League of Legends Match Prediction

Alex Siegel – 001193405

Introduction

Since its release in 2009, League of Legends has become one of the most widely played and recognizable video games, averaging over four million players daily. Individuals from around the world compete to ascend the game's ranked ladder and establish themselves as top players. In pursuit of this goal, players continually develop new strategies to win matches, with the most successful ultimately competing at national and international levels.

Objective

This project aims to predict the outcome (win or loss) of a League of Legends match using a machine learning classification model. By analyzing in-game performance statistics, the model identifies which factors most strongly influence a player's chance of winning. Although a match consists of two opposing teams of five players each, only individual player performance data is used. Factors that depend on multiple teammates, such as team composition, are excluded, as a player cannot control their teammates' actions and would most likely want to know what they themselves can do to increase their chances of winning.

Dataset

The dataset, downloaded from Kaggle (https://www.kaggle.com/datasets/nathansmallcalder/lol-match-history-and-summoner-data-80k-matches?resource=download), comes directly from Riot Games, the developer of League of Legends. The data is separated into seven *.csv* files and connected to each other through primary and foreign keys. Of these seven files, TeamMatchTable.csv and ItemTbl.csv are not used.

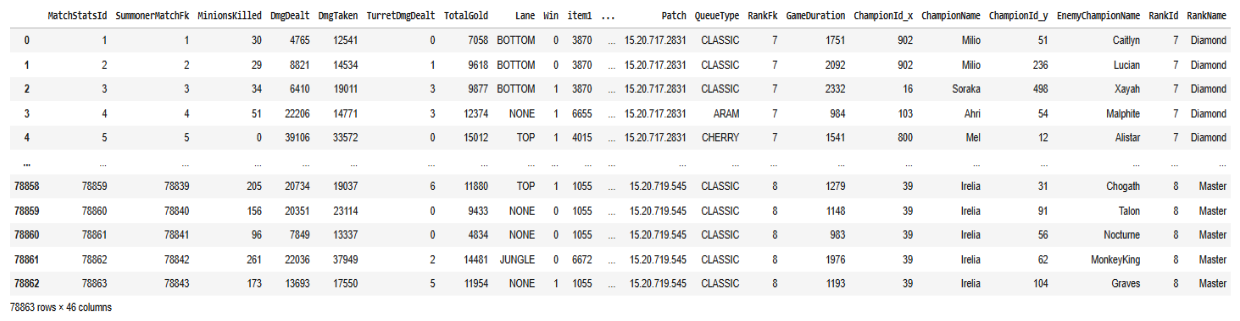
Hypothesis

The objective of a League of Legends match is to destroy the enemy team's nexus, which can only be reached after several layers of turrets are taken down. Therefore, I hypothesize that the most influential feature in the dataset is TurretDmgDealt, which represents the number of enemy turrets a player has demolished during a match.

Throughout a match, players accumulate gold to purchase items that can increase their champion's damage output. Higher damage output improves a player's ability to kill enemy champions, and each elimination awards additional gold that can be used to buy even more items. Therefore, I hypothesize a positive correlation between the amount of gold a player has (TotalGold), the damage they deal to enemy champions (DmgDealt), and the number of enemy champions they slay (kills).

Preliminary Analysis

The dataset contains 78,863 entries divided across seven different files, although only five are used in this project. A relational database model links each file through a primary key and several foreign keys. After merging the selected files, the dataset forms a single combined DataFrame with 46 columns, as shown in Figure 1.



