A Greedy Algorithm Approach for Feature Selection to Maximize Predictive Accuracy with Minimal Model Complexity

I Made Wiweka Putera - 13523160
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: wiwekaputera@gmail.com , 13523160@std.stei.itb.ac.id

***Abstract*—Datasets with too many features are a common problem in machine learning, making models slow, inaccurate, and hard to understand. This paper tests a method to select only the most important features, with the goal of achieving high prediction accuracy using a much simpler model. We use a greedy algorithm called Sequential Forward Selection (SFS). This method starts with an empty set and, one by one, adds the single feature that provides the biggest improvement to the model's accuracy. To test this method, we used a large public dataset of Dota 2 matches, aiming to predict the winner of a match based on data from the first five minutes. A Logistic Regression classifier was used to evaluate the performance of each feature subset. The experiment was successful. The analysis showed a point of diminishing returns, where using approximately 15-20 features achieved nearly the same performance as using 45. By selecting a practical subset of features, we significantly reduce model complexity without a meaningful loss in accuracy, achieving a final accuracy of 87.48% on unseen data. This study validates the greedy approach as a powerful strategy for finding efficient, high-performing models in complex domains like esports analytics.**

Keywords— greedy algorithm, feature selection, wrapper method, machine learning, logistic regression, esports analytics.

I. Introduction

In artificial intelligence (AI), an agent is basically something that can look at its situation and make a decision to reach a goal [1]. You can think of our machine learning model as one of these agents. Its 'situation' is all the data from a Dota 2 match. It 'looks' at this data by using the input features, like a player's gold or number of deaths. The 'decision' it makes is its prediction of which team will win the game. A 'good' or 'rational' agent is one that is effective at its job. The job of our agent is to be as accurate as possible, so its main goal is to find the perfect set of features that helps it make the most correct predictions.

A common problem that reduces an agent's performance is "high dimensionality," which arises from an excessive number of features [2]. This "Curse of Dimensionality" presents several distinct problems. First, with more features, the volume of the data space increases exponentially, making the available data sparse. Second, a high number of features increases the likelihood that the model will find spurious correlations in the training data, leading it to learn from random noise rather than true underlying patterns—a problem known as overfitting. This results in a model that performs well on data it has already seen but fails to generalize to new, unseen data. Finally, models with excessive features are more computationally expensive to train and harder for humans to interpret [2]. To fix these problems, we use feature selection.

The main problem this paper solves is one of optimization. If a dataset has N features, there are 2^N possible combinations of features to test. Trying all of them, a method called brute-force search, is simply impossible for computers to finish with most real datasets [3]. Therefore, we need a smarter, more practical approach. Our goal is to find a method that can identify a small set of features that achieves two things at once: (1) maximizes the model's predictive accuracy, and (2) keeps the model simple by using as few features as possible.

This paper proposes using a greedy algorithm as a practical and effective way to solve this optimization problem. A greedy algorithm works by making the best possible choice at each step to build a final, high-quality solution [3]. To show how it works in a real scenario, we use a "wrapper-based method," where our greedy algorithm selects features and uses a Logistic Regression model to test how good they are [4].

We chose to apply this method to an esports dataset for two main reasons. First, the global esports market is experiencing rapid growth, making it a timely and relevant field for data analysis. Second, esports data is famously high-dimensional and complex, providing a challenging and suitable test case for our algorithm [5]. It is important to note that while Dota 2 is our case study, the feature selection methodology discussed in this paper is general and can be applied to other complex, high-dimensional datasets in fields such as finance or bioinformatics.

II. Theoretical Foundation

A. The Greedy Algorithm Paradigm

The greedy algorithm is a straightforward method for solving optimization problems. Its core principle is to make a locally optimal choice at each stage with the hope of finding a global optimum [3]. The algorithm builds a solution step-by-step, and at each step, it makes a choice that appears to be the best at that moment, without considering the future consequences of that

choice. This immediate best choice is called the local optimum. The hope is that by making a series of locally optimal choices, the algorithm will arrive at a global optimum—the best overall solution.

