Applied Machine Learning Spring 2026, NBI - Final Project
Project statement on individual contributions

All participants contributed evenly

Jakob Holst Svenningsen, cxh771,
Mikkel Keller Andreasen, trq281,
Sofie Havn, gdx888,
Kasper Schultz-Nielsen, nmd596,
Tobias Emil Gundel, dkz483

Machine Learning insights in football data

Final Project Presentation

By Mikkel, Tobias, Kasper, Sofie & Jakob

Setting the stage

1) Can we classify whether the match is played in a mens or womens competition?
2) Can we predict which shots will turn into a goal?
3) Can we predict player positions (e.g. defender) based on movement patterns?

What does the data look like?

Football data

Mens Euro 2024:
51 matches

Womens Euro 2025
31 matches

MU21 Euro 2025
31 matches

For each match:

Trajectory data

- **Time series data** tracking positions of up to 22 shirt numbers on the field and the ball
- Sampling frequency varies from 25 Hz to 50 Hz, Average 31.86 Hz
- Each timestamp records positions in coordinates on the pitch

Event data

EXAMPLE TRAJECTORY DATA →

label	half	time_half	total_time_	home_4_x	home_4_y	…	ball_x	ball_y
ENG-FRA	1	12.00	12.00	34.2	18.7	…	52.1	31.4

What does the data look like?

Football data

Mens Euro 2024:
51 matches

Womens Euro 2025
31 matches

MU21 Euro 2025
31 matches

For each match:

Trajectory data

- **Time series data** tracking positions of up to 22 shirt numbers on the field and the ball
- Sampling frequency varies from 25 Hz to 50 Hz, Average 31.86 Hz
- Each timestamp records positions in coordinates on the pitch
- 2 CSV files for each match

Event data

- **Time series data** (by def) containing an event stream of actions
- One entry with timestamp for a event.
- Entry also contains, type, playerPosition, teamFormation, start + end positions, duration and various features describing the type of event

EXAMPLE EVENT DATA →

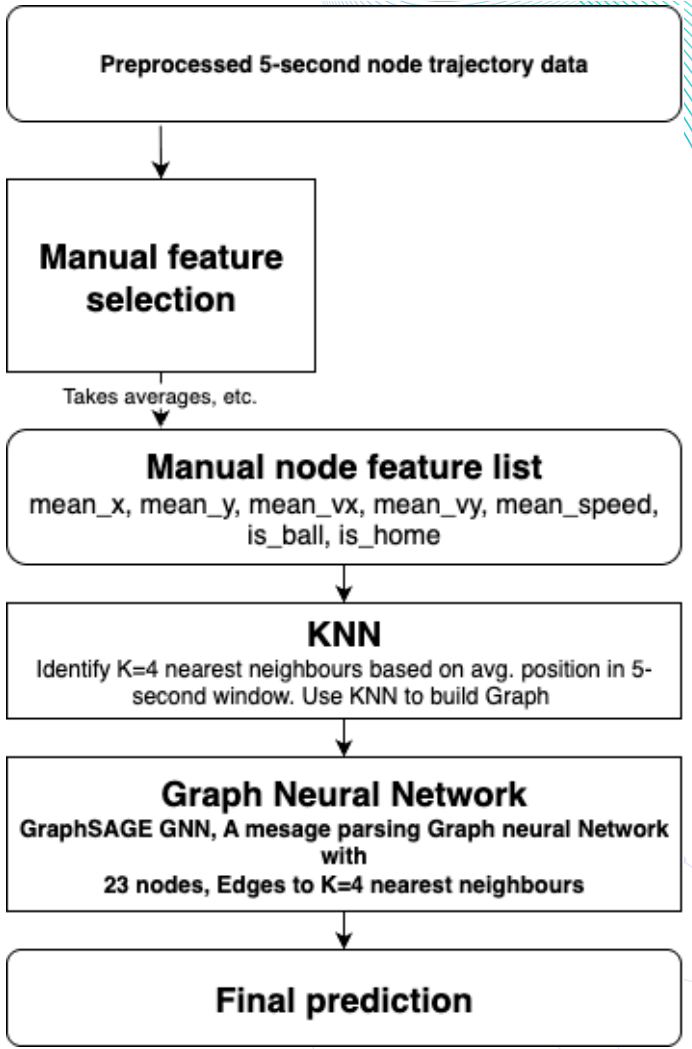
Match	Period	Time	Team	Position	Event Type	Duration	X	Y	Next X	Next Y	…
GER - SCO (5–1)	1H	00:01	GER	CF	Pass	26.0	51	50	31	18	…

Research question 1

Given an n-second sequence of ball and player **trajectories** can we classify whether the underlying match is played in a men's or women's competition?

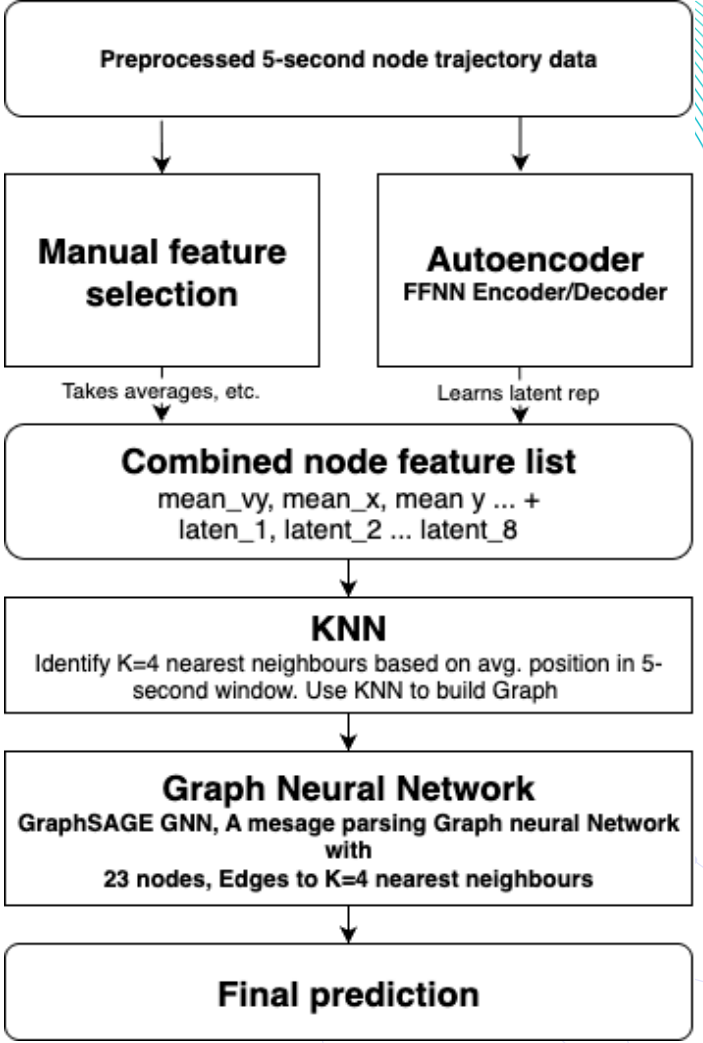
GNN Node feature selection with manual features or Autoencoder

- Local player interactions shape tactics → GNN is reasonable modelling tool
- 5-second window data → Increase samples (≈ 1080 pr match) and mitigate sensor fluctuations/errors
- Manual features based on domain knowledge
- These may fail to capture important information → Autoencoder (AE)
- AE does not improve model, why?
  - Learned patterns may not help classification
  - Information limit reached by manual features



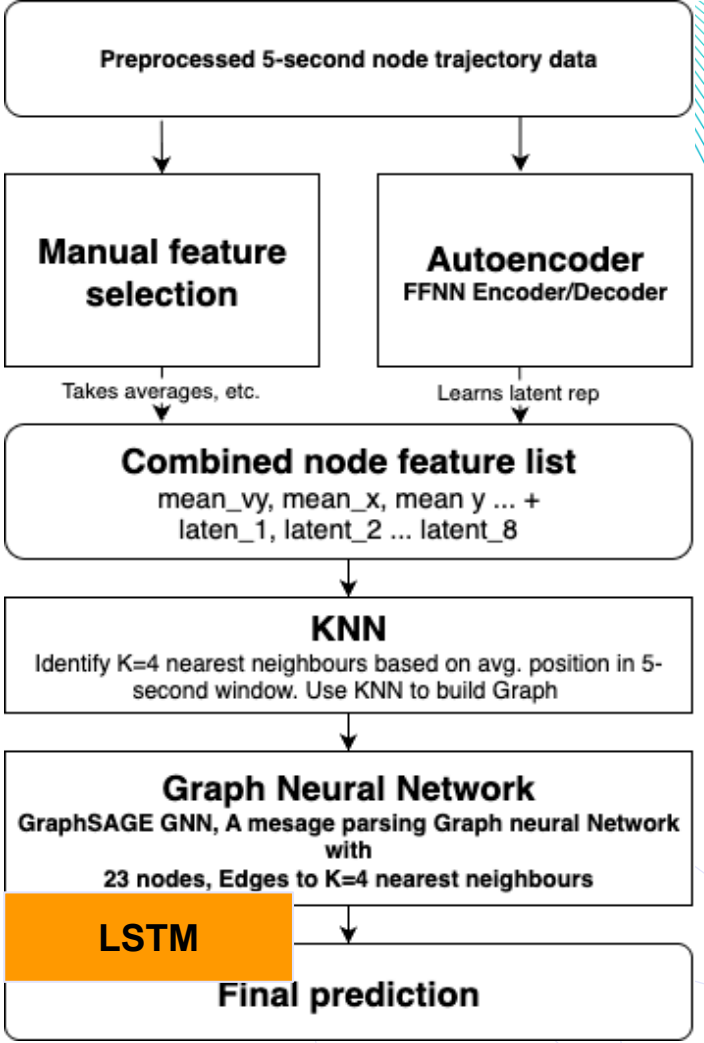
GNN Node feature selection with manual features or Autoencoder