	MatchStatsId	SummonerMatchFk	MinionsKilled	DmgDealt	DmgTaken	TurretDmgDealt	TotalGold	Lane	Win	item1	...	Patch	QueueType	RankFk	GameDuration	ChampionId_x	ChampionName	ChampionId_y	EnemyChampionName	RankId	RankName
0	1	1	30	4765	12541	0	7058	BOTTOM	0	3870	...	15.20.717.2831	CLASSIC	7	1751	902	Milio	51	Caitlyn	7	Diamond
1	2	2	29	8821	14534	1	9618	BOTTOM	0	3870	...	15.20.717.2831	CLASSIC	7	2092	902	Milio	236	Lucian	7	Diamond
2	3	3	34	6410	19011	3	9877	BOTTOM	1	3870	...	15.20.717.2831	CLASSIC	7	2332	16	Soraka	498	Xayah	7	Diamond
3	4	4	51	22206	14771	3	12374	NONE	1	6655	...	15.20.717.2831	ARAM	7	984	103	Ahri	54	Malphite	7	Diamond
4	5	5	0	39106	33572	0	15012	TOP	1	4015	...	15.20.717.2831	CHERRY	7	1541	800	Mel	12	Alistar	7	Diamond
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
78858	78859	78839	205	20734	19037	6	11880	TOP	1	1055	...	15.20.719.545	CLASSIC	8	1279	39	Irelia	31	Chogath	8	Master
78859	78860	78840	156	20351	23114	0	9433	NONE	0	1055	...	15.20.719.545	CLASSIC	8	1148	39	Irelia	91	Talon	8	Master
78860	78861	78841	96	7849	13337	0	4834	NONE	0	1055	...	15.20.719.545	CLASSIC	8	983	39	Irelia	56	Nocturne	8	Master
78861	78862	78842	261	22036	37949	2	14481	JUNGLE	0	6672	...	15.20.719.545	CLASSIC	8	1976	39	Irelia	62	MonkeyKing	8	Master
78862	78863	78843	173	13693	17550	5	11954	NONE	1	1055	...	15.20.719.545	CLASSIC	8	1193	39	Irelia	104	Graves	8	Master

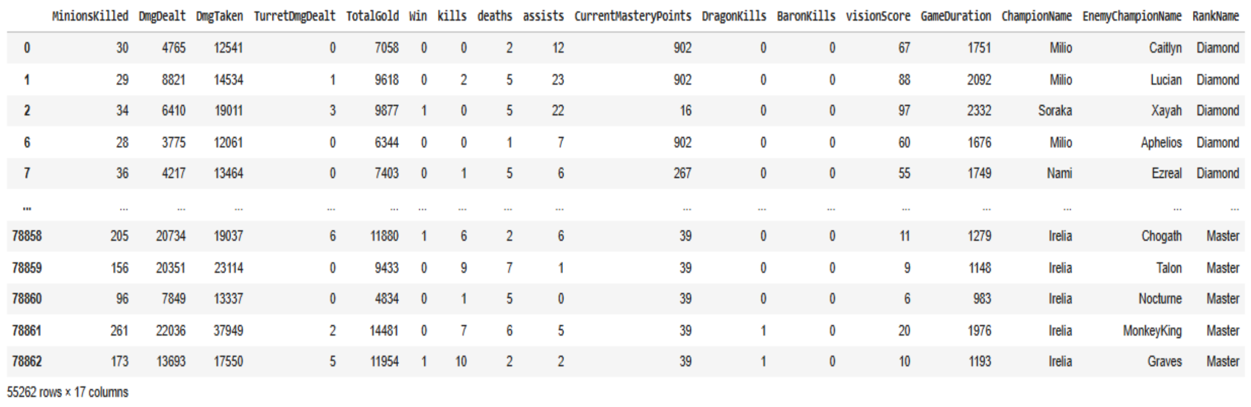
78863 rows × 46 columns

Figure 1: DataFrame before dropping features

Filtering Data

The combined DataFrame contains performance data from multiple game modes, as indicated by QueueType. This project focuses exclusively on predicting the outcome of the most common game mode: Summoner's Rift. Therefore, only entries with QueueType equal to "CLASSIC" are retained, while all other game modes are dropped.

Many features are removed prior to visualization, including all IDs and foreign keys; hyper-specific features such as Item1–Item6, PrimaryKeyStone, SummonerSpell; and Patch, since players have no control over the game version. The QueueType column is also dropped since all entries are "CLASSIC." The resulting DataFrame contains 55,262 rows and 17 columns, as shown in Figure 2.