The formal elements of a greedy algorithm are as follows :

1. Candidate Set: At each step of the SFS algorithm, this is the set of all features that have not yet been added to the final solution.
2. Solution Set: This is the set of features that have already been selected in previous steps. It starts empty and grows by one feature at each iteration.
3. Selection Function: This is the heart of the greedy choice. In our wrapper method, the selection function involves training and evaluating our Logistic Regression model with a temporary feature set (the current solution set plus one candidate feature). The function "selects" the candidate feature that results in the highest model accuracy.
4. Feasibility Function: In this problem, the feasibility check is simple. A choice is feasible as long as there are still features in the candidate set to evaluate.
5. Objective Function: The overall goal is to maximize the final model's predictive accuracy, which the algorithm attempts to achieve by maximizing the accuracy at each individual step.

A key weakness of greedy algorithms is that they are not guaranteed to find the true global optimum for all problems. To illustrate, consider the classic "change-making problem". Given a set of coins (e.g., 1, 7, 10), and a target amount of 15, a greedy algorithm would first take the largest coin (10), leaving 5. It cannot take 7, so it must take five 1-unit coins, for a total of six coins (10, 1, 1, 1, 1, 1). This is a locally optimal choice at each step, but the true global optimum is to take two 7-unit coins and one 1-unit coin (a total of three coins). This "short-sighted" nature is a known limitation [3]. However, for extremely complex problems like feature selection, where the search space is too vast for exhaustive methods, a greedy approach is a fast and practical way to find a solution that is "good enough" and often very close to the true optimum.

B. Feature Selection Strategies

Feature selection methods are generally grouped into three categories based on how they interact with the machine learning model [2, 4].

1. Filter Methods: These methods act as a pre-processing step, ranking features based on statistical properties like correlation or mutual information before any model is trained. For example, a filter might calculate the correlation of each feature with the target variable and discard any features below a certain threshold. They are very fast and model-agnostic, but their primary weakness is that they evaluate features in isolation, potentially missing complex relationships where a combination of individually weak features is strongly predictive.
2. Wrapper Methods: These methods use a specific machine learning model to evaluate subsets of features. They "wrap" the feature selection process around the model, effectively using the model's performance as the objective function for the search. This approach is often more accurate because it considers feature interactions and is tailored to the model's performance. However, because it requires training thousands of models (one for each subset), it is much slower and more computationally expensive. The greedy search we use in this paper is a type of wrapper method.
3. Embedded Methods: These methods perform feature selection as part of the model training process itself. A common example is LASSO (L1 regularization) regression, which adds a penalty to the model's loss function based on the magnitude of the feature coefficients. This penalty forces the coefficients of less important features towards zero, effectively removing them from the model. They offer a good balance between the speed of filters and the accuracy of wrappers but are intrinsically tied to the specific model they are embedded in.

C. Sequential Forward Selection as a Greedy Search

The task of finding the best subset of features can be framed as a classic AI search problem [3]. In this context:

- States: Each possible subset of features is a state in the search space.
- Initial State: The empty set of features, {}.
- Actions: The actions available are adding a feature to the current subset.
- Goal State: The state (subset of features) that maximizes the objective function (model accuracy).

With N features, the size of this state space is 2^N, making an uninformed, brute-force search that checks every state computationally infeasible. Instead, we must use an intelligent search strategy. Sequential Forward Selection (SFS) is a form of greedy local search, also known as a hill-climbing algorithm [1]. It starts at the initial state and, at each step, moves to the best immediate neighboring state—the one that provides the largest increase in accuracy—without ever backtracking. This approach drastically reduces the search complexity compared to brute-force. The specific steps of this greedy search process are as follows:

1. Begin at the initial state with an empty set of selected features.
2. In the first iteration, evaluate all possible actions by creating a temporary subset for each individual feature and training a model. Select the single feature (action) that results in the best model performance. This becomes the new current state.
3. In each subsequent iteration, evaluate all neighboring states by temporarily adding each of the remaining features to the current set.

4. Permanently add the feature that provides the largest improvement in model performance to the solution set, thus moving to the new best state.
5. Repeat this process until a stopping criterion is met, such as reaching a desired number of features.

While this greedy approach risks getting stuck in local optima, as discussed previously, it is often a highly effective heuristic for this type of problem. A local optimum in feature selection occurs when the algorithm makes a choice that seems best at the time, but which prevents a better combination from ever being discovered. For example, two features might be weak individually but very strong together. SFS might discard both early on because neither provides a large individual benefit, thus getting "stuck" in a suboptimal solution path.

