NBA Betting

How to beat the bookmakers

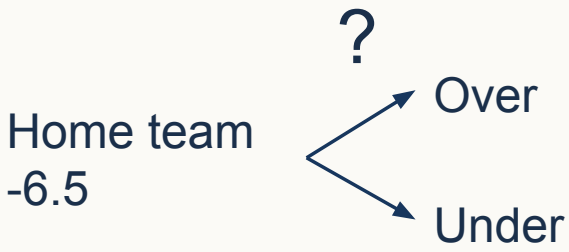
Alfred, Louie and Sebastian

Outline

- Introduction to NBA data
- Regression model on match outcome
- Classification model on match winner
- Betting market exploration
- Betting strategies
- Reinforcement learning - betting bot

Outcome prediction types

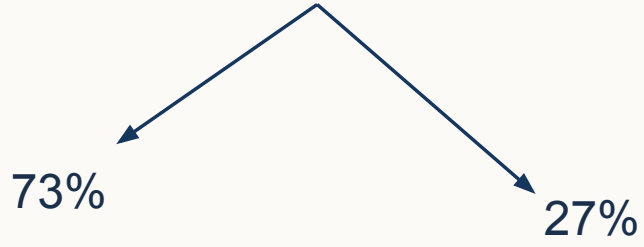
Winning margin (“spread”):



Who wins (“moneyline”):

Home team	vs	Away team
x1.39	vs	x3.57

Implied winning probability



The data

- Official NBA Data: 1946-
- NBA Data from stats page: 1986-
- Betting data: 2007-2025
- Injury reports: 2018-2025

Home team

Game #	Offensive rating	Field goal pct.	….
Game 1	109.3	52.7%	
Game 2			
…	…	…	
…	…	…	
Game N-1			

Away team

Game #	Offensive rating	Field goal pct.	….
Game 1	102.5	48.2%	
Game 2			
…	…	…	
…	…	…	
Game N-1			

Game ID	Home off rating	Home def rating	…	Away off rating	Away def rating	…		
Game N								

Regression model to predict: *winning margin*

Split

Training seasons: 2020-21, 2021-22, 2022-23 (3068 games)

Validation season: 2023-24 (1074 games)

Test season: 2024-25 (1073 games)

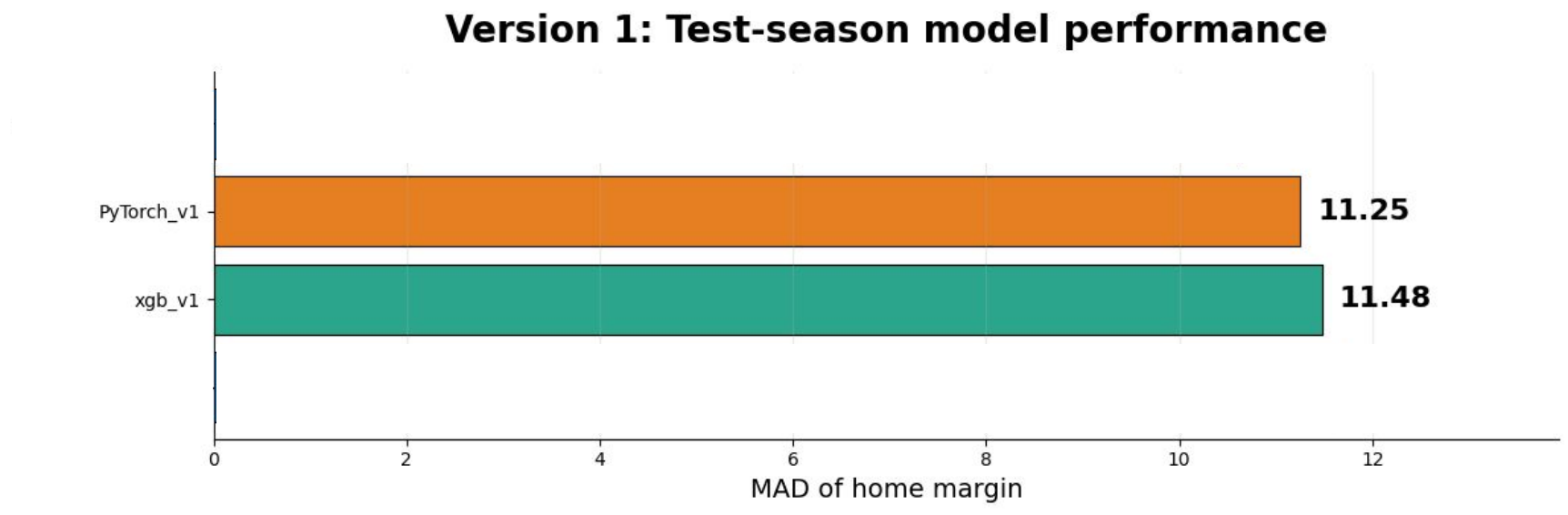
General strategy

- Bayesian HP optimization
- Tried multiple GBDT´s and NN´s
- Chose just XGBoost and PyTorch
- Performance evaluation: MAD (Actual diff in score of game - Predicted diff in score)