	MinionsKilled	DmgDealt	DmgTaken	TurretDmgDealt	TotalGold	Win	kills	deaths	assists	CurrentMasteryPoints	DragonKills	BaronKills	visionScore	GameDuration	ChampionName	EnemyChampionName	RankName
0	30	4765	12541	0	7058	0	0	2	12	902	0	0	67	1751	Milio	Caitlyn	Diamond
1	29	8821	14534	1	9618	0	2	5	23	902	0	0	88	2092	Milio	Lucian	Diamond
2	34	6410	19011	3	9877	1	0	5	22	16	0	0	97	2332	Soraka	Xayah	Diamond
6	28	3775	12061	0	6344	0	0	1	7	902	0	0	60	1676	Milio	Aphelios	Diamond
7	36	4217	13464	0	7403	0	1	5	6	267	0	0	55	1749	Nami	Ezreal	Diamond
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
78858	205	20734	19037	6	11880	1	6	2	6	39	0	0	11	1279	Irelia	Chogath	Master
78859	156	20351	23114	0	9433	0	9	7	1	39	0	0	9	1148	Irelia	Talon	Master
78860	96	7849	13337	0	4834	0	1	5	0	39	0	0	6	983	Irelia	Nocturne	Master
78861	261	22036	37949	2	14481	0	7	6	5	39	1	0	20	1976	Irelia	MonkeyKing	Master
78862	173	13693	17550	5	11954	1	10	2	2	39	1	0	10	1193	Irelia	Graves	Master

55262 rows × 17 columns

Figure 2: DataFrame after dropping features.

Train-Test Split

Before visualization, the data is split into training and testing data, using “Win” as the target feature. The result is 44,209 (80%) instances for training and 11,053 (20%) for testing. The test data will not be touched until it is time to evaluate the models. All visualizations, preprocessing, and model training will be done only with the training data.

Visualizations

To better understand the dataset, a histogram is generated for each numeric column, and a bar chart is created for each categorical column.

