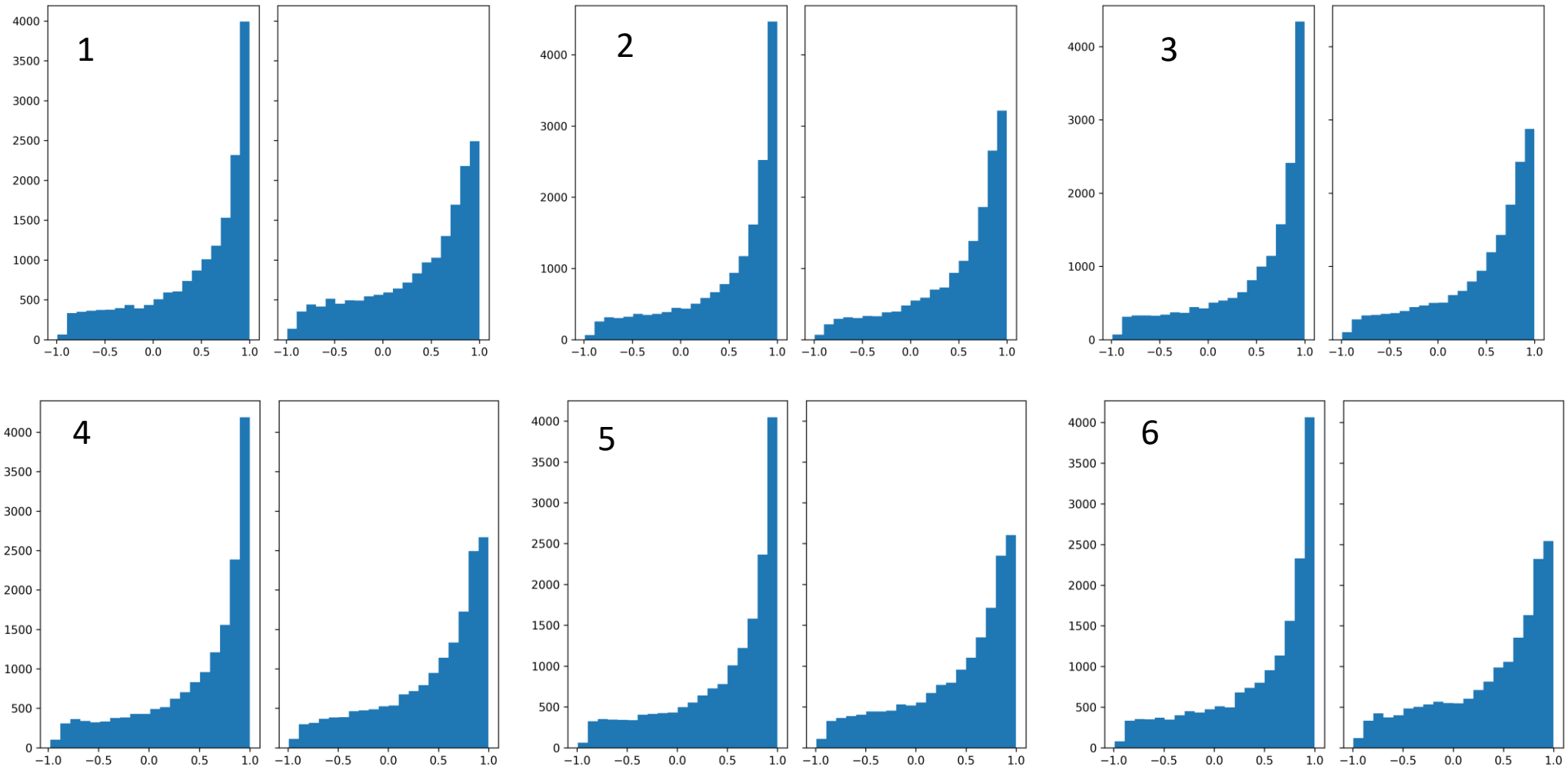
Ball-player position correlation

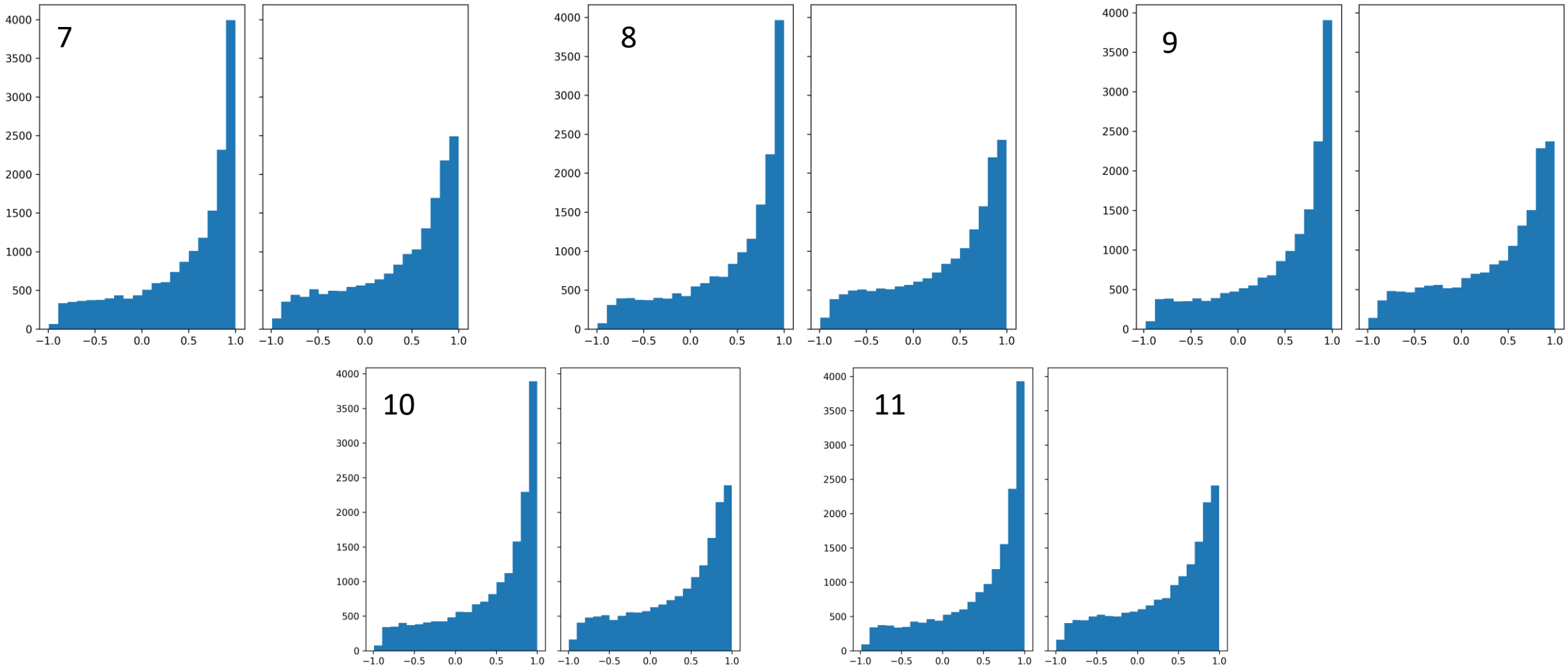
The following two slides show correlation between ball and player positions across all moments analyzed, for the team labeled “Team A”. This is to see if anything has a distribution which problematizes the choice of how we defined a feature. Correlation profiles are quite similar for all players, so it is not necessarily too enlightening. But due to more similar distances to ball, it is likely still better than alternatives

1 is closest player, 11 is player furthest away. On X-axis, we see pearson correlation. On Y-axis, we see frequency

Average player-ball distances across all points in time and all time series for team A. All quite similar within 1 std.

