Level Difficulty and Player Skill Prediction in Human Computation Games

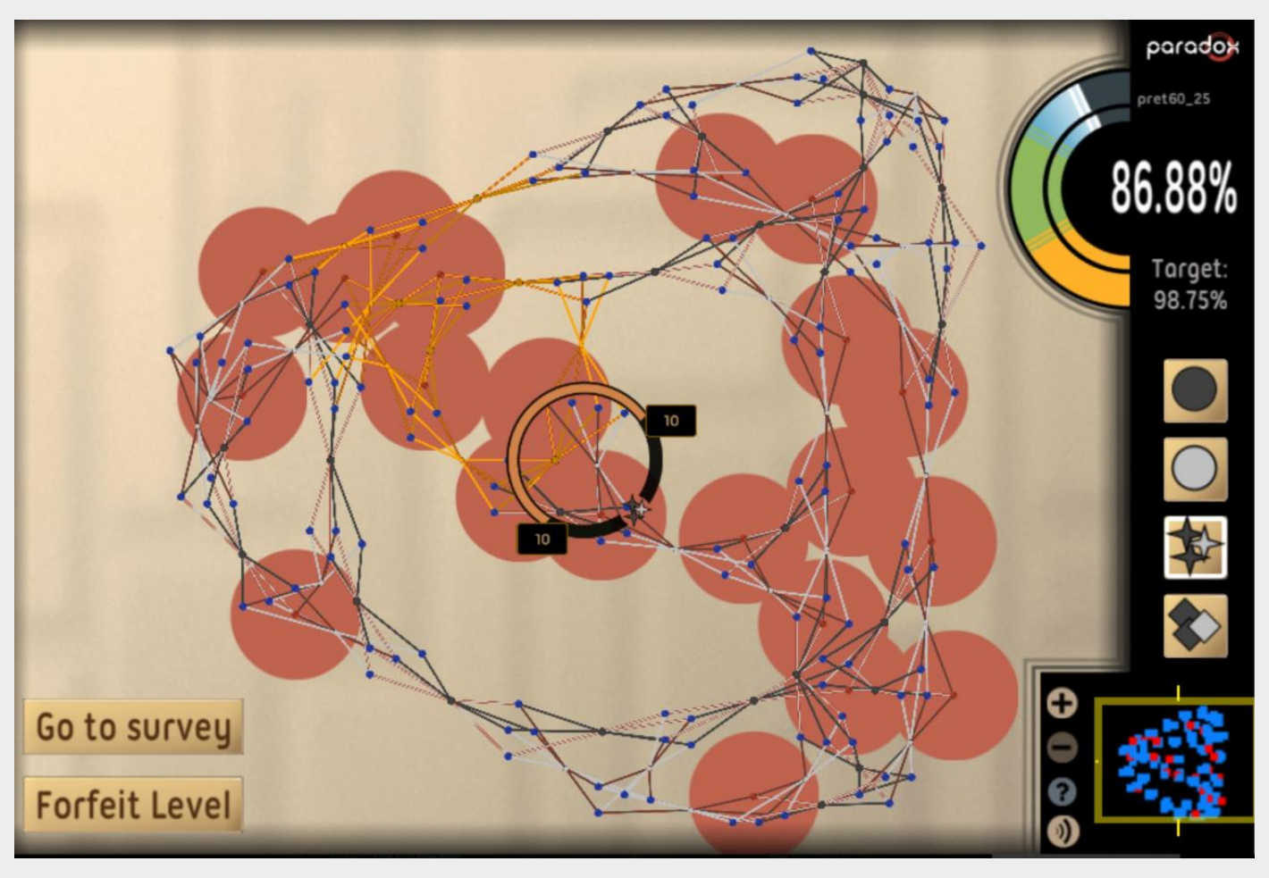
Anurag Sarkar and Seth Cooper
Northeastern University