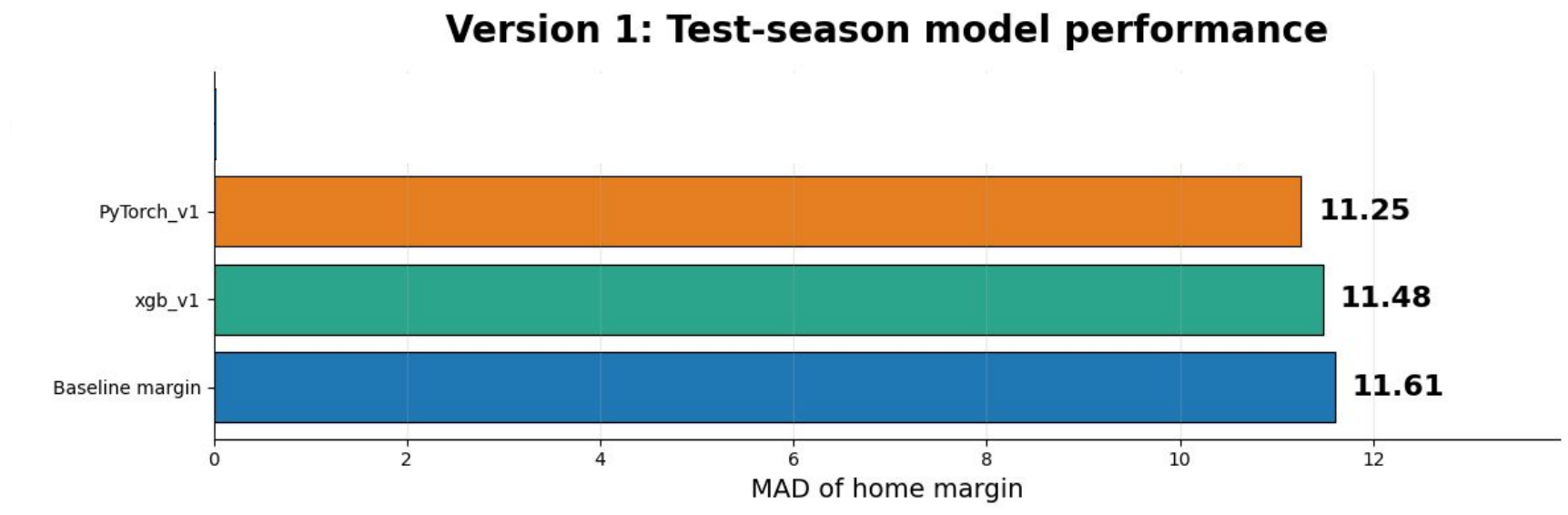
Version 1

Throw all variables into a regression model and optimize

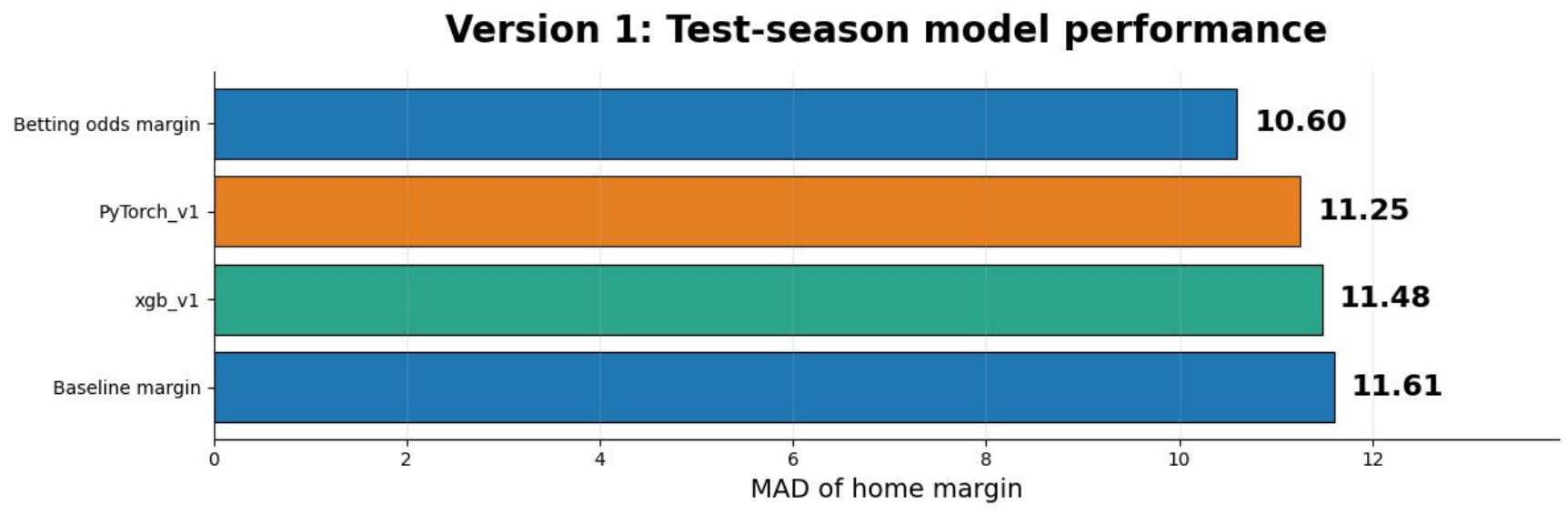
Version 1 performance



Version 1 performance



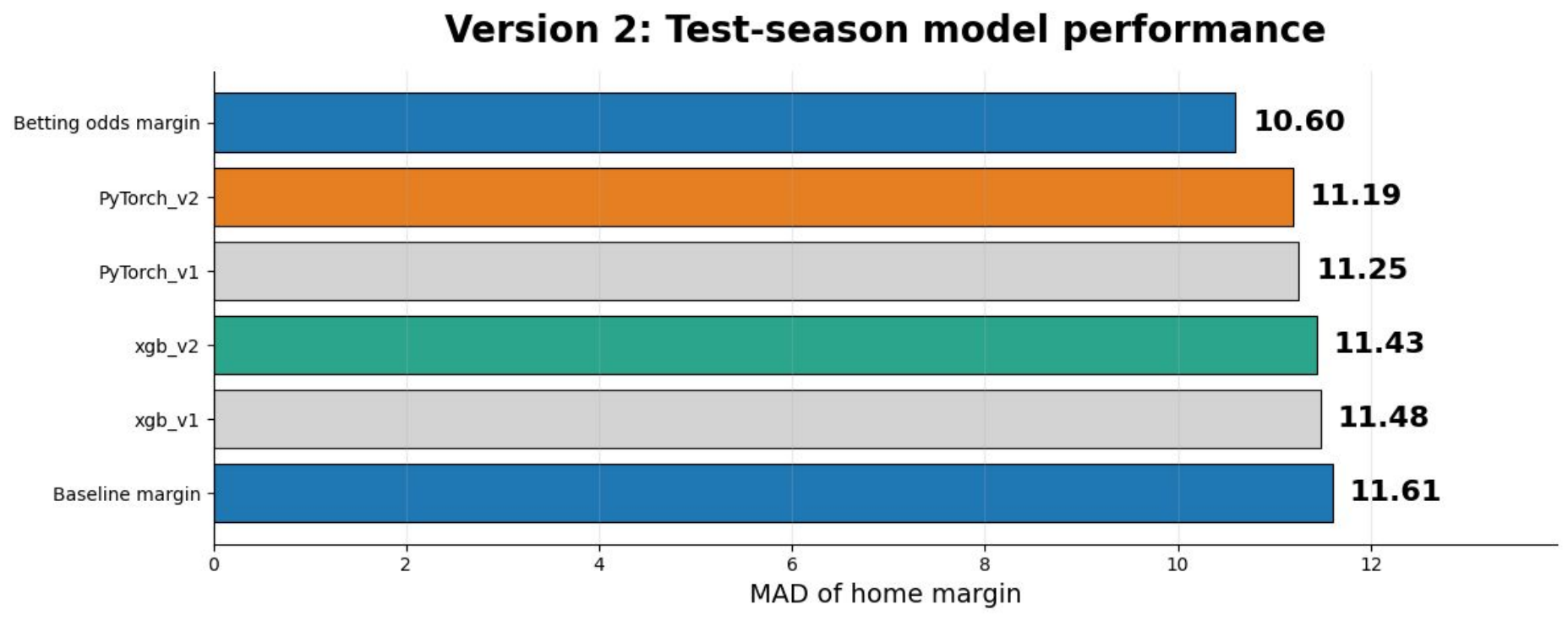
Version 1 performance



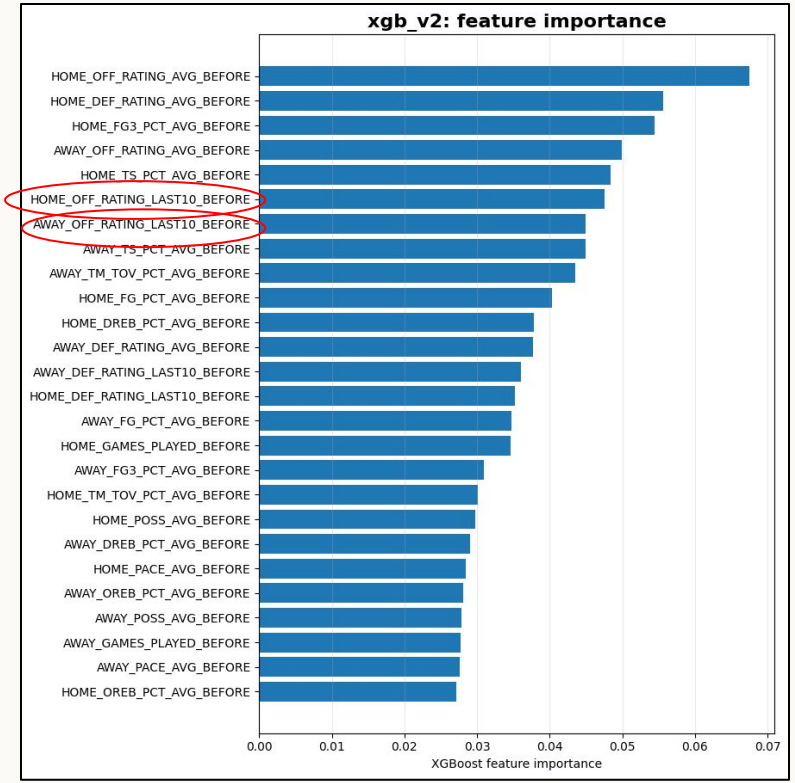
Version 2

Rolling features added - last 10 game off/def efficiency

Version 2 performance:



Version 2 - feature importance



Version 3

Injury impact

Version 3 - injury impact

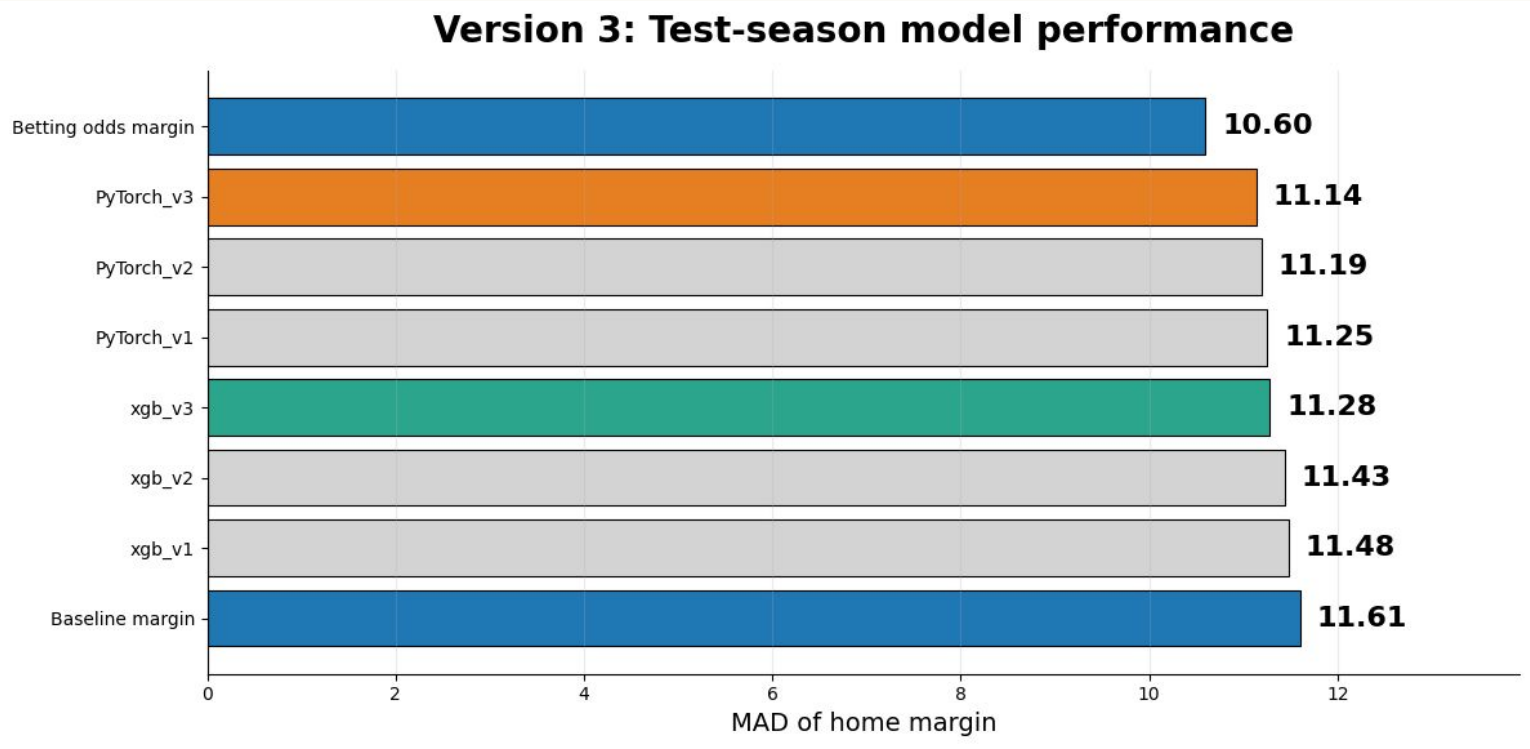
Injury report:

“LAL vs DEN - 27-11-2022 - LeBron James – Out – Left ankle soreness”

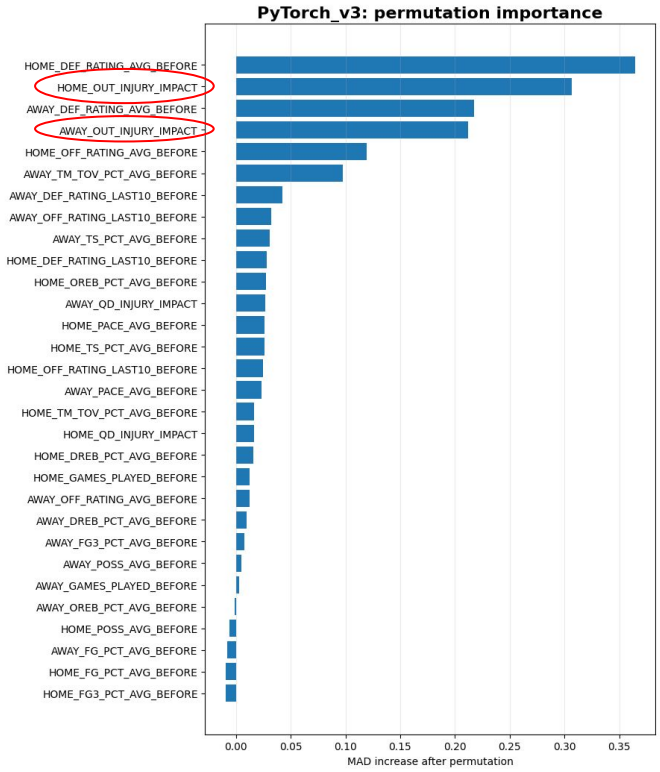
“Missing value” = Player Impact Estimate x Average minutes played

