Data Science and Engineering (DSE) Record, Volume 7, issue 1.

Auto Battle Game Outcome Prediction with Machine Learning

Sittakon Phommee [1] and Juggapong Natwichai [2]

[1] Master's Degree Program in Data Science, Chiang Mai University, Chiang Mai, Thailand
sittakon_phomme@cmu.ac.th
[2] Department of Computer Engineering, Faculty of Engineering, Chiang Mai University, Chiang Mai, Thailand
juggapong.n@cmu.ac.th

Abstract. This research aims to compare the performance of machine learning algorithms for classifying match outcomes in Teamfight Tactics (Set 13) using a dataset of 78,412 samples collected from the Riot API. Four additional engineered features are implemented and compared three encoding techniques Label Encoding, One-Hot Encoding, and Bag-of-Words, in combination with four classification algorithms: k-Nearest Neighbors (k-NN), Support Vector Machine (SVM), Random Forest, and XGBClassifier. The experimental results indicate that the Bag-of-Words technique achieved the highest performance across all algorithms and effectively reduced the impact of data sequence variance. Among the algorithms, XGBClassifier delivered the most accurate predictions, with an Accuracy of 85.25% and an F1-Score of 0.85. Furthermore, feature importance analysis revealed that the newly engineered feature, Total Cost, is the most significant factor influencing match outcomes.

Keywords: Auto Battle Game, Machine Learning, Classification, Game Analytics.

1 Introduction

In recent years, the gaming industry has experienced rapid growth and has become a major sector in global entertainment. Beyond providing entertainment, modern games generate significant revenue through live streaming, competitive tournaments, and data-driven analysis. One of the most popular auto-battler games is Teamfight Tactics (TFT), developed by Riot Games in 2019 [1]. TFT has gained widespread attention through international competitions with substantial prize pools [2] [3] [4], reflecting its growing influence in both global and regional esports scenes.

The complexity of TFT lies in its round-based system, where players must navigate a vast decision space involving over 60 unique units and 25 distinct "Traits." These Traits provide specialized bonuses and synergies that significantly influence combat outcomes. To gain a competitive edge, players must analyze the intricate relationships between unit statistics and active synergies. The availability of granular match data through the Riot API provides a critical opportunity for researchers to apply Artificial Intelligence (AI) and Machine Learning (ML) techniques to identify the key factors that determine player rankings.

However, predicting player rankings in TFT remains a challenging task due to the high-dimensional nature of the data, the mixture of categorical and numerical features, and the inherent uncertainty in the game environment, such as randomness in unit availability and opponent decisions. Therefore, effective predictive models must be capable of capturing complex, non-linear relationships among multiple factors.

This study aims to develop machine learning models for predicting player rankings in TFT using match data obtained from the Riot API [5]. In addition, the performance of different algorithms, including k-Nearest Neighbors, Random Forest, Support Vector Machine, and XGBoost, is compared to identify the most accurate approach for this task.

2 Literature Review

2.1 Teamfight Tactics (TFT)

Teamfight Tactics (TFT) as depicted in Figure 1 is an auto-battler game developed by Riot Games and released in 2019. The game emphasizes strategic planning, resource management, and decision-making under uncertainty. Players do not directly control combat; instead, they select, organize, and position units on a board, after which battles are executed automatically in each round.

TFT is an eight-player multiplayer game in a Player versus Player (PvP) format. Each player starts with equal health points, and the objective is to survive as the last remaining player or achieve the highest possible ranking. The game follows a round-based system, where outcomes affect both player health and in-game resources, primarily gold, which is used to purchase units, level up, and refresh the shop.

Fig. 1. Preparation phase in TFT

Units are obtained from a randomized shop, with probabilities depending on the player's level. Each unit has unique attributes and abilities that influence team composition and strategy. Additionally, units are associated with traits, representing their origin or role. Combining units with shared traits grants bonuses or special effects, forming a key strategic element of the game.

Furthermore, units can be upgraded through a star-level system ranging from 1 to 4 stars. By combining three identical units, players can increase their star level, significantly enhancing their attributes and abilities. This system plays a crucial role in determining overall team strength.