- Local player interactions shape tactics → GNN is reasonable modelling tool
- 5-second window data → Increase samples (≈ 1080 pr match) and mitigate sensor fluctuations/errors
- Manual features based on domain knowledge
- These may fail to capture important information → Autoencoder (AE)
- AE does not improve model, why?
    - Learned patterns may not help classification
    - Information limit reached by manual features



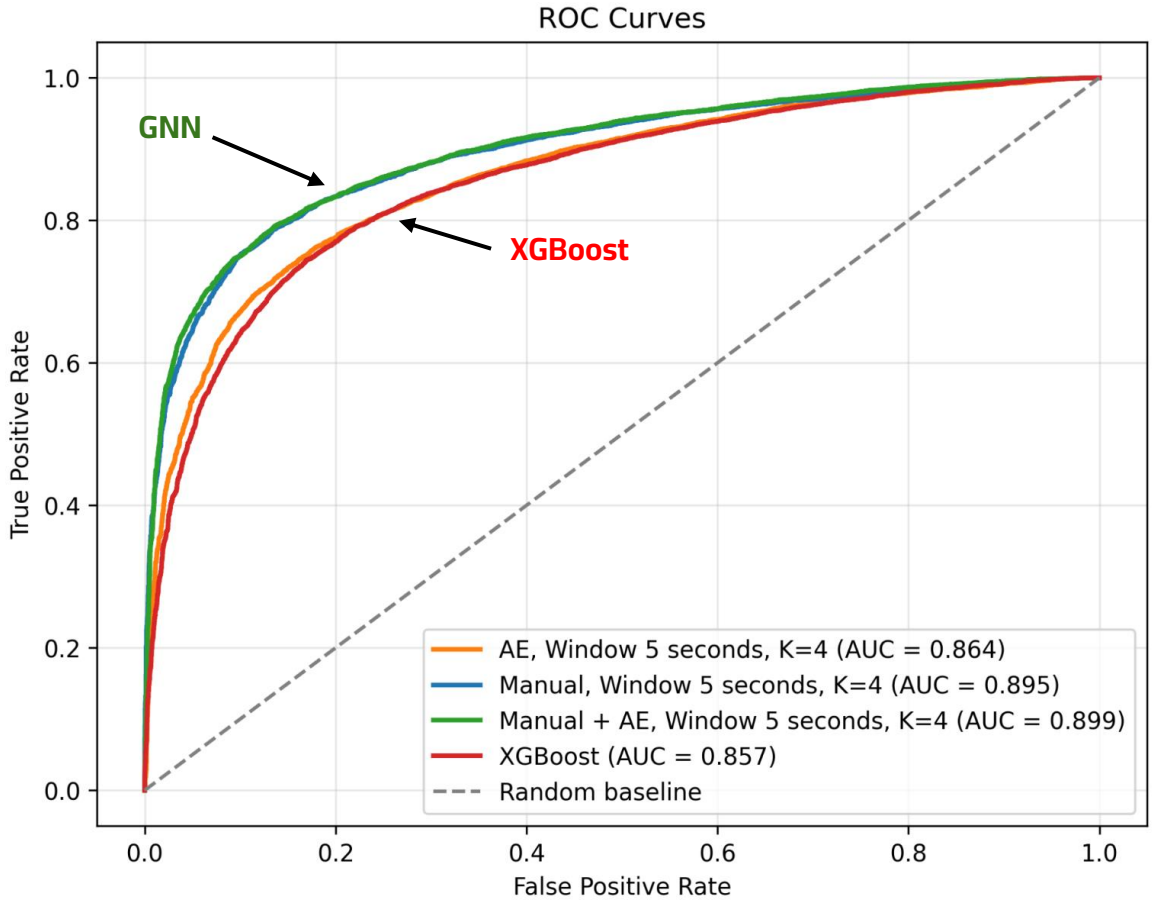
GNN Node feature selection with manual features or Autoencoder

- Local player interactions shape tactics → GNN is reasonable modelling tool
- 5-second window data → Increase samples (≈ 1080 pr match) and mitigate sensor fluctuations/errors
- Manual features based on domain knowledge
- These may fail to capture important information → Autoencoder (AE)
- AE does not improve model, why?
  - Learned patterns may not help classification
  - Information limit reached by manual features

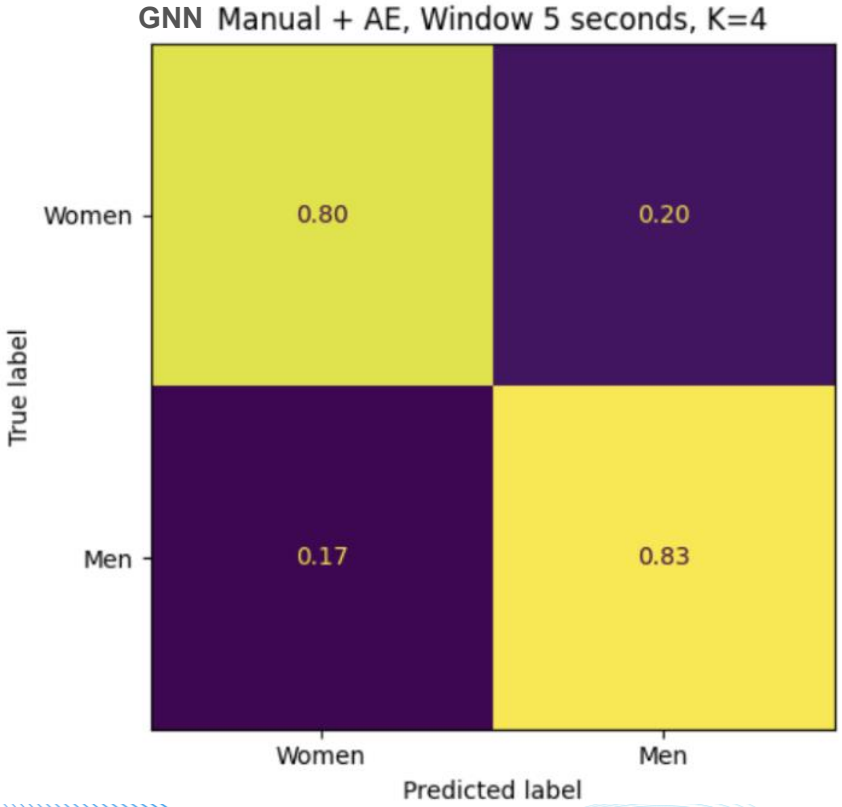


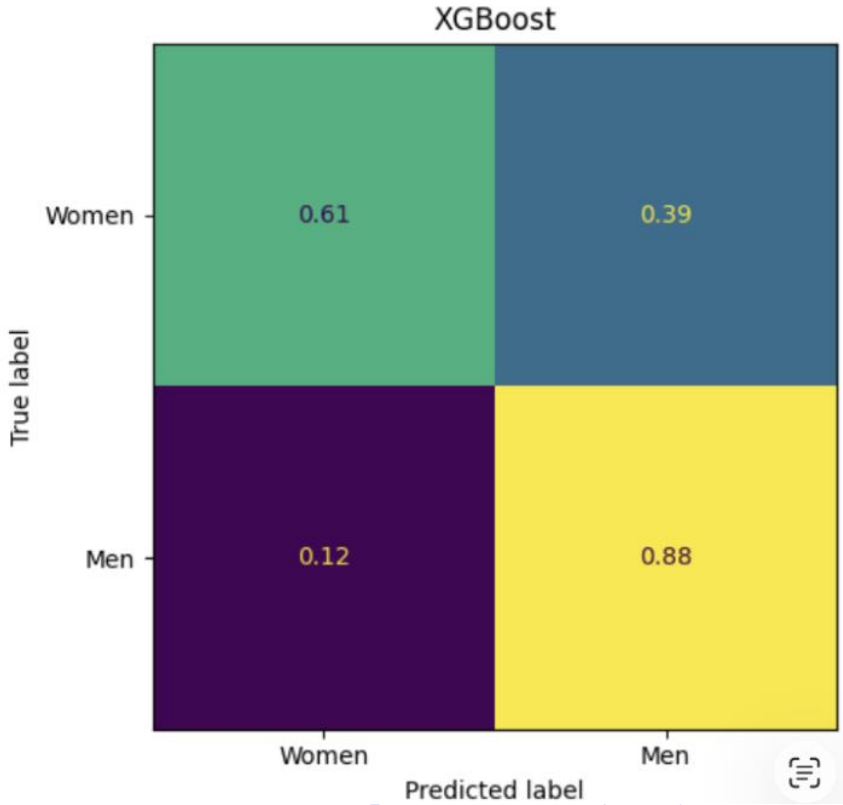
Mens/Womens classification ROC-AUC Curves

- **Note:** for XGBoost data was tabularized with avg, std, min and max over all nodes in a graph
- Graph-based approach outperforms XGB → Local interactions matter
- A lot of signal captured globally when feeding XGB tabular data
- Graph-based approaches outperforms XGBoost
- Most important features are positions and not speed as one would may intuitively think



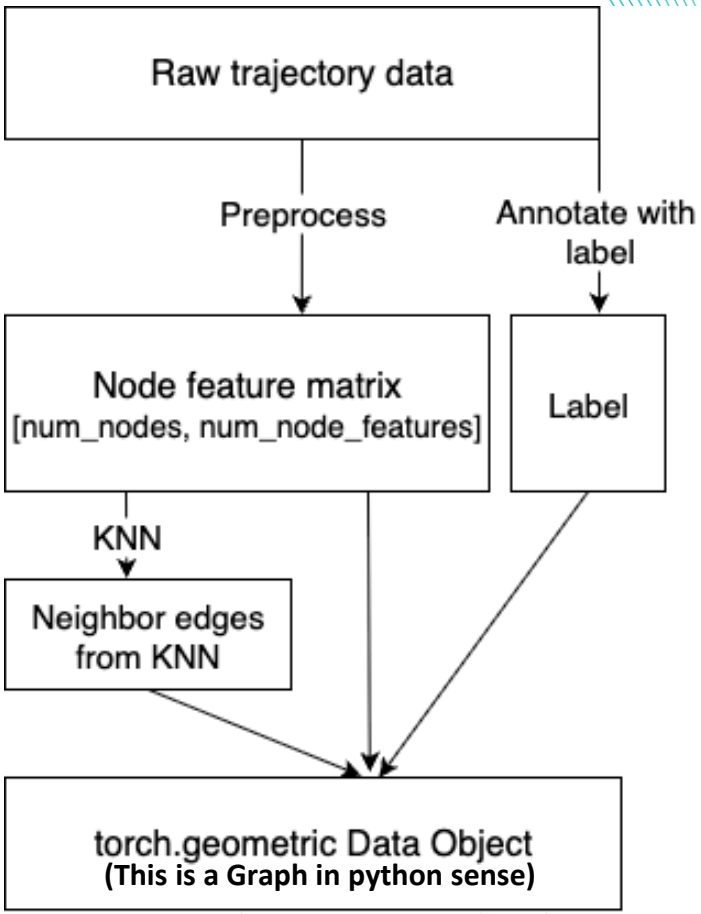
Mens/Womens classification (normalized) confusion matrices





The curse of pre-processing data for a GNN

- Samples are full of noise
  - NaN
  - duplicate samples
  - sensor errors
- Sensor errors cause exploding velocities
- Handling substitutions across windows
- Pitch normalization and accounting for half-time switches
- Resample windows to fixed size before inputting to AE
- Domain knowledge!
- Getting data in a format that a GNN can actually work with



Research question 2

Given **only event data** from matches can we classify whether the underlying match is played in a men's or women's competition?

