Joyner, Jackson

[Email address]

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

lolesports MVP Prediction Analysis

Table of Contents

**Introduction**

In 2009, Riot Games founded by Brandon Beck and Marc Merrill released a game that would soon become one of the largest video games in the world, League of Legends (commonly referred to as League). While the game was growing to astronomical levels, the interest in a professional level of league was growing at similar rates. This was due to having one of the best in game ranking systems in modern video games. Players joined up and started teams to compete in various tournaments across the world, competing against the best of the best players. As this grassroots tournament scene grew, so did Riot Games’ efforts to support the players in this pro scene. In 2011, the first world championship took place at a gaming convention Dreamhack in Sweden, thus LoLEsports was born.

Throughout the years of LoLesports, there have been 11 different leagues created, with four leagues eventually reaching a franchised league. Each league consists of 8-10 teams that compete throughout the year to earn a spot at the world championship. Throughout the regular season, each league grants a Most Valuable Player (MVP) of their league. In 2021, the North American league, the LCS, granted their MVP to a player named Spica, the Jungler for Team Solo Mid (TSM). A player by the name of Closer, the Jungler for 100 Thieves (100T) was a close second. Personally, I believe that Closer was more deserving of the title. This idea led to finding a statistical method to determine the MVP of a given league for the regular season.

There are some inherent issues when working with League data, as much of the dataset is colinear. As the game progresses, the player passively gains important resources that can be used to gain advantages in damage and damage reduction. This leads to collinearity of Damage, Gold, XP, Total CS and more. While managing the collinearity, is there a statistical way to determine the MVP of a league?

**What is League of Legends?**

League of Legends is an online 5 vs. 5 multiplayer game that was released in 2009 by Riot Games. The objective of the game is to destroy the enemy nexus by dealing damage to it. There are a series of objectives that need to be taken for a team to be able to destroy the enemy base. League of Legends has built a huge following by being a free game, that offers cosmetic transactions to earn revenue. As the game was growing, there was a huge push for this game to be played at a professional level, and in 2011, the first world championship was held, and LoLEsports was born.

Just like traditional sports, League of Legends players play separate positions. The positions are referred to by the lane that the player is in. 

Figure 1: Map of League of Legends

The top lane, the left side of the map, is filled with one player from each team. These players typically play tank or brawlers who have good damage and can take a hit. The mid lane (diagonal lane in the map) player typically plays mages or assassins. These characters have low health and high damage stats. There are two players per team in the bottom lane. One plays a marksman, which is a high-ranged damage dealer and the other plays a support character who is either a healer, or a tank. Then the player who doesn’t have a designated lane to play in is the jungler. In between each lane is the jungle, which is the jungler’s territory. They go from lane to lane helping other teammates. These players typically play high mobility characters that excel in helping others.

As the game progresses, the lanes get waves of minions that grant gold and experience for characters that kill them. The jungle has mobs that grant the same resources in a more condensed fashion. While fighting minions and mobs, the players are also battling their lane opponent, often killing each other players use their gold and experience to buy items and level up. Items grant bonus attributes to their characters, that powers them up. Leveling up grants increased abilities effectiveness. This allows for players to try different strategies, go for different objectives, all effecting the outcome of the game.

**Data Description**

In 2014, there was a huge push for LoLEsports data to be released to the poublic, and Riot Games agreed to do so. They released an API that allowed people to request data for researching and analysis. This was primarily used by teams and organizations trying to push into the top leagues. Around this time, Tim Sevenhuysen was working with a team in Europe named Fnatic as a data analyst. He used this API to create a website for the public to use called Oracleselixer.com. This website would become the go-to resource for all LoLEsports statistics for fans and up-and-coming analysts alike. In January, I pulled match data from the last 8 years of LoLEsports games from oracleselixers.com to begin this analysis. This match data contains 12 lines of statistics per game; 5 lines for the blue side team players, 5 lines for the red side team players, one line for the blue side team, and one line for the red side team.

This data ended up being 400,000+ rows of data from the years 2014 to 2022. I wanted to only use the data from the top four leagues in this analysis, so the 400,000+ dropped down to (Need to pull the 2022 data still so this will be updated at a later date). From this, I needed to drop any rows with null valuables. This is ok with this data, as there are many more instances of that player playing games. This could be an issue with low game count players, but not enough to break the analysis.

