Analysing Game Mechanics Used to Predict Performance:

Can a player's focused game mechanic predict the total damage they deal during a match?

Victoria Putri Setiawan Halim, Nicholas Setyawan, Jeff Mario Kartanegara, Stephanie Daniella Hartono

[redacted]

May 19, 2022

Word Count: 2055

1 Introduction

League of Legends is a competitive Multiplayer Online Battle Arena game, a video game genre where ten players are split into two teams and go head-to-head against each other in an enclosed space. The goal is to gather resources, such as gold and experience by defeating minions, jungle monsters, and opposing players in combat, and finishing off the opposing team's buildings. Winning requires a team to destroy the opposing team's *Nexus* (also known as the enemy's base). Multiple factors take account in each match: the draft of characters (referred to as champions), strategy, playstyle, and more. The varied factors all contribute to deciding which team would win.

2 Research Question

In every match, players in each team allocate themselves to 5 distinct roles. *TopLaner* and *Jungler* roles in particular are responsible for the majority of the damage dealt in a match, which is a crucial factor since higher damage outputs would mean more dominance over the opposing team.

Different players focus on varying aspects of the game. To maximise total damage dealt in a match, players must balance four different game mechanics:

1. "*Farming*": Levelling up and earning gold to purchase useful game items.
2. *"Combat"*: Battling enemy champions to weaken their offensive power.
3. "*Objective*": Damaging the opposing team's offensive structures and the *Nexus*.
4. "*Support*": Assisting teammates and keeping them alive.

Thus, this research aims to **predict the total damage a player would inflict in a match, given the game mechanics they decide to focus on**.

3 Target Audience

Like most modern competitive multiplayer games, League of Legends has a large competitive scene. All players are able to compete on an online ladder ranking, and professional players compete with their teams against other highly skilled teams in tournaments. The largest tournament is the annual League of Legends World Championship and is one of the most-streamed competitions worldwide. The outcome of this report will help determine the game mechanics competitive teams should focus on to maximise their damage output, and ergo win. It can also aid non-professional players in emphasising a playstyle, assisting them in climbing the online competitive ranked ladder.

4 Dataset and Features

The dataset chosen for this experiment was collected by Andrew Suter[1] and contains the end-game data of every *Challenger* ranked (Top 50 in their server) player's matches throughout January 2022. The dataset spans across three regions: North America (NA), Europe (EU), and

[1] Original dataset: LoL Challenger Soloq Data (Jan, Kr-Na-Euw) | Kaggle

Korea (KR). All the data points collected are distinct and were randomly selected per match to ensure independence. Selected features of the dataset are described in *Table 1* below.

Feature	Description
d_spell	Amount of times the summoner spell[2] on the *d* key is used
f_spell	Amount of times the summoner spell on the *f* key is used
champion	The character the player is playing
side	The team the player is on
role	The role the player is playing as
assists	Number of times the player helped a teammate kill an enemy
damage_turrets	Amount of damage dealt to turrets [3]
damage_objectives	Amount of damage dealt to objectives[4]
damage_building	Amount of damage dealt to buildings[5]
deaths	Number of deaths
gold_earned	Amount of gold earned
kda	The player's **k**ill **d**eath **a**ssist ratio[6]
kills	Number of kills
level	Player's champion level
time_cc	Time spent crowd controlling[7]
damage_total	Total damage dealt by a champion
damage_taken	Total damage a champion took
minions_killed	Number of minions[8] killed
vision_score	Player's vision score[9]

Table 1: Selected Features from the Dataset

Support	Objective	Combat	Farming

[2] Helpful abilities players can use on their champions
[3] Enemy structures that deal damage to the player
[4] Includes enemy's offensive structures, the Nexus, and enemy 'Epic Monsters'
[5] Includes enemy's offensive structures and the Nexus
[6] Ratio is calculated using the formula: *(kills + assists) / deaths*
[7] Abilities that limit the mobility/combat ability of enemies, such as *stun*, *slow*, *charm*, etc.
[8] Offensive enemy units that are continuously spawned by the Nexus
[9] How much vision a player has influenced in the game

assists	damage_objectives	kda	gold
vision_score	damage_turrets	kills	level
time_cc	damage_building	deaths	gold_earned
	turret_kills	damage_taken	minions_killed

Table 2: Categorization of Features into Predefined Game Mechanics

The *d_spell* and *f_spell* data have been removed from the dataset since they are too vague; they do not specify which summoner spell was used. "*side*" was also removed since it is irrelevant to the research.