Histograms

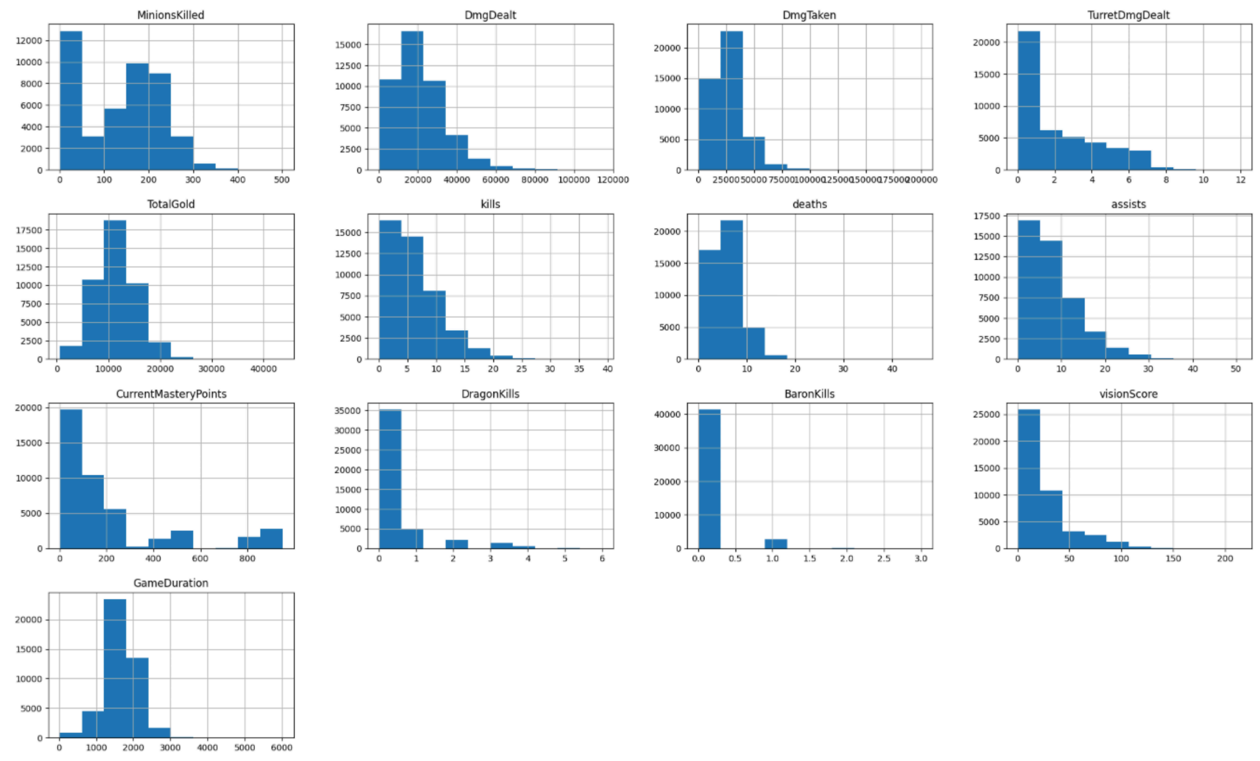


Figure 3: Histograms of numeric columns

As shown in Figure 3, many of the histograms are right-skewed, with most values concentrated near zero. These features need to be transformed to represent a more uniform distribution. However, both BaronKills and DragonKills are excluded from this process due to their low-integer nature and the underlying game mechanics that constrain their possible values.

Baron and Dragons are optional map objectives in a match that provide powerful buffs to the team. Slaying Baron is strongly associated with increased win probability, but there are too few instances of a player defeating Baron multiple times to assess any additional impact. Furthermore, only approximately half of matches last long enough for more than one Baron to spawn, as the earliest possible spawn time for a second Baron is 26 minutes (1560 seconds). Figure 3 reflects this limitation as most values for BaronKills are either 0 or 1, with very few exceeding this. Therefore, this feature will be binarized, with a value of 1 indicating that a player has slain Baron and 0 indicating that a player has not.

Similarly, defeating dragons provides a stacking buff, and a team gains a significantly stronger "Dragon Soul" effect after defeating four dragons. A more powerful dragon spawns six minutes after this effect is claimed, but instances of a team securing more than four dragons are rare. Additionally, a match outcome is often highly predictable once this effect is claimed, suggesting that subsequent dragons have limited additional impact. For these reasons, DragonKills will be capped at four.

Bar Charts

Both ChampionName and EnemyChampionName contain a large number of categories, corresponding to the 172 champions in the game. In contrast, RankName contains only 11 categories, with a majority of the samples belonging to the Master tier. As a result, these columns will require different transformations.

Correlations

My second hypothesis predicts a positive correlation between the TotalGold, DmgDealt, and kills features. To investigate this, a correlation matrix was generated using the DataFrame.corr function in Pandas. This analysis was performed prior to any data transformations.

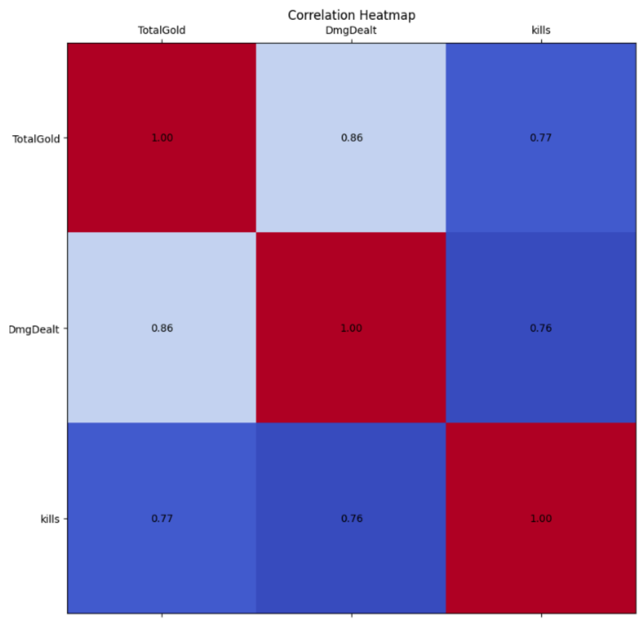


Figure 4: Correlation heat map of TotalGold, DmgDealt, and kills

As shown in Figure 4, all three features display correlation coefficients greater than 0.75, indicating a strong positive relationship. Players who accumulate more total gold tend to deal increased damage to enemy champions, likely securing more kills as a result. Similarly, players who achieve more kills earn additional gold, reinforcing this relationship. These results provide solid support for my second hypothesis, but further examination will be conducted during subsequent model analysis.