Data Dictionary

Below is a data dictionary for the LoLEsports statistics used in this analysis

|  |  |
| --- | --- |
| GameID | Unique ID for each match |
| Datacompleteness | Identifier if there are null values |
| League | Identifier of which league the match took place in |
| Year | Year in which the game was played |
| Split | Which split of the season the match was played |
| Side | Which side of the map the player started on |
| Player | Players in game name |
| Team | Team name |
| Pos | Position |
| GP | Games Played |
| W% | Win Percentage |
| CTR% | Counter Pick rate: percentage of games in which this player/champion was picked after their lane opponent |
| K | Kills |
| D | Deaths |
| A | Assists |
| KDA | Kill Death Assist Ratio |
| KP | Kill Participation: percentage of teams kills in which player earned a Kill or Assist |
| KS% | Kill Share: players percentage of their teams total kills |
| DTH% | Average share of teams deaths |
| FB% | First Blood rate: for players/champions, percent of games earning a First Blood participation |
| Result | Result of the game: 1 = win 0 = loss |
| GD10 | Average gold difference at 10 minutes |
| XPD10 | Average experience difference at 10 minutes |
| CSD10 | Average creep score difference at 10 minutes |
| CSPM | Average monster + minions killed per minute |
| CS%P15 | Average share of teams total CS post 15 minutes |
| DPM | Average damage to champions per minute |
| DMG% | Damage Share: Average share of teams total damage to champions |
| D%P15 | Share of teams damage to champions post 15-minutes |
| EGPM | Earned gold per minute - excludes starting gold and inherent gold generation |
| GOLD% | Gold Share: Average share of teams total gold earned - excludes starting gold and inherent gold generation |
| STL | Neutral Objectives Stolen |
| WPM | Wards placed per minute |
| WCPM | Wards cleared per minute |
| Double Kills | Amount of double kills in the match |
| Triple Kills | Amount of triple kills in the match |
| Quadra Kills | Amount of quadra kills in the match |
| Penta Kills | Amount of penta kills in the match |
| Total CS | Total creep score for the match |
| GD15 | Average gold difference at 15 minutes |
| XPD15 | Average experience difference at 15 minutes |
| CSD15 | Average creep score difference at 15 minutes |

**Descriptive Stats**

When looking at League data, there are hard floors to most statistics, as a player can’t have negative stats like gold, xp, damage, kills, deaths, assists and more. Items like these are at a baseline of 0 at the start of the game and only go up as the game progresses. With that in mind, there are some interesting stats to look at when it comes to League of Legends.

Typically, when reporting statistics in League of Legends, it is common practice to compare the results of players against other players of the same position. All visualizations will be done in this fashion, as each role has a different purpose, and each position may have varying degrees of utilization of these statistics.

KDA is a ratio of kills and assists to deaths per game. This is a common statistic used to discuss the usefulness of a player in a game, calculated by (kills + assists) / deaths. Sometimes, when a player doesn’t die in game, the ratio would report as an infinite number, as you are trying to divide by zero, in these specific cases, you would just report kills + assists as the KDA ratio. The average KDA is 4.98, with a minimum KDA being 0, a median of 3.33 and a max of 33.

Chart, box and whisker chart

Description automatically generated

Figure 2: KDA Box Plot

As Figure 2 shows, positions like jungle, mid and bot have a higher average KDA than top and support players. Bot lane has the largest IQR, as it typically is a more kill and assist position, where supports and top lane players are typically more supportive and die more frequently. With that in mind, it is important to note that supports have the highest KDA at 33. This performance came from the European support, Kold who played for Splyce in 2017. He sported a score line of 5/1/28 (Kills/Deaths/Assists) in a playoff game against Vitality.

A statistic that showcases the extreme differences in positions is CSPM. This statistic is the creep score per minute. A creep grants the person who last hit it gold, and anyone around the kill, experience. In a typical game of League, bot, mid and top dominate this statistic as they are the players who are getting cs often. Junglers and supports are going to show much lower numbers in this statistic. Overall, the average CSPM for all players is 6.25, the median is 7.12, and the max is 15.50.

Chart, box and whisker chart

Description automatically generated

Figure 3: CSPM by Position

Figure 3 really showcases the stark differences between positions when it comes to CSPM. The support position has the overall lowest CSPM as they are the players who don’t kill creeps. There was a point in time, where the supports were the players who were getting cs, and that is shown by the data above the top whisker of the box plot. With junglers moving around the map so much, they typically report lower CSPM values, and that shows in Figure 3.