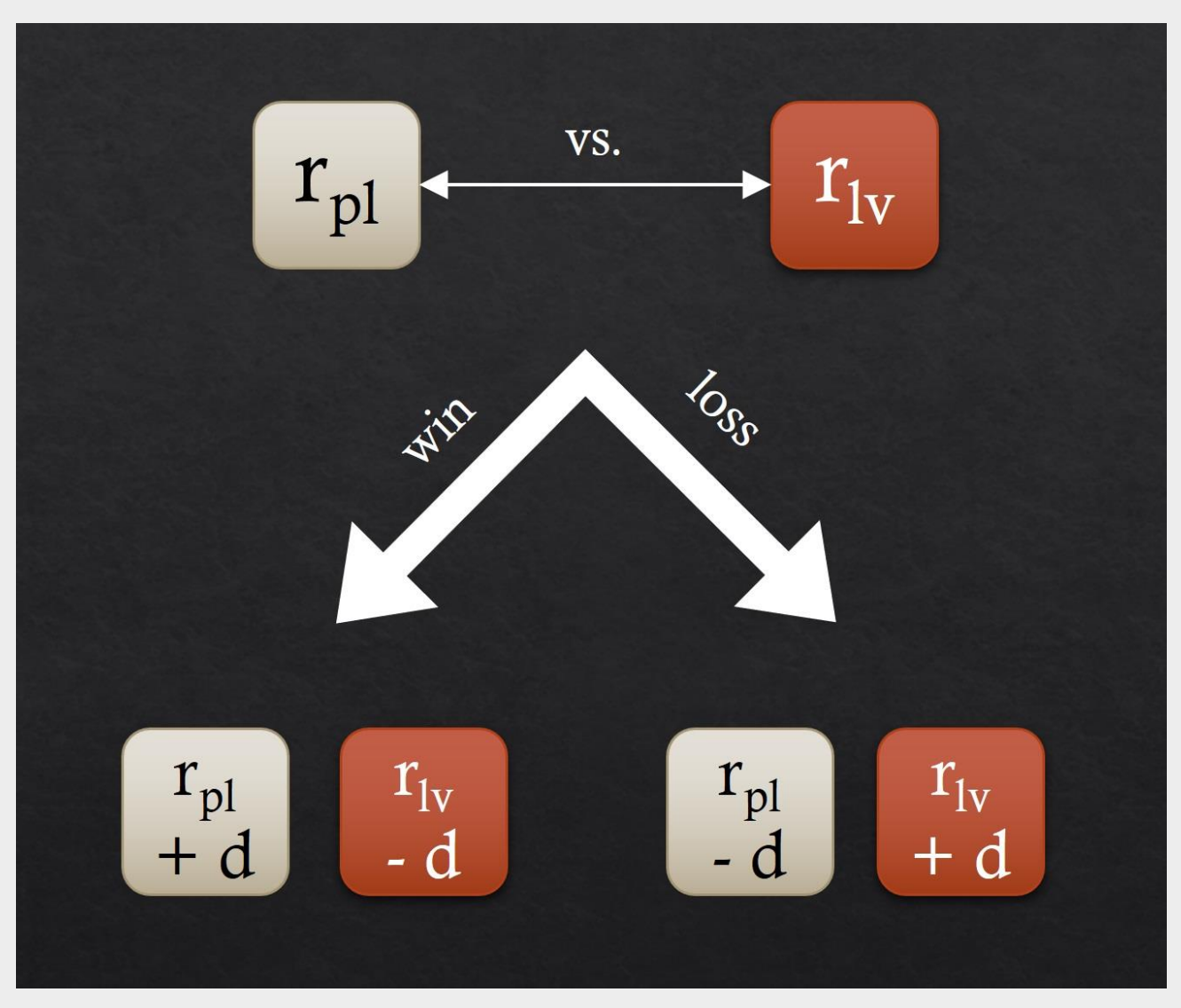
Introduction

- Human computation games (HCGs) often suffer from poor engagement, potentially due to difficulty balancing issues, as levels model real problems and thus can't be arbitrarily modified in advance.
- Player rating systems (e.g. Glicko/Elo) can improve engagement in HCGs by matching players with levels of appropriate difficulty without actually modifying the levels [1].
- However, such systems start players and levels with default ratings. In HCGs, we may have information about players (from tutorials) and levels (from underlying properties) that could inform initial ratings that are more indicative of both player skill and level difficulty.
- **Thus, we examined if player and level features could be used to predict player and level ratings that are closer to their actual, eventual ratings than the default ratings.**

Background

For our study, we used **Paradox** [2], a 2D puzzle HCG designed for crowdsourced formal verification. Each level in the game represents an underlying MAX-SAT problem.

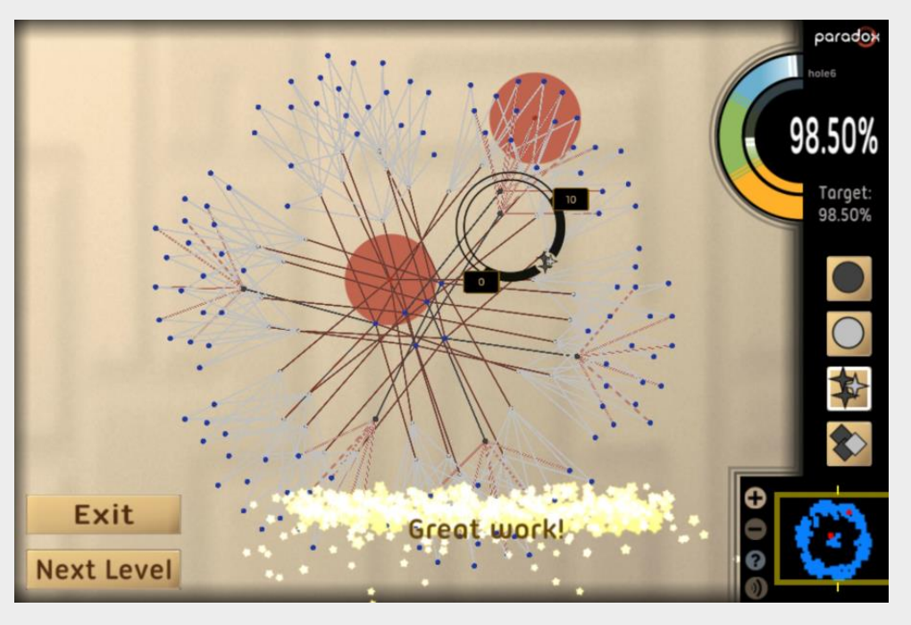




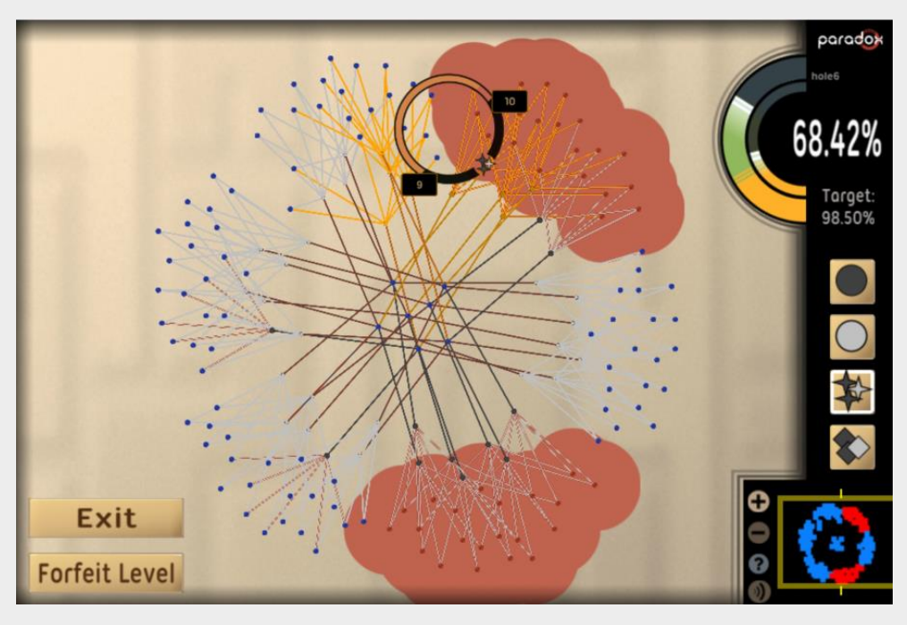
We ran the **Glicko-2 rating system** [3] on gameplay data involving players recruited through MTurk for generating ratings for players and levels. Ratings correspond to skill for players and difficulty for levels, and once generated, may be used to match players of certain skill with levels of comparable difficulty.