Data Preprocessing

The features were examined for missing values using the DataFrame.info function. The resulting output indicates 44,209 non-null values for all features, so no values are missing and imputation is not required. Additionally, the target data is relatively balanced, with 50.9% of values corresponding to a win (1) and 49.1% to a loss (0). With these outcomes, the training data is prepared for the preprocessing pipeline. Each column type will require a different set of transformations as shown below.

Symmetric Columns

Features that roughly exhibit a uniform distribution do not require a logarithmic transformation. For these columns, the preprocessing pipeline only includes the StandardScaler function, which scales the data to have a mean of 0 and a standard deviation of 1.

Skewed Columns

Right-skewed features are transformed using NumPy's log1p function. This function is preferred over the standard logarithm because it safely handles zero values, as log(0) is undefined. After this transformation, StandardScaler is called to standardize these features.

Baron Kills

As mentioned earlier, the BaronKills will be binarized to better represent the data. This process uses a custom binarize function that checks if a value is greater than zero and returns the Boolean as an integer. FunctionTransformer is called to apply the binarize function to the feature. Lastly, the feature is also standardized using StandardScaler.

DragonKills

DragonKills also requires a custom cap_value function, using np.clip to limit the feature to 4. Like BaronKills, FunctionTransformer applies the function and is followed by StandardScaler.

ChampionName and EnemyChampionName

Using a One-Hot Encoder for ChampionName and EnemyChampionName would produce 171 new columns, greatly reducing all models' efficiency. Instead, a Target Encoder is applied, replacing each entry with its mean target value. This process converts the categorical feature into a numerical one without increasing the number of features.

Rank Name

A standard One-Hot Encoder is employed for RankName, creating a new column for each tier. This method is beneficial as there is a limited number of ranks and any unknown categories can easily be handled.

Preprocessing Pipeline

All transformation processes are implemented in a single ColumnTransformer. Each step takes one or multiple features and applies the correct transformations to it. While the numeric columns use pipelines to apply multiple functions, the categorical steps each call their transformation function directly in the ColumnTransformer.

Model Selection, Training, & Hyperparameter Search

The three classification models chosen for this project are Logistic Regression, Random Forest Classifier, and MLP Classifier. Due to the complex nature of the features, a linear model, such as the Logistic Regression model, may not be the most optimal solution. Therefore, Random Forest or MLP Classifier may be the better model option at the cost of computational resources.

Prior to fine-tuning, each model's mean cross-validation score is checked to establish a performance baseline. The mean score for each model is shown below using a cv of five.

Model	**Base Accuracy**
Logistic Regression	0.8586
MLP Classifier	0.8774
Random Forest Classifier	0.9393

Figure 5: Cross_val scores before fine-tuning

Fine-Tuning

The baseline performance for all three models is already quite accurate, with Random Forest Classifier having the highest mean accuracy score of .93. Despite this, all models will be fine-tuned to ensure proper evaluation. Using a dictionary of hyperparameter values, GridSearchCV will test each combination and return the cross-validation score of the best hyperparameters. The hyperparameters and definitions of each model are exhibited below.

The Logistic Regression model uses the "saga" solver algorithm with a maximum iteration of 5000. The hyperparameters tested are the penalty/regularization methods (L1, L2, elastic net), C values for each penalty, and the L1 ratio, which controls the mix between L1 and L2 penalties when using elastic net. C is the inverse of regularization, so a higher C value means penalties are punished less, which may result in overfitting.

MLP Classifier uses a number of hidden layers, each with a specified amount of neurons. Since the number of features is 26, the model is tested using two or three hidden layers, each containing either 13 (n_features/2), 26 (n_features), or 52 (n_features*2) neurons. The alpha value is also tested, which controls the strength of the L2 regularization penalty.

The Random Forest model is built by combining the outputs of multiple decision trees, as indicated by n_estimators. Each tree can have a max depth determined by the max_depth hyperparameter. In order for a node to split, it must have at least the number of samples specified in min_samples_split. Lastly, the min_samples_leaf hyperparameter determines the minimum number of samples needed to be a leaf node.

After fine-tuning all three models, the results can be compared with the previous cross validation scores to see if there is a significant difference.