Add column “missing value” and “maybe missing value” columns to model

Version 3 – performance



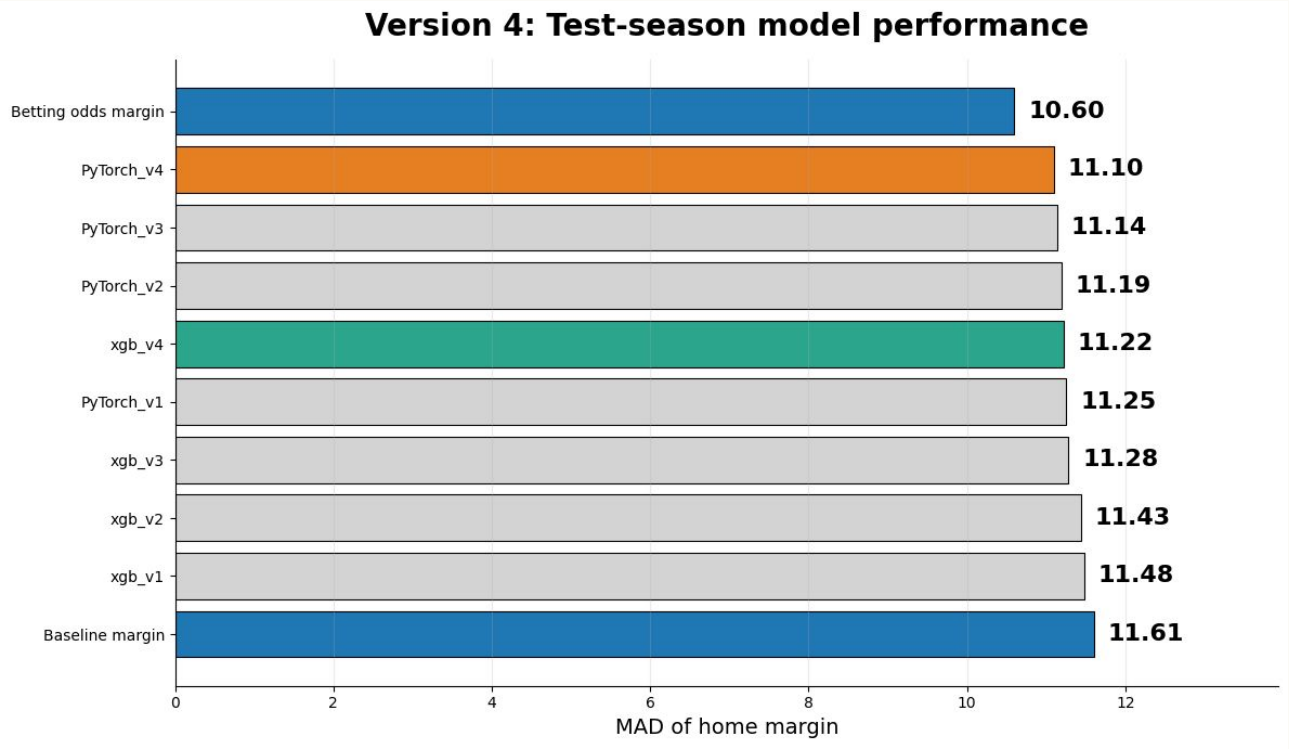
Version 3 - permutation importance



Version 4

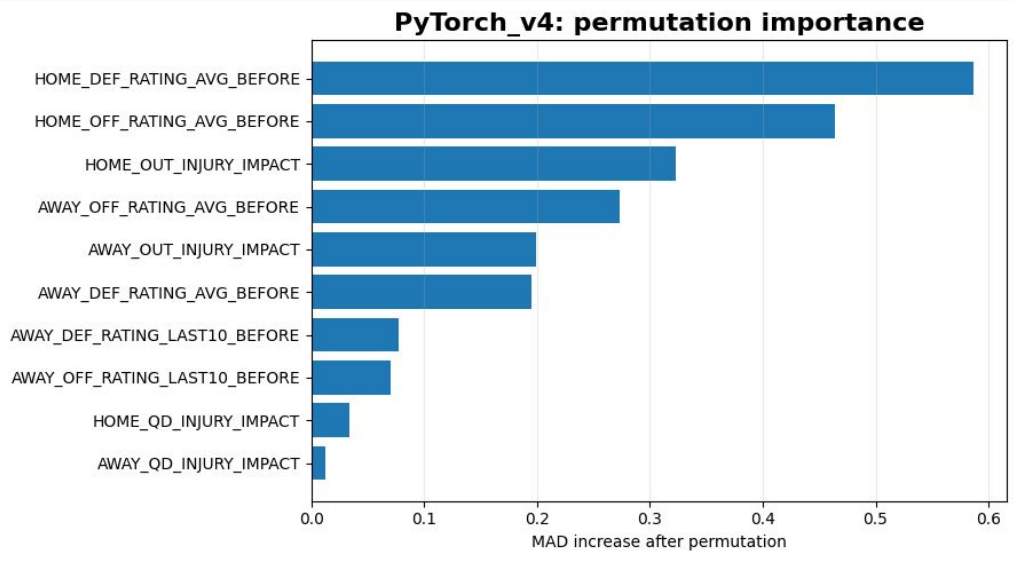
Reducing number of features

Version 4 – performance



Best model: a PyTorch neural network!

- Only 10 variables
- 1 hidden layer, 16 nodes wide
- Other architectures?



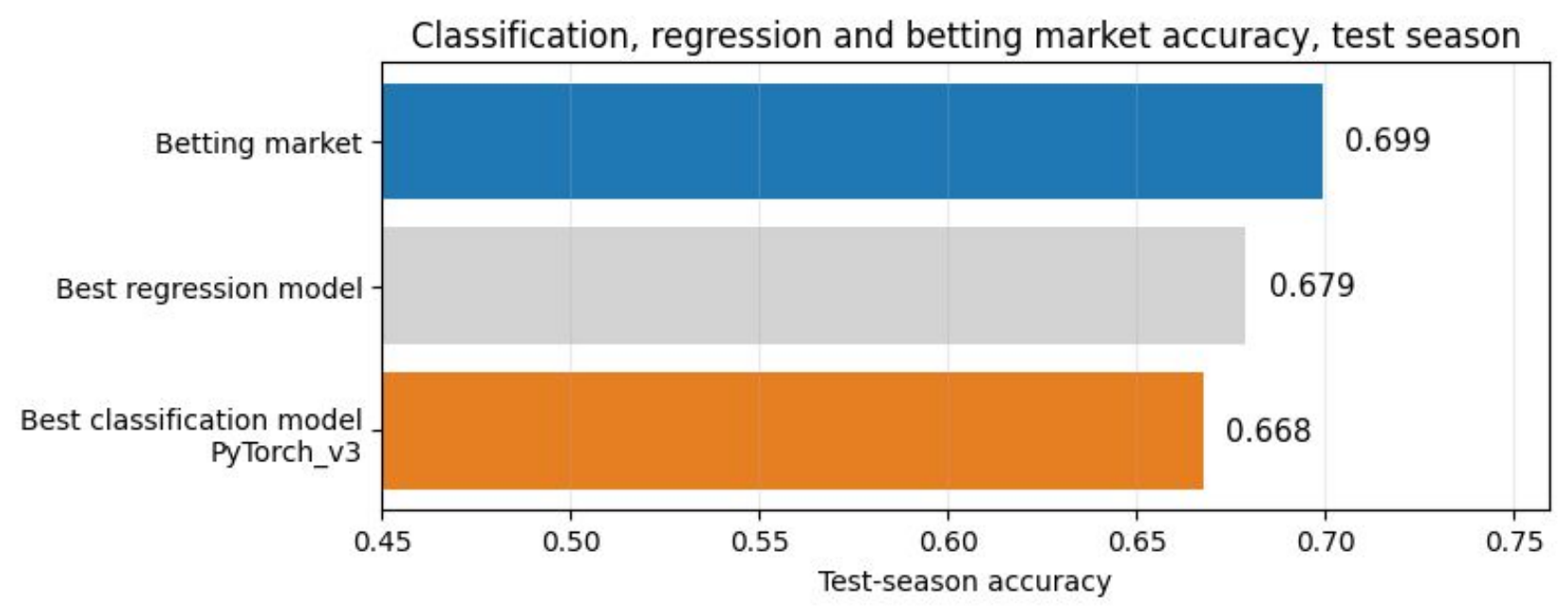
Classification

Classification - results

Same method as for regression. Best model compared with betting market:

Model	Log loss	ROC AUC
Betting market implied prediction model	0.577	0.761
Best model (PyTorch)	0.594	0.741

Classification - results



Predicting mistakes in the betting market

Regression on betting market error

- **Betting market error = (actual score - betting market prediction)**
- Models get many inputs, including our best regression model and team names etc.
- Same strategy

DID NOT WORK

Strategy searching - An elaborate preprocessing

Are there certain betting strategies that are advantageous?

Can we use ML to exploit this?

For example: Always bet on underdogs (bad odds, but huge potential profit).

Expanding the strategy search

Combine strategies:

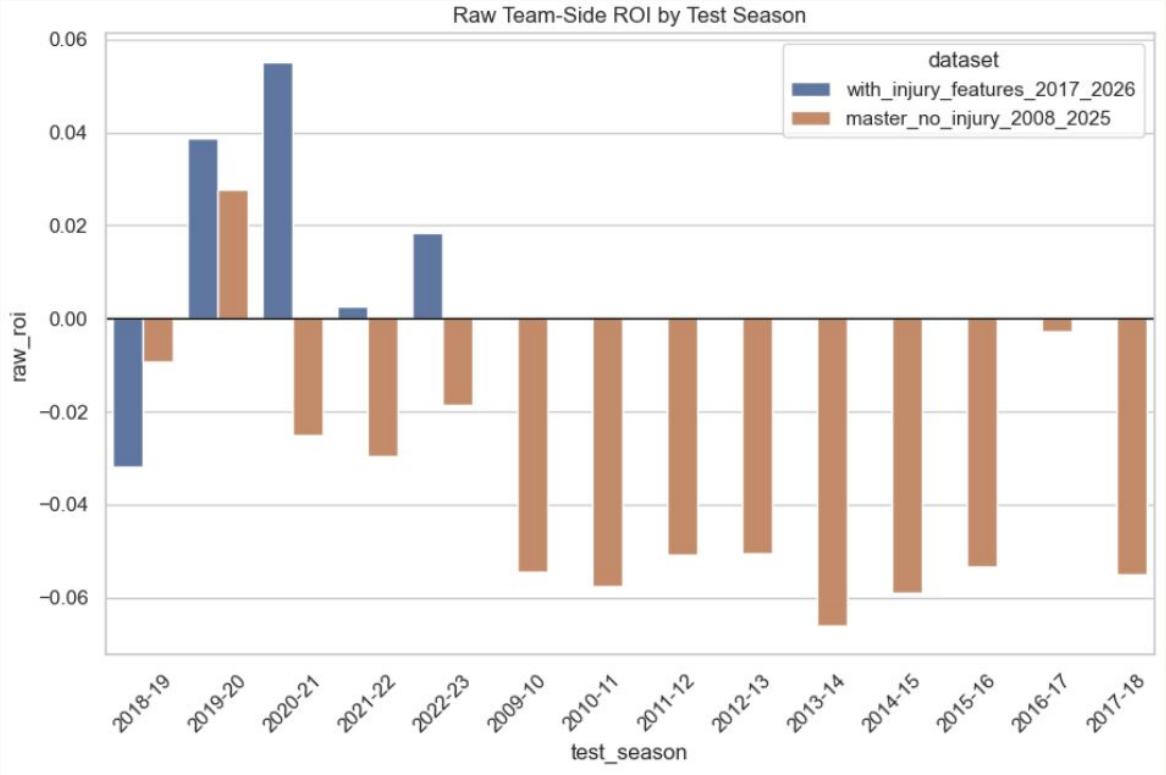
Roughly 200 game strategies X 12 market strategies = 2400 combined strategies

Game strategies by percentiles = [0.1, 0.2, 0.3, 0.7, 0.8, 0.9]

- “Is this teams defense rating at this point in the 70th percentile?”

Results of expanded strategies

Not really anything meaningful



Applying ML

Can Machine Learning help us choose among potential bets?

Applying ML

Can Machine Learning help us choose among potential bets?

Split	Seasons	Total games in raw season
Strategy train	2017-18, 2018-19	2,460
Validation	2019-20	1,059
Hidden / test	2020-21, 2021-22	2,310

Expanding the strategy search

Combine strategies:

Roughly 200 game strategies X 12 market strategies = 2400 combined strategies

Game strategies by percentiles = [0.1, 0.2, 0.3, 0.7, 0.8, 0.9]

- “Is this teams defense rating at this point in the 70th percentile?”

Expanding the strategy search

Combine strategies:

Roughly 200 game strategies X 12 market strategies = 2400 combined strategies

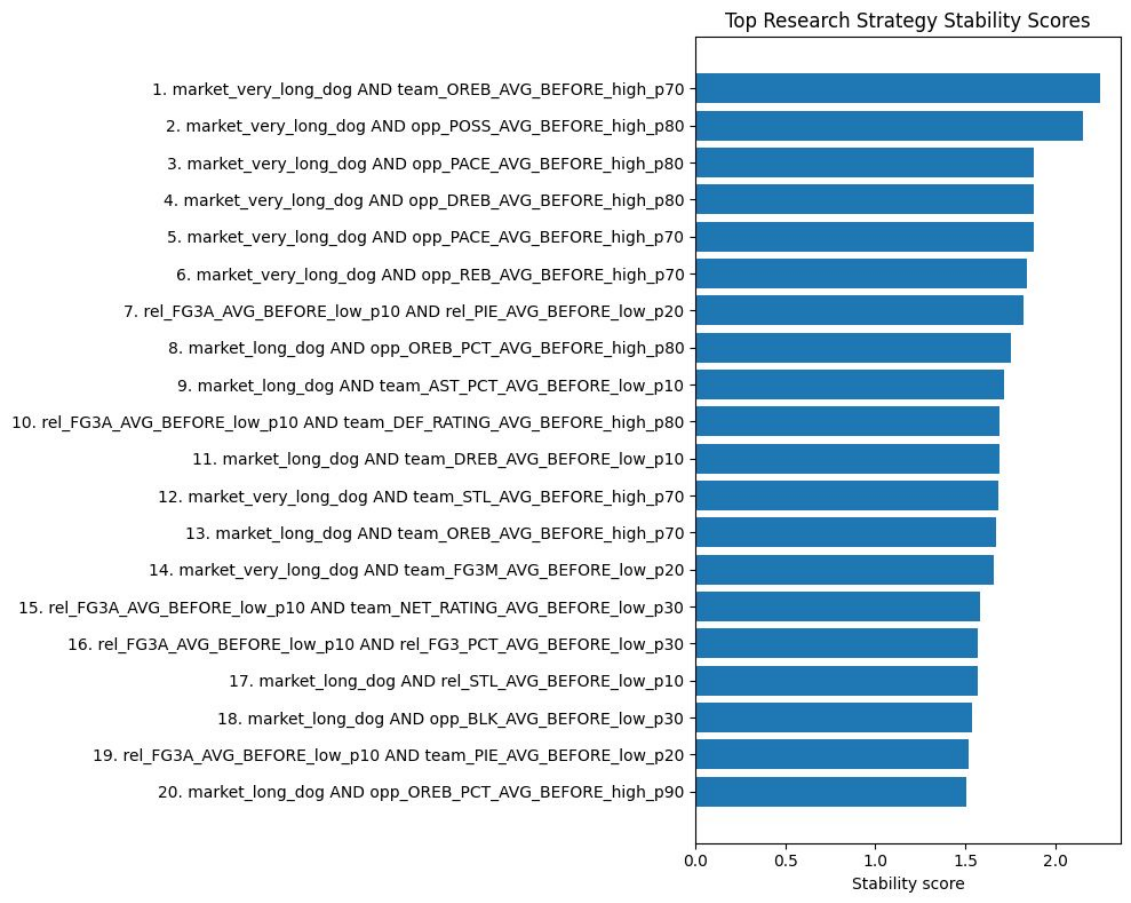
Game strategies by percentiles = [0.1, 0.2, 0.3, 0.7, 0.8, 0.9]

- “Is this teams defense rating at this point in the 70th percentile?”

$$\text{Stability Score} = \frac{\text{Mean ROI across Seasons} \times \sqrt{\text{Number of Bets}}}{1 + \text{Standard Deviation of ROI across Seasons}}$$

Scores of expanded strategies

Choose the top 40!



