Position, Power, Probability: Interpreting Win Prediction in a Strategy Game

Jeffrey Mu
jeffrey_mu@brown.edu
Brown University

Abstract

Win prediction in battle arena games is often confounded by factors like player skill and coordination. We study a battle arena-like tactical setting where combat resolves automatically, isolating the effects of team composition and unit positioning on match outcomes. We evaluate several machine learning models and find modest improvements over a majority-class baseline. We apply interpretability methods and find importances concentrated in a few unit slots and attributes, and further address directions toward recommendation-oriented modeling.

Keywords

strategy games, human-centered AI, interpretable machine learning, model evaluation, decision support, match outcome prediction

GitHub Repository:
https://github.com/jeffreymu1/interpretable-win-predictions/

1 Introduction

In many competitive game settings, match outcomes depend on underlying mechanics and human decisions. Predicting outcomes in these systems is valuable for both forecasting upcoming matches and for diagnosing which battle features consistently drive success [9, 16]. Multiplayer online battle arena (MOBA) games have become a common testbed for this classic machine learning recommendation problem, including match-outcome prediction and increasingly sophisticated learning-based agents [5, 8, 17]. MOBAs like Dota 2, Honor of Kings, and Mobile Legends attract over 100 million daily players, both in casual and competitive settings. However, most MOBA prediction settings inherently combine game mechanics with human-controlled factors [16, 20], such as player skill, team coordination, and the ability to control characters, making it difficult to isolate the implications of game mechanics on outcome predictability.

To separate these effects, this project studies a MOBA-like tactical combat setting where combat is resolved without player control, instead depending heavily on unit configuration, unit location, and stochastic combat mechanics. This yields a challenging but structured binary classification problem, involving a high-dimensional feature space with many interacting units and attributes. We evaluate and interpret the results of several linear and non-linear machine learning models to identify which in-game attributes consistently drive match success, and recommend how several features may positively contribute to interaction-derived results.

1.1 Dataset

Each row in our dataset corresponds to a single player-versus-player match in-game, with fixed team compositions for an attacking player (four on-field units with positions $a_1, \ldots, a_4$ and two supporting units as_1, as_2) and a defending player (four on-field $d_1, \ldots, d_4$ and two supporting ds_1, ds_2), along with the binary match outcome, or *Outcome*, indicating whether the attacking player won. In addition to raw unit attributes (e.g., HP, attack, healing), we construct complementary unit interaction features based on a *time-to-kill* (TTK) approximation between several units. Letting A_i denote the attack statistic of attack unit i, and H_j denote the max HP of target unit j, we define an effectiveness multiplier $m(c_i^{atk}, c_j^{def})$ from a color/type matchup table, where c_i^{atk} is unit i's attack type and c_j^{def} is unit j's defense type. The pairwise TTK from attacker i to target j is then

$$\mathrm{TTK}_{i \to j} = \frac{H_j}{A_i \, m(c_i^{atk}, c_j^{def})}. \tag{1}$$

To summarize a unit's expected performance, we average over the four opposing on-field units

$$\mathrm{TTK}_{a_k \to D} = \frac{1}{4} \sum_{j \in D} \mathrm{TTK}_{a_k \to j}, \quad \mathrm{TTK}_{d_k \to A} = \frac{1}{4} \sum_{i \in A} \mathrm{TTK}_{d_k \to i}. \tag{2}$$

We also compute team-level aggregates by averaging across all unit pairs

$$\overline{\mathrm{TTK}}_{A \to D} = \frac{1}{16} \sum_{i \in A} \sum_{j \in D} \mathrm{TTK}_{i \to j}, \quad \overline{\mathrm{TTK}}_{D \to A} = \frac{1}{16} \sum_{j \in D} \sum_{i \in A} \mathrm{TTK}_{j \to i}, \tag{3}$$

and use their difference $\Delta_{\text{strikers}} = \overline{\mathrm{TTK}}_{D \to A} - \overline{\mathrm{TTK}}_{A \to D}$ as a compact proxy of attacker advantage. We compute analogous TTKs for supporting units to opposing on-field units, and exclude units that cannot deal direct damage by masking TTK contributions. For a comprehensive list of dataset features, see `data-dictionary.md`.