Methods

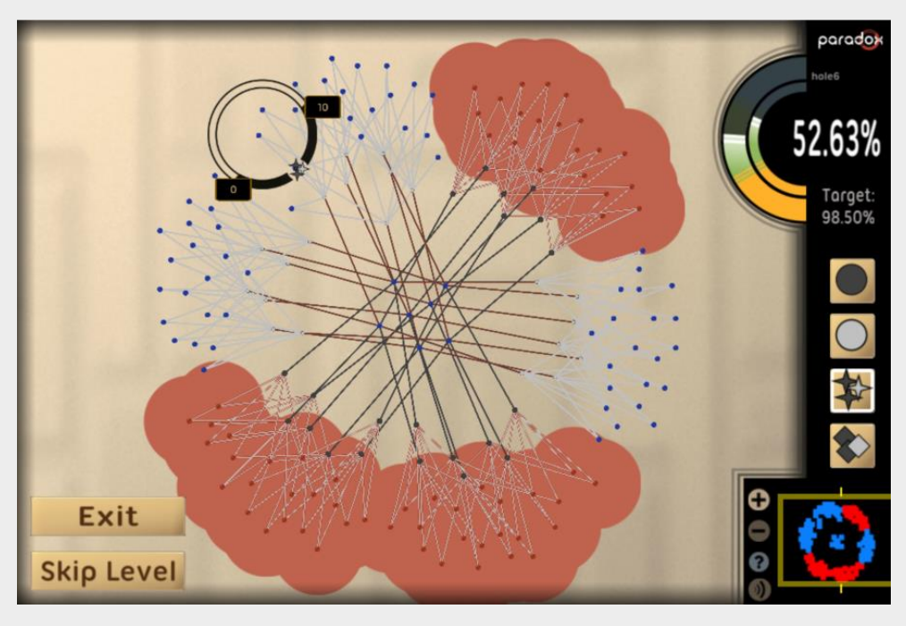
Each player-level pairing was treated as a match with one of 3 outcomes:



Level Completed => Win for Player



Level Forfeited => Win for Level



Level Skipped => Ignored from analysis

Such matches between players and levels were used to generate their actual ratings. For ratings prediction, we used these training features:

Level Features

- # of variables per clause
- # of clauses per variable
- If all clauses satisfied when all variables set to True/False
- % of clauses satisfied when all variables set to True/False
- # of total/variable/clause nodes in level graph
- # of edges in level graph
- % of edges in MST
- Size of graph periphery
- Average shortest path length