5 Method

5.1 Pre-processing

5.1.1 Merging Datasets

The original dataset stored all the match data in three separate CSV files, organised by region (NA, EU, KR). Region is irrelevant to this research, so all the match data was merged into a single *DataFrame*.

5.1.2 Noise Removal

The dataset was filtered to only include champions that:

Fall under the *TopLane_Jungle* role.

- Since, damage dealing is most relevant to *Top Laner* or *Jungler* Champions.

Were played more than 53 times as a *Top Laner* or *Jungler*[10].

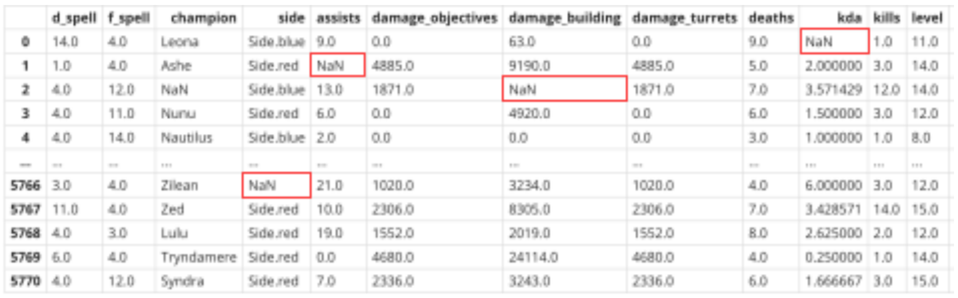
- Each champion tends to be played in certain roles to maximise their unique abilities. For instance, champion *'Lulu'* is almost always played as a supportive role, but is played as a *TopLaner* in rare instances.
- Champions unsuited as a *TopLaner* or *Jungler*, but were played as either of those roles, are likely to introduce noise into the research data. Thus this research will only focus on champions typically played as *TopLaner* or *Jungler* roles.

5.1.3 Numerical Imputation

The current dataset has missing values, represented by *NaN*, as seen in Figure 1. It is safe to assume that these values are *Missing Completely at Random*. Thus numerical imputation was chosen to deal with this issue to preserve as much data as possible for data processing, which, in

[10]Calculated from the average number of times a champion is played

turn, will help the model become more reliable. This will be done by replacing missing data points with the mean of the entire column.



	d_spell	f_spell	champion	side	assists	damage_objectives	damage_building	damage_turrets	deaths	kda	kills	level
0	14.0	4.0	Leona	Side.blue	9.0	0.0	63.0	0.0	9.0	NaN	1.0	11.0
1	1.0	4.0	Ashe	Side.red	NaN	4885.0	9190.0	4885.0	5.0	2.000000	3.0	14.0
2	4.0	12.0	NaN	Side.blue	13.0	1871.0	NaN	1871.0	7.0	3.571429	12.0	14.0
3	4.0	11.0	Nunu	Side.red	6.0	0.0	4920.0	0.0	6.0	1.500000	3.0	12.0
4	4.0	14.0	Nautilus	Side.blue	2.0	0.0	0.0	0.0	3.0	1.000000	1.0	8.0
...	...	...	...	...	...	...	...	...	...	...	...	...
5766	3.0	4.0	Zilean	NaN	21.0	1020.0	3234.0	1020.0	4.0	6.000000	3.0	12.0
5767	11.0	4.0	Zed	Side.red	10.0	2306.0	8305.0	2306.0	7.0	3.428571	14.0	15.0
5768	4.0	3.0	Lulu	Side.red	19.0	1552.0	2019.0	1552.0	8.0	2.625000	2.0	12.0
5769	6.0	4.0	Tryndamere	Side.red	0.0	4680.0	24114.0	4680.0	4.0	0.250000	1.0	14.0
5770	4.0	12.0	Syndra	Side.red	7.0	2336.0	3243.0	2336.0	6.0	1.666667	3.0	15.0