Typically, higher amounts of gold leads to higher damage. This is seen by looking at the relationship between CSPM and DPM. Figure 4 showcases DPM per position. With an overall mean DPM of 372.80, standard deviation of 216.61, and a max overall DPM of 1773.93. The same positions are dominating in DPM as CSPM. This is caused by the relationship between gold and damage. As gold increases so does potential damage. If a character is able to purchase an item that doubles their damage, having the gold to do so is important. Thus, you see the characters with the highest average DPM having the highest CSPM, as cs grants the majority of a player’s gold.

Chart, box and whisker chart

Description automatically generated

Figure 4: DPM by Position

To give the positions that don’t focus primarily on damage, let’s look at WPM. This is a statistic that looks at the amount of wards a player places per minute in a game. Wards are items that provide vision in a small area. This has very little combat incentives and is primarily used to gather information around the map. WPM sports an overall mean of .70, a standard deviation of .47, and a maximum 3.75 WPM.

Chart, box and whisker chart

Description automatically generated

Figure 5: WPM by Position

As you can see in Figure 5, the supports and junglers are the players who are doing the heavy lifting when it comes to WPM. This is caused by the role of the support is to do things that help the team in combat and non-combat scenarios. They build items that help them place wards, so it makes sense for them to be in the lead in this statistic. Junglers place high in this statistic as they aren’t quite as stationary as other players in the game. For their play, they need control of the map, and to get control of a map, you need vision of the map.

One of the best statistics to determine performance in LoLEsports is Earned GPM. Gold is one of the most important factors of player performance in a game of League. Gold allows for players to become more powerful in the game through the purchase of items, which give statistical advantages to the players. Items can range in price from 400 gold to 3600 for a full item. Having a higher Earned GPM gives you earlier access to these important items. With an overall mean of 223.21, standard deviation of 78.92, and a max of 556.56, Earned GPM showcases the importance of gold in League of Legends.

Chart, box and whisker chart

Description automatically generated

Figure 6: Earned GPM by Position

**Analysis**

For the analysis of LoLEsports data, I decided to run a logistic regression, as I was going to be testing against a binary variable against continuous variables. This section will contain the initial logistic regression, the principal component analysis, and the final logistic regression. The principal component analysis was performed to reduce the collinearity seen in the data that would inhibit a model’s effectiveness.

Part of the research into how MVP models are created was looking into impact scores, and how different variables affect the score. One way to get an impact score would be to use a fantasy score, like you find in fantasy football. Luckily, there is a fantasy game mode for League called Fantasy LCS. While the dataset didn’t have this statistic already in it, it’s an easy calculation to make. In a Medium article [1], Jonathan Hockey explains that the scoring system for Fantasy LCS is this: (3\*Kills) + (1.5 \* Assists) + (.02 \* Total CS) – Deaths. Starting with this baseline impact score, I ran a multitude of different logistic regressions on the data set to determine if there was a better impact score achievable.

The best Impact Score model I could build without care of multi-collinearity uses the following variables. When testing for result, adding kills, assists, deaths, total cs, damage share, DPM, CSPM, and WPM, you see an R-Squared of .592, which is a decent R-squared. While this could be considered a good model, there is an issue with the model, there is multi-collinearity displayed in the data.

To combat this issue, I ran some of the data through a Principal Component Analysis to help reduce the multicollinearity in the data. There were three different principal components produced, to reduce the multicollinearity.

The first principal component found was using the variables Gold, XP, and Damage at 15. I chose these variables to reduce into one factor as these three variables are good indicators of early game performance for a player.

A picture containing window, building, shoji

Description automatically generated

Figure 7: Pairs Plot of PCA15 variables

Figure 7 showcases the three variables and how they interact with each other. It is clear that there is a positive correlation between the three variables, which leads perfectly to a PCA reduction on these three variables. This reduction showcases an explained variance ratio of .8628, or 86% of the variance in these three variables can be explained by this reduced factor.

The variables Earned Gold Share and Damage Share showcase how the players performed compared to their teammates in game. These are fairly important metrics to determine how efficient a player was with their in-game resources. Figure 8 showcases a positive correlation of these variables. After standardizing these variables, we reduced these two factors into one, getting an explained variance ratio of .8999. This factor helped reduce collinearity of the data in our final model.