2 Exploratory Data Analysis

Analysis of the dataset shows a relatively balanced target class with similar proportions of attacker wins and losses (Fig. 1a). Additionally, wins tended to be more frequent and clustered at higher ranks, a strong indicator of match outcome (Fig. 1b). However, because features like ranking would not be readily available or known to players before a match begins, we pay close attention to discard features deemed *post-outcome*. We additionally compute Pearson's R between groups of *(individual features, unit positions)* and *Outcome*. We find that across various unit attributes and position groups, the 3rd on-field attacking position and the 1st supporting attacking position were most significantly correlated to match outcome (Fig. 1d).

During exploratory data analysis, we note several categorical and numeric features with missing values, which comprised around 3% of the total dataset. We note that although categorical missing values spanned multiple features, numeric missing values were

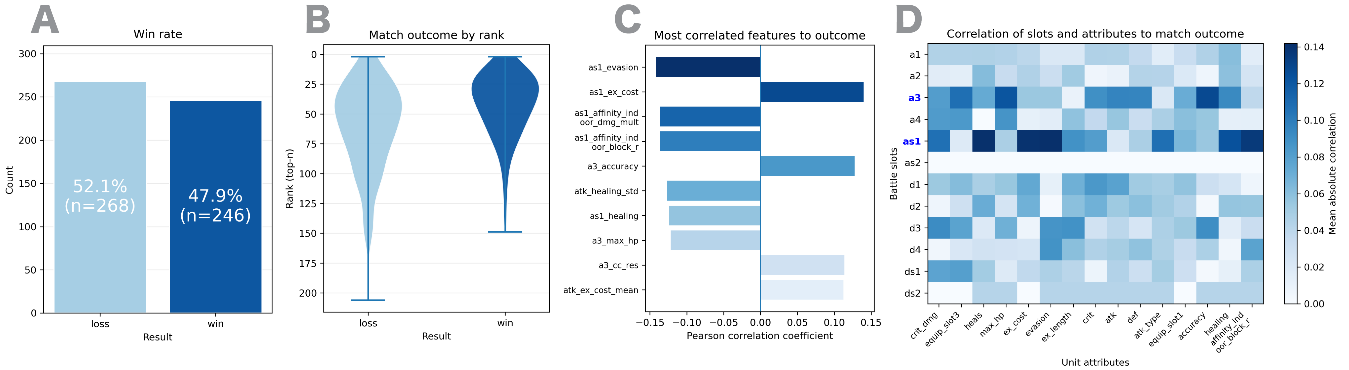

Figure 1: EDA plots. (a) Target class. (b) Wins by rank. (c) Most correlated features. (d) Feature groups by pearson's r.

concentrated solely within features related to the time-to-kill proxy. We also observe several features perfectly collinear to other dataset features. We further examine the connection between utilizing these features for model prediction versus discarding them in `Results` (Section 4).

3 Methods

3.1 Data acquisition and preprocessing

From 514 rows and 635 total features of the dataset, we discard any post-outcome attributes unavailable to players before matches. We further select features representing attributes of all units represented on a team (post- versus pre-scout information, addressed in *Outlook*), and discard any remaining features with no variation. This preprocessing step yields a dataset of 514 rows and 571 features.

We further separate features into categorical and numeric groups, based on the relevant feature information provided by the database. Within categorical features, we categorize missing values in a single feature as their own new value group via scikit-learn's `SimpleImputer` with `strategy='constant'`. Within numeric features, a further analysis of time-to-kill features yields missing values only for units that cannot deal damage. Thus for these missing values, we naturally encode time-to-kill as an arbitrary large value (10^6), representing infinite time-to-kill.

3.2 The machine learning pipeline

To measure the effect of random splits and states on the performance of each model, we iterate through five random states, splitting the dataset via a stratified train-test split with a 10% test holdout. We subsequently run the preparation pipeline consisting of `StandardScaler` and `OneHotEncoder` for numeric and categorical features respectively. We then cut up the training set into five stratified folds consisting of training and cross-validation sets. For each of these folds, we will train a machine learning model on the designated training subset of data, searching over numerous parameters and scoring evaluation metrics including accuracy, ROC-AUC, precision, recall, and F1 scores. Of these metrics, we focus primarily on accuracy as a core evaluation metric, as maximizing accuracy maximizes correct predictions across both classes over a dataset with stratified *Outcome*.