Tried two ways to represent a match

- Feature Engineering - used with Multi-Layer Perceptron (MLP) and XGBoost
  - 3 chained event patterns and count them (normalized pattern frequencies)
  - Feature matrix: One row pr match (113 x 1058)

acceleration__to__pass__to__pass	acceleration__to__pass__to__shot	acceleration__to__pass__to__shot_against
0.0005688282138794084	0.0005688282138794084	0.0

- Sequence Learning - Used with Gated Recurrent Unit (GRU)
  - A sequence of 40 events
  - Feature matrix: many rows pr match (9384 x 40)
  - Model predicts on window level -> predictions aggregated to match level
  - The number in the feature matrix corresponds to an eventtype ("pass", "acceleration" etc..)

	0	1	2	3	4	5	6	7	8	9	...	30	31	32	33	34	35	36	37	38	39
0	13	13	13	13	13	3	3	6	19	13	...	3	3	13	10	1	20	3	3	13	13
1	3	13	13	13	13	1	10	13	3	3	...	13	13	13	20	13	13	10	3	3	13

What did we try?

Models

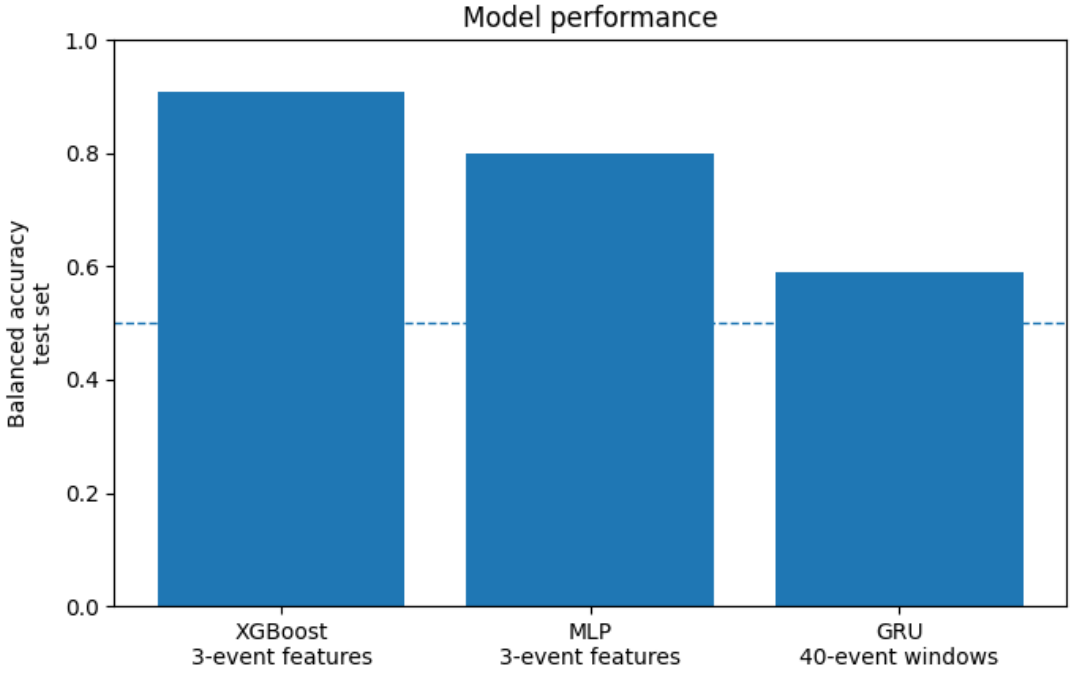
- XGBoost
  - Tree model
  - Strong for tabular data
  - baseline
- MLP (Multi-layer perceptron)
  - feedforward nn
  - suitable for small datasets
- GRU (Gated Recurrent Unit)
  - recurrent nn
  - learns Temporal event patterns

Model robustness (small dataset):

- Stratified K-fold cross validation (K=5) used for MLP/XGBoost
- stratified Maintains class balance across folds
- Stratified Group K-fold cross validation (K=5) for GRU
- keeps all windows from the same match (group) in the same fold
- MLP (std ≅ 0.04), GRU(std ≅ 0.0176), XGBoost (std ≅0.112)

Evaluation

- XGBoost has highest accuracy but also highest std between folds



* balanced accuracy score from sklearn defined as " the average of recall obtained on each class "
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

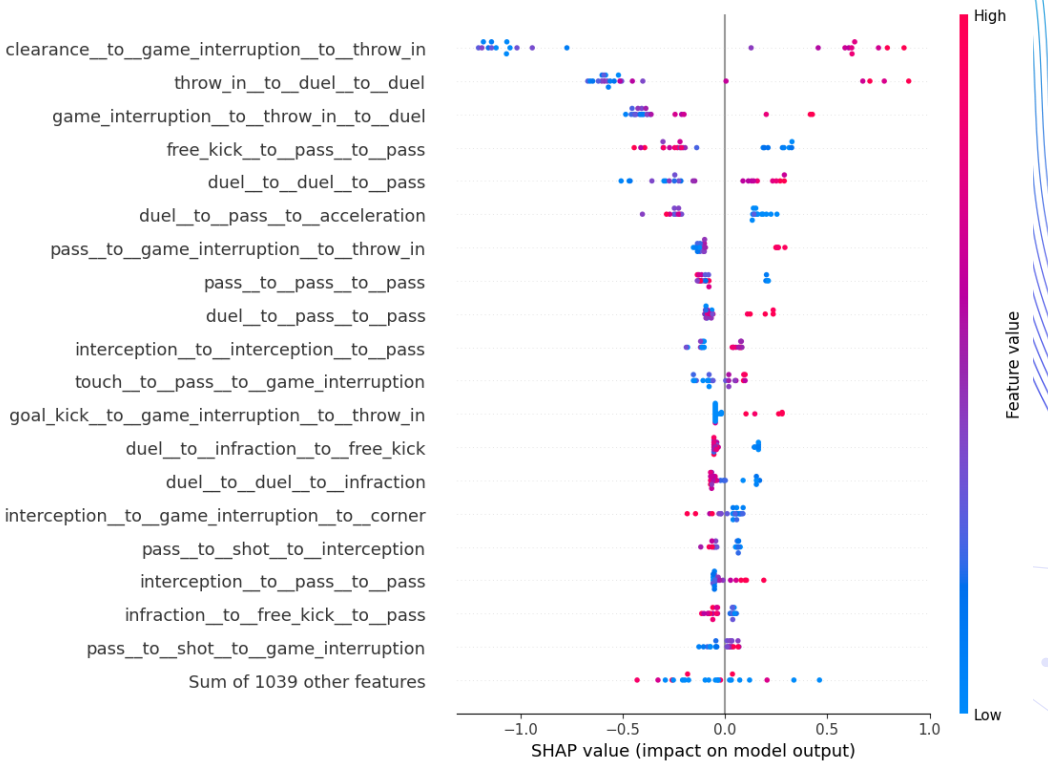
What did we (or the model) Learn?

From XGBoost

- Shap value > 0 (Women)
- Shap value < 0 (men)

So can we predict if men or women played the match, based only on event data? soft yes

- more game interruptions and throw ins in womens games
- more free kicks in mens games (maybe more aggressive?)
- soft because,
  - relative high accuracy
  - model seems to learn something
  - but unstable models (maybe more data?)



link to event glossary: https://dataglossary.wyscout.com/

Research question 3

How well can we predict the probability of a shot becoming a goal using event and tracking data?