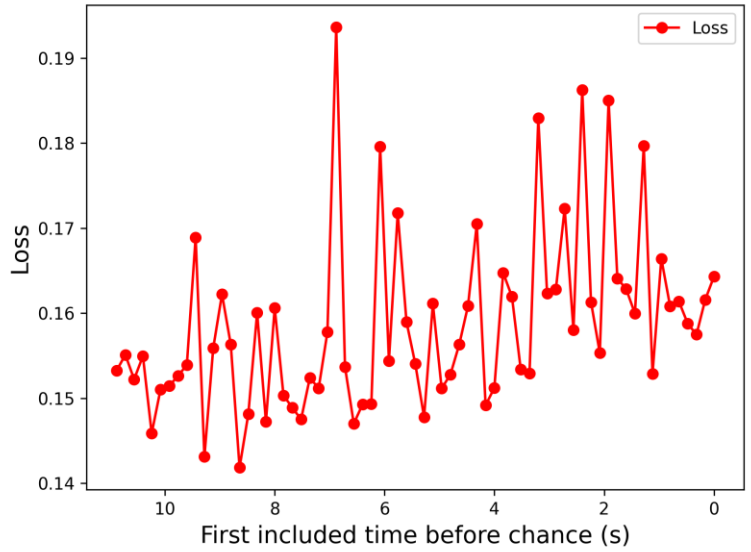


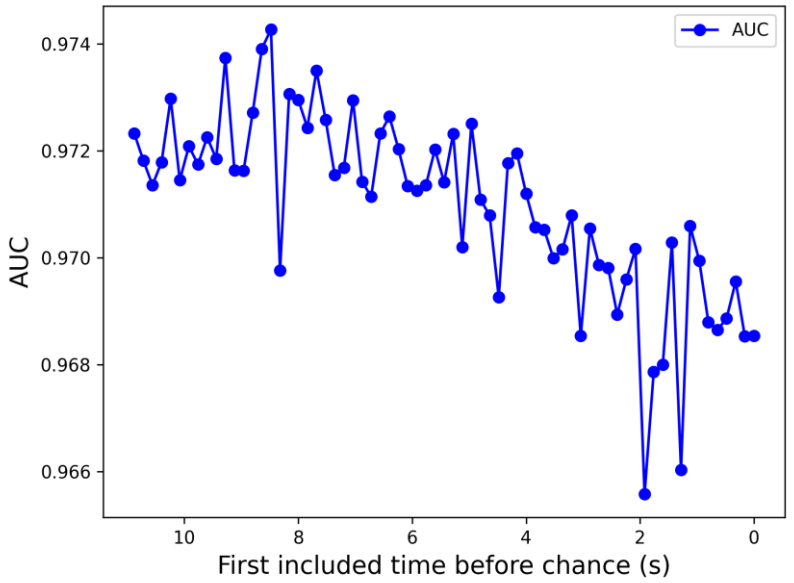

1
2
3
4
5
6

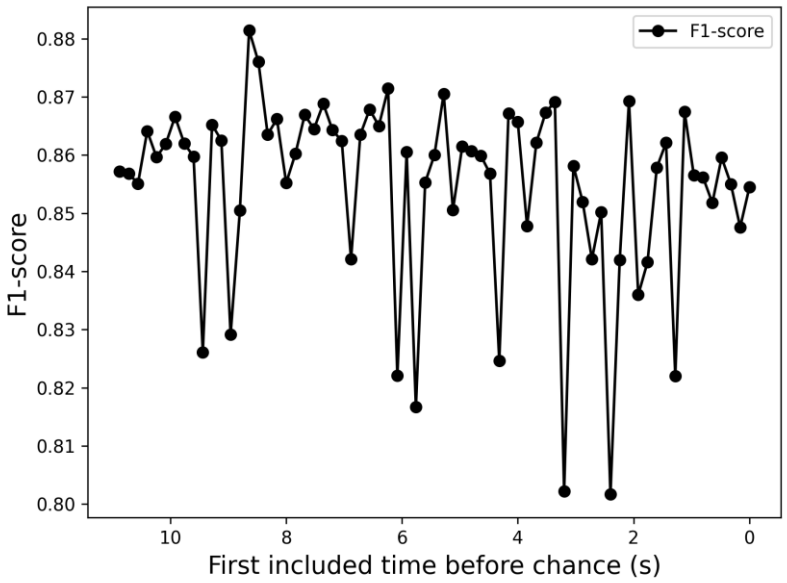

7
8
9
10
11

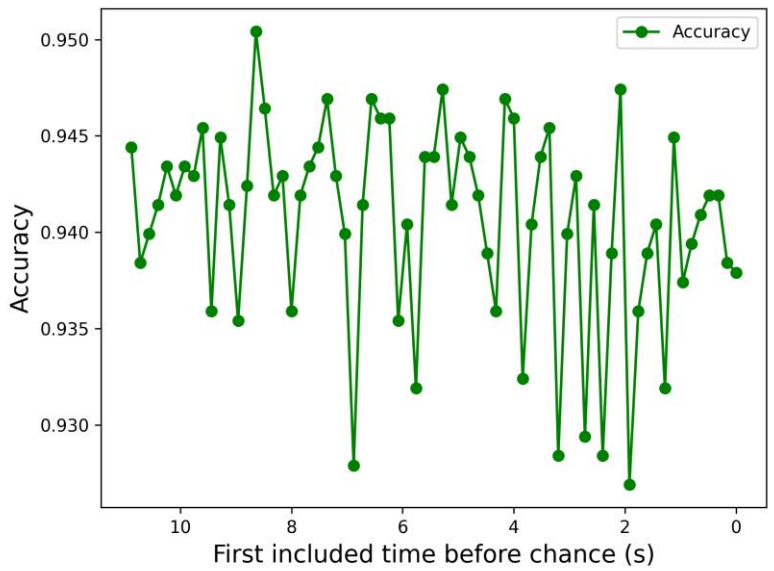
Comments on time scales

- When we looked into the time scales of loss function drop-off, we wanted to ensure that the drop-off was not just caused by the smaller quantity of information given. So we did machine learning on the same process but in reverse, removing more and more information further away from the recorded chance. Results are shown on the next slide.