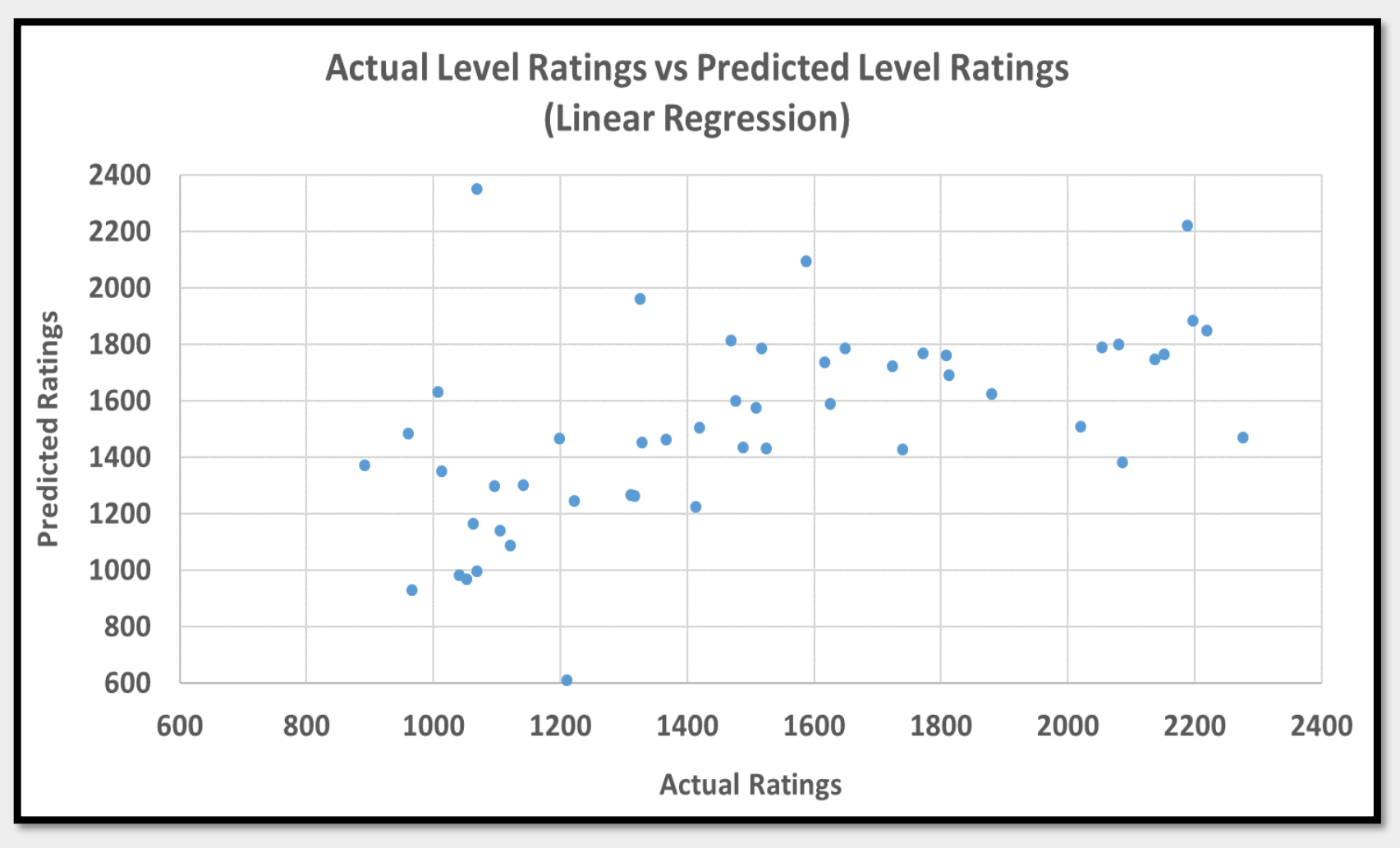
Player Features

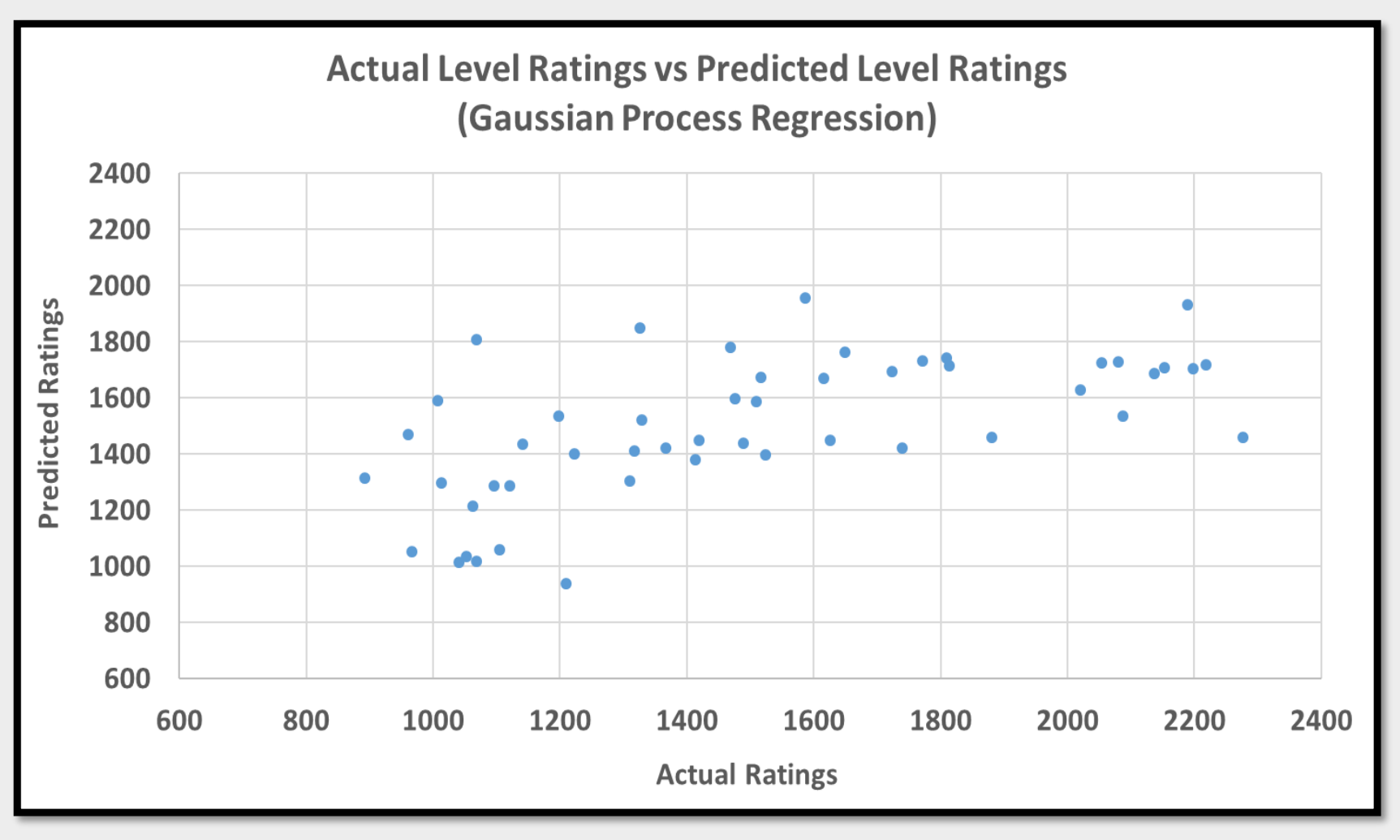
- Total time to complete all tutorial levels
- # of moves to complete all tutorial levels
- Amount by which player exceeds total target score for all tutorial levels

We used the following methods, with leave-one-out cross validation, to evaluate different approaches to predicting ratings:

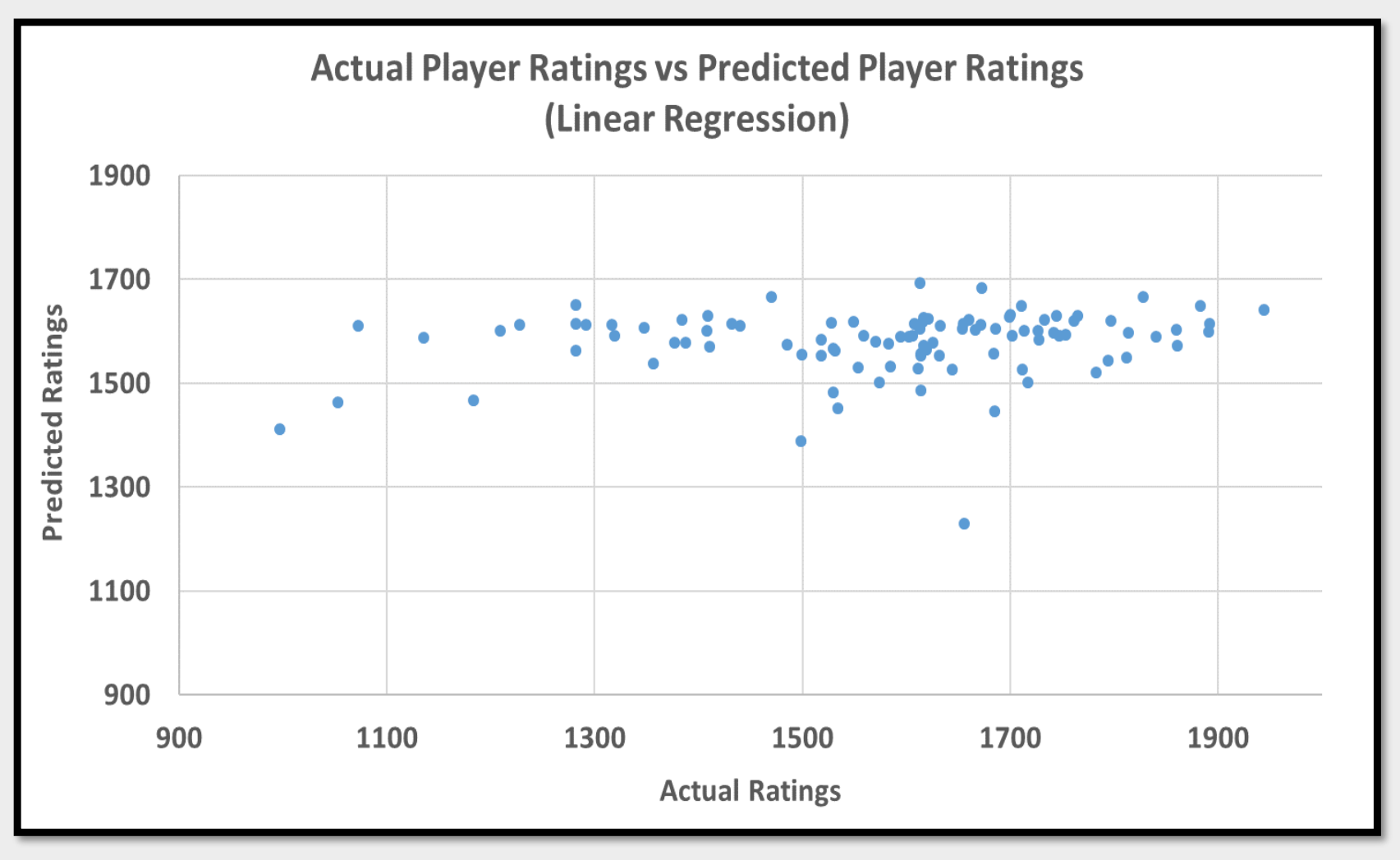
- ***Default:*** Always predict default rating of 1500 (baseline)
- ***Average:*** Predict average of all training ratings (baseline)
- ***LR:*** Use **linear regression** [4] on features to predict ratings
- ***GPR:*** Use **Gaussian process regression** [4] on features to predict ratings