Data Preprocessing and Feature Generation

```
Train matches: 78
Test matches: 27
Train shots: 1324
Test shots: 453
Train goalrate: 0.114
Test goalrate: 0.104
```

tournament	all_matches	train_matches	test_matches
H_EURO2024	47	35	12
Q_EURO2025	29	21	8
U21_EURO2025	29	22	7

Step 1: Feature set comparison
- 5 different feature sets

Step 2: Cross-validation
- 5-fold StratifiedGroupFold
- Grouped by MatchId to avoid leakage

Step 3: Hyperparameter tuning
- Gridsearch over C-values
0.03, 0.1, 0.3, 1, 3, 10

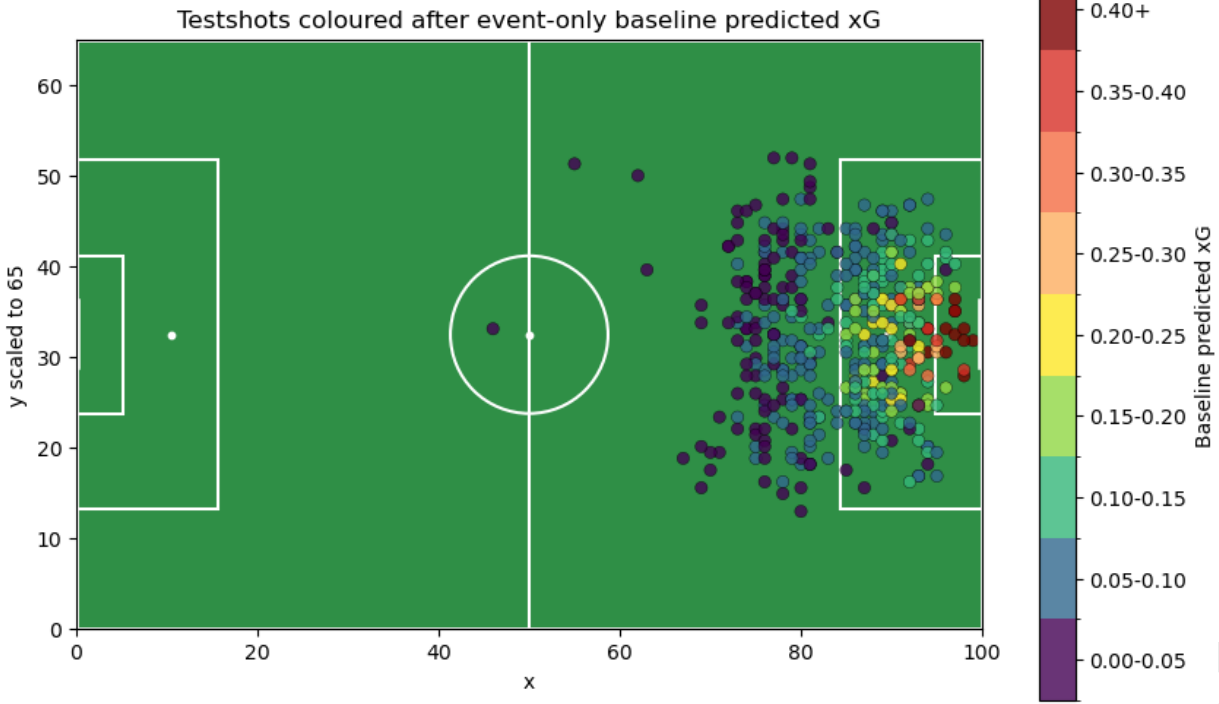
Standard Logistic Regression classifier

	model	best_C	best_cv_log_loss
0	M4: + previous_event_type	0.10	0.310707
1	M5: all event features	0.10	0.310905
2	M2: geometry + time + binary features	0.03	0.312965
3	M3: + bodyPart	0.03	0.313241
4	M1: geometry only	0.10	0.316195

Event features:

x	y	shot_distance_m	shot_angle_deg	bodyPart	playerPosition	matchPeriod	previous_event_type	after_set_piece	is_header	isGoal
83	48	17.901735	23.049903	right_foot	LAMF	1H	pass	0	0	1
86	51	14.715719	27.908372	right_foot	RAMF	1H	duel	0	0	1
82	52	18.948868	21.813183	left_foot	CF	1H	pass	0	0	0

The models predicted xG shown on the pitch



Comparison with the existing_xG in the provided data

	model	log_loss	brier_score	roc_auc
0	Existing xG	0.302838	0.084935	0.724688
1	Our baseline model	0.298917	0.086060	0.767687

Results from adding tracking features

	model	log_loss	brier_score	roc_auc
0	Event + Engineered logistic regression	0.290012	0.083275	0.801960
1	Event + tracking logistic regression	0.290728	0.084440	0.792317
2	Event-only logistic regression	0.298917	0.086060	0.767687

Top-k selection

	model	log_loss	brier_score	roc_auc
0	All engineered features logistic regression	0.290012	0.083275	0.801960
1	Best top-k logistic regression	0.291943	0.083593	0.798082

Exploring other models

Model tuning - GridSearch:

"model__learning_rate":

"model__max_iter"

"model__max_leaf_nodes"

"model__l2_regularization"

"model__min_samples_leaf"

Extra Trees and Random Forest:

"model__n_estimators"

"model__max_depth"

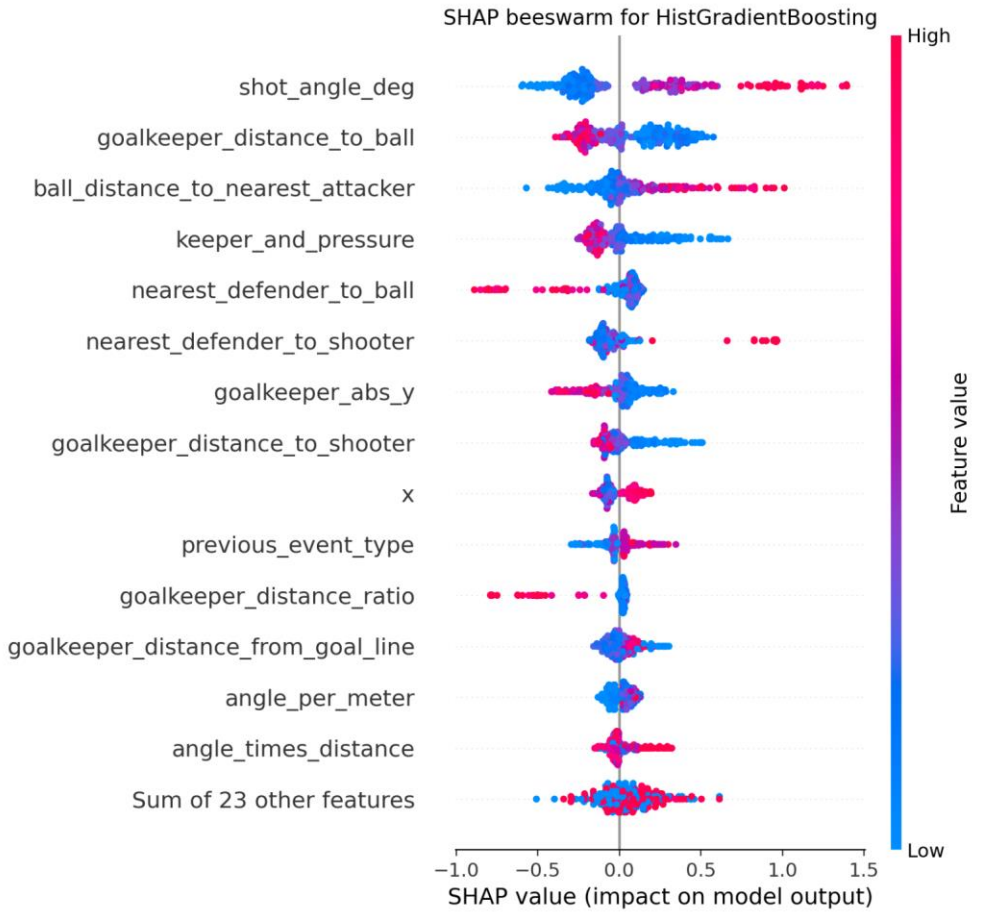
"model__min_samples_leaf"

"model__max_features"

	model	log_loss	brier_score	roc_auc
0	HistGradientBoosting	0.273264	0.079058	0.804895
1	Extra Trees	0.282740	0.081569	0.778482
2	Random Forest	0.290907	0.084065	0.767163
3	Existing xG	0.302838	0.084935	0.724688

Split perturbation analysis

	model	mean_log_loss
0	HistGradientBoosting	0.303754
1	Logistic regression	0.306958
2	Existing xG	0.320680



Research question 4

Can we predict player positions based on movement patterns?

Problem setting and reference model

Problem: Tracking data identifies players by shirt number but does not describe player informations

- Position labels are not available in data - but can we predict positions from movement patterns?
  - Movement patterns: x = x - mean(x), y' = y - mean(y)

- We only use data from Mens Euro 2024 (avoids effects from domain differences)

- Naive reference model based on domain knowledge:
  - Positions based on shirt numbers: 1 = GK, (2, 3 ,4 ,5) = DF, (6, 7, 8) = MF, (9, 10, 11) = FW, shirtnumber > 11 => random.choice(DF, MF, FW)

	label	half	time_passed_in_half	total_time_passed	home_3_x	home_3_y	home_7_x	home_7_y	home_20_x	home_20_y	...
0	Albania - Spain	1	0.00	0.00	-8.584832	-25.231071	0.544622	-31.981096	-2.093903	-9.673864	...
1	Albania - Spain	1	0.04	0.04	-8.544409	-25.217894	0.669641	-31.950325	-2.091313	-9.667552	...
2	Albania - Spain	1	0.08	0.08	-8.499372	-25.201967	0.804644	-31.921465	-2.092380	-9.652243	...