Results show that performance tends to be worse when less information is included, but not to the same extent as when we moved away from the chance. Loss function used was still binary cross entropy

Appendix II: Goal or not

Performance of different models

LSTM (2 hidden layers: 30,15) Learning rate: 0.07	
Best results of on unseen data	
Accuracy	89.10 %
ROC AUC-score	60.1%
F1 Score	**0.0%**

GRU (2 hidden layers: 20,32) Learning rate: 0.07	
Best results on unseen data	
Accuracy	90.13%
ROC AUC-score	54.4%
F1 Score	**0.0%**

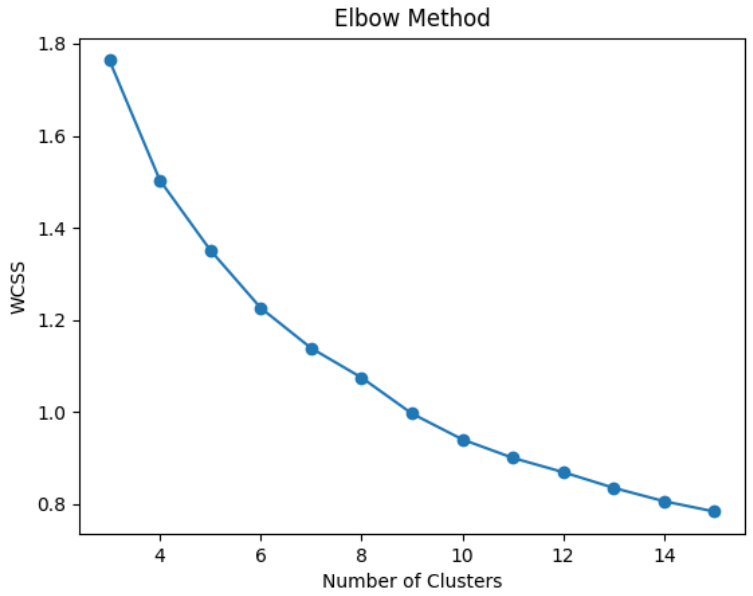
Similarly poor performances. Remember generally accuracy is not useful at all here due to the asymmetric data distribution.

Tried both MinMax scaler and Standard Scaler: Basically identical results.