Figure 1: Table Demonstrating NaN values in the Dataset

5.1.4 Normalisation

Normalisation was used to scale all numerical features in the dataset to the interval [0, 1] to prepare the data for feature selection by normalised mutual information. To do so, the following formula was used:

$$X_{Normalised} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

5.2 Feature Selection

Considering too many features for the research would just contribute to noise. Thus, we must only select features sufficiently correlated with *damage_total* for further data processing. Normalised Mutual Information (NMI) was the chosen measure of correlation since its range of [0, 1] provides easy interpretability.

A feature was deemed to share a significant relationship with *damage_total* if its NMI with the variable exceeds 0.3; this threshold was chosen since it filters out an appropriate amount of features. Below are the features which were selected for further processing on the basis of NMI.

Feature	Mutual information with *damage_total*
gold_earned	0.8429
level	0.6329
damage_building	0.3938
damage_taken	0.3314

Table 3: Selected features and their mutual information with damage_total

5.3 Train-Test Split

The filtered dataset was shuffled and split into training and testing sets (75/25 split). The models will be built using the train set, and then evaluated using the test set. This split was done to more accurately test the performance of the models created against unseen data.

A train-test split is required for these types of investigations as it is a way to test the performance of the models created against unseen data. As the current dataset is moderately large, it is appropriate to use this method to help with the evaluation later on in the method. It is not recommended to use a train-test split if the database is relatively small, as it runs the risk of overfitting the data.

5.4 Ordinalization

All the selected features, along with *damage_total*, were ordinalized so results could be interpreted more easily and so the predictive models will not overfit the training set. *damage_total* was ordinalized into three different classes depending on the amount of damage dealt: "*low*", "*mid*", and "*high*". The features selected in *Section 5.2* were also ordinalized to the same "*low*", "*mid*", "*high*" classes, but with a numeric representation instead (0, 1, 2).

5.4.1 Bin Formation

For this research it was assumed that the numeric features followed a normal distribution. It was also assumed that the majority of *damage_total* values would fall into the "*mid*" category, with fewer instances of the two extremes "*high*" and "*low*".

With these two assumptions, each numeric feature was ordinalized by the method below:

1. Find the mean (μ) and standard deviation (σ) of the numeric feature using the **training** set.
2. Form the bins for each ordinal category, where:
    a. "low" (0): $[0, \mu-\sigma)$
    b. "mid" (1): $[\mu-\sigma, \mu+\sigma]$
    c. "high" (1): $(\mu+\sigma, 600{,}000]$
3. Use the bins to ordinalize the training and test sets
4. Repeat until all features are ordinalized

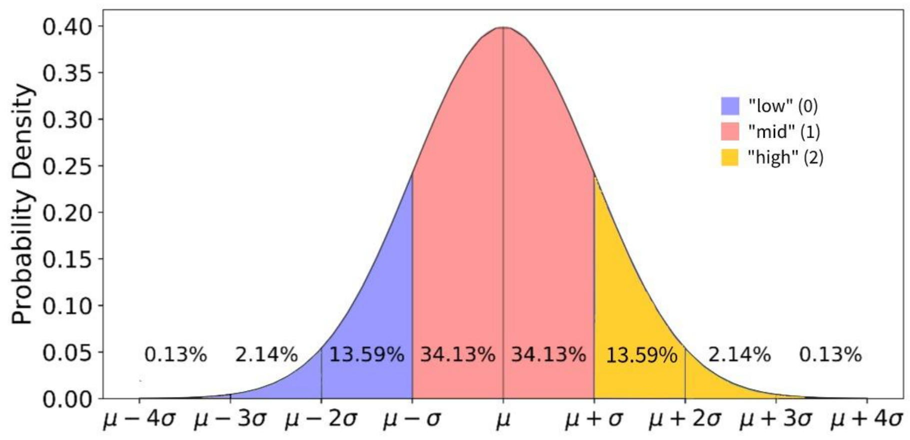


Figure 2: Normal distribution curve, modified to show how features will be binned (Galarnyk, 2018)

5.5 Predictive Models

5.5.1 Decision Tree

The training set was used to model a decision tree which predicts the class of *damage_total*, given input data containing the selected features (*gold_earned*, *level*, *damage_taken*, *damage_building*). The choice of a decision tree is because they effectively visualise classification as compared to other classification models. Additionally, they are easy to understand, and are more flexible with numerical and categorical data.