Trying to beat the naive model: 2-layered clustering

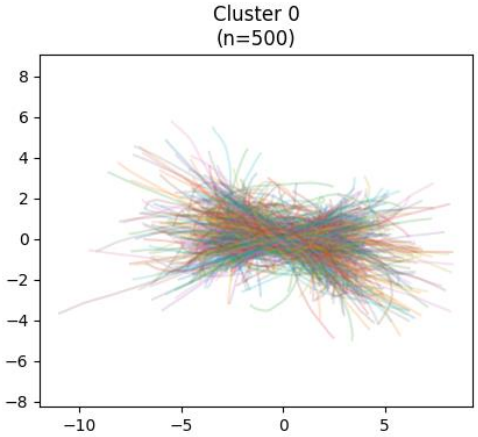
Challenges of clustering full trajectories

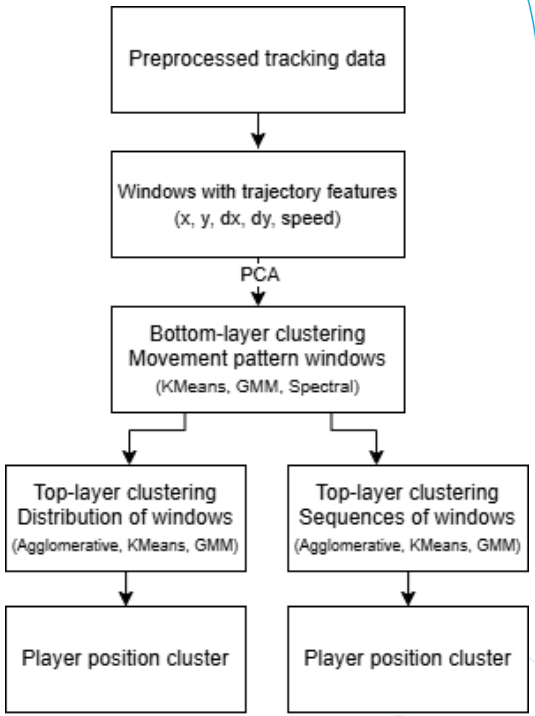
- Full trajectories are high-dimensional and computationally expensive to cluster.
- Substitutions lead to trajectories of unequal length.
- Movements at different times might not be clustered together (no time-invariance)

Modelling trajectory windows

- Fixed length windows: Lower dimensionality and consistent trajectory length
- Partially addresses time-invariance - local movement patterns

Example of movement cluster:





Evaluation of 2-layered clustering

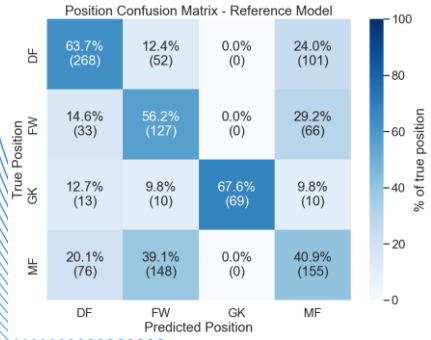
Key findings:

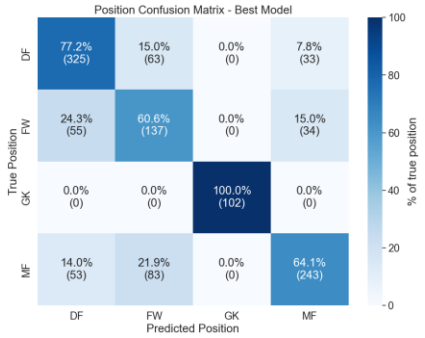
- Reference model performed better than most initial models (AMI reference_model= 0.228)
- Clustering time-series data is challenging!
- KMeans worked surprisingly well when clustering movement patterns (bottom layer)
- Approach with sequence-based top-layer outperformed distribution profiles

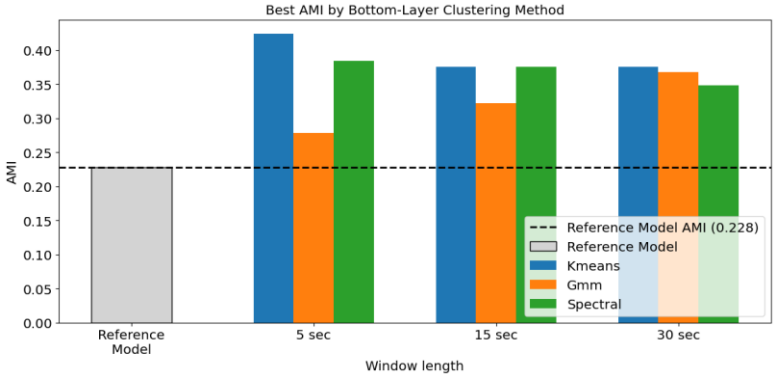
Best model parameters (AMI = 0.424):

Bottom: KMeans, Top = KMeans, N_movement_clusters = 12, sequence_length = 2, window_length = 5 seconds

Models evaluated by scraping actual positions from: https://en.wikipedia.org/wiki/UEFA_Euro_2024_squads







Alternative Approach:

Variational Autoencoder for processing trajectory windows as oppose to clustering

Refinement of data pre-processing

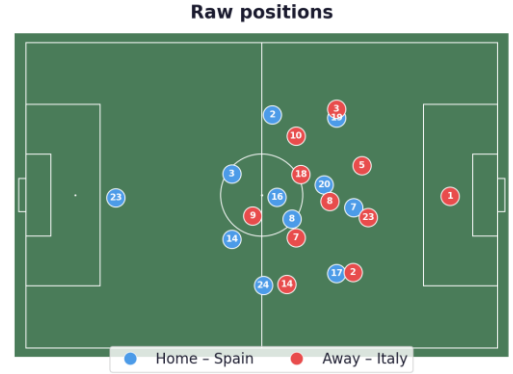
Comparability of player positions
- Ensuring all players attack in the same direction
- Ensuring left and right flank players are comparable

Strict playing-time filter (40+ min per half)

Two approaches to forming trajectory windows
- Fixed-interval windows (non-overlapping)
- Anchored in events from Event data

Each window origin-normalised to (0,0)

Window length
- Tested lengths: 5, 10 and 20 seconds
- Best performing: 10 seconds



Processing Trajectory Windows

Variational Autoencoders (VAE)

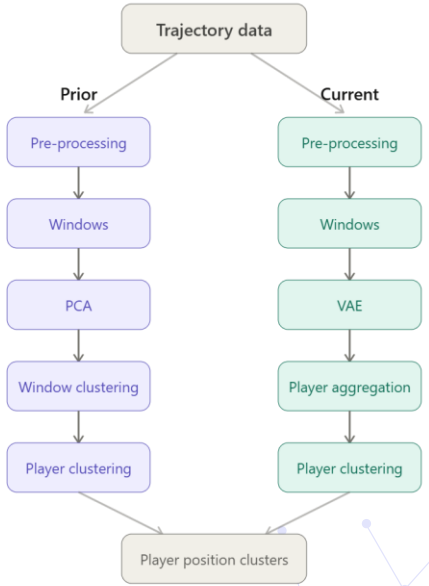
- Tested Autoencoders: Feed Forward, Conv, LSTM, ConvLSTM and Conditional VAE
  - Best performing: ConvLSTM
- More flexibility relative to clustering
  - 12 discrete clusters vs. 16 continuous latent dimensions

Hyper Parameters

- Tested latent dimensions: 2D, 8D, 16D and 32D
  - Best performing: 16D
- Batch size: 256
- Learning rate: 1e-4
- Epochs: 50
- KL annealing epochs: 20

Aggregating Embeddings

- We aggregate each embedding by
  - Percentiles: {0.5%, 1%, 5%, 25%, 50%, 75%, 95%, 99%, 99.5%}
  - Mean
  - Standard deviation
- This gives each player a single representation per game of (16 x 11) dimensions



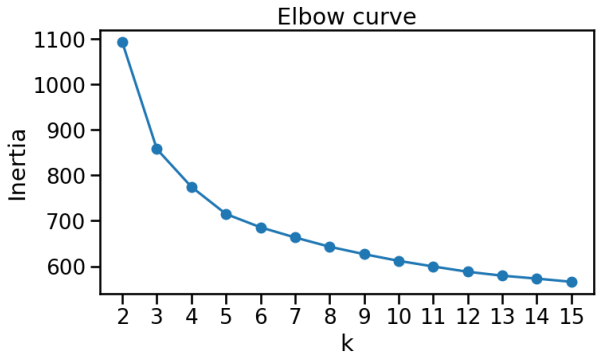
Clustering

We use the aggregated embeddings as input to clustering algorithms

- Tested algorithms: K-means, GMM, HDBSCAN and Agglomerative clustering
- k=4 chosen (when possible) to match the 4 broad player positions (GK, DF, MF, FW)
- Best performing: K-means

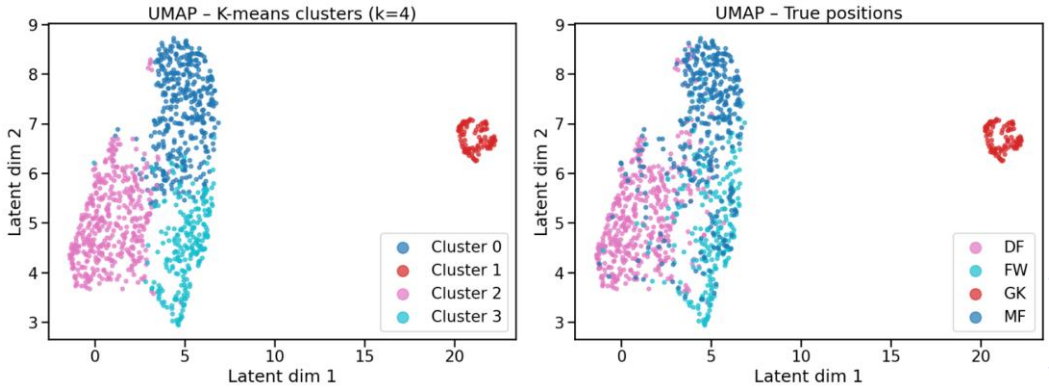
Elbow plot

- K-means on the best model
- No clear elbow, hence no indication of a natural cluster size



UMAP visualization

- Aggregated embeddings -> 2D coordinates which can be plotted
- Reveals clear structural patterns correlated with player positions
- There is some noise in the true labels, which K-means doesn't capture



Did we succeed?

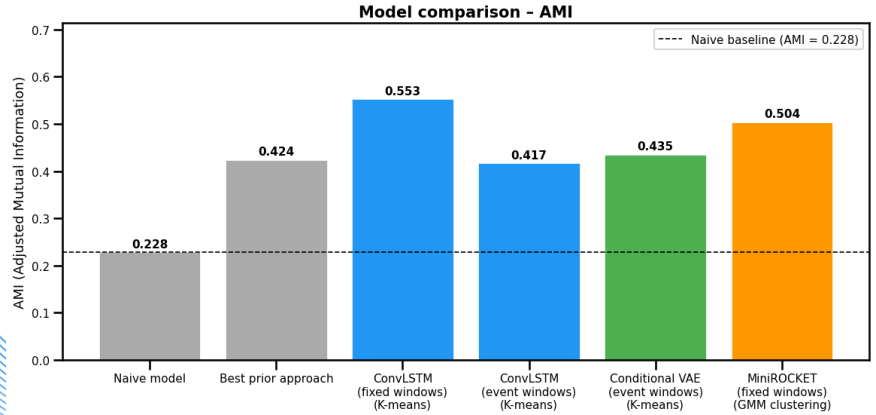
New approach yielded notable improvements

Can it be improved further?