Betting on all “candidate bets”

1589 bets

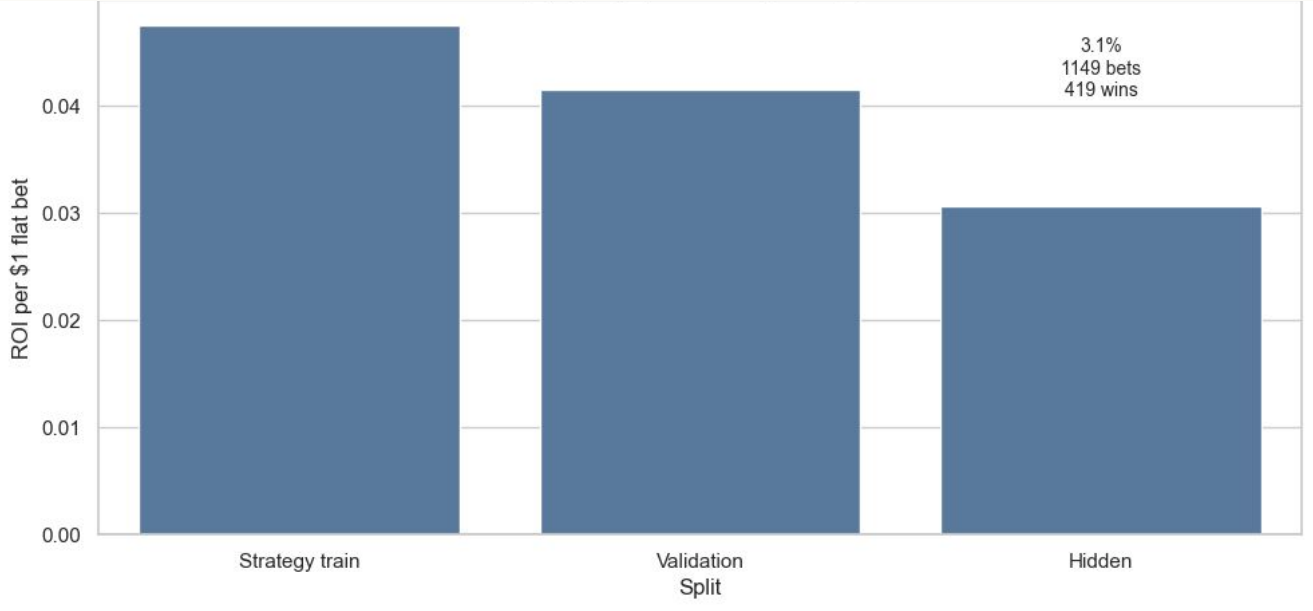
584 wins (36.7%)

768 bets

290 wins (37.8%)

1149 bets

419 wins (36.4%)



Betting on Expected profit - A Classification problem

$$\text{Expected Profit} = P(\text{Win}) \times \text{Profit From Bet} - (1 - P(\text{Win}))$$

Rank bets according to highest expected profit

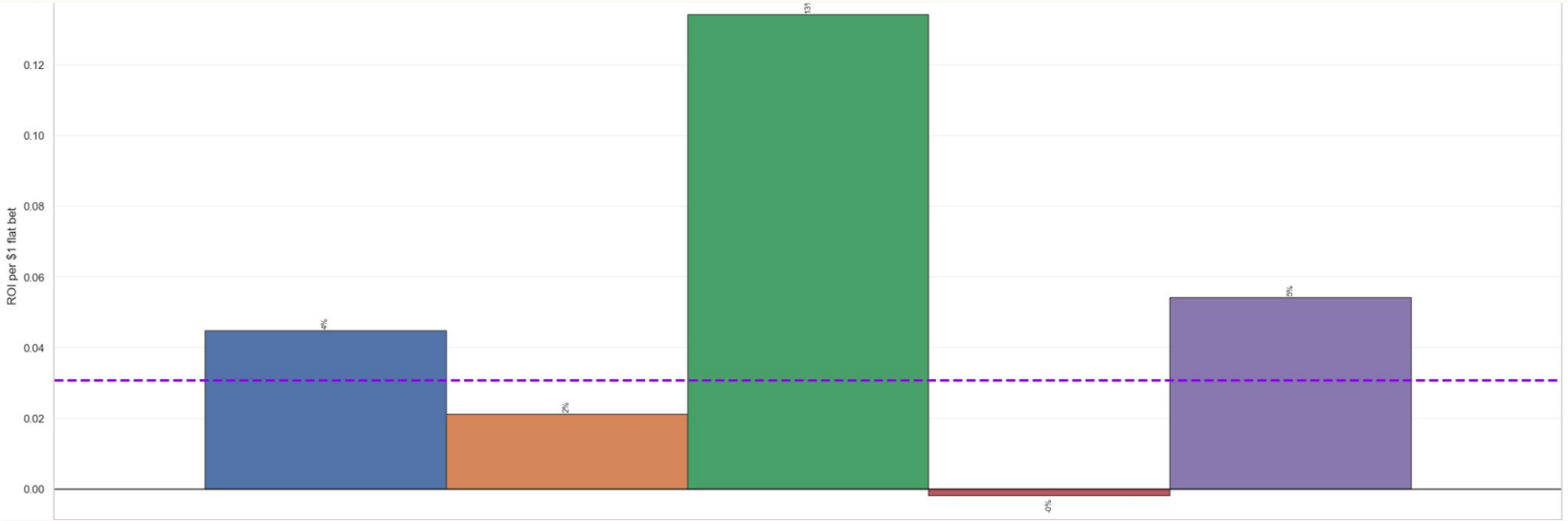
Betting on Expected profit

ML model color legend

- Ridge
- Random forest
- Gradient boosting
- Hist gradient boosting
- XGBoost

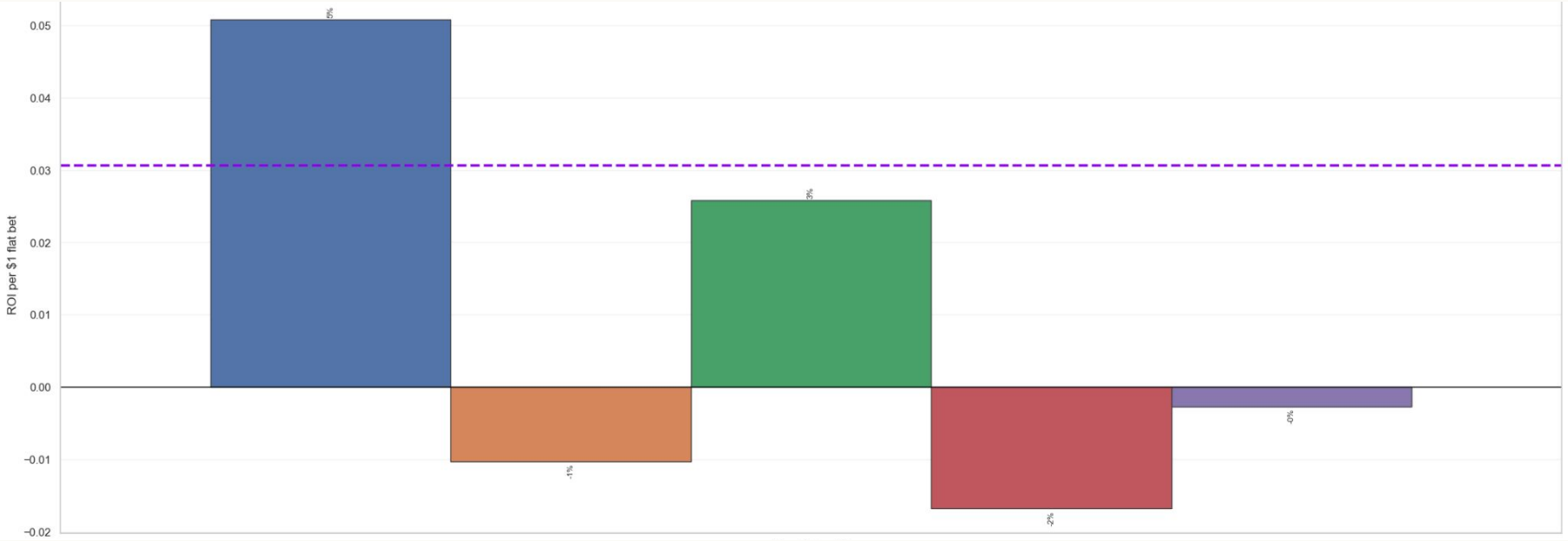
Top 20%

230 bets



Betting on Expected profit

Top 40% 460 bets



Actually just a classification problem, win/no win.

Actually just a classification problem, win/no win.

Can we make ML train explicitly on profit using regression?

Actually just a classification problem, win/no win.

Can we make ML train explicitly on profit directly using regression?

Probably very susceptible to noise

Similar looking data will have vastly different profit_true:

Eg: -1, -1, -1, +3.5, -1, -1, +7, -1

Preprocessing on flat profit regression - two approaches

1 Cap the flat profits, remove extreme outliers

→ Only use profit in range [-1, +5]

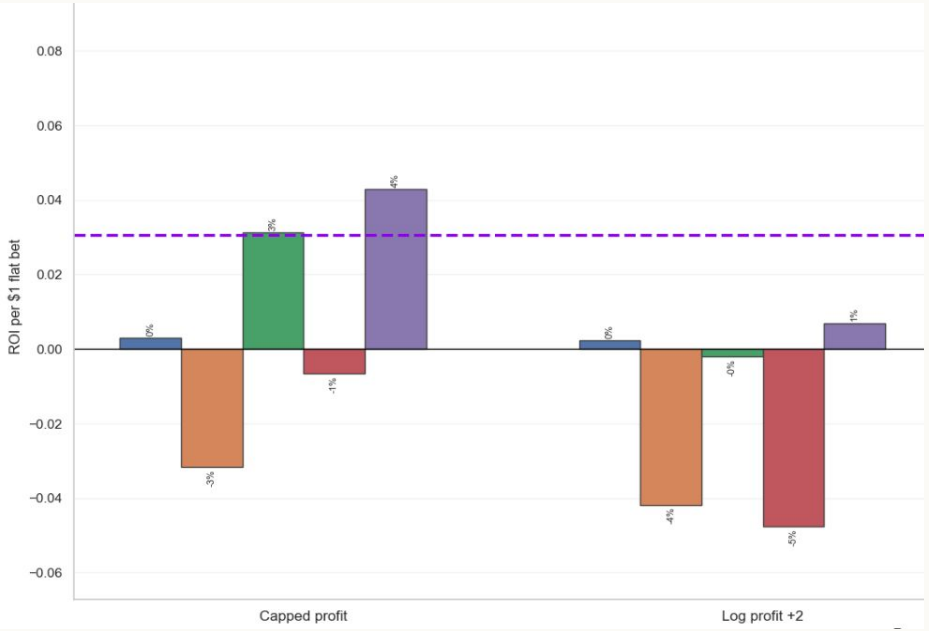
2 log (flat profit + 2)