D. Alternative Search Strategies

To overcome the local optima problem of simple greedy search, more complex search strategies exist. Stochastic hill-climbing, for example, introduces randomness by occasionally picking a suboptimal move to escape a local peak and explore other areas of the search space. A more advanced version, simulated annealing, allows "bad" moves more frequently at the beginning of the search and gradually reduces this randomness, analogous to the cooling of a metal.

Another common greedy strategy is Sequential Backward Elimination (SBE). It is the opposite of SFS: it starts with all features and, at each step, greedily *removes* the one feature whose removal causes the smallest drop (or largest increase) in performance. While SFS and SBE are both greedy, they can sometimes arrive at different final feature sets. For this paper, SFS was chosen due to its conceptual simplicity and efficiency, as it starts with smaller, faster-to-evaluate models.

E. Logistic Regression

For this study, we use Logistic Regression as our classification model. It is a simple, fast, and widely used algorithm for binary classification problems where the outcome has two classes, like "win" or "loss". It works by passing a linear combination of the input features through a sigmoid function, which outputs a probability between 0 and 1. Because it is computationally efficient and its results are relatively easy to interpret, it is an excellent choice for a wrapper method, where the model must be trained thousands of times [2].

III. METHODOLOGY

A. Dataset

The experiment uses data from "Dota 2: Predicting match outcome," a Kaggle community competition dataset hosted by WIN.gg [6]. Dota 2 is a complex multiplayer online battle arena (MOBA) game where two teams of five players, the Radiant and the Dire, compete to destroy the opposing team's base. This dataset captures the state of over 90 features at the 5-minute mark of each game. These features include crucial economic indicators like gold (in-game currency to buy items) and experience (used to level up hero abilities), and player actions such as kills (defeating an opponent) and deaths. The target variable for our prediction is binary: whether the "Radiant" team won the match. The high dimensionality of this data makes it an ideal test case for demonstrating the value of feature selection.

B. Data Preprocessing

Before training the model, several preprocessing steps were performed. First, any rows with missing values in the radiant_win target column were removed to ensure data integrity. The dataset includes features for each player's chosen character, known as a hero. As these are categorical identifiers, they were excluded to focus the selection on purely numerical game-state variables. Next, any remaining missing values in the feature columns were addressed. These primarily related to "first blood"—a bonus given for the first kill of the match—which would have missing values if the event did not occur within the first 5 minutes. These missing values were filled with 0, as the absence of the event is itself a piece of information. Finally, all numerical features were standardized using StandardScaler from scikit-learn. This process scales the data to have a mean of 0 and a standard deviation of 1, which is crucial for the optimal performance of a Logistic Regression model as it prevents features with larger scales from disproportionately influencing the model's coefficients.

C. Experimental Setup

The experiment was implemented in Python using the pandas library for data manipulation, scikit-learn for the Logistic Regression model and data scaling, and mlxtend for the Sequential Forward Selection implementation. The preprocessed dataset was split into a training set (80%) and a testing set (20%). This split uses a fixed random_state for reproducibility and stratification. Stratification ensures that both the training and testing sets have the same proportion of wins and losses as the original dataset, which is crucial for preventing bias when training the model and evaluating its performance.

D. Greedy Agent Implementation

The core of this research is the implementation of the SFS algorithm.

- Model: A LogisticRegression classifier is used as the estimator.
- Search: The SequentialFeatureSelector is configured for a forward search.
- Objective: The search aims to find the feature subset that maximizes the model's accuracy, evaluated using 5-fold cross-validation on the training set.
- Process: The SFS algorithm was run to select feature subsets ranging from 1 up to a maximum of 45 features.

E. Evaluation Metrics

While Accuracy is the primary performance metric, other evaluation tools provide deeper insight. The confusion matrix breaks down predictions into four categories: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). This allows for the calculation of:

- Precision: Measures the accuracy of positive predictions (TP / (TP + FP)). High precision means that when the model predicts a win, it is very likely correct.
- Recall (Sensitivity): Measures the model's ability to find all actual positive instances (TP / (TP + FN)). High recall means the model is good at correctly identifying all the matches that were actually won.