Before modelling, 10-fold cross validation was used to determine the decision tree depth which gives the highest accuracy; this was found to be four. The decision tree model can be found below in *section 4*.

5.5.2 k-Nearest Neighbours

A *k*-Nearest Neighbours (*k*-NN or KNN) algorithm was also used to create a predictive model. This algorithm is based on finding the data points that have the shortest distance to the current stored records. A parameter that needs to be chosen for this algorithm is *k*; which represents the number of other data points to include in the classification of data. It is crucial to pick a *k*-value of the right size, as a *k* value that is too small may be impacted by small noises in data, and a *k*-value that is too big may include points that belong to other classes.

To pick the *k* value used in the algorithm, the following pseudocode was used:

```
for k value in a range from 1 to 10
        calculate knn
        fit it onto training data
        map it onto testing data
        calculate accuracy
```

Figure 3: Pseudocode to Find k-value

It was decided to check *k*-values between 1 and 10 to prevent computations from becoming extremely expensive and unnecessary for the scope of this research. To better understand the pseudocode above, the following visualisation was generated:

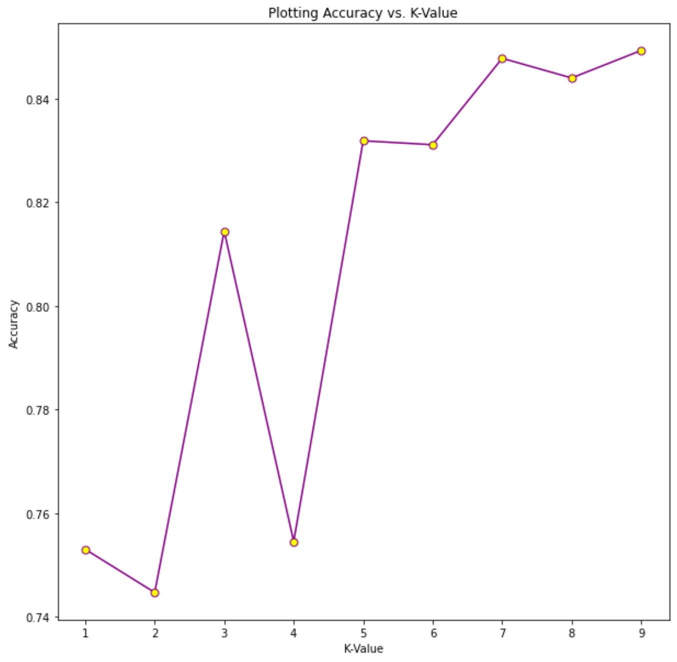


Figure 4: Graphing the Accuracy of the Model Depending on the k-value

5.6 Model Evaluation

5.6.1 Confusion Matrices

Confusion matrices were used to evaluate the performance of the classification models. They compare the classes given by the predictive model and their true class. The diagonal boxes (spanning from the top-left to the bottom-right boxes) represent the correct classifications from the model (the model predicted the same class as the actual data).