LOSS → 0

WIN → slightly larger positive number

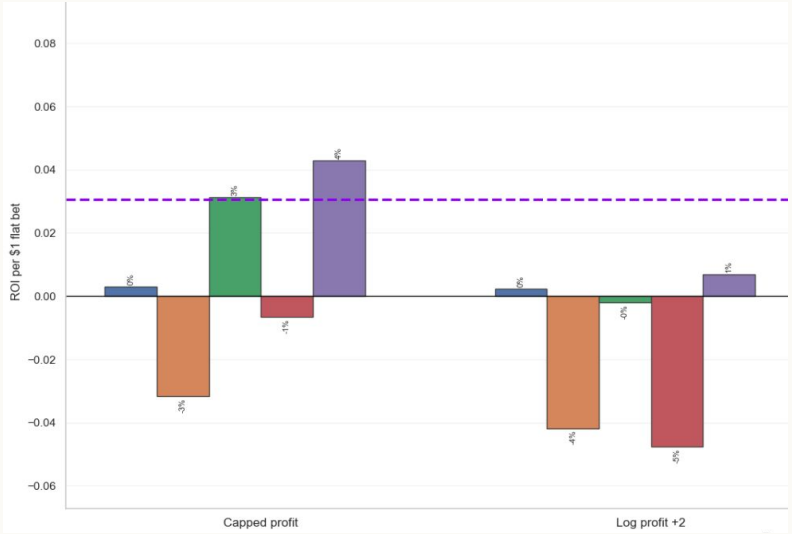
Preprocessing on flat profit bets

Top 40% 460 bets

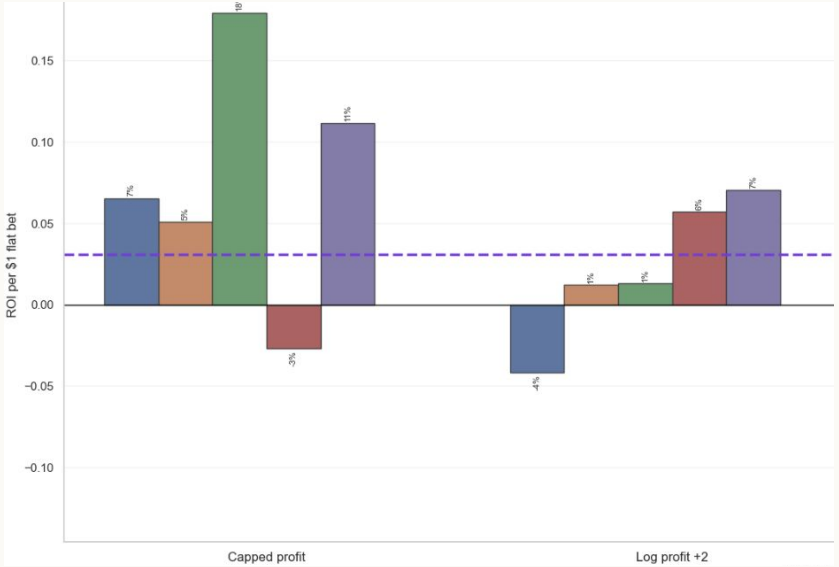


Preprocessing on flat profit bets

Top 40% 460 bets



Top 20% 230 bets



Reinforcement learning

Contextual bandit on winning margin

Contextual bandit

Inputs:

For each game:

- Margin prediction from best GBDT and NN model
- Betting market prediction
- Injury impact difference
- …

Contextual bandit

Inputs:

For each game:

- Margin prediction from best GBDT and NN model
- Betting market prediction
- Injury impact difference
- …

Choose action:

1. No bet
2. Bet home 0.5 units
3. Bet home 1 units
4. Bet home 2 units
5. Bet away 0.5 units
6. Bet away 1 units
7. Bet away 2 units

Contextual bandit

Inputs:

For each game:

- Margin prediction from best GBDT and NN model
- Betting market prediction
- Injury impact difference
- …

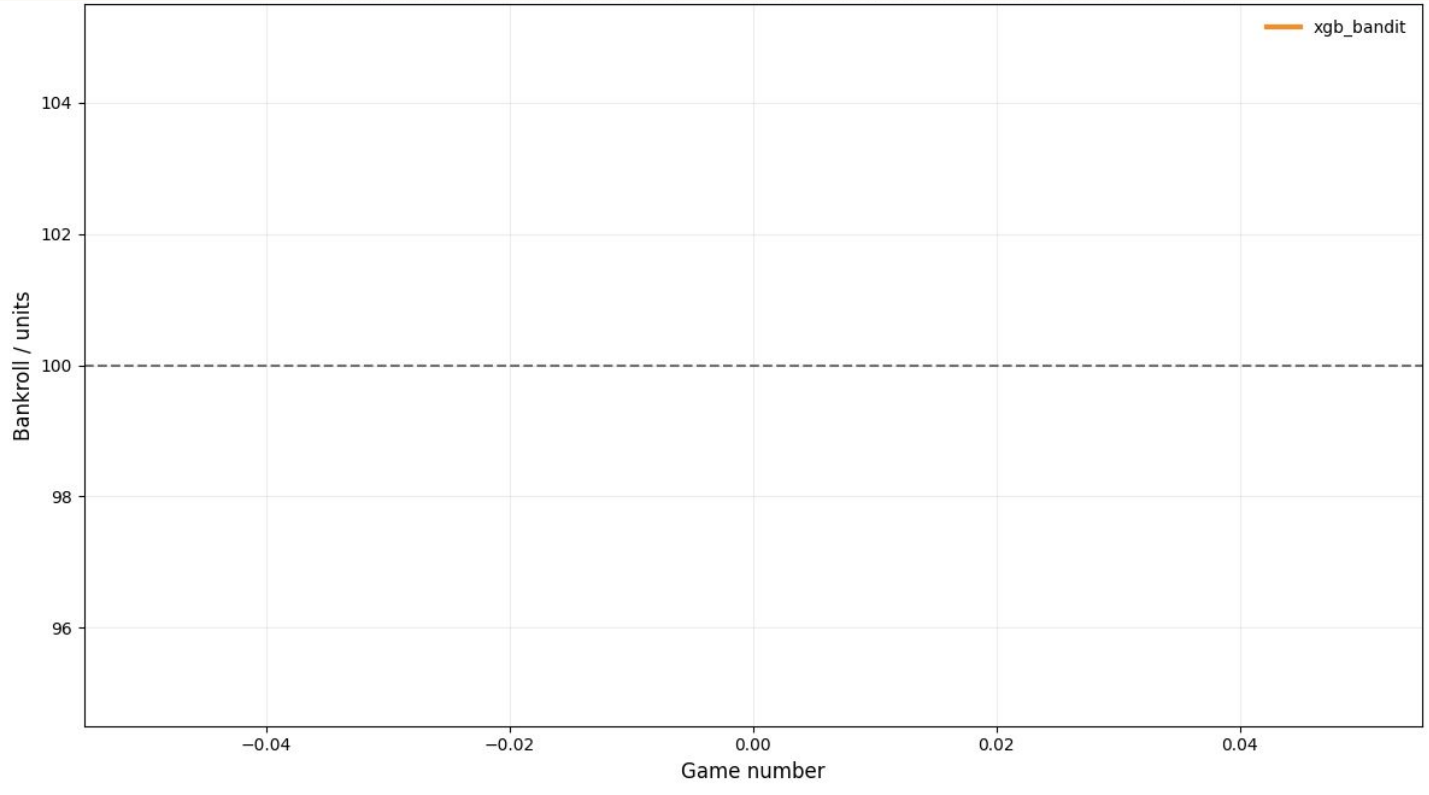
Choose action:

1. No bet
2. Bet home 0.5 units
3. **Bet home 1 units**
4. Bet home 2 units
5. Bet away 0.5 units
6. Bet away 1 units
7. Bet away 2 units

Get reward:

- If win: +0.91
- If loss: -1

Contextual bandit



Contextual bandit

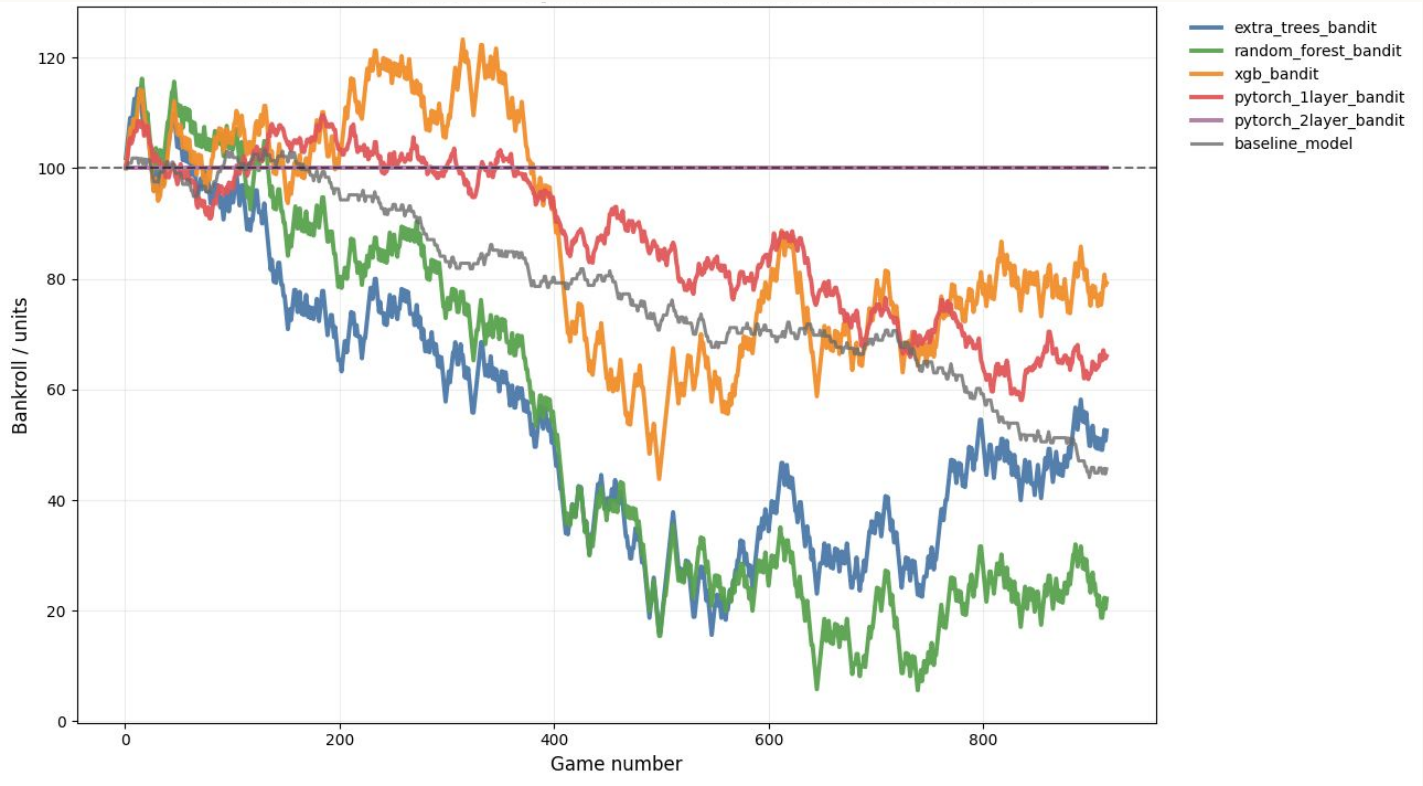
224% profit?!

Contextual bandit

~~224% profit?!~~

Leakage!

Contextual bandit



Conclusion

Good luck beating the bookmakers!