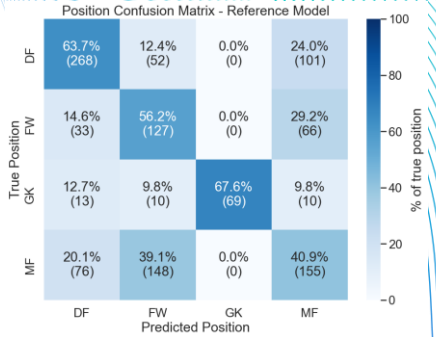
- We haven't done much *hyperparameter optimization*, but doing it could improve the model further
- However, we think this is a strong model and a strong approach to a complex problem

To answer the question - Did we succeed?

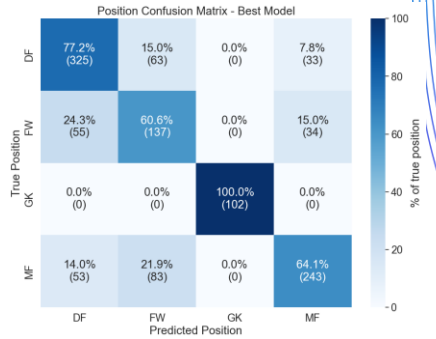
- Yes, we would say so (and say it proudly)



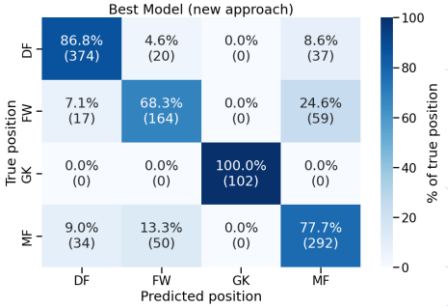
Naive approach
AMI: 0.228



Best
Previous approach
AMI: 0.424



Best
Current approach
AMI: 0.553



Appendix

Appendix A

model descriptions for Question can we predict if a match is played by women or men, based only on event data?

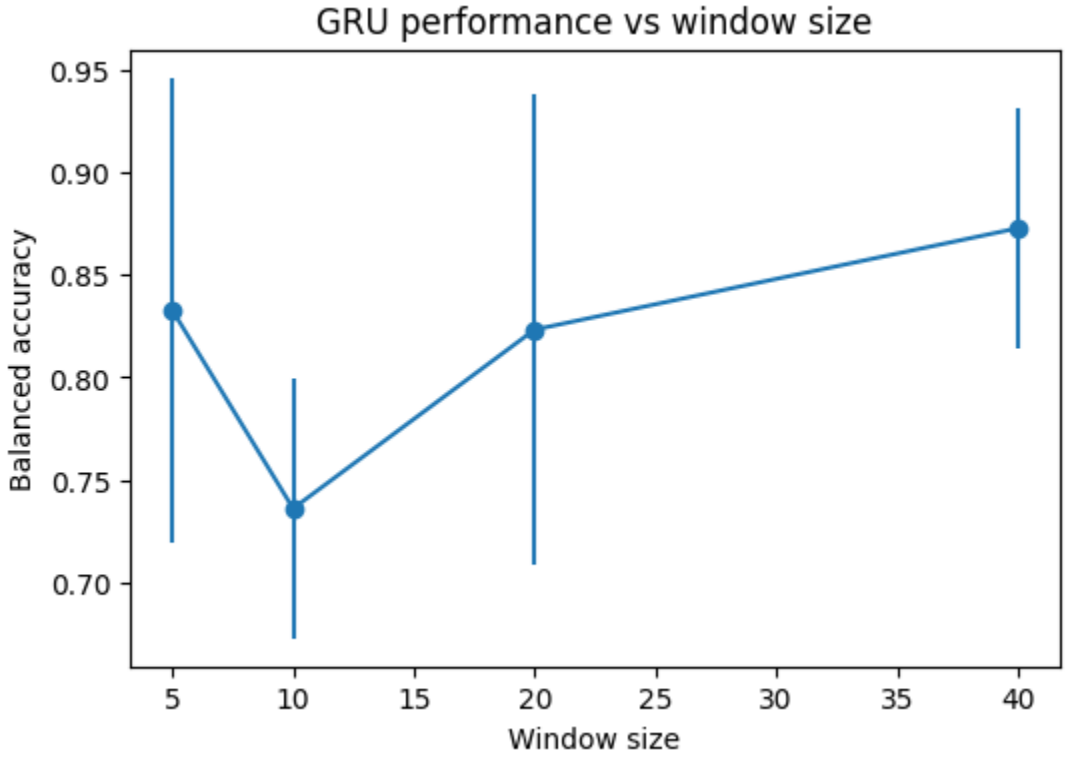
A.1 Models

```
model_1 = XGBClassifier(
        n_estimators=100,
        max_depth=3,
        learning_rate=0.05,
        subsample=0.8,
        colsample_bytree=0.8,
        eval_metric="logloss"
    )


model_2 = MLPClassifier(
        hidden_layer_sizes=(128, 32),
        alpha=0.01,
        max_iter=500,
        learning_rate=0.001
        random_state=42
    )
```

GRU:

```
emb_dim=32, hidden_dim=64, dropout=0.3)

self.gru = nn.GRU(
            input_size=emb_dim,
            hidden_size=hidden_dim,
            batch_first=True,
            bidirectional=True
        )
self.classifier = nn.Sequential(
            nn.Dropout(dropout),
            nn.Linear(hidden_dim * 2, 64),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(64, 1)
        )
```

A.2 GRU, window size and stability (vertical lines are STD)



Appendix B

Model descriptions for question
Given an n-second sequence of ball and player trajectories can we classify whether the underlying match is played in a men's or women's competition?

Appendix B.1 - Overview of model experiments

Model	Window Size (s)	Epochs	Hidden Channels	Learning Rate	K Neighbors	Conv Layers	Linear Layers	Final Test Accuracy	Best Epoch
Man Features	5	50	32	0.001	4	2	2	**0.8215**	41
Manual + AE	5	50	32	0.001	4	2	2	**0.8213**	39
AE Only	5	50	32	0.001	4	2	2	**0.7888**	23

Model	n_estimators	max_depth	learning_rate	subsample	colsample_bytree	objective	eval_metric	Final Test Accuracy
XGBoost	100	4	0.05	0.8	0.8	binary:logistic	logloss	**0.7733**

Appendix B.2 - Structure of the Graph Neural Network

```
GraphSAGEClassifier(
  (conv1): SAGEConv(8, 32, aggr=mean) # neighbours
  (conv2): SAGEConv(32, 32, aggr=mean) # Allows information to propagate from neighbours neighbours
  (lin1): Linear(in_features=32, out_features=32, bias=True)
  (lin2): Linear(in_features=32, out_features=2, bias=True)
)
```

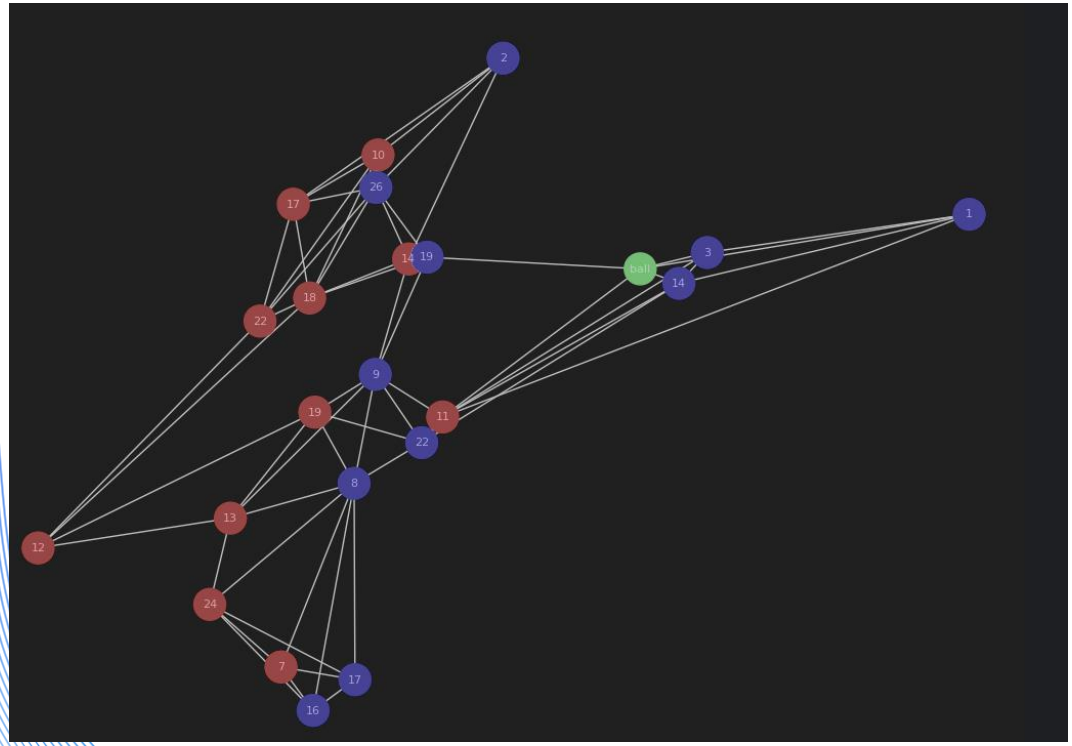
Input features	: 8
Hidden channels	: 32
Output classes	: 2
GraphSAGE layers	: 2
Linear layers	: 2

Appendix B.3 - Structure of the Autoencoder

```
[2 rows x 200 columns]
Train trajectory tensor shape: torch.Size([1240741, 200])
All trajectory tensor shape: torch.Size([2084130, 200])
TrajectoryAutoencoder(
  (encoder): Sequential(
    (0): Linear(in_features=200, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=8, bias=True)
  )
  (decoder): Sequential(
    (0): Linear(in_features=8, out_features=128, bias=True)
    (1): ReLU()
    (2): Linear(in_features=128, out_features=200, bias=True)
  )
)
```

- Latent space size = 8 → key trajectory patterns per node.
- Input trajectories are resampled to 50 timesteps to standardize variable-length windows. This turned out to be a big challenge as the frequency of sampling varies.
- Trained unsupervised using MSE loss, minimizing reconstruction error of node trajectories.
- Encoder output (latent features) are then appended to features to act as node features for the classification task
- Match-level train/validation/test split used to avoid data leakage from windows originating from the same match.