Another powerful tool is the Receiver Operating Characteristic (ROC) Curve, which plots the True Positive Rate against the False Positive Rate at various classification thresholds. The Area Under the Curve (AUC) summarizes the ROC curve into a single value, representing the model's ability to distinguish between the positive and negative classes. An AUC of 1.0 indicates a perfect classifier, while an AUC of 0.5 indicates performance no better than random chance. For this study, given the balanced nature of the dataset, accuracy remains the primary metric, but these other tools provide valuable context for the final model's performance.

IV. Results and Analysis

A. *Performance Trend*

The simulation was conducted to evaluate the performance of the Logistic Regression model as the number of features selected by the greedy algorithm increased. The relationship between the number of selected features and the model's accuracy is shown in Figure 1.

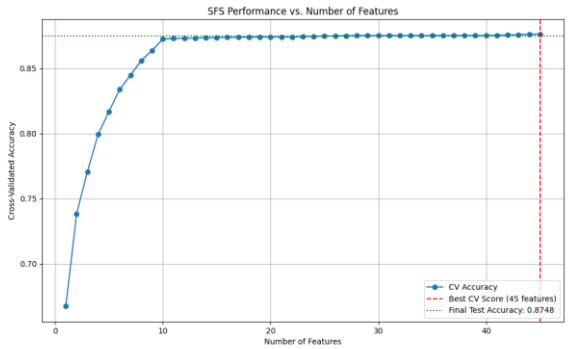

Fig. 1. Model prediction accuracy as more features are added by the SFS algorithm. The line shows the average cross-validated accuracy, and the shaded area represents the standard deviation. The horizontal line shows the final accuracy on the unseen test set.

As seen in Figure 1, the model's cross-validated accuracy increases sharply with the first 10-15 features. The performance continues to improve, but at a much slower rate, until it reaches a plateau. This clearly illustrates the concept of diminishing returns, where each additional feature provides less and less benefit.

B. *Optimal Feature Subset*

The SFS process determined that the absolute peak cross-validated accuracy of 0.8752 was achieved with a subset of 45 features. A final model trained on this peak feature set was evaluated on the held-out test set, achieving a final accuracy of 0.8748. The performance of this final model on the test set is detailed in the confusion matrix in Figure 2.

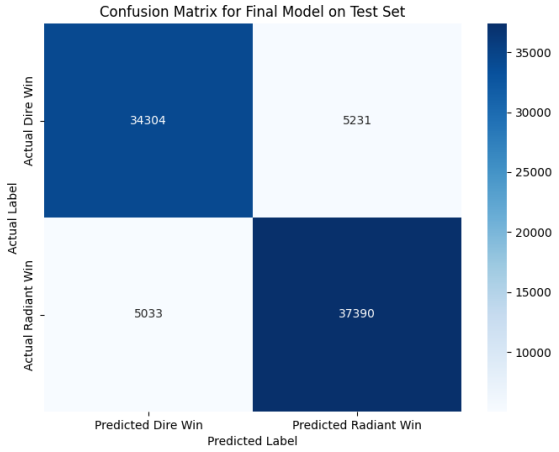

Fig. 2. A confusion matrix showing the performance of the final model on the test set.

Figure 2 provides a detailed breakdown of the final model's predictions on the unseen test data. The model correctly predicted 34,304 Dire wins (True Negatives) and 37,390 Radiant wins (True Positives). It made two types of errors: incorrectly predicting a Radiant win 5,231 times when Dire actually won (False Positives), and incorrectly predicting a Dire win 5,033 times when Radiant actually won (False Negatives). This balanced distribution of errors indicates that the model is not significantly biased towards predicting one outcome over the other.

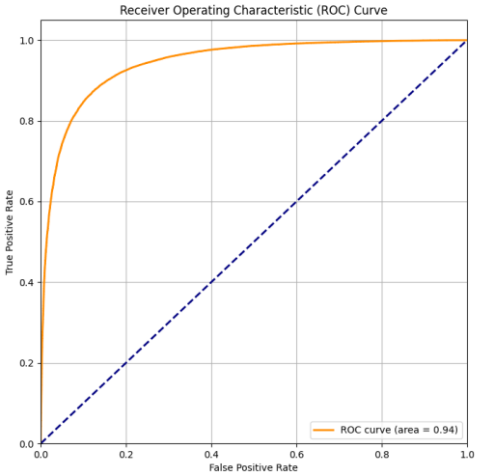

Fig. 3. Receiver Operating Characteristic (ROC) curve for the final model.