Model	**Fine-Tuned Accuracy**	**Improvement (%)**
Logistic Regression	0.8590	+0.05%
MLP Classifier	0.8916	+1.62%
Random Forest Classifier	0.9397	+ 0.04%

Figure 6: GridSearchCV best score after fine-tuning

For two of the three models, fine-tuning had minimal impact on the scores, with the only noticeable increase being the MLP Classifier. While it is fairly common for neural network models to produce stronger results after fine-tuning, this increase could also indicate a complex, non-linear relationship between the features. However, the computational cost of fine-tuning the models outweighs the benefit of slightly increased accuracy. Regardless, all three models are trained and ready for evaluation.

Evaluation

Using the best hyperparameters for each model, the match outcome is predicted using the test data. Afterwards, the Accuracy, Precision, Recall, and F1 Scores are all calculated for the three models and each confusion matrix is drawn. The scores for each model can be found below in Figure 7.

Model	Accuracy	Precision	Recall	F1
Logistic Regression	0.8554	0.8601	0.8562	0.8582
MLP Classifier	0.9046	0.9266	0.8833	0.9044
Random Forest Classifier	0.9597	0.9646	0.9563	0.9604

Figure 7: Evaluation metrics for test data predictions

All three models achieved a strong performance, each predicting match outcome with over 85% accuracy and having a good balance between precision and recall. In this context, precision is slightly more important than recall, since incorrectly predicting that a player will win may cause them to behave more aggressively or take unnecessary risks during a match.

The training and test accuracy were both high and closely matched, indicating that all three models generalize well to unseen data. Logistic Regression performing the worst of the three models suggests that the underlying relationships between features and the target may be complex and non-linear. Upon inspecting the confusion matrices, all models produced slightly more false negatives than false positives, indicating more complex boundary decisions that favored precision over recall.

Result

Of the three models evaluated, Random Forest Classifier is the recommended choice due to its high accuracy and balanced metrics. Logistic Regression is also a reasonable alternative for scenarios requiring faster speed at the cost of slightly lower accuracy. Although the MLP Classifier performed well, the model required the most time and computational power, making it less suitable for this problem.

Analysis

The first hypothesis predicted TurretDmgDealt as the most influential feature in determining match outcomes. To evaluate this, both the Logistic Regression coefficients and the Random Forest feature importances are retrieved. Despite having slightly lower predictive accuracy, Logistic Regression provides interpretable coefficients, with a positive coefficient indicating that increasing the feature value is associated with a higher likelihood of winning, while a negative coefficient indicates the opposite. In contrast, the Random Forest model measures the relative importance of each feature, but does not indicate whether a feature increases or decreases the likelihood of winning – only how influential the feature is within the model's decision process.