Appendix

Further investigations

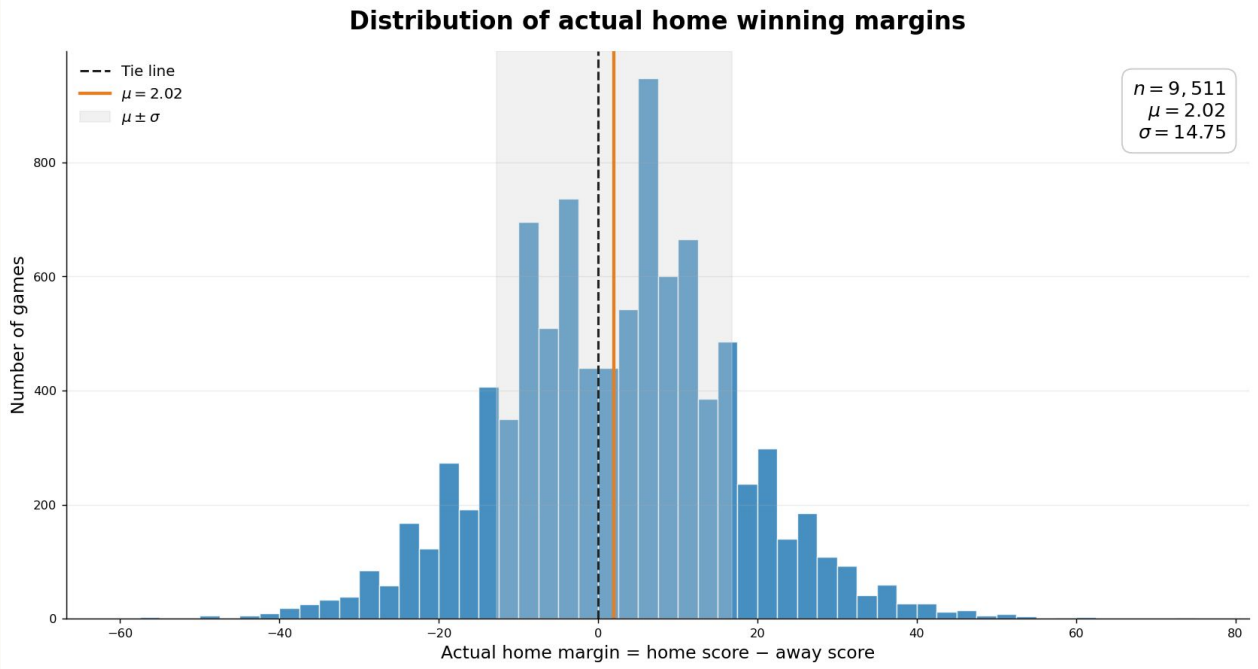
- More DATA! Especially data with odds for more seasons
- Better outlier handling: use other target/loss function, removal of outliers etc.
- Better model architecture/ more HP optimization
- Add more information to model: back-to-back games, more complex recent form data or averages (maybe time series could work, if it doesnt get confused by all the variance)
- Work on less optimized bets, such as specific “player props”, fx. how many points a player scores in a game
- Clustering on player/team embeddings to see if model can use matchups of different styles to predict score i.e small ball team vs big players.
- Clustering on where the market is wrong (this was tried, but didn’t show any noticeable results)

Winning margin data spread

As seen NBA game outcomes have a very large spread of σ=14.75

A lot of the observed variance is likely due to factors not present in the dataset. This puts a limit on the performance of our regression models.

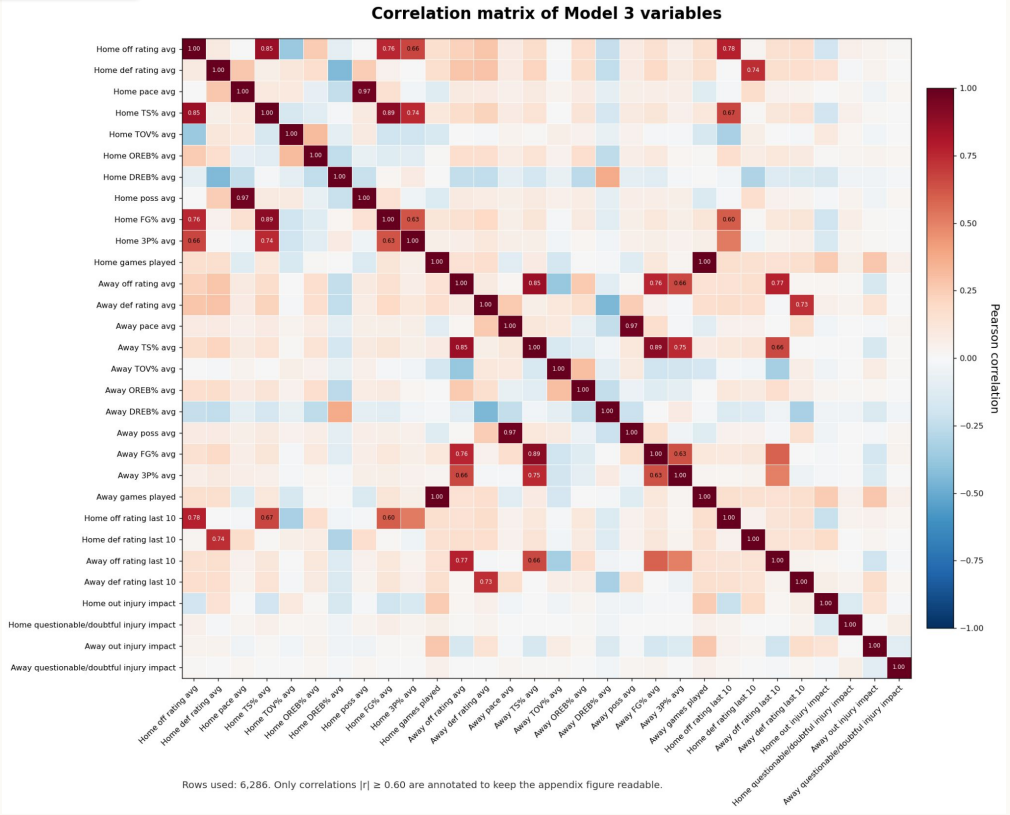
We are aiming to find the features that can lessen the variance as much as possible.



Correlation matrix

The correlation matrix clearly shows that most variables have small, but some with a quite large correlation.

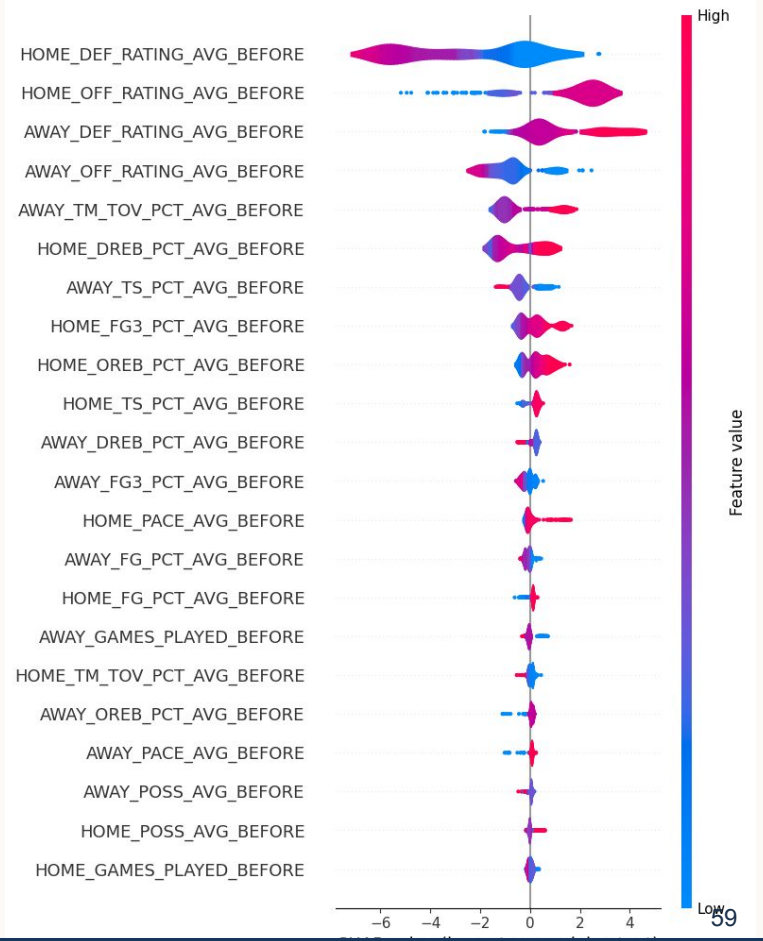
This should have taken more into account, when selecting variables.



Version 1 - XGBoost analysis

The SHAP value violin indicates - as should be expected - that variables such as the offensive and defensive rating (how efficient the team is at generating points and not allowing points) are dominating the regression model.

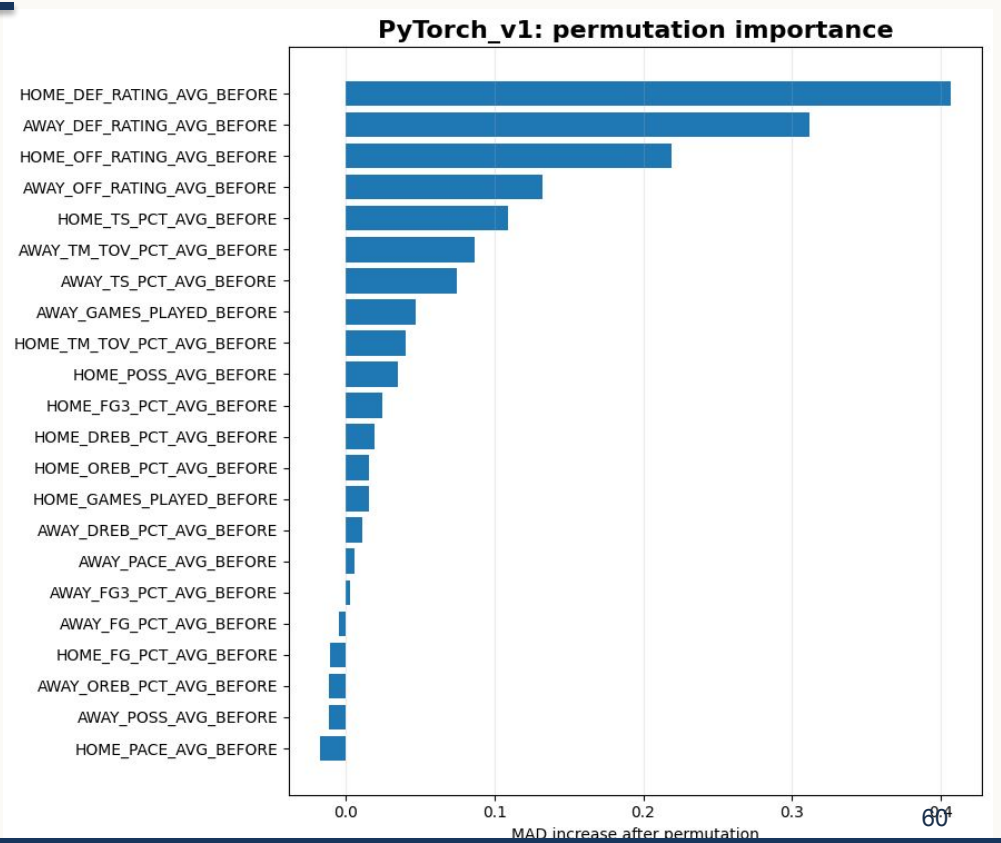
Other variables such as number of games played before the current game are essentially negligible.



Version 1 - PyTorch analysis

Same pattern shows up for the permutation invariance for the PyTorch model. Many not very useful variables.

And we see negative MAD change after permutation of some variables that indicate overfitting of unimportant variables!



Version 3 - injury impact - extra details

Player Impact estimate formula: $\mathbf{PIE = PIE_{raw} / (\sum_{\text{all players in game}} PIE_{raw})}$

$$PIE_{raw} = PTS + FGM + FTM - FGA - FTA + 0.5DREB + OREB + AST + STL + 0.5BLK - PF - TO$$

(PTS = Points, FGM = Field Goals Made, FTM = Free Throws Made, FGA = Field Goal Attempts, FTA = Free Throw Attempts, DREB = Defensive Rebounds, OREB = Offensive Rebounds, AST = Assists, STL = Steals, BLK = Blocks, PF = Personal Fouls, TO = Turnovers)

Players can be either “Out” meaning they do not play, or “Questionable”/”Doubtful” meaning they might play. So in total four columns are added to the data, a sum of the “Out”-players PIE and a sum of the “Questionable”/”Doubtful”-players PIE for both teams.