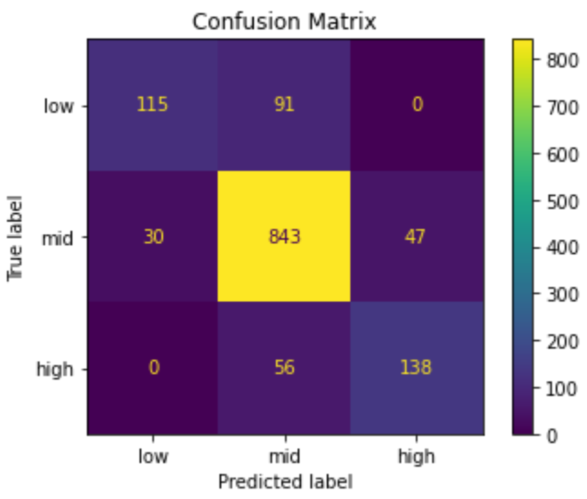


Figure 5: Confusion Matrix for Decision Tree Model

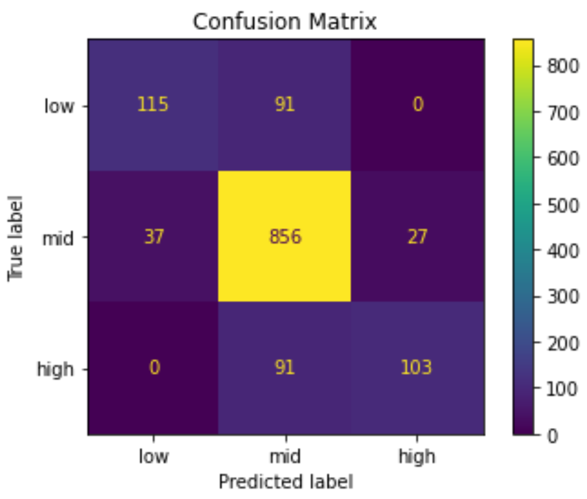


Figure 6: Confusion Matrix for k-Nearest Neighbours Model

Both models make most accurate predictions for *damage_total* values that fall into the "*mid*" category. However, both lack accuracy for predicting the minority classes "*high*" and "*low*". The decision tree slightly outperformed the KNN Model by that it was able to more accurately classify "*mid*" values.

5.6.2 Performance Metrics

The models' accuracy score was used as a performance metric since it indicates how often they predict the correct class for the test set. However, accuracy can be misleadingly high in imbalanced problems such as in this research, where *"low"* and *"high"* are minority classes. *f1* score, which is the harmonic mean between the calculated precision[11] and recall[12], takes into account this class imbalance. Thus it was also used as a performance metric.

Both metrics range from 0 to 1; as the value approaches 1, it can be said that the model is accurate/classifies the data well.

Statistic	Decision Tree Model	KNN Model
Accuracy	0.8303	0.8136
f1	0.7554	0.6185

Table 4: Comparing the Performance Metrics of Both Models

The decision tree model was found to have the higher accuracy and *f1* score. All in all this indicates that it predicts the correct classes, despite the class imbalances, more often.

[11] Agreement of the true class labels with those of the classifier's
[12]How many positive predictions the model made compared the the actual number of positives from the data

6 Results and discussion

6.1 Interpretation of decision tree

The decision tree reveals that *gold_earned* and *level* seem to be the deciding factors of having a high *damage_total* by far. Low *gold_earned* (<=0.5) only allows a player to deal mid damage at most. High *gold_earned* guarantees that a player will at least deal mid damage, and increases the chance that a player will deal high damage by about 70%. High *gold_earned* coupled with high *level*, greatly increases the probability of dealing high damage.