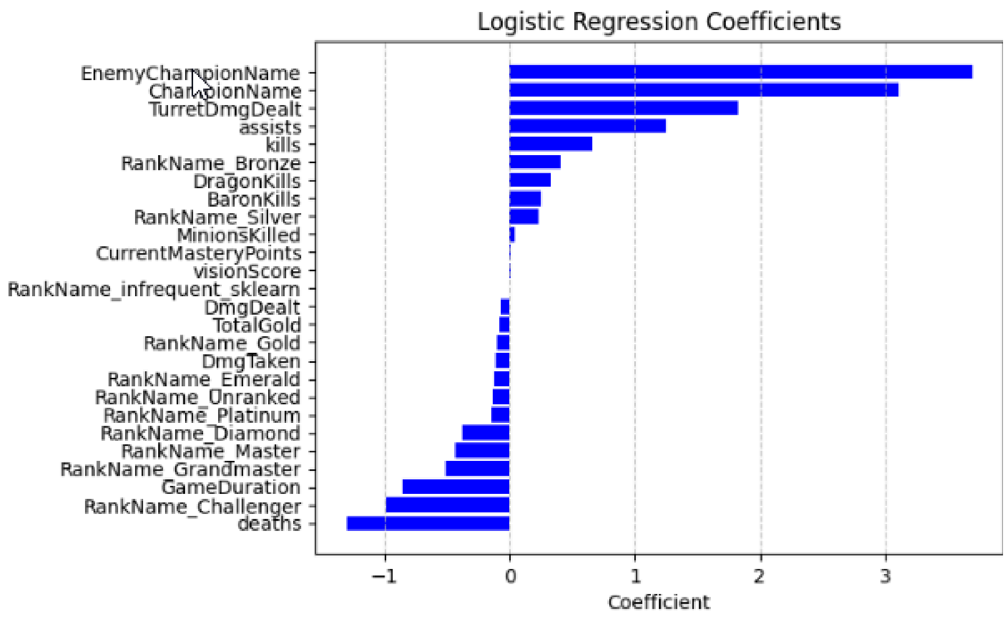


Figure 8: Logistic regression coefficients

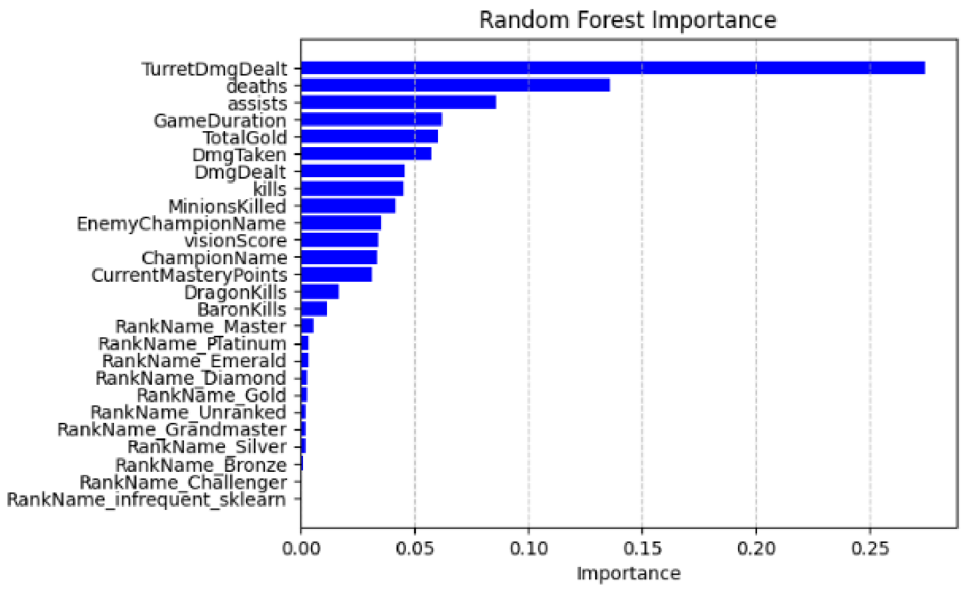


Figure 9: Feature Importance

The Random Forest model identifies TurretDmgDealt as the most important feature in predicting match outcomes, while its Logistic Regression coefficient indicates that increasing this value presents the highest chance of winning. This result provides evidence in support of Hypothesis 1.

Although TotalGold, DmgDealt, and kills are strongly correlated, the Linear Regression model (Figure 8) suggests that kills has a positive effect on winning matches, while TotalGold and DmgDealt appear to have very slight negative effects. In contrast, the Random Forest model identifies all three features as among the most important factors in determining match outcome. While Random Forest does not indicate the direction of influence, it is unlikely that higher gold earnings or greater damage dealt would reduce a player's chance of winning. This discrepancy suggests that the Logistic Regression model may have been assigned suboptimal weights, which could contribute to its lower accuracy. The correlation heat map presented earlier together with the Random Forest feature importances provide strong evidence in support of Hypothesis 2.

Conclusion

Players who focus on taking turrets, assisting their teammates, and minimizing deaths are more likely to win their matches. Interestingly, the Random Forest analysis shows that conquering objectives, like Dragons or Baron, is less predictive of victory than expected. Furthermore, a player's rank has minimal effect on match outcome, encouraging players to continue improving without being discouraged by their current rank.

Since a League of Legends team is typically composed of five players with very little pre-existing chemistry, individual strategies that emphasize consistent objectives can help players win more reliably without depending heavily on teammates. By leveraging data from Riot Games' API, these models could be refined further and provide even more personalized guidance.

A significant amount of preprocessing was necessary due to the nature of the features. Columns such as Items 1-6 were dropped because their impact depends heavily on the champions involved, requiring a much larger dataset for meaningful analysis. Nevertheless, this project offers strategies for players to succeed across a wide range of champions and keystone choices.

With these insights, every player has the opportunity to improve their performance and aim for the top, making the game more engaging and rewarding.