The ROC curve in Figure 3 further validates the model's strong performance. The curve bows significantly towards the top-left corner, far from the diagonal line representing a random-guess model. The calculated Area Under the Curve

(AUC) of 0.94 indicates excellent discriminatory power. An AUC of 0.94 means that if we randomly select one Radiant-win match and one Dire-win match, there is a 94% chance that our model will correctly assign a higher win probability to the actual winner.

C. Selected Features

The SFS algorithm provides a ranked list of features based on the order they were selected. This ranking reveals which game state variables the greedy agent identified as the most powerful individual predictors early in the process. Table I presents the top 10 features chosen by the algorithm. Examining this list gives us our first insight into the underlying factors that drive match outcomes. While these features are specific to Dota 2, the principle of identifying and ranking key performance indicators (KPIs) through a greedy selection process is a widely applicable strategy in many other data-driven fields, from financial modeling to medical diagnostics.

TABLE I. Top 10 Features Selected by The Greedy Algorithm

Order	***Feature Name***	***Description***
1	r1_deaths	Deaths of Radiant player 1
2	r1_items	Items of Radiant player 1
3	r2_level	Level of Radiant player 2
4	r2_lh	Last hits of Radiant player 2
5	r2_kills	Kills by Radiant player 2
6	r2_deaths	Deaths of Radiant player 2
7	r3_level	Level of Radiant player 3
8	r3_gold	Gold of Radiant player 3
9	r3_lh	Last hits of Radiant player 3
10	r3_kills	Kills by Radiant player 3

V. Discussions

A. Analysis of Diminishing Returns

The most critical finding from this experiment is the clear evidence of diminishing returns shown in Figure 1. While the technical peak in performance was found with a large number of features, the graph shows that a vast majority of the model's predictive power is gained from a much smaller subset. For instance, a model with only 15-20 features already achieves an accuracy that is very close to the peak performance.

This observation is central to the paper's objective. It demonstrates that pursuing the absolute maximum accuracy leads to a model that is unnecessarily complex. A more practical approach, which is enabled by this analysis, is to select a feature set from the "elbow" of the performance curve. By choosing a subset of ~20 features instead of 45, we can create a model that is significantly faster, easier to interpret, and less prone to overfitting, while sacrificing a negligible amount of predictive accuracy. This trade-off is highly desirable in real-world applications.

B. Interpretation of Key Features

Analyzing the features selected by the algorithm provides insight into the game itself. Figure 3 shows the coefficients assigned by the final Logistic Regression model to the top 15 features, indicating the weight and direction of their influence.

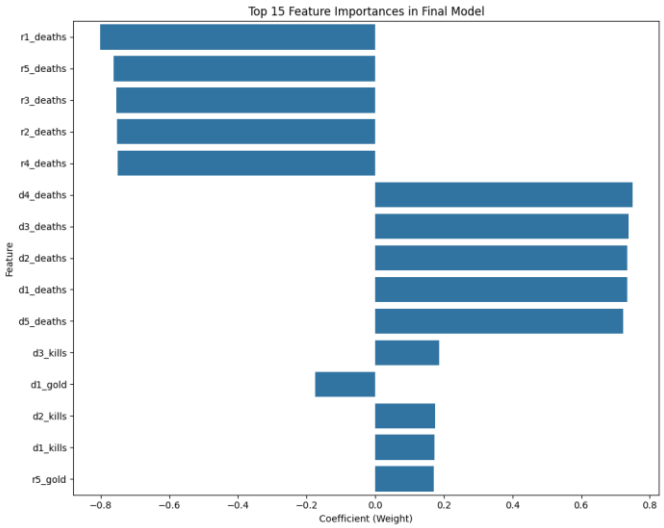

Fig. 4. Feature importance from the final model's coefficients. Positive values predict a Radiant win, while negative values predict a Dire win.