2.2 Other Literature

In the domain of Multiplayer Online Battle Arena (MOBA) games, such as League of Legends and DotA 2, several studies have explored the application of machine learning techniques for game outcome prediction. Do et al. [6] employed a deep neural network to predict match outcomes in League of Legends based on champion experience, achieving an accuracy of 75.1% immediately after the champion selection phase. Meanwhile, Jailson et al. [7] compared model performance across different stages of the game and found that LightGBM achieved the highest accuracy of 81.62% during the mid-game phase (60%–80% of game duration), whereas Logistic Regression and Gradient Boosting performed better in the early stages.

Furthermore, Hitar-Garcia et al. [8] enhanced predictive performance by incorporating additional features, such as player–champion proficiency, team synergy, and player effectiveness against opponents. Their results showed that prediction accuracy exceeded 70% even with limited data and without relying on external sources. Similarly, Tuzcu et al. [9] analyzed both team-level and player-level data and demonstrated that team-based features significantly improved prediction performance, achieving up to 98% accuracy using AdaBoost after feature selection. In addition, the accuracy of Logistic Regression increased from 89% to 98%, and Gradient Boosting improved from 96% to 98%.

For DotA 2, Kalyanaraman et al. [10] proposed a hybrid approach that combines Logistic Regression with a Genetic Algorithm to identify influential hero combinations, resulting in a significant improvement in prediction accuracy. Their findings showed that Logistic Regression achieved the highest accuracy at 75.2%, followed by the Genetic Algorithm at 74.1%, and Augmented Regression at 68.4%, while Augmented Regression achieved the highest recall at 90.9%. Moreover, Conley et al. [11] highlighted that hero synergy and counter-picking strategies are crucial factors, particularly among high-skill players. Logistic Regression achieved an accuracy of 69.8%, while the performance of the K-Nearest Neighbors model improved with increasing data size, reaching nearly 70%. Additionally, Semenov et al. [12] compared multiple algorithms based on the hero selection phase and found that Factorization Machines (FM) achieved the best performance (ROC AUC of 0.706), as they are effective in capturing interactions between pairs of heroes. Their study also indicated that prediction becomes more challenging among high-skill player groups.

3 Data and Methodology

3.1 Data

The dataset for this study was sourced from the Riot API, the official developer gateway provided by Riot Games. We specifically collected match data from

Teamfight Tactics (TFT) Set 13, focusing exclusively on players in the Master Rank. This high-skill tier was selected to ensure that the data reflects optimized strategic decision-making and a profound understanding of game mechanics. The raw data was retrieved in JSON format, containing comprehensive details of each match, including the compositions of all eight participants, unit statistics, and final placements as shown in Table 1.

1) Data Preprocessing After collecting the raw data in JSON format, the data were transformed into a tabular format. The following features were defined

1.Placement: The final ranking of the player in a match (1–8), used as the target variable. The rankings were converted into a binary classification problem, where ranks 1–4 were labeled as 1 (win) and ranks 5–8 as 0 (loss).

2.Unit & Tier: Information on each unit used by the player along with its star level (up to 4 stars), representing individual unit strength.

3.Items: The list of items equipped by each unit, indicating enhancements to unit performance (each unit can equip up to three items).

4.Trait Synergies (Traits): Information on activated traits, where values range from 0 (inactive) to 4 (maximum strength level: Bronze, Silver, Gold, and Prismatic).

Table 1. Example of Data Structure

Placement	Unit_1	Tier	Item_1	Item_2	Item_3	…	Trait_1	Trait_2	...
1	Draven	2	Item_X	0	0	…	1	0	…

2) Feature Engineering In addition to the extracted features from the Riot API, four derived features were constructed to represent the economic strength and structural efficiency of the team

1. Number of Units (num_units): The total number of units in the team, representing team size.

2. Average Stars (avg_star): The average star level of all units, representing overall unit quality.

3. Total Items (num_items): The total number of items equipped, reflecting resource utilization.

4. Total Team Cost (total_cost): The sum of unit costs based on star level and rarity, representing the economic strength and overall team investment.

3) Feature Encoding To enable machine learning models to process heterogeneous data types, three encoding strategies were applied and compared

1. Label Encoding This method was applied to categorical features such as unit names and item names by assigning unique integer values. Ordinal features (e.g., unit tier and trait level) were preserved in their numerical form to maintain their inherent order. This approach helps prevent an increase in data dimensionality.