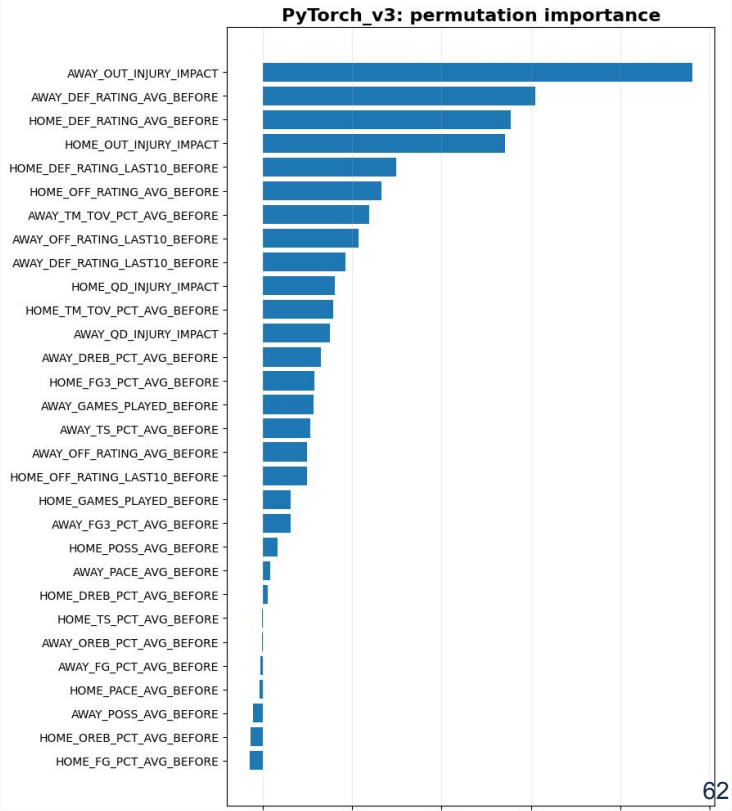
Philosophy and method of limiting number of features for version 4

Similarly to version 1, performing permutation importance on PyTorch_v3 we see a large amount of probably unnecessary variables.

To select the variables for the final model (v4) we used therefore used recursive feature selection. With this method we could reduce the number of features one at time and be able analyse the changes it made each time on the validation set results.

This was however mixed with some manual selection using domain knowledge and the correlations of variables, where both the XGBoost SHAP/gain values and the PyTorch feature permutation results where taken into account.

This was done to get the most sensible final model variables.



Parameters of best winning margin regression model

Version 4 Regression (PyTorch_v4)

- Model: PyTorch MLP Regression
- Architecture: 1 hidden layer (16 nodes)
- Blocks: Linear → ReLU → BatchNorm → Dropout
- Loss: L1 / MAD
- Optimizer: Adam
- Learning rate: 0.00473
- Weight decay: 0.00981
- Training: 120 epochs, patience 8

Parameters of best classification model

Classification (PyTorch_v3)

- Model: PyTorch MLP Classifier
- Architecture: 2 hidden layers (32 → 16 nodes)
- Activation: ReLU
- Loss: Binary Cross Entropy
- Optimizer: AdamW
- Learning rate: 0.00164
- Weight decay: 0.000762
- Training: 300 epochs, patience 30

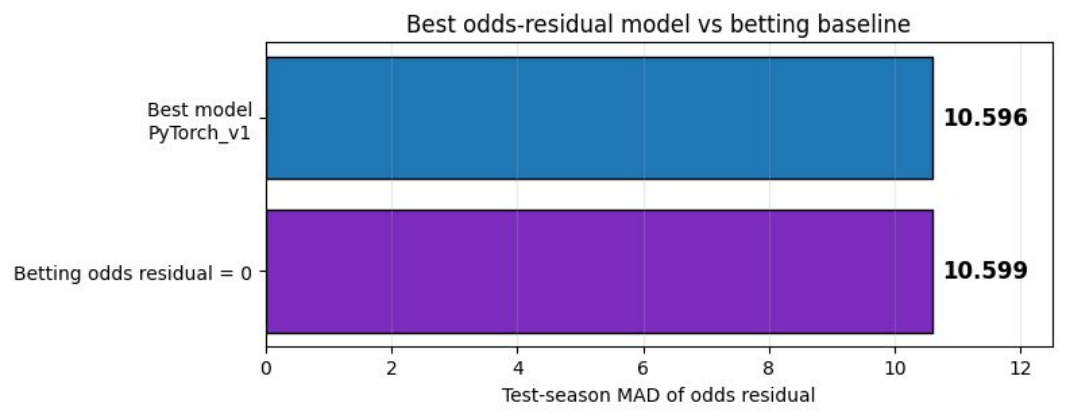
Regression on betting market error: DETAILS

This model gets many inputs, including the same team stats, our best regression model and team names. The same HP bayesian optimization is done, but not with an extensive enough feature selection we have to admit. Both an XGBoost and PyTorch solution was attempted.

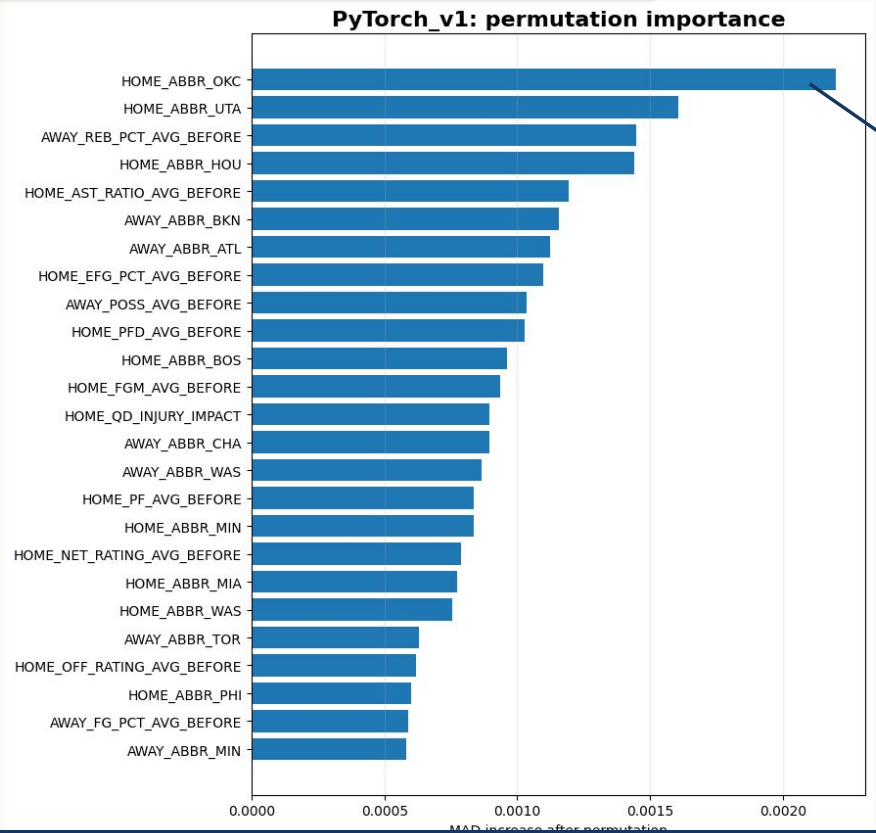
For the model to indicate it is better than the market, it has to get a MAD of less than the earlier betting market regression model, as it otherwise could simply just accept the betting model as the best model by setting *(actual score - betting market prediction) = 0* for all games and get a MAD score of exactly what the betting market gets.

Turns out our model does beat the betting market MAD! Barely:

This is most likely just randomness. Next slide investigates what happens inside the best model.



Regression on betting market error: DETAILS



Best variable:
"Home team named OKC":
MAD increase = 0.0023

This means that permuting the best variable only increases MAD by a miniscule amount, and this is most likely just randomness as some variable has to be the highest.

Meaning the model does not find any meaningful pattern among the basic data it has.

Parameters of best market error regression model

Market Error Regression (PyTorch_v1)

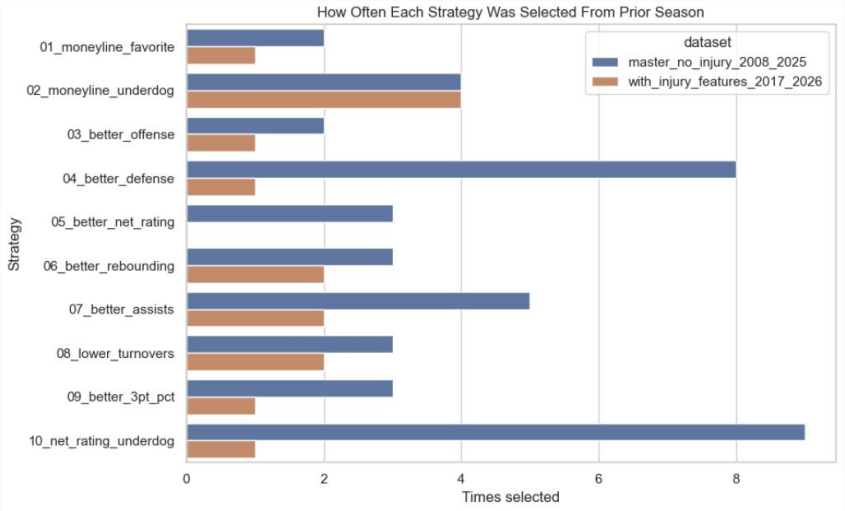
- Model: PyTorch MLP Regression
- Architecture: 1 hidden layer (128 nodes)
- Activation: ReLU
- Loss: L1 / MAD
- Optimizer: AdamW
- Learning rate: 0.000460
- Weight decay: 0.00916
- Training: 200 epochs, patience 35

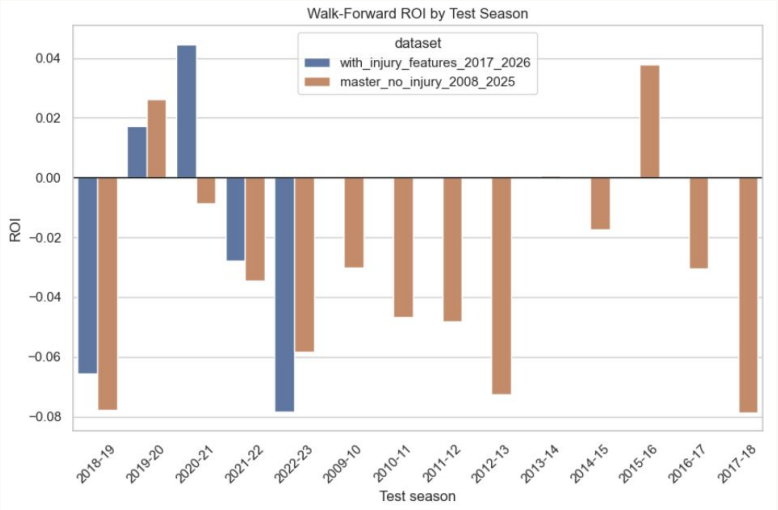
Strategy search - Pipeline

- Generate and run strategies
- Keep those with highest and most solid ROI evaluated over seasons
- **Only use bets where one of top strategies may be applied (Candidate universe / Betting candidates)**
- Train ML regression model to predict Bookmaker error / P(WIN) / Profit
- Select bets from ML analysis
- Simulate betting on these games

Initial strategy test

This was an initial strategy test, only trying out 10 stand_alone strategies/criteria. It does seem that our more complicated expanded strategy search was an improvement, at least for what concerns stability of ROI.