We choose Logistic Regression with L2 and ElasticNet penalties [7, 21], XGBoost [3], Random Forest Classifier [1], Support Vector Classifier [4], and k-Nearest Neighbors [6] to train. We tune hyperparameters until those yielding best performance were within a reasonable parameter range. For XGBoost, we enforce 50 early-stopping rounds over *n_estimators*. As the dataset is relatively small and high-dimensional, for models that utilize regularization, we expect that greater regularization would lead to better performance. For example, a regularization penalty closest to Lasso regression would zero out coefficients of many features in a linear model, helpful for a dataset with numerous, sometimes collinear features. Additionally, to make the model generalizable and reduce the possibility of overfitting or fitting noise, we expect that tree-based models like XGBoost would prefer lower `learning_rate` and `max_depth` hyperparameter values in most training splits.

Algorithm 1 Supervised machine learning pipeline

```
Require: Data $(X, y)$; $r$ seeds; $\tau$ holdout size; $k$ CV folds; $\mathcal{M}$ models with $\mathcal{G}_m$ grids
Split features into numeric/categorical groups; encode missing continuous ttk entries $\to 10^6$
Prep: StandardScaler(numeric) + SimpleImputer(constant, "missing") + OneHotEncoder(categorical)
for all $r \in \mathcal{R}$ do
    $(X_{tr}, X_{te}, y_{tr}, y_{te}) \leftarrow$ stratified train_test_split$(X, y; \tau, r)$
    Evaluate baseline on $(X_{te}, y_{te})$
    for all $m \in \mathcal{M}$ do
        $\pi \leftarrow$ Pipeline(Prep, $m$); CV $\leftarrow$ StratifiedKFold$(k, r)$
        $\pi^\star \leftarrow$ GridSearchCV$(\pi, \mathcal{G}_m$, CV) fit on $(X_{tr}, y_{tr})$
        Evaluate $\pi^\star$ on $(X_{te}, y_{te})$, Acc/ROC-AUC/Prec/Rec/F1
    end for
end for
Aggregate metrics, make confusion matrix (best model)
Explainability (best model): Permutation Importance; SHAP TreeExplainer; local SHAP
```

4 Results

4.1 Classification accuracy

For this binary classification task, predicting the majority class yields a baseline CV accuracy of 0.521, close to a coin flip. Of all

Table 1: Hyperparameter grids used for model selection.

Model	Parameter	Values searched
Logistic Regression (L2)	C	{0.0001, **0.001**, 0.01, 0.1, 1, 10}
	class_weight	{**`None`**, `balanced`}
Logistic Regression (ElasticNet)	C	{0.01, **0.1**, 1, 10, 100}
	l1_ratio	{0.5, 0.8, 0.9, **0.95**, 1.00}
	class_weight	{**`None`**, `balanced`}
Random Forest Classifier	n_estimators	{400, **500**, 600, 700, 800}
	max_depth	{None, 2, **3**, 5, 8}
	min_samples_leaf	{3, **5**, 8}
	max_features	{**`sqrt`**, 0.25, 0.5}
	class_weight	{**`None`**, `balanced`}
Support Vector Classifier (RBF)	C	{0.1, 0.5, **1**, 2, 5, 10}
	γ	{**`scale`**, 0.01, 0.1, 0.5, 1}
	class_weight	{**`None`**, `balanced`}
k-Nearest Neighbors	n_neighbors	{1, **2**, 3, 5, 9}
	weights	{**`uniform`**, `distance`}
	p	{**1**, 2}
XGBoost (early stopping)	max_depth	{0, **1**, 2, 3}
	learning_rate	{0.03, **0.1**, 0.3, 0.6, 1}
	subsample	{0.1, 0.2, 0.3, **0.6**, 1.0}
	colsample_bytree	{0.8, 0.9, 0.95, 0.98, **1.0**}
	reg_lambda	{0.001, 0.01, 0.1, **1.0**, 10.0}

models we train, the CV accuracy of the Random Forest Classifier (0.580 ± 0.010) had highest mean, followed closely by XGBoost (0.576 ± 0.016) and Support Vector Classifier (0.566 ± 0.007). For the test set, Support Vector Classifier (0.569 ± 0.055) outperformed Random Forest Classifier (0.558 ± 0.080) and Logistic Regression with L2 penalty (0.55 ± 0.075). Across all evaluation metrics, highest-performing means were found in either Random Forest Classifier or Support Vector Classifier, and both models performed above baseline on all but one metric each (Table 2).