2. One-Hot Encoding To avoid introducing artificial ordinal relationships, one-hot encoding was applied. Each unit and item was represented as a binary feature (1 for presence, 0 for absence). Although this significantly increases the feature space, it improves distance-based learning for models such as Support Vector Machine (SVM) and K-Nearest Neighbors (KNN).

3. Bag-of-Words (BoW) Inspired by natural language processing techniques, the team composition was treated as a document, while units and items were treated as words. Instead of preserving positional information, this method counts the occurrence frequency of units and items within a team. This representation emphasizes overall team composition, which aligns with the TFT gameplay mechanics where outcomes depend on the combination of units rather than their positional indexing.

3.2 Methodology

3.2.1 Model Training and Evaluation Process

To ensure a fair and systematic comparison of model performance, the experimental procedure was divided into the following steps:

1) Data Splitting

The dataset was divided into five equal parts using a 5-fold cross-validation approach. In each iteration, four parts were used as the training set, while the remaining part was used as the validation set. This process was repeated five times so that each subset served as the validation set once.

The average performance across all five folds was used as the representative performance of each model. This approach helps reduce bias that may arise from a single data split.

2) Baseline Model Construction and Comparison

Four machine learning algorithms k-Nearest Neighbors (k-NN), Random Forest, Support Vector Machine (SVM), and XGBoost were trained using default parameters. For SVM, a Linear SVC was applied.

Each model was trained on datasets processed using different encoding techniques. The goal was to identify the most suitable combination of data representation and algorithm that provides the best initial performance.

3) Hyperparameter Tuning and Final Evaluation

After identifying the best-performing model (Champion Model), further performance improvement was conducted through hyperparameter tuning using the Random Search method.

The optimized model was then evaluated on the testing set to compare performance before and after tuning, allowing for a clear assessment of improvement.

3.2.2 Evaluation Metrics

Since this study is a binary classification problem, model performance was evaluated using metrics derived from the confusion matrix.

1. Accuracy: The overall proportion of correct predictions (both Top 4 and Bottom 4) relative to the total number of cases.

2. Precision: The ratio of true Top 4 predictions to the total number of instances predicted as Top 4, indicating the model's exactness.

3. Recall: The ratio of correctly predicted Top 4 instances to the actual number of Top 4 finishes, reflecting the model's ability to capture all positive outcomes.

4. F1-Score: The harmonic mean of Precision and Recall, serving as a balanced metric to evaluate the model's overall effectiveness, especially in ensuring a trade-off between precision and coverage.

4 Experiments And Results

4.1 Model Comparison.

The performance of four models k-NN, Random Forest (RF), SVM, and XGBoost was evaluated in combination with three feature encoding techniques: Label Encoding, One-Hot Encoding, and Bag-of-Words (BoW). The comparative results are presented in Table 2.

Table 2. Example of Data Structure

Model	Feature Encoding	Accuracy (Train)	Accuracy (Test)	Precision	Recall	F1-Score
k-NN	Label Encoding	0.8460	0.7717	0.77	0.77	0.77
	One-Hot Encoding	0.6659	0.5635	0.60	0.56	0.52
	Bag-of-Words	0.7994	0.6998	0.71	0.70	0.69
RF	Label Encoding	1.000	0.8293	0.83	0.83	0.83
	One-Hot Encoding	1.000	0.8208	0.82	0.82	0.82
	Bag-of-Words	1.000	0.8361	0.84	0.84	0.84
SVM	Label Encoding	0.8691	0.8356	0.84	0.84	0.84
	One-Hot Encoding	0.8673	0.8200	0.82	0.82	0.82
	Bag-of-Words	0.8955	0.8393	0.84	0.84	0.84
XGBoost	Label Encoding	0.8536	0.8354	0.84	0.84	0.84
	One-Hot Encoding	0.8487	0.8370	0.82	0.82	0.82
	Bag-of-Words	0.8560	0.8437	0.84	0.84	0.84

The results indicate that XGBoost combined with Bag-of-Words achieved the highest performance, with a test accuracy of 0.8437 and an F1-score of 0.84. This suggests that XGBoost is particularly effective in handling categorical data transformed into vector representations within the context of Teamfight Tactics (TFT) data. Meanwhile, SVM demonstrated consistent performance across all encoding techniques, reflecting its robustness against variations in data representation.