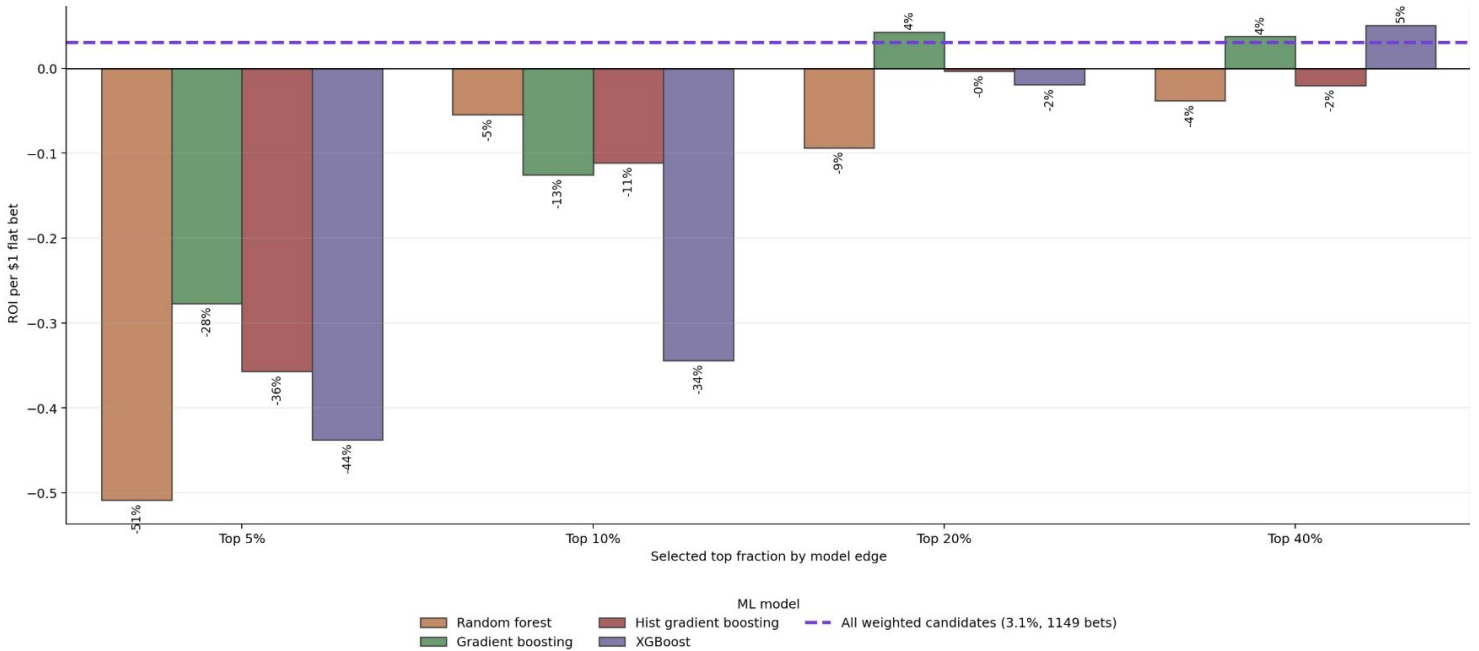
Strategy searching - example bets

```
================================================================================
Game: CLE at UTA  |  Date: 2021-03-29  |  Season: 2020-21
BET: CLE (away) at moneyline +1100
Result: LOSS  |  Profit: -1.000 units
Bookmaker implied probability: 0.083
Raw signal count: 26  |  Weighted score: 25.626

Why this was a candidate bet:
- frozen_signal_001: market_very_long_dog AND team_OREB_AVG_BEFORE_high_p70
  Bet row team moneyline >= +400 AND team_OREB_AVG_BEFORE >= research p70 (10.71)
- frozen_signal_004: market_very_long_dog AND opp_DREB_AVG_BEFORE_high_p80
  Bet row team moneyline >= +400 AND opp_DREB_AVG_BEFORE >= research p80 (36.03)
- frozen_signal_006: market_very_long_dog AND opp_REB_AVG_BEFORE_high_p70
  Bet row team moneyline >= +400 AND opp_REB_AVG_BEFORE >= research p70 (45.88)
- frozen_signal_007: rel_FG3A_AVG_BEFORE_low_p10 AND rel_PIE_AVG_BEFORE_low_p20
  rel_FG3A_AVG_BEFORE <= research p10 (-7.614) AND rel_PIE_AVG_BEFORE <= research p20 (-0.03932)
- frozen_signal_008: market_long_dog AND opp_OREB_PCT_AVG_BEFORE_high_p80
  Bet row team moneyline >= +250 AND opp_OREB_PCT_AVG_BEFORE >= research p80 (0.2885)
- frozen_signal_010: rel_FG3A_AVG_BEFORE_low_p10 AND team_DEF_RATING_AVG_BEFORE_high_p80
  rel_FG3A_AVG_BEFORE <= research p10 (-7.614) AND team_DEF_RATING_AVG_BEFORE >= research p80 (111.2)
- frozen_signal_012: market_very_long_dog AND team_STL_AVG_BEFORE_high_p70
  Bet row team moneyline >= +400 AND team_STL_AVG_BEFORE >= research p70 (8.273)
- frozen_signal_013: market_long_dog AND team_OREB_AVG_BEFORE_high_p70
  Bet row team moneyline >= +250 AND team_OREB_AVG_BEFORE >= research p70 (10.71)
- frozen_signal_014: market_very_long_dog AND team_FG3M_AVG_BEFORE_low_p20
  Bet row team moneyline >= +400 AND team_FG3M_AVG_BEFORE <= research p20 (9.714)
```

```
================================================================================
Game: ORL at NYK  |  Date: 2021-10-24  |  Season: 2021-22
BET: ORL (away) at moneyline +600
Result: WIN  |  Profit: 6.000 units
Bookmaker implied probability: 0.143
Raw signal count: 25  |  Weighted score: 24.746

Why this was a candidate bet:
- frozen_signal_001: market_very_long_dog AND team_OREB_AVG_BEFORE_high_p70
  Bet row team moneyline >= +400 AND team_OREB_AVG_BEFORE >= research p70 (10.71)
- frozen_signal_002: market_very_long_dog AND opp_POSS_AVG_BEFORE_high_p80
  Bet row team moneyline >= +400 AND opp_POSS_AVG_BEFORE >= research p80 (103.5)
- frozen_signal_004: market_very_long_dog AND opp_DREB_AVG_BEFORE_high_p80
  Bet row team moneyline >= +400 AND opp_DREB_AVG_BEFORE >= research p80 (36.03)
- frozen_signal_006: market_very_long_dog AND opp_REB_AVG_BEFORE_high_p70
  Bet row team moneyline >= +400 AND opp_REB_AVG_BEFORE >= research p70 (45.88)
- frozen_signal_007: rel_FG3A_AVG_BEFORE_low_p10 AND rel_PIE_AVG_BEFORE_low_p20
  rel_FG3A_AVG_BEFORE <= research p10 (-7.614) AND rel_PIE_AVG_BEFORE <= research p20 (-0.03932)
- frozen_signal_010: rel_FG3A_AVG_BEFORE_low_p10 AND team_DEF_RATING_AVG_BEFORE_high_p80
  rel_FG3A_AVG_BEFORE <= research p10 (-7.614) AND team_DEF_RATING_AVG_BEFORE >= research p80 (111.2)
- frozen_signal_013: market_long_dog AND team_OREB_AVG_BEFORE_high_p70
  Bet row team moneyline >= +250 AND team_OREB_AVG_BEFORE >= research p70 (10.71)
- frozen_signal_015: rel_FG3A_AVG_BEFORE_low_p10 AND team_NET_RATING_AVG_BEFORE_low_p30
  rel_FG3A_AVG_BEFORE <= research p10 (-7.614) AND team_NET_RATING_AVG_BEFORE <= research p30 (-2.238)
- frozen_signal_016: rel_FG3A_AVG_BEFORE_low_p10 AND rel_FG3_PCT_AVG_BEFORE_low_p30
  rel_FG3A_AVG_BEFORE <= research p10 (-7.614) AND rel_FG3_PCT_AVG_BEFORE <= research p30 (-0.01303)
- frozen_signal_017: market_long_dog AND rel_STL_AVG_BEFORE_low_p10
```

Bookmaker error regression using strategy search

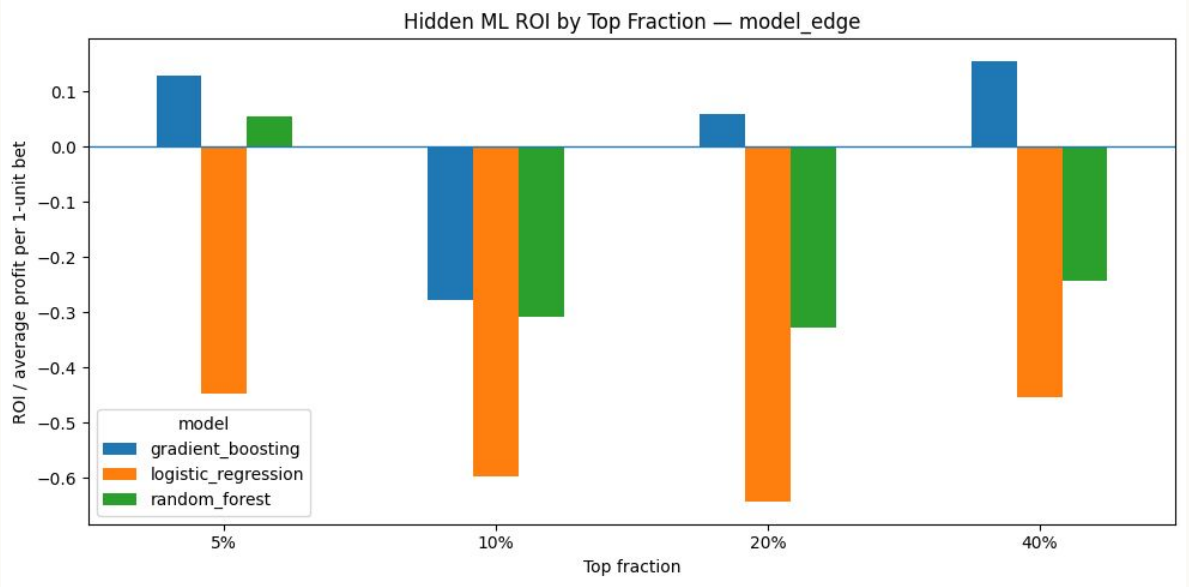
This was the result from trying to rank our bets after most likely bookmaker error. It does not convert well to profit, evidently.



Premier League football - A quick attempt

We attempted constructing a similar strategy search and ML optimization on soccer data, but soccer was much harder to predict. Even just predicting who won with classification was much much worse. Probably due to fewer goals occurring in soccer compared to basketball.

But this is a surface level exploration of simply applying the same strategy to football data.



Contextual bandit - details and extra results

	model	best_optuna_value	validation_reward_mae	validation_reward_mse	validation_final_bankroll	validation_ROI	validation_number_of_bets	best_params
0	xgb_bandit	-0.936957	0.943926	1.506533	113.772727	0.007991	892	{'n_estimators': 560, 'learning_rate': 0.04313...
1	random_forest_bandit	-0.940796	0.950123	1.466890	83.272727	-0.010016	835	{'n_estimators': 250, 'max_depth': 6, 'min_sam...
2	extra_trees_bandit	-0.945719	0.953428	1.425816	83.272727	-0.010547	793	{'n_estimators': 300, 'max_depth': 8, 'min_sam...
3	pytorch_2layer_bandit	-0.953757	0.953758	1.365973	100.000000	NaN	0	{'hidden_size_1': 48, 'hidden_size_2': 24, 'dr...
4	pytorch_1layer_bandit	-0.955799	0.957782	1.396202	84.409091	-0.021624	808	{'hidden_size': 8, 'dropout': 0.09134614558548...

Parameters of one of the contextual bandit

One example of one the bandits (as all gave similar results)
Contextual Bandit (Random Forest - from SK learn)

- Bandit: Offline Contextual Bandit
- Reward model: Random Forest Regressor
- Actions: 7 betting actions
- Trees: 250
- Max depth: 6
- Min split: 90
- Min leaf: 70
- Metric: Reward MAE (reward function defined by betting odds)

Work contribution - STATEMENT

All participants contributed evenly