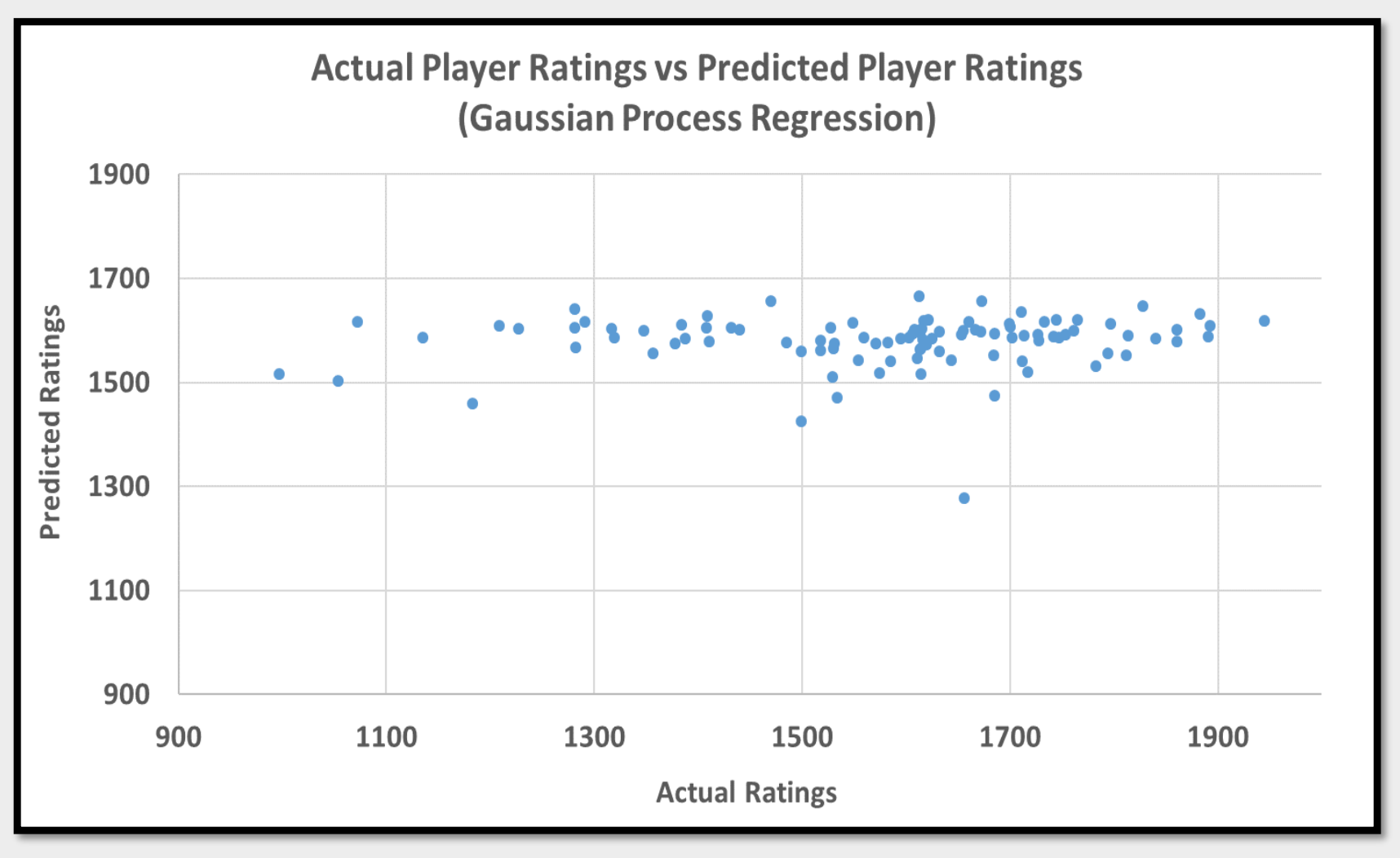
Results



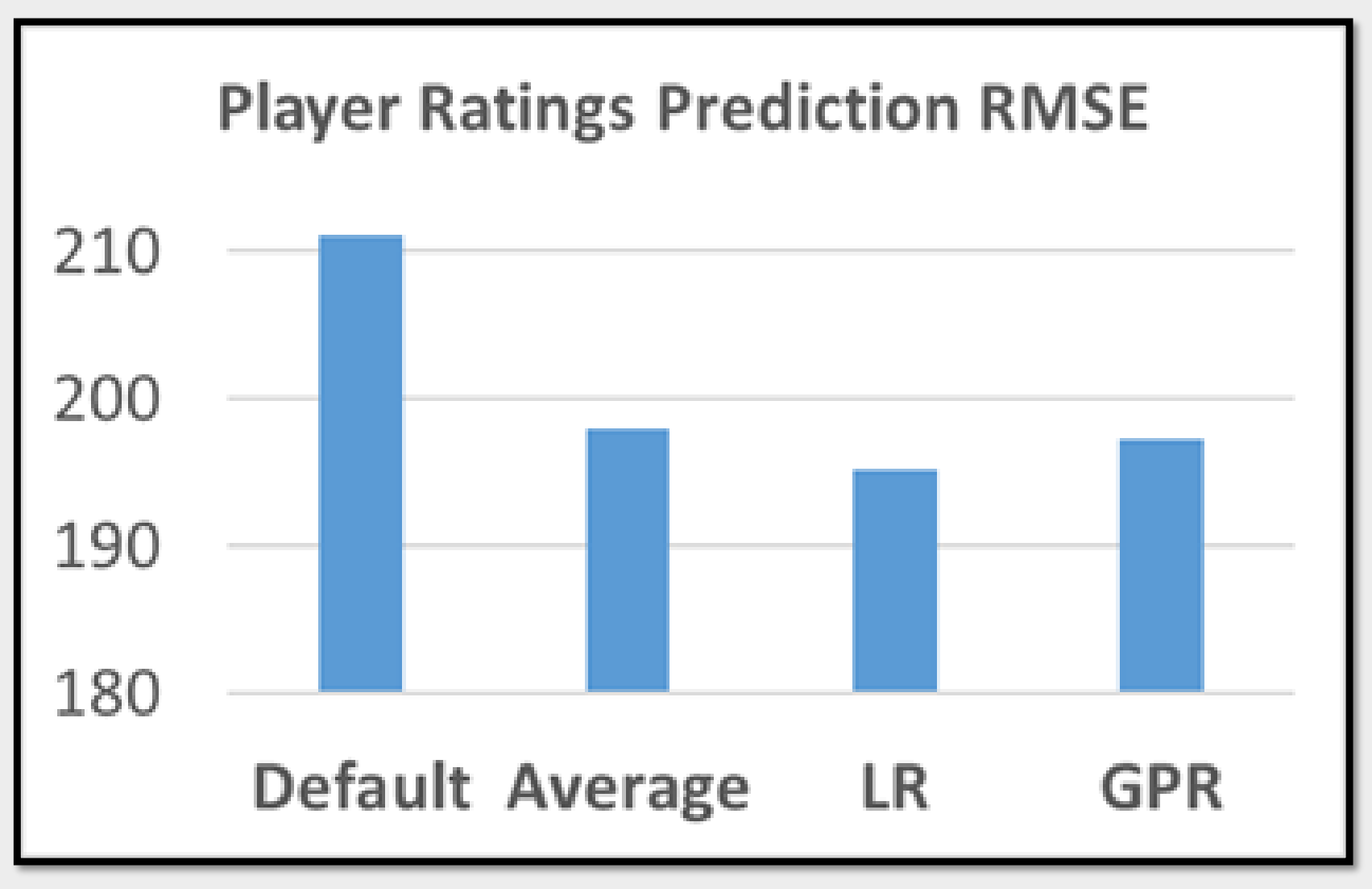