We noted previously that several features in the dataset were collinear with other non-target features, and considered dropping these features to assist with outcome prediction. However, training all models over both the full dataset and the dataset without collinear features yielded mostly negligible mean accuracy differences. Furthermore, four models of six obtained higher accuracy over the full dataset in both the CV set and the test set, so we consider the full dataset with collinear features when proceeding with analysis.

Table 2: Performance summary for baseline scores and best two models. Full metrics are provided in the appendix.

Metric	Baseline	RFC	SVC
CV Accuracy	0.522 ± 0.000	**0.580 ± 0.010**	0.566 ± 0.007
Test Accuracy	0.519 ± 0.000	0.558 ± 0.080	**0.569 ± 0.055**
Test ROC-AUC	0.500 ± 0.000	**0.582 ± 0.065**	0.577 ± 0.076

We focus on the trained Random Forest Classifier, which achieved best performance on the training set. We tuned the number of trees, maximum depth, minimum samples per leaf, maximum features, and class weighting. The best configuration used 500 trees, max depth 3, min samples per leaf 5, `max_features=sqrt`, and no class weights.

In terms of our primary evaluation metric (accuracy), the Random Forest correctly predicted losses and wins in roughly 60% of cases, but performance was highly unstable. The confusion matrix shows symmetric errors, indicating no strong bias toward either class. The ROC-AUC was 0.582 ± 0.065, only marginally above chance, and the uncertainty makes the improvement difficult to interpret. Reducing this variance will likely require more data or additional model refinement.

4.2 Predictive features

In order to analyze the relative impact of different match features on the model's prediction, we compute three global feature importance metrics: perturbation importance via ablating individual features, Random Forest mean decrease in impurity (MDI), and global SHAP [12] values. We further evaluate several local SHAP force plots for randomly sampled points in the test set to analyze important feature values and their impact on predictions.

Although the model's signal concentrated in two unit slots—the 3rd position on-field attacker and the 1st position supporter attacker—the overall interpretability results were noisy, with large standard deviations. Still, these two slots were consistently identified by both permutation importance and global SHAP, albeit through different feature constructions (Fig. 2).

At the attribute level, *evasion* emerged as the strongest driver in global SHAP. In permutation importance, importance was more

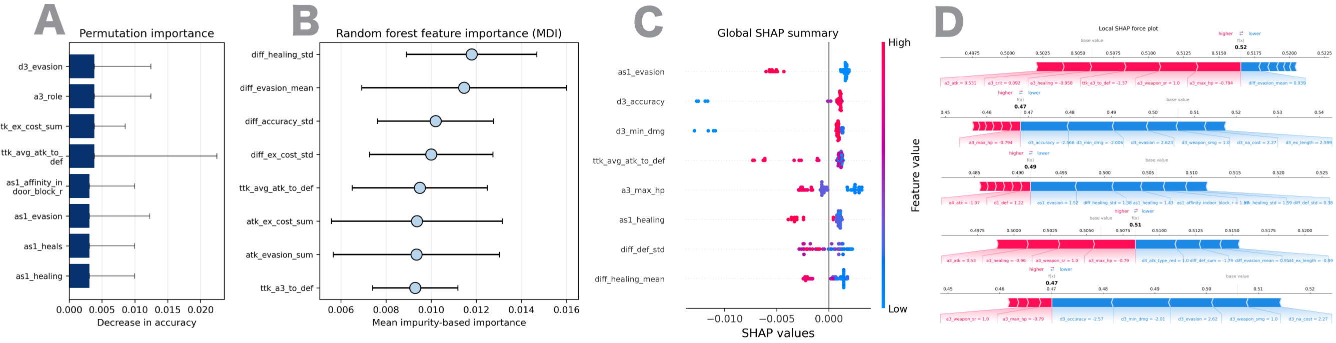

Figure 2: Predictive features for the Random Forest. (a) Feature permutation importance. (b) Mean decrease in impurity across trees. (c) Global SHAP feature importances. (d) Local SHAP explanations for five test set examples.

diffuse across features; however, the between-team difference in evasion showed high variance and may nonetheless have the largest effect on impurity-based importance. A second robust signal was the team-averaged time-to-kill, which appeared consistently across all three feature-importance analyses. Most other attributes varied substantially across runs and methods.