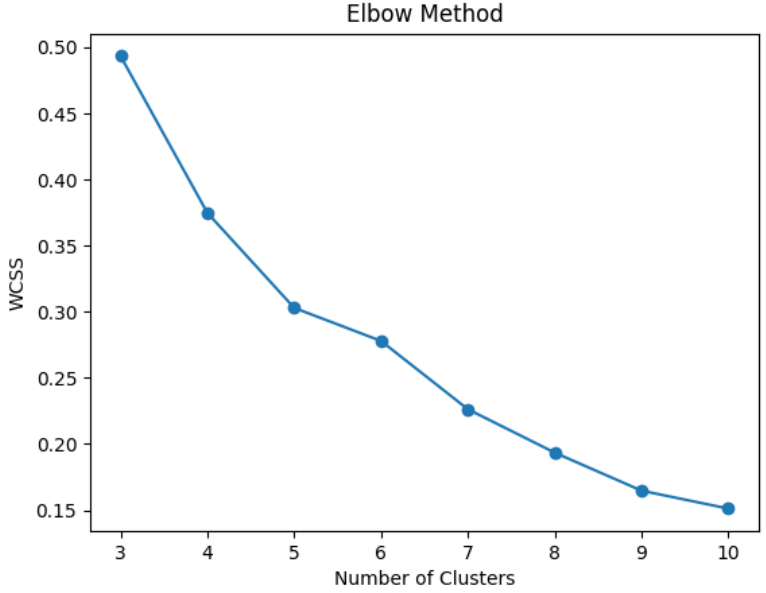
Appendix III: Clustering

*Dynamic Time Swarping

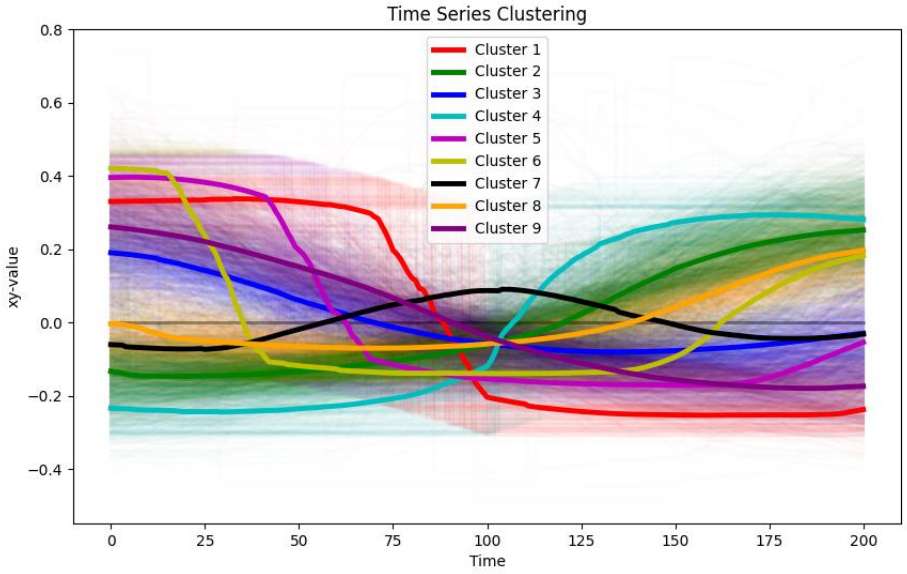
Results: Ball



Elbow method using Euclidean metric



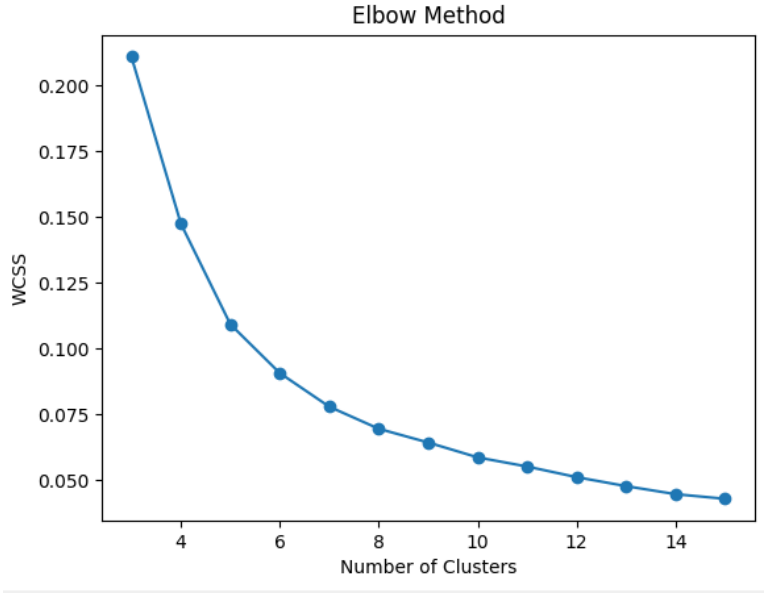
Elbow method using DTW metric*



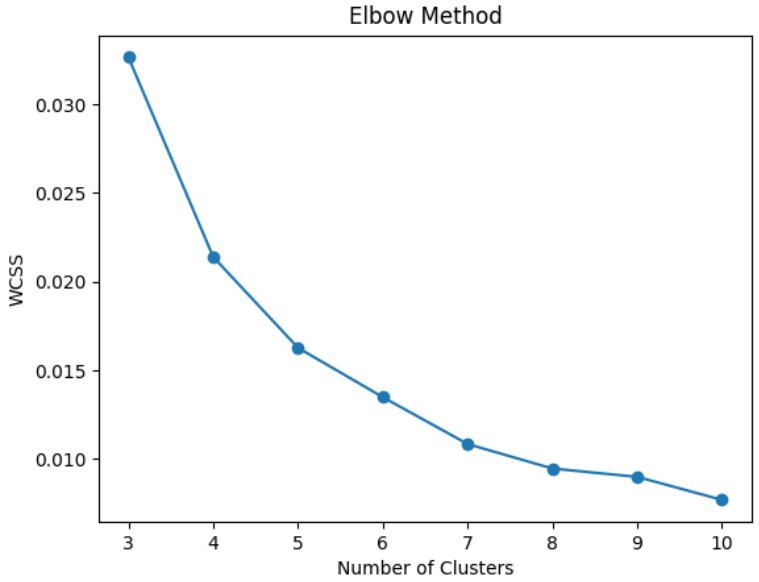
Clustering with mean

Explained Variance Ratio	**94.15%**
Highest chance of goal	**14.31%**
Silhoutte score	**0.22**