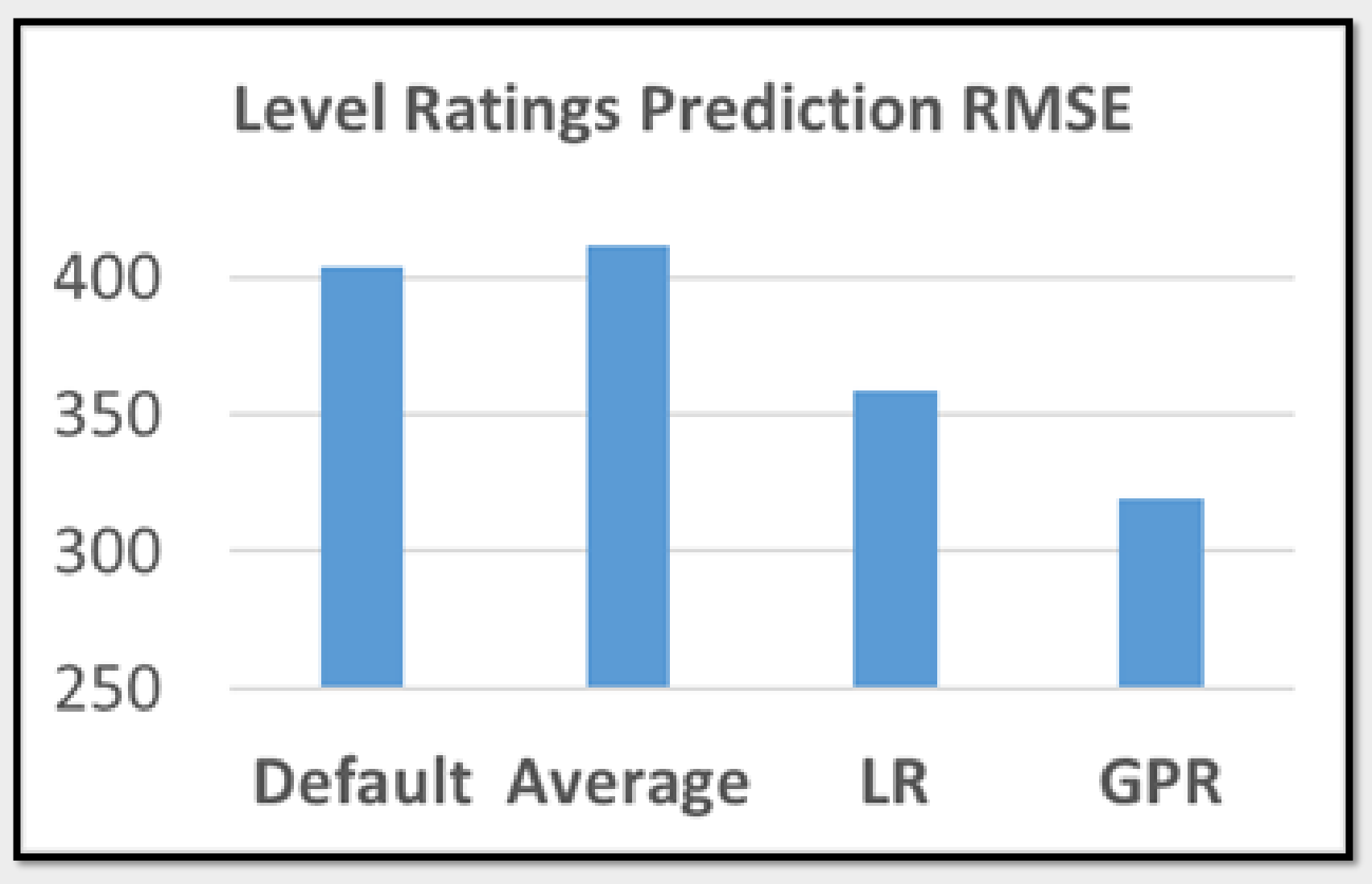
Scatter plot of actual vs regression predicted ratings for levels