Local SHAP values help explain why the model predicts a given match outcome by highlighting which features mattered for a specific matchup. Importantly, we did not observe a single monotonic pattern of feature importance. Instead, contributions frequently changed sign and magnitude across games, suggesting the model is relying on matchup-specific interactions rather than memorizing a monotonic relationship.

Qualitatively, game-clip inspection aligned with this interpretation: players often placed a high-damage attacker in the third on-field slot, and the same unit appeared repeatedly in the second special supporting slot, while the first special slot varied across teams and matches.

5 Outlook

We set out to develop a supervised machine learning prediction and recommendation system to assist players in generating team formations to counter enemies. As the dataset was relatively small, a first extension of this work could scrape more data from a larger database of player battles to improve predictive power. Several extensions also follow naturally to this prediction task, namely developing a classic machine learning recommendation system used to maximize the chances of winning in a single round against an opponent. In real in-game battles, players also have limited access to opponents' team formation dependent on their rank; players ranked higher may only be allowed information about the first slot and supporting defending units in their first battle. An effective recommendation system would account for this issue by splitting up first matches and subsequent matching into pre- and post-scout features, and hiding features irrelevant to each one respectively.

To more effectively interpret our model, we consider Missing at Random (MAR) and Missing Completely at Random (MCAR) masking perturbations to our dataset to better understand the effect of missing values on predictions. As missing values of features may depend on subsequent or adjacent features, we postulate that a thorough analysis of different missing-value imputation techniques would lead to a stronger prediction system, and could possibly help generate stronger, more tactical team formations.

Acknowledgments

The author would like to thank Andras Zsom for his support throughout the project, as well as course TAs Shiyu Liu and Devraj Raghuvanshi.

References

[1] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32. doi:10.1023/A:1010933404324

[2] Kevin Chen, Petar Baugas, Vladlen Koltun, et al. 2018. The Art of Drafting: A Team-Oriented Hero Recommendation System for Multiplayer Online Battle Arena Games. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys)*. https://arxiv.org/abs/1811.07657

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 785–794. doi:10.1145/2939672.2939785

[4] Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Machine Learning* 20, 3 (1995), 273–297. doi:10.1007/BF00994018

[5] Stefan Costa, Marion Brandt, and Michael Weinstock. 2021. Feature Analysis of League of Legends to Victory Prediction Based on Pre-Game Information. In *Proceedings of the IEEE Conference on Games (CoG)*.

[6] Thomas M. Cover and Peter E. Hart. 1967. Nearest Neighbor Pattern Classification. In *Proceedings of the IEEE Transactions on Information Theory*, Vol. 13. 21–27. doi:10.1109/TIT.1967.1053964

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2 ed.). Springer, New York, NY. doi:10.1007/978-0-387-84858-7

[8] Victoria Hodge, Anders Drachen, and Peter Cowling. 2017. Win Prediction in Esports: Mixed-Rank Match Prediction in Multi-player Online Battle Arena Games. arXiv:1711.06498 https://arxiv.org/abs/1711.06498

[9] Victoria J. Hodge, Sam Devlin, Nick Sephton, Franziska Block, Anders Drachen, and Peter I. Cowling. 2019. Win Prediction in Multi-Player Esports: Live Professional Match Prediction. *IEEE Transactions on Games* (2019). doi:10.1109/TG.2019.2948469

[10] A. Júnior and Carlos Campelo. 2023. League of Legends: Real-Time Result Prediction. arXiv:2308.03747 https://arxiv.org/abs/2308.03747

[11] Zhenyu Ke, Pei Mu, Peng Cui, Chao Zhang, Yuguang Chen, and Wenwu Zhang. 2022. Team Fight Models for Match Prediction in Dota 2. In *Proceedings of the IEEE Conference on Games (CoG)*. https://arxiv.org/abs/2208.00866

[12] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[13] Michael Marshall, Georgios Chalkiadakis, et al. 2022. Enabling Real-Time Prediction of In-game Deaths through Telemetry Data in Counter-Strike: Global Offensive. In *Proceedings of the Foundations of Digital Games (FDG)*.