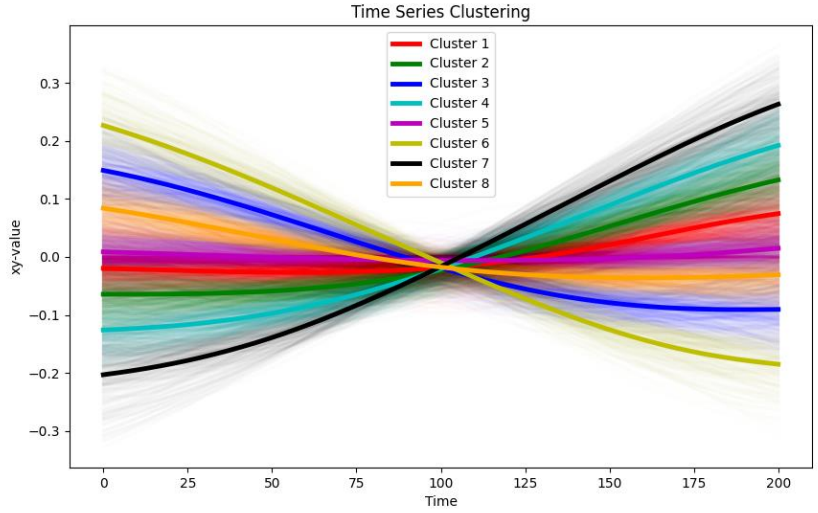
Results: XGPlayer



Elbow method using Euclidean metric



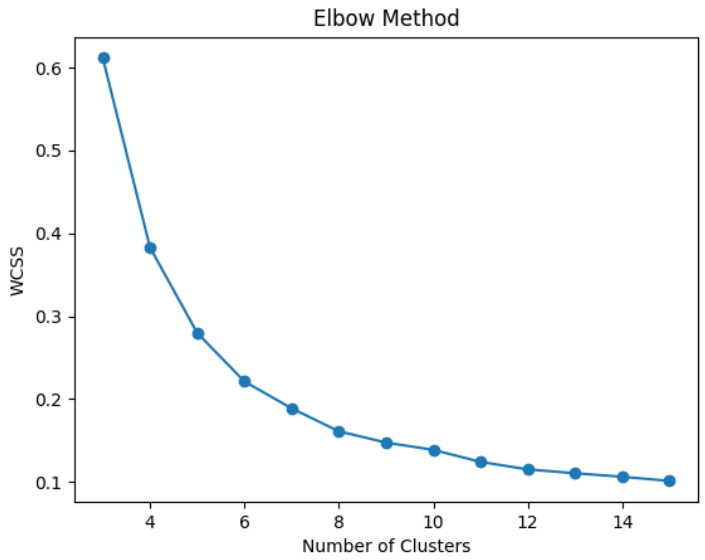
Elbow method using DTW metric



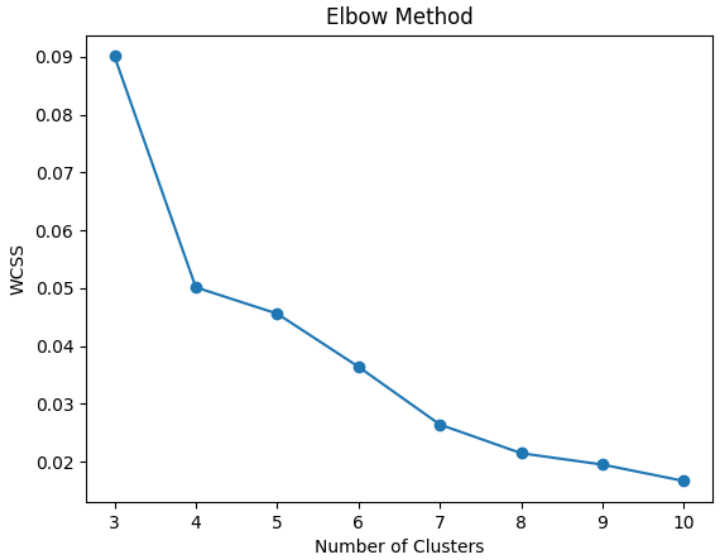
Clustering with mean

Explained Variance Ratio	**99.32%**
Highest chance of goal	**18.46%**
Silhoutte score	**0.34**