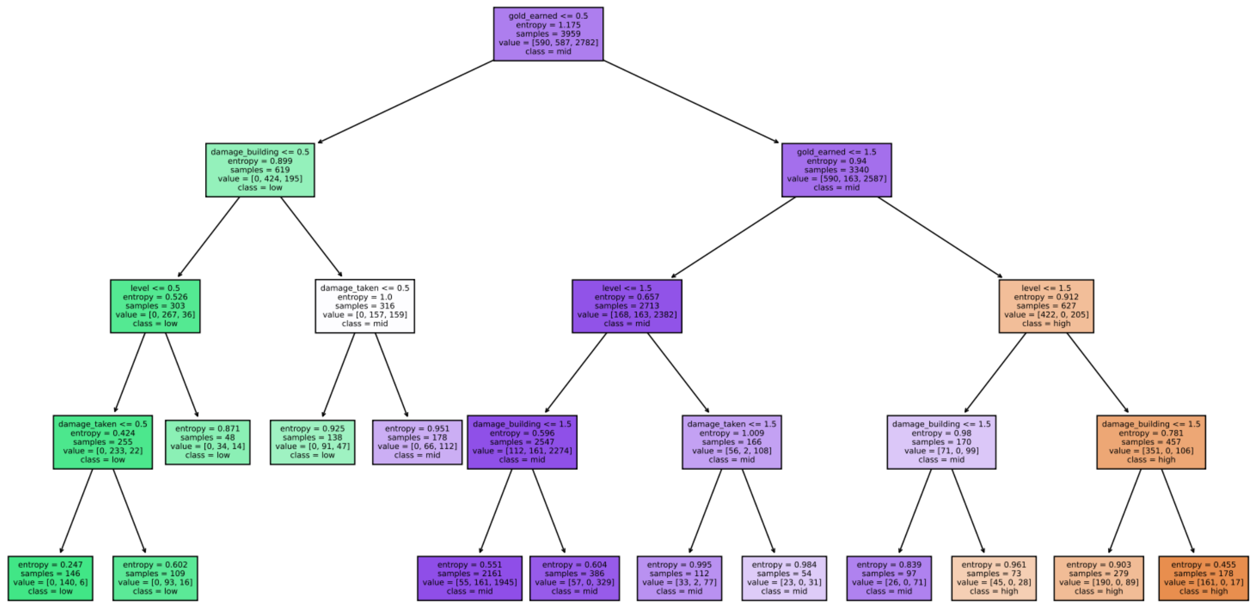


Figure 7: Decision Tree Model

damage_building is far less pivotal in determining a player's *damage_total*, but still supplement it nonetheless; players with higher *damage_building* are more likely to deal slightly higher damage. *damage_taken* is the least influential variable of the four, and does not show a clear relationship to *damage_total*.

6.2 Application to research

In descending order, the four most influential variables that affect the total damage at the end of the game are: *gold_earned*, *level*, *damage_building*, and *damage_taken*. As was described on *Section 1* and *Table 2*, these variables can be classified into broader game mechanics.

Objective	Combat	Farming
damage_building	damage_taken	gold
		level

Table 5: Game mechanics the four most influential variables fall into

Focusing on *farming* generates a significantly larger net total damage than the other playstyles. This is since when players focus more on *farming*, they can equip their champions with better items, allowing them to deal significantly more damage in the later parts of the game ("late-game phase").

The late-game phase is considered to be high stakes as a single action can change the results of the game. At this phase, it is expected that some of the champions (also known as the *carry*[13]) would have reached their maximum potential damage, as sufficient time has passed to allow them to purchase *premium* performance-boosting items, and to level up. As a result of having the ability to deal more damage, the champion will be able to destroy enemy buildings, and eventually capture the *Nexus*, in a much swifter manner.

Thus to maximise the chances of winning a team fight, and in turn the match at the late-game phase, teams should have a carry champion that can "*farm*" efficiently.

7 Limitations and improvements

7.1 Limitations of the results

One of the main limitations of this investigation was the assumptions made during the data processing pipeline. It was assumed that all numeric features followed a normal distribution, and were binned according to this assumption. This could have caused some of the data to be ordinalized into inappropriate categories. Future research should seek cooperation with a domain expert to determine what is considered as "*low*", "*mid*", and "*high*" values for each feature.

Additionally, to be able to evaluate our models better, other types of performance metrics could have been taken into consideration. This, in turn, will help researchers evaluate the efficacy of the model. In the future, to help compare the models created in this experiment, a Receiver Operating Characteristic curve could have been used to give more enriching and detailed information for consideration, as it plots data points under the True Positive and False Positive against different thresholds (Vidiyala, 2020).

[13] The term *carry* comes from the phrase "easily carrying the team to victory"

References

Galarnyk, M. (2018, June 4). *Explaining the 68-95-99.7 Rule for a Normal Distribution*. Medium; Towards Data Science. https://towardsdatascience.com/understanding-the-68-95-99-7-rule-for-a-normal-distribution-b7b7cbf760c2

Suter, A. (2022). *LoL challenger soloq data (jan, kr-na-euw)*. Www.kaggle.com. https://www.kaggle.com/datasets/andrewasuter/lol-challenger-soloq-data-jan-krnaeuw

Vidiyala, R. (2020, July 26). *Performance metrics for classification machine learning problems*. Medium. https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007