A collage of skyscrapers

Description automatically generated with medium confidence

Figure 8: Pair Plot for PCAShare Variables

A picture containing window, shoji, building

Description automatically generated

Figure 9: Pair plot of PCAPM factor variables

The final factor that I produced used the variables DPM, CSPM and Earned GPM. These are variables that standardize damage and gold for the duration of the game. Generally, a higher value in CSPM leads to a higher value of Earned GPM and DPM. Figure 9 showcases this relationship well. This leads to collinearity, so reducing these variables to one is needed for the model. Reducing these variables to one produces an explained variance ratio of .9186. This factor is our strongest created factor and should prove beneficial to our model.

OLS Regression Results

==============================================================================

Dep. Variable: result R-squared: 0.623

Model: OLS Adj. R-squared: 0.623

Method: Least Squares F-statistic: 2.546e+04

Date: Fri, 08 Apr 2022 Prob (F-statistic): 0.00

Time: 11:41:48 Log-Likelihood: -22092.

No. Observations: 92600 AIC: 4.420e+04

Df Residuals: 92593 BIC: 4.426e+04

Df Model: 6

Covariance Type: nonrobust

==============================================================================

coef std err t P>|t| [0.025 0.975]

------------------------------------------------------------------------------

const 0.3208 0.003 124.265 0.000 0.316 0.326

x1 0.0462 0.001 85.803 0.000 0.045 0.047

x2 -0.0884 0.001 -142.910 0.000 -0.090 -0.087

x3 0.0490 0.000 175.679 0.000 0.048 0.050

x4 -0.0586 0.001 -41.961 0.000 -0.061 -0.056

x5 -0.2168 0.002 -123.806 0.000 -0.220 -0.213

x6 0.2804 0.002 122.468 0.000 0.276 0.285

==============================================================================

Omnibus: 86.387 Durbin-Watson: 1.099

Prob(Omnibus): 0.000 Jarque-Bera (JB): 86.616

Skew: -0.075 Prob(JB): 1.55e-19

Kurtosis: 2.999 Cond. No. 20.9

=============================================================================

These reduced factors are incredibly beneficial for the final model, paired with removing the variable total\_cs, all collinearity issues were non-existent in the model. The final model, testing for game result by Kills, Deaths, Assists, and the three reduced factors (PCA15, PCAShare, PCApm). This model received an R-squared value of .623. Each variable is statistically significant with p-values all lower than .000. Since this model is a logistic regression, the coefficients need to be exponentiated, which you can see in Table X.

Table X: Odds Multipliers of Model Coefficients

const 1.378232

Kills 1.047269

Deaths 0.915350

Assists 1.050260

PCA15 0.943106

PCAShare 0.805085

PCApm 1.323606

These new odds multipliers for the model help explain the results of the model. As the amount of kills a player earns goes up, the odds of winning go up by 4.7%. This would also indicate that as a player dies, their odds of winning go down by 8.5%. With the reduced variables PCA15, PCAShare and PCApm, the odds multipliers are based on those reduced variables, not the actual variables that made them. As PCA15 increases by one, the odds of winning go down by 5.7%. It is the combination of Damage at 15, Gold at 15, and XP at 15 increasing that makes the odds of winning drop by 5.7%

For PCAShare, as the combination of damage share and gold share increase, the likely hood of winning reduces by 19.5%. This could be an indication of team performance is more important than individual performance when it comes to a win. When a player is having a stand out game, it could indicate that others on the team are having a lack luster performance, thus hurting the team more than the standout performance is helping the team. As for PCApm, as the combination of DPM, EGPM, and CSPM increase, the likelihood of a win increases by 32.4%.

PCApm is the first reduced factor that produces a positive effect on the model. As PCApm increases, our odds of wining increase by 32.3%. As DPM, CSPM and EGPM increase, the PCApm would increase, meaning that a player with a higher DPM, CSPM, and EGPM is more likely to win than a player with lower values in those metrics. This indicates that players who are more efficient with their in-game time are more likely to win.

When looking at the 2022 data, we can test this model out and get some idea of how credible this model is.

Things I still need to do:

Collect the 2022 data and fit it to the model

Expand explanations on the data (could use some guidance on this)

Add the Appendix, abstract, conclusion, references