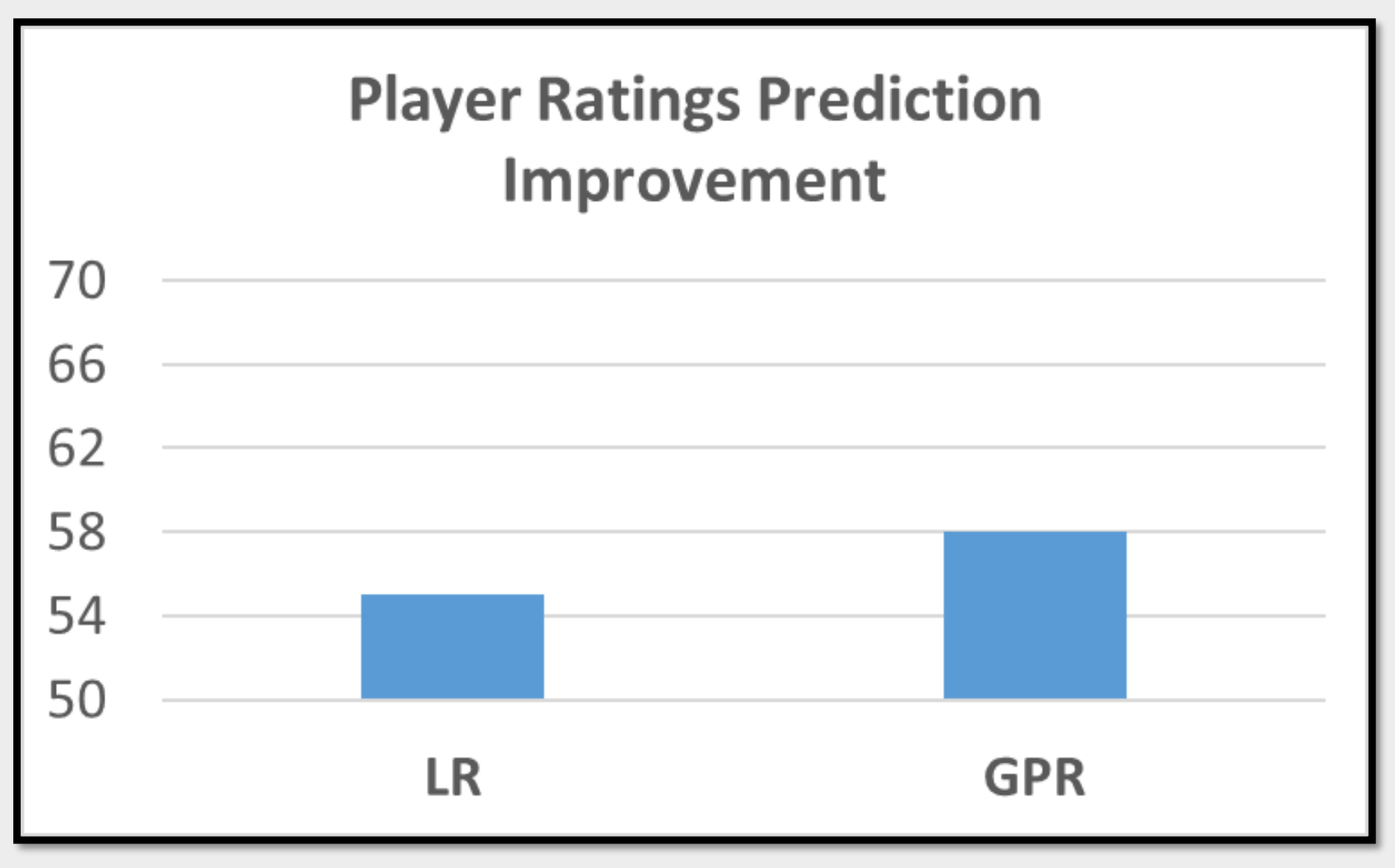


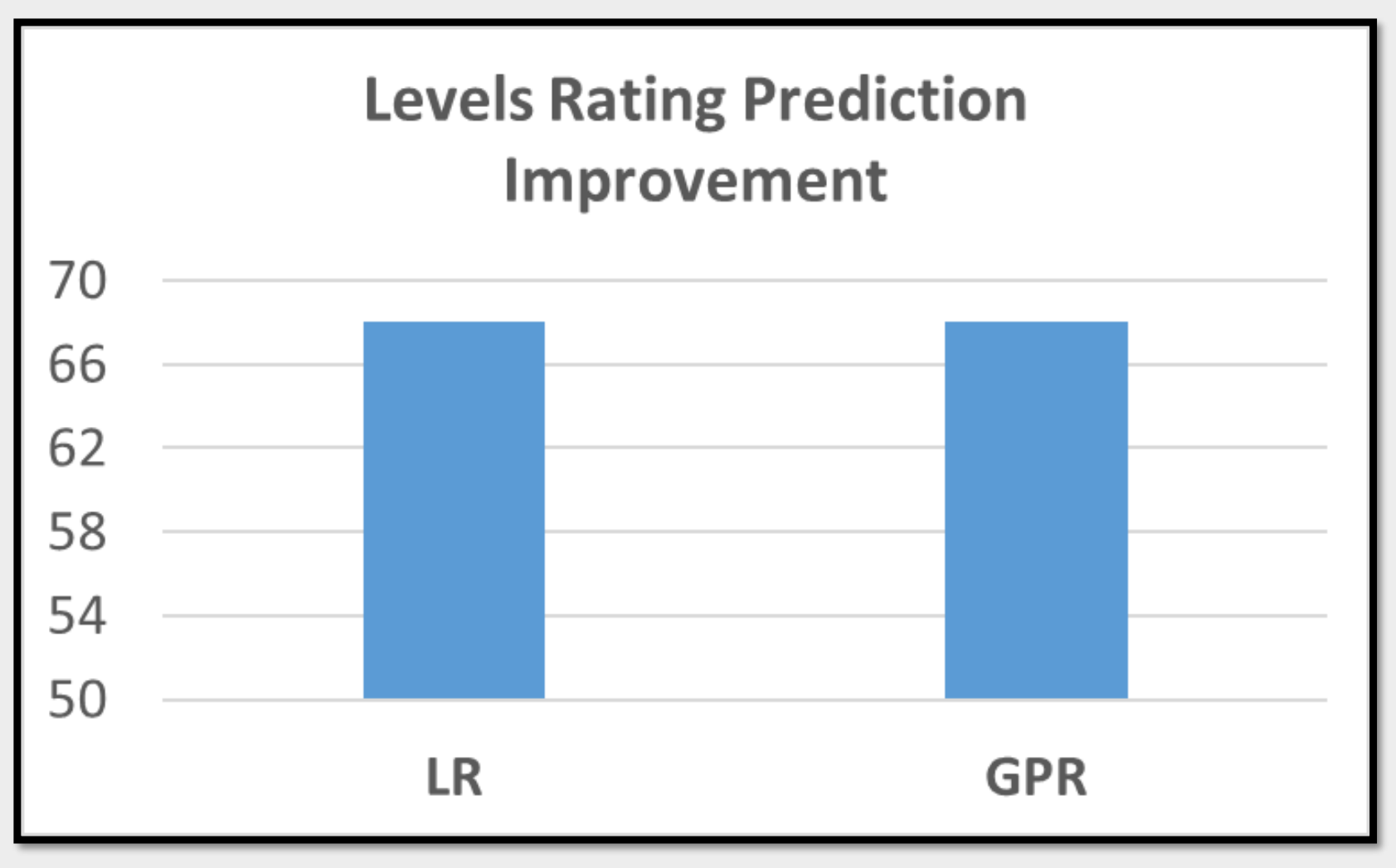
Scatter plot of actual vs regression predicted ratings for players





Root mean square error in ratings prediction for players and levels





Percentage of regression predictions closer to actual ratings than average predictions

Conclusion

- Regression on player and level features can help **predict more accurate initial ratings** for both players and levels than the default ratings, with better results for levels than players.
- This may allow players and levels to reach their actual ratings more quickly by starting them out with predicted rather than default ratings.
- Future work could involve applying this technique to other games with graph-based levels and validating if using predicted ratings increases engagement by matching players with appropriate levels sooner.

References

[1] Sarkar, A.; Williams, M.; Deterding, S.; and Cooper, S. 2017. Engagement effects of player rating system-based matchmaking for level ordering in human computation games. In *Proceedings of the 12th International Conference on the Foundations of Digital Games.*

[2] Dean, D.; Gaurino, S.; Eusebi, L.; Keplinger, A.; Pavlik, T.; Watro, R.; Cammarata, A.; Murray, J.; McLaughlin, K.; Cheng, J.; and Maddern, T. 2015. Lessons learned in game development for crowdsourced software formal verification. In *Proceedings of the 2015 USENIX Summit on Gaming, Games and Gamification in Security Education.*

[3] Glickman, M.E. 2001. Dynamic paired comparison models with stochastic variances. *Journal of Applied Statistics* 28(6):673-689.

[4] Pedregosa, F.; Varoqaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research* 12:2825-2830.

Acknowledgments

This work was supported by a Northeastern University TIER 1 grant. This material is based upon work supported by the National Science Foundation under Grant No. 1652537. We would like to thank the University of Washington's Center for Game Science for initial Paradox development.