Regarding Random Forest (RF), although it achieved competitive test accuracy (approximately 0.82–0.83), a clear issue of overfitting was observed, as the training accuracy reached 1.000 in all cases. This indicates that while the model captures the training data perfectly, it faces challenges in generalizing to unseen datasets. Conversely, k-NN yielded the lowest performance, specifically when paired with One-Hot Encoding, where the F1-score dropped to 0.52. This highlights the inherent limitations of k-NN in managing high-dimensional and sparse feature spaces.

In conclusion, the findings demonstrate that both model selection and feature encoding techniques significantly impact prediction performance, with the XGBoost-BoW architecture emerging as the most effective approach for this study."

4.2 **Analysis of Engineered Features**

To investigate the impact of the newly engineered features namely num_units, avg_star, num_items, and total_cost a comparative experiment was conducted using the best-performing model (XGBoost with Bag-of-Words) on both the baseline dataset and the enhanced dataset.

Table 3. Performance comparison between baseline and enhanced datasets.

Dataset	Accuracy (Train)	Accuracy (Test)	Precision	Recall	F1-Score
Baseline Dataset	0.8477	0.8350	0.83	0.83	0.83
Enhanced Dataset	0.8560	0.8437	0.84	0.84	0.84
Difference	+0.0083	+0.0087	+0.01	+0.01	+0.01

The results, presented in Table 3, show that incorporating the additional features leads to a consistent improvement across all evaluation metrics. Specifically, test accuracy increased from 0.8350 to 0.8437, while the F1-score improved from 0.83 to 0.84. Similar improvements were observed in precision and recall, indicating that the added features enhance the model's overall predictive capability.

Importantly, the gap between training and testing accuracy remains relatively stable, suggesting that the inclusion of these features does not introduce overfitting and that the model maintains good generalization performance.

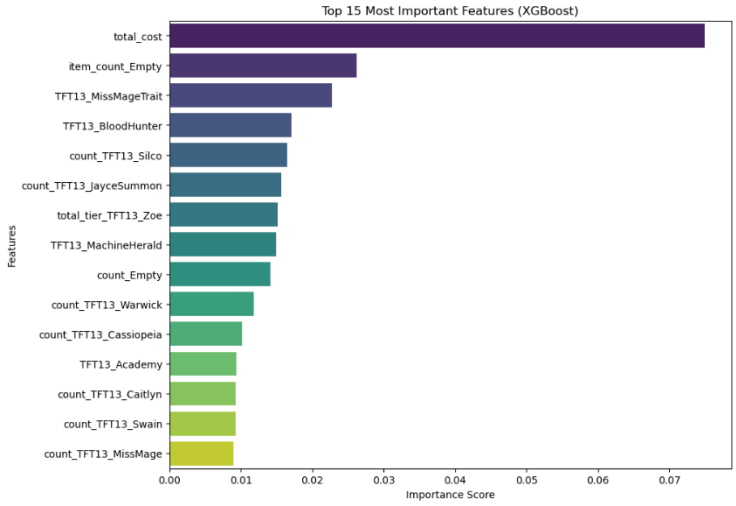

Figure 2. Feature importance of the enhanced dataset.

Further analysis of feature importance as shown in Figure 2 reveals a notable shift after the inclusion of engineered features. In particular, total_cost emerges as the most influential feature, significantly outperforming other variables. This indicates that the overall team cost serves as a strong structural indicator of team strength and has a substantial impact on match outcomes.

In summary, the introduction of engineered features especially total_cost not only improves model performance but also reshapes the importance structure of the input variables, enabling the model to better capture underlying patterns in the data.

4.3 Model Optimization

To further improve model performance, hyperparameter tuning was applied to the XGBoost model using Randomized Search with 5-fold cross-validation. The search was conducted over a predefined parameter space, as summarized in Table 4, to balance model complexity and prevent overfitting.

Table 4. Performance comparison between baseline and enhanced datasets.

Metric	Pre-Tuning	Post-Tuning	Difference
Accuracy (Train)	0.8560	0.8641	+ 0.0081
Accuracy (Test)	0.8437	0.8525	+ 0.0088
Precision	0.84	0.85	+ 0.01
Recall	0.84	0.85	+ 0.01
F1-Score	0.84	0.85	+ 0.01

The comparison results before and after tuning are shown in Table 4. The tuned model demonstrates consistent improvements across all evaluation metrics. Test

accuracy increased from 0.8437 to 0.8525, while the F1-score improved from 0.84 to 0.85. Precision and recall also increased by approximately 0.01.