[14] Antonio Luis Cardoso Silva and Mauro Roisenberg. 2018. Continuous Outcome Prediction of League of Legends Competitive Matches Using Recurrent Neural Networks. In *Proceedings of SBGames*.

[15] Rebecca Smithies, David James, et al. 2021. Random forests identify key metrics for predicting winning in Rocket League. *Scientific Reports* (2021). doi:10.1038/s41598-021-98879-0

[16] Kaixin Wang, Xinxin Li, et al. 2020. Match Tracing: A Unified Framework for Real-time Win Prediction and Performance Evaluation in Esports. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM)*. doi:10.1145/3340531.3412727

[17] Nanzhi Wang, Lin Li, Linlong Xiao, Guocai Yang, and Yue Zhou. 2018. Outcome Prediction of DOTA2 Using Machine Learning Methods. In *Proceedings of 2018 International Conference on Mathematics and Artificial Intelligence (ICMAI '18)*. doi:10.1145/3208788.3208800

[18] Vasilios Xenopoulos, Josip Marić, Diederik M. Roijers, et al. 2022. ESTA: An Esports Trajectory and Action Dataset. arXiv:2209.09861 https://arxiv.org/abs/2209.09861

[19] Fan Yang, Yan Yu, Minghua Chen, et al. 2020. Interpretable Real-Time Win Prediction for Honor of Kings. arXiv:2008.06313 https://arxiv.org/abs/2008.06313

[20] Yifan Yang, Tian Qin, and Yu-Heng Lei. 2016. Real-time eSports Match Result Prediction. arXiv:1701.03162 https://arxiv.org/abs/1701.03162

[21] Hui Zou and Trevor Hastie. 2005. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67, 2 (2005), 301–320. doi:10.1111/j.1467-9868.2005.00503.x

A Appendix

We report the trained Random Forest confusion matrix aggregated over five random seeds.

Table 3: Random Forest Classifier, five seeds. Values are percent of all test examples.

		True	
		loss	**win**
Predicted	**loss**	32.31 ± 5.51%	19.62 ± 5.51%
	win	24.62 ± 9.16%	23.46 ± 9.16%

We also report baseline scores and model performances for five random seeds, and across all evaluation metrics.

Table 4: Full evaluation metrics across all models, five seeds.

Metric	Baseline	LR-EN	LR-L2	XGB	RFC	SVC	kNN
CV Accuracy	0.522 ± 0.0	0.561 ± 0.019	0.562 ± 0.015	0.576 ± 0.016	**0.580** ± **0.010**	0.566 ± 0.007	0.553 ± 0.013
Test Accuracy	0.519 ± 0.0	0.546 ± 0.062	0.550 ± 0.075	0.485 ± 0.053	0.558 ± 0.080	**0.569** ± **0.055**	0.508 ± 0.065
Test ROC-AUC	0.500 ± 0.00	0.566 ± 0.072	0.562 ± 0.085	0.470 ± 0.067	**0.582** ± **0.065**	0.577 ± 0.076	0.485 ± 0.081
Test Precision	0.000 ± 0.000	0.520 ± 0.087	0.520 ± 0.097	0.446 ± 0.082	0.534 ± 0.101	**0.551** ± **0.055**	0.476 ± 0.089
Test Recall	0.000 ± 0.000	0.456 ± 0.161	0.464 ± 0.178	0.464 ± 0.239	0.488 ± 0.191	**0.520** ± **0.177**	0.320 ± 0.155
Test F1 Score	0.000 ± 0.000	0.480 ± 0.131	0.485 ± 0.146	0.437 ± 0.164	0.502 ± 0.143	**0.524** ± **0.124**	0.369 ± 0.141