Appendix B.4 - Example graph of 5-

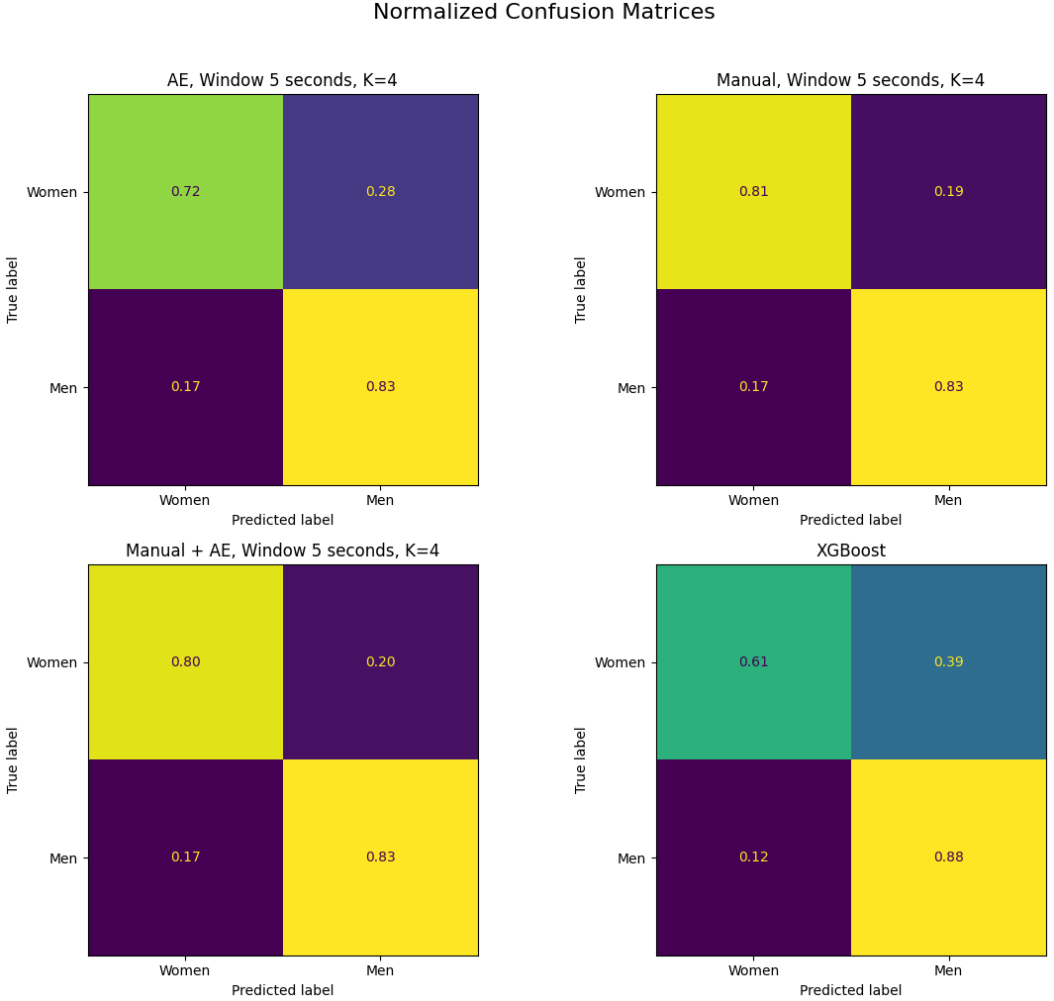


The visualization serves as a sanity check of the procedure to construct graph objects. Looking up the shirt numbers of the match Slovakia-Ukraine in UEFA mens EURO 2024, we can actually see that number 12 was indeed Marek Rodák, a Goalkeeper, which football semantically provides a reasonable spatial interpretation of what we visually can see

Pytorch geometric object Data(x=[23, 8], edge_index=[2, 184], y=[1], node_ids=[23], match_id='2036182_Slovakia_Ukraine', window_id=0)

Appendix B.5 - Normalized Confusion matrices for all four models

The three GNN models obtain a significantly better per-class accuracy, likely because message parsing will pull outliers to the local context.



Appendix B.6 - XGBoost Feature Importance and Permutation importance of GNN with manual + latent node features

feature	importance XGBoost
mean_x	0.260016
mean_y	0.253722
is_ball	0.147503
mean_speed	0.131883
mean_vy	0.079080
is_home	0.071721
mean_vx	0.056075

Interesting finding
The positions of the play, and thereby width etc. matter more for classification than e.g. speed

GNN Permutation importance

is_home: 0.1587
latent_0: 0.1171
latent_1: 0.1171
latent_5: 0.0854
latent_3: 0.0642
latent_2: 0.0629
mean_x: 0.0355
latent_4: 0.0344
latent_7: 0.0342
latent_6: 0.0252
is_ball: 0.0119
mean_y: 0.0088
mean_vx: 0.0037
mean_speed: 0.0037
mean_vy: -0.0014

Comparing with ROC-AUC etc, results suggest latents carry redundant information also conveyed in manual features

Note: Regarding GNN permutation importance, the relative ordering is informative, but the absolute values should be interpreted carefully since the message-passing nature of the GNN does not allow features to be considered independently. Importance therefore reflects both the direct feature effects but also their influence on neighborhood aggregation.

Appendix C

Model descriptions for question:

How well can we predict the probability of a shot becoming a goal using event and tracking data?

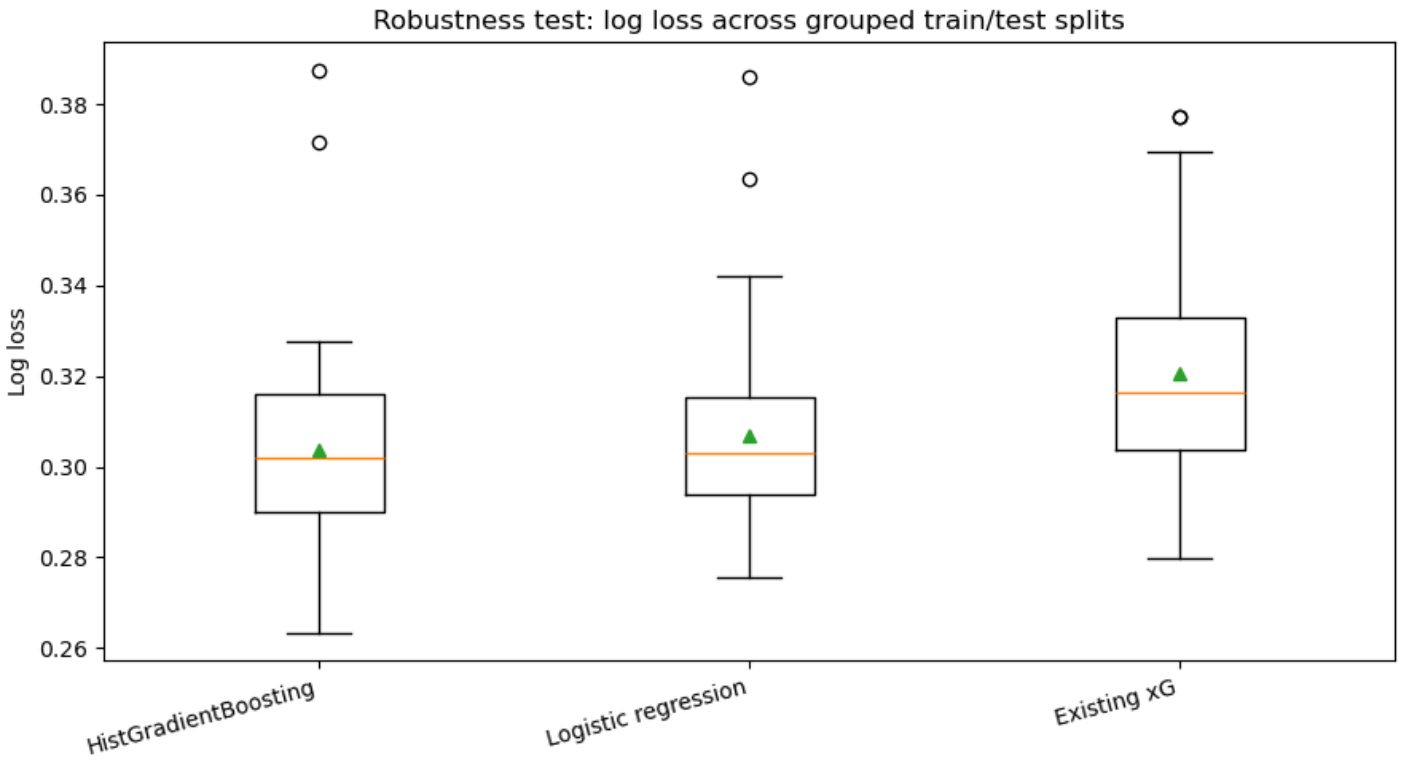
Appendix C.1 - Overview of logistic regression (baseline model)

Model	Feature set	Tuning	Best C / k	Test log loss	Brier	ROC-AUC
Existing xG	Provided xG benchmark	None	-	0.302838	0.084935	0.724688
Event-only Logistic Regression	Event features only	GridSearch over C	C = 0.10	0.298917	0.086060	0.767687
Event + Tracking Logistic Regression	Event + raw tracking features	Manual C comparison	C = 0.03	0.290728	0.084440	0.792317
Engineered Logistic Regression	Event + tracking + engineered features	Manual C comparison	C = 0.03	0.290012	0.083275	