The plot of feature importances in Figure 3 powerfully confirms real-world Dota 2 knowledge. The features with the largest coefficients (both positive and negative) are overwhelmingly related to player deaths. For example, r1_deaths has a large negative coefficient, meaning that as the number of deaths for Radiant Player 1 increases, the probability of a Radiant win sharply decreases. Conversely, d4_deaths has a large positive coefficient, indicating that as Dire Player 4 dies more, the probability of a Radiant win increases. This demonstrates that the model has learned the most fundamental principle of early-game Dota: securing kills and avoiding deaths is the strongest driver of victory. The model is not just finding abstract correlations; it is quantifying the game's core "snowball" mechanic.

C. Practical Implications and Applications

The findings of this study have direct practical applications. A lightweight model using only the ~20 most important features could be deployed as a real-time win probability predictor. Such a tool would be invaluable for esports broadcast production, providing viewers with live, data-driven insights. It could also be used as a coaching tool, allowing teams to analyze the early stages of their practice games to identify which key performance indicators are most correlated with their success or failure. The simplicity of the model means it can run quickly and efficiently, making these applications feasible.

D. Limitations of the Study

It is important to acknowledge the limitations of this study. First, the analysis is limited to data from the first five minutes of the game. While predictive, this snapshot does not capture

mid-game or late-game dynamics that can turn the tide of a match. Second, by excluding the categorical hero features, we have ignored the significant impact of hero matchups and team composition, which is a key area of strategic depth in Dota 2. A more complex model would be needed to incorporate this information effectively. Finally, the choice of Logistic Regression, while good for this study's purpose, is a relatively simple linear model. Other, more complex algorithms like Gradient Boosting or Neural Networks might be able to find more intricate patterns in the data.

VI. Conclusion and Future Work

A. Conclusion

This paper successfully implemented a greedy algorithm for feature selection and demonstrated its effectiveness on a complex esports dataset. The study showed that this approach can effectively identify a small, powerful subset of features, achieving a balance between high predictive accuracy and minimal model complexity. The methodology not only confirmed existing domain knowledge by selecting logically important features but also provided a quantitative framework for understanding the trade-off between model complexity and performance. The analysis revealed a clear point of diminishing returns, proving that a significantly simpler model can achieve near-peak performance. This validates the greedy strategy as a practical and powerful tool for building efficient and interpretable models from high-dimensional data, and provides a clear path for creating practical predictive tools for Dota 2 analytics.

B. Limitations of the Study

Future research could expand on this work in several directions. One could apply this methodology to different machine learning models, such as Support Vector Machines or Neural Networks, to see how the optimal feature set changes depending on the classifier. Another path would be to incorporate the categorical hero data, perhaps using embedding techniques to represent hero matchups numerically. Finally, a dynamic version of the agent could be developed to analyze how the importance of features changes as a match progresses beyond the first 5 minutes.

Video Link at Youtube

https://youtu.be/U-KJHtviN10

Github Repository Link

https://github.com/wiwekaputera/greedy-feature-selection-dota2

Acknowledgment

The author wishes to express sincere gratitude to all parties who have contributed to the completion of this paper. First and foremost, praise and gratitude to God Almighty for the blessings and grace throughout this journey. Special thanks are extended to the lecturers of the IF2211 Algorithmic Strategies course, Dr. Ir. Rinaldi, M.T., Mr. Monterico Adrian, S.T., M.T., and Dr. Nur Ulfa Maulidevi, S.T, M.Sc., for their invaluable knowledge and guidance. The author hopes this paper will be beneficial and make a positive contribution.

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

[2] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157-1182, 2003.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[4] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273-324, 1997.

[5] M. Schubert, A. Drachen, and T. Mahlmann, "Esports analytics through encounter detection," in *MIT Sloan Sports Analytics Conference*, 2016.

[6] WIN.gg, "Dota 2: Predicting match outcome," Kaggle, 2019. [Online]. Available: https://www.kaggle.com/competitions/dota-2-prediction/data.

Declaration

I hereby declare that this paper is my own original work and is not an adaptation or translation of another's work, and is not plagiarism.

Bandung, June 24, 2025

I Made Wiweka Putera (1352310)