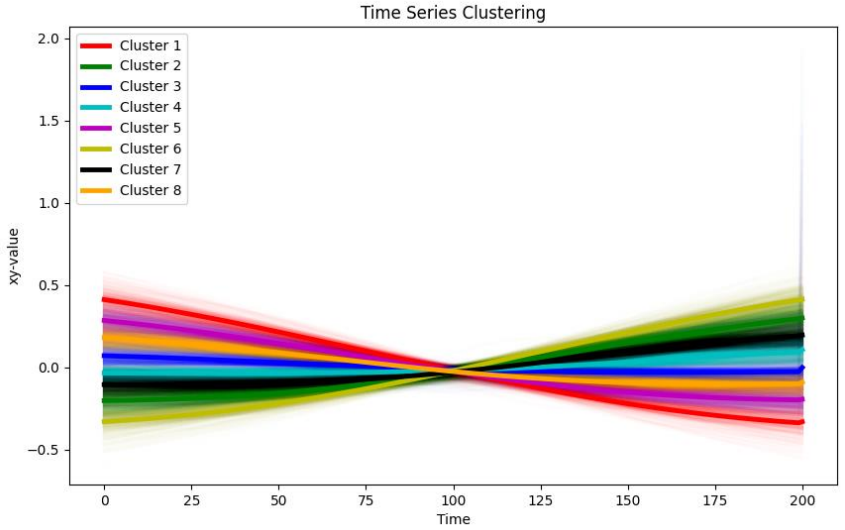
Results: TeamA



Elbow method using Euclidean metric



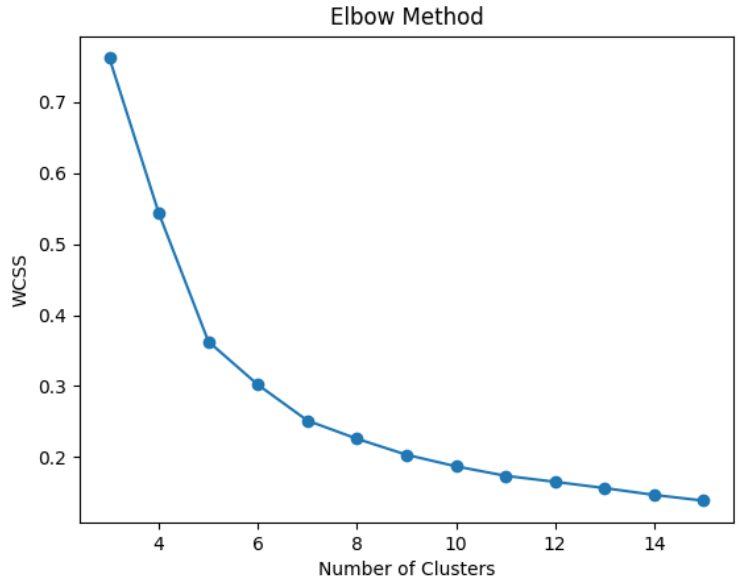
Elbow method using DTW metric



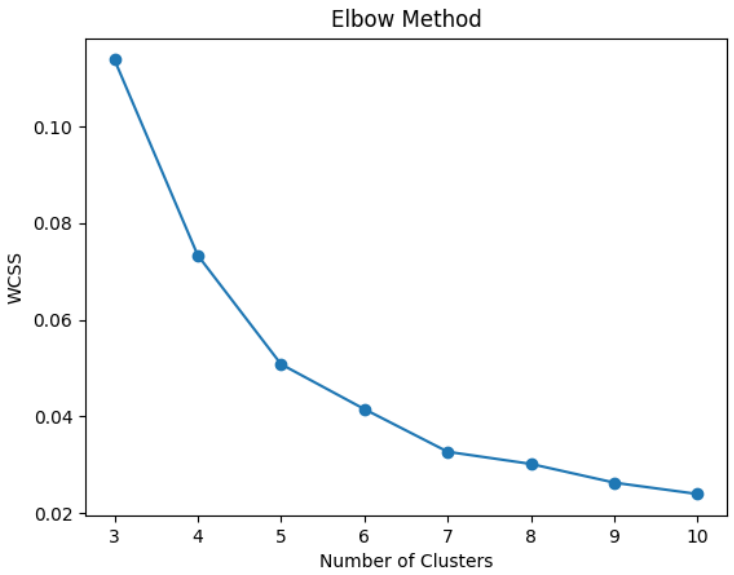
Clustering with mean

Explained Variance Ratio	**95.22%**
Highest chance of goal	**14.72%**
Silhoutte score	**0.39**