Appendix C.2 - Overview of tree models

Model	Main features	Hyperparameter search	Best hyperparameters	Test log loss	Brier	ROC-AUC
Random Forest	All engineered features	GridSearchCV	`n_estimators=300, max_depth=8, min_samples_leaf=5, max_features=0.7`	0.290907	0.084065	0.767163
Extra Trees	All engineered features	GridSearchCV	`n_estimators=300, max_depth=8, min_samples_leaf=5, max_features=0.7`	0.282740	0.081569	0.778482
HistGradientBoosting	All engineered features	GridSearchCV	`learning_rate=0.05, max_iter=75, max_leaf_nodes=7, l2=0.1, min_samples_leaf=40`	0.273264	0.079058	0.804895

Appendix C.3 - perturbation split test



How to read the boxplot:

- Each box shows log loss across 50 match-level split perturbations
- Lower log loss is better
- Orange line = median
- Green triangle = mean
- Box = middle 50% of results
- Whiskers = typical range
- Circles = unusually high/low split results

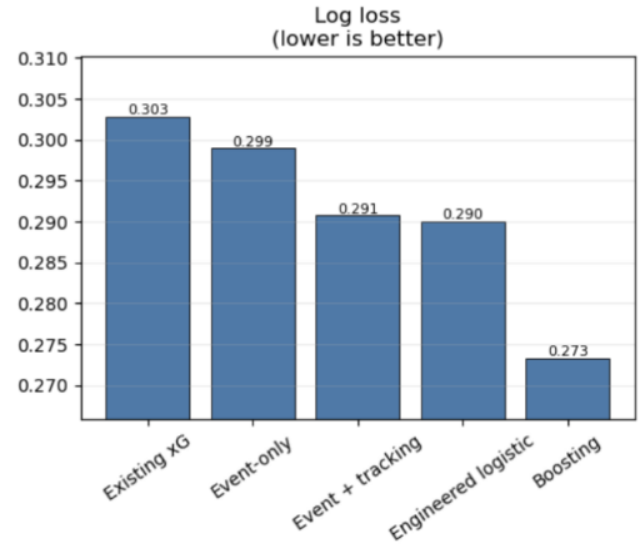
Key takeaway:
HistGradientBoosting has the lowest average log loss, but the overlap with Logistic Regression shows that performance depends on the match split.

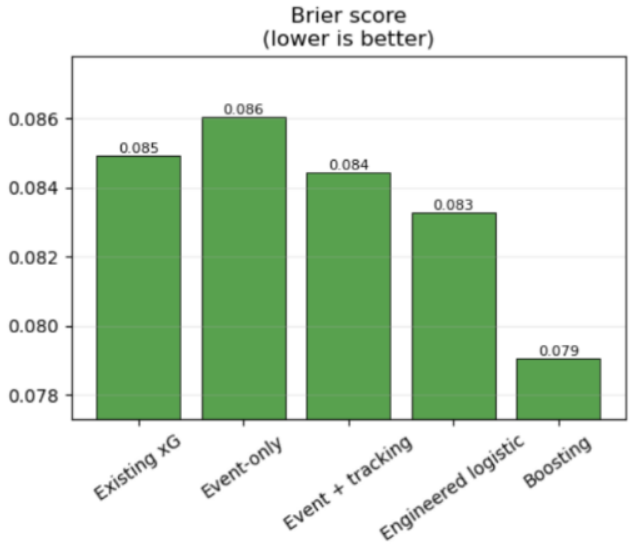
Appendix C.4 - Permutation importance test

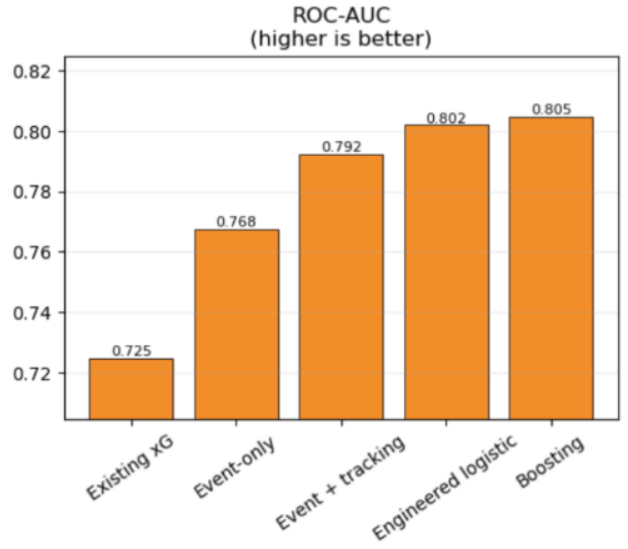
	feature	mean_log_loss_increase	std
3	shot_angle_deg	0.030980	0.008039
8	ball_distance_to_nearest_attacker	0.022904	0.003986
16	goalkeeper_distance_to_ball	0.015218	0.005608
35	previous_event_type	0.006116	0.004046
9	nearest_defender_to_ball	0.005803	0.002235
31	keeper_and_pressure	0.004758	0.003452
33	playerPosition	0.004199	0.001021
17	goalkeeper_distance_to_shooter	0.003686	0.002213
0	x	0.002691	0.001409
6	is_header	0.002604	0.000473
18	goalkeeper_distance_from_goal_line	0.002588	0.001462
29	goalkeeper_distance_ratio	0.002253	0.001980
28	goalkeeper_abs_y	0.001822	0.001964
2	shot_distance_m	0.001579	0.000726
23	angle_per_meter	0.001406	0.001102

```
perm_result = permutation_importance(
    best_model,
    X_test,
    y_test,
    scoring="neg_log_loss",
    n_repeats=10,
    random_state=42,
    n_jobs=1,
)
```

Appendix C.5 - Performance comparison across models





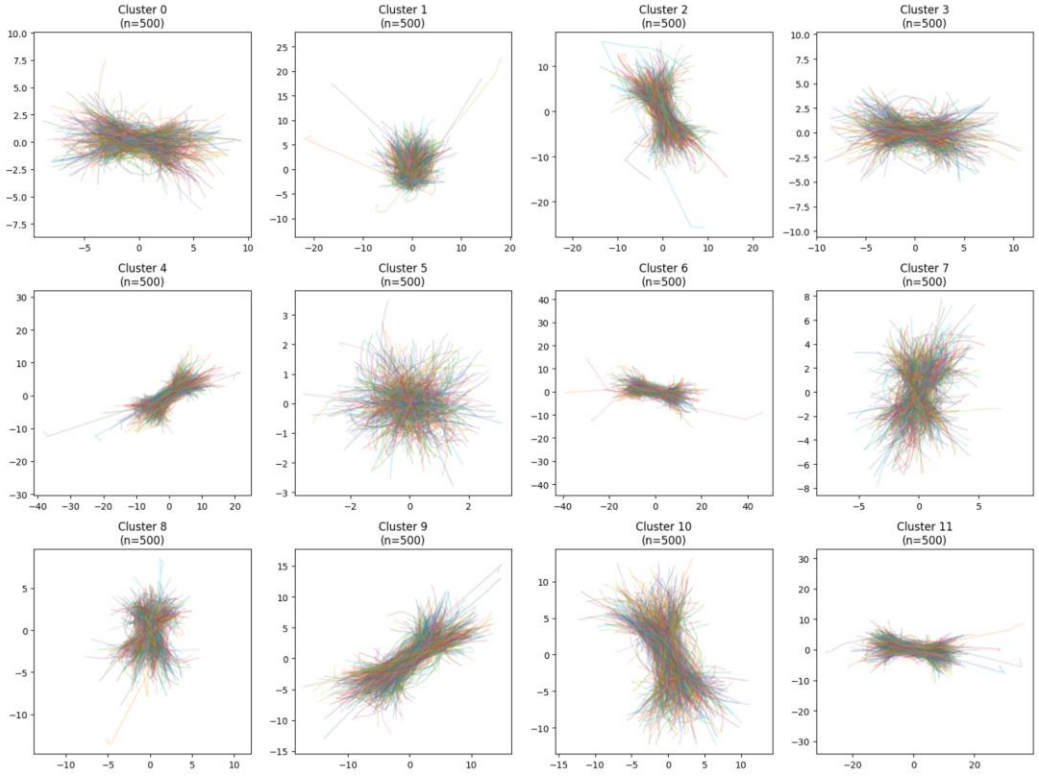


Appendix D

Model descriptions for question:
Can we predict player positions based on movement patterns?

Appendix D.1 - Movement clusters - best model

Movement clusters for best model (2-layered clustering), showed with a maximum of 500 trajectories pr. cluster



Appendix D.2
Most common sequences for each player cluster

Most common sequences of movement clusters for best model:

	cluster_0	cluster_1	cluster_2	cluster_3
0	(5, 5) (12.9%)	(5, 5) (44.3%)	(5, 5) (12.1%)	(5, 5) (9.8%)
1	(0, 0) (4.5%)	(0, 5) (4.7%)	(3, 3) (4.5%)	(0, 0) (3.1%)
2	(7, 7) (3.3%)	(5, 0) (4.3%)	(7, 7) (3.7%)	(7, 7) (3.0%)
3	(7, 5) (2.8%)	(5, 7) (4.3%)	(7, 5) (2.7%)	(3, 3) (2.4%)
4	(0, 5) (2.6%)	(7, 5) (4.3%)	(8, 8) (2.5%)	(7, 5) (2.2%)

Appendix D.3
Best model - Granular Confusion Matrix

We also scraped more granular player position labels (12 positions instead of 4), allowing us to inspect how well our clustering captures finer positional distinctions. The confusion matrix below shows how our k=4 K-means clustering aligns with these finer distinctions.
It clearly makes clusters for the four general positions: GK, DF, MF and FW. It is also apparent that it is good at clustering GK, CB, CM and ST - the most central players for each of those four general positions . It has some trouble with the positions which are between two categories e.g. AM, which is split between the MF and FW cluster. It is also apparent that LB and RB, as well as RW and LW, are similarly appointed to different clusters, indicating that the data pre-processing was successful and that symmetric players are “the same” in the pre-processed data.