Moreover, the gap between training and testing accuracy decreased to 0.0116, indicating improved model stability and better generalization to unseen data.

These results highlight the effectiveness of hyperparameter tuning in enhancing model performance. By identifying optimal parameter configurations, the model becomes better aligned with the underlying data distribution, leading to more accurate and reliable predictions.

5 Conclusion and Discussion

5.1 Conclusion

The experimental results demonstrate that the XGBoost model combined with Bag-of-Words (BoW) encoding achieves the highest performance in classifying TFT match outcomes. Following hyperparameter optimization, the model exhibited significant improvements in both accuracy and stability, reaching a Test Accuracy of 85.25% and an F1-Score of 0.85. This indicates a well-calibrated balance between precision and recall. Furthermore, the tuning process effectively mitigated overfitting, reducing the generalization gap between training and testing accuracy to a mere 1.16%. Consequently, the XGBoost-BoW architecture is identified as the most suitable approach for distinguishing between Top 4 and Bottom 4 placements in this study.

The success of the BoW technique stems from its ability to represent feature frequency while maintaining a manageable feature space, unlike One-Hot Encoding, which suffered from the "curse of dimensionality" particularly evident in the k-NN model's poor performance. Although Random Forest achieved perfect training accuracy, its tendency to overfit the noise in gaming data limited its reliability. On the other hand, SVM demonstrated commendable resilience, proving that it can effectively map the complex, non-linear relationships inherent in team compositions and itemizations.

5.2 Future Work

Future research could focus on incorporating high-impact game mechanics such as the Augment system and situational dynamics (e.g., leveling timing and economic management). Transitioning from traditional statistical models to Deep Learning architectures, such as Word Embeddings or Transformers, could further enhance the model's ability to learn intricate team synergies and contextual sequences. Additionally, exploring Ensemble Learning techniques to combine the strengths of multiple algorithms remains a promising path for achieving higher predictive stability.

References

[1] Teamfight Tactics open beta goes live in North America tomorrow. Available: https://www.pcgamer.com/teamfight-tactics-open-beta-goes-live-in-north-america-tomorrow/

[2] Interested in attending the TFT Vegas Open? Available: https://teamfighttactics.leagueoflegends.com/th-th/news/esports/so-you-want-to-attend-the-tft-vegas-open/

[3] Magic n' Mayhem Tactician's Crown Overview. Available: https://teamfighttactics.leagueoflegends.com/th-th/news/esports/magic-n-mayhem-tacticians-crown-primer/

[4] Riot Games announces "Open Tournament 2024: Stand Your Ground." Available: https://workpointtoday.com/riot-games/

[5] Riot API. Available: https://developer.riotgames.com/apis

[6] T. D. Do, S. I. Wang, D. S. Yu, M. G. McMillian, and R. P. McMahan, "Using Machine Learn-ing to Predict Game Outcomes Based on Player-Champion Experience in League of Legends," in ACM International Conference Proceeding Series, Association for Computing Machinery, Aug. 2021.

[7] B. Jailson S Junior and E. Claudio C Campelo, "League of Legends: Real-Time Result Prediction."

[8] J. A. Hitar-Garcia, L. Moran-Fernandez, and V. Bolon-Canedo, "Machine Learning Methods for Predicting League of Legends Game Outcome," IEEE Trans. Games, vol. 15, no. 2, pp. 171–181, Jun. 2023.

[9] A. Tuzcu, G. Ay, A. Umay Uçar, and D. Kılınç, A Machine Learning Based Predictive Analysis Use Case For eSports Games. 2023.

[10] K. Kalyanaraman, "To win or not to win? A prediction model to determine the outcome of a DotA2 match," La Jolla, 2015.

[11] K. Conley and D. Perry, "How Does He Saw Me? A Recommendation Engine for Picking He-roes in Dota 2," Stanford, 2013. [Online]. Available: http://www.youtube.com/watch?v=BAC9B9z

[12] A. Semenov, P. Romov, S. Korolev, D. Yashkov, and K. Neklyudov, "Performance of Machine Learning Algorithms in Predicting Game Outcome from Drafts in Dota 2," 2016.