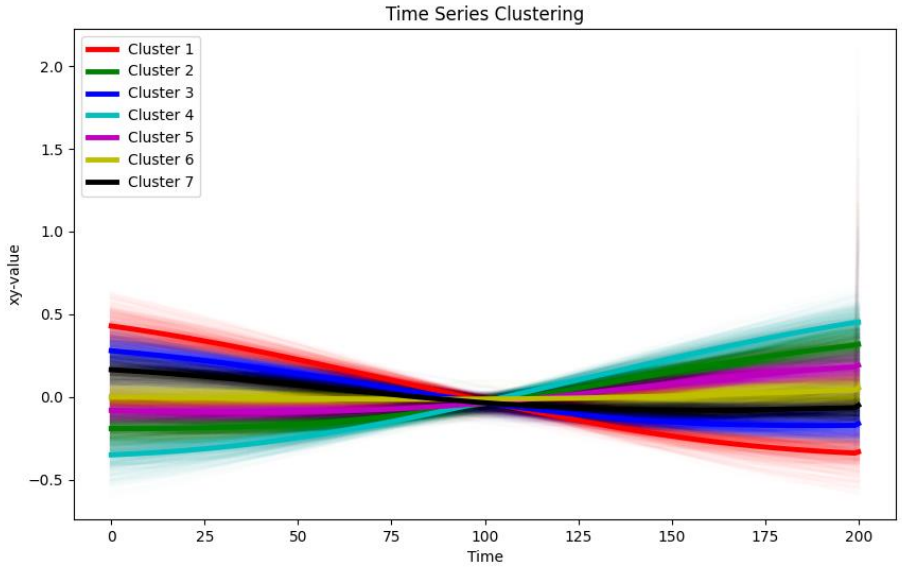
Results: TeamOpp



Elbow method using Euclidean metric



Elbow method using DTW metric

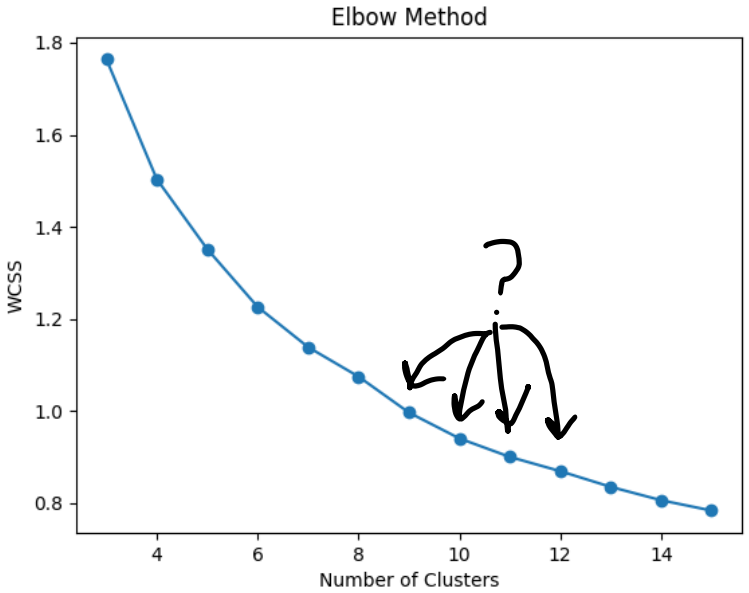


Clustering with mean

Explained Variance Ratio	**98.97%**
Highest chance of goal	**12.90%**
Silhoutte score	**0.40**

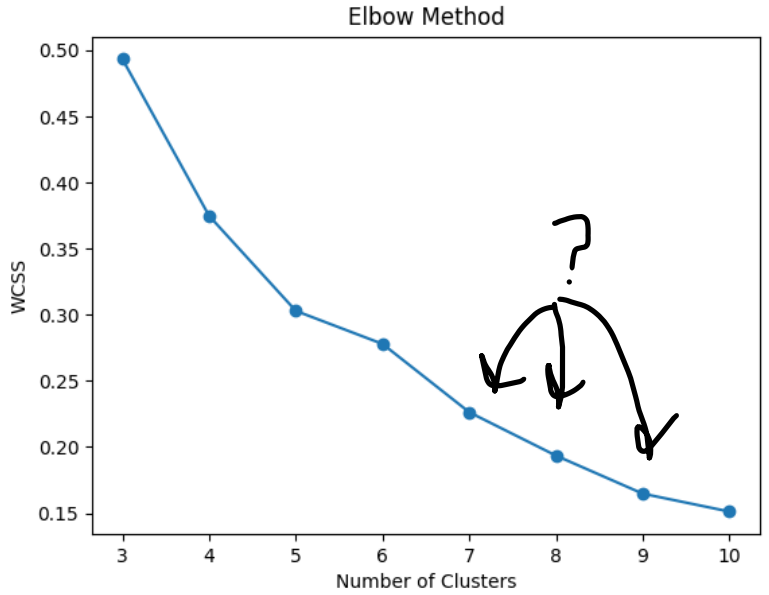
Explainer: Choice of N_clusters

We looked at both ”Elbow-plots”, for Euclidean and DTW metrics, where we determined N_clusters by regarding both plots.
Example:



Elbow method using Euclidean metric

+



Elbow method using DTW metric*

≈ 9

(Extra) Step 4.5: Reverse PCA (Didn’t work)

There does exist a reverse PCA function, however, too much info is lost.