Group 48
Benjamin Lathrop, Noah Bond, Jiacheng Shang, Zhenyu Wu, Garrett Norden
https://github.com/garrettnorden20/machine-learning-project

## Background

League of Legends is a free to play PC game in which two teams of five players work together to defeat their opponents and destroy the enemy team's base. There are a wide variety of characters to choose from, as well as a variety of objectives in the game that can provide players with money. Because of the wide array of variables in League of Legends, our group decided it would be the perfect place for a machine learning project.

## Problem Definition

Our goal is to forecast the winner of a match given all the data in the match (such as the characters on each team, damage done per minute, etc). There are many ways this model could be used. Teams wanting to experiment with new team compositions without wanting to hurt their rankings could see how changes in heroes could affect their predicted win rate. Players could also see the most important elements of a match - i.e., the features that drive the prediction the most. Many of the viewers of competitive matches could also use these predictions to make more informed betting decisions.

## Dataset

The dataset was obtained from Kaggle, and contains data for 184,070 League of Legends ranked solo games. This is direct data from the League of Legends servers, and as such will be generally accurate across all skill and gameplay levels.The data is separated in different csv's for match data, player data, and champion data, so we imported them and appended them into the same dataframe.

## Data Cleaning

To prevent the data from being skewed by matches that ended in disconnects, forfeits, or general "game-throwing," we dropped all games lasting less than 15 minutes. In addition, some matches had duplicate roles, so we dropped any matches that had duplicates. We also divided most features by the duration of a match, in order to display them as rates per minute to make them more compatible with each other.
In total, before cleaning there were 184,070 matches, and after cleaning that was reduced to 148,638. That's around 19% total reduction. This means our dataset was still largely useful and would give us accurate information

## Feature Selection

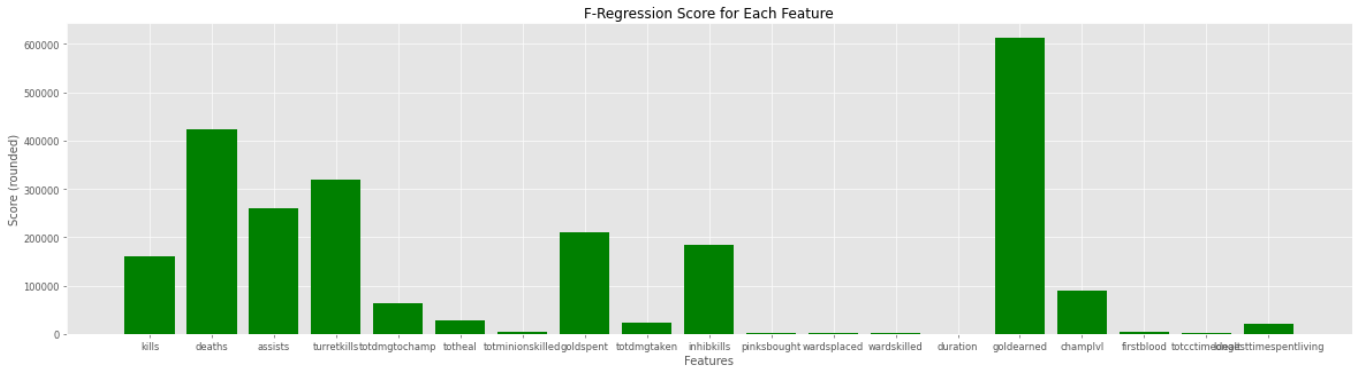
After those drops, the features needed to be addressed. There were around 50 columns in the "stats" csv to begin with, pertaining to stats concerned with the gameplay itself, such as "kills", "deaths", "gold earned", and "physical damage dealt." However, this is a very high amount, so before doing the actual feature reduction, we manually dropped many of the clearly non-important columns.

After cleaning these columns, we still had around 20 left, which still is a relatively high amount, so we looked towards methods for feature reduction.

We mainly tested two different methods for reducing our features - SelectKBest, and PCA.

SelectKBest picks the top features based on a score function. We experimented with several different scoring functions, mainly f_regression. F regression was chosen as a method to directly compare whether our model is actually improving in its fit by doing a comparison against the F value, where all the regression coefficients are equal to zero. This method will help to tell us when our changes in regression coefficient improves the model.

Contrasting with KBest feature selection, we also tried PCA dimensionality reduction on the dataset. The main reasoning for this was the overabundance of features in our dataset. Through PCA dimensionality reduction, we hope to isolate the most important features in order to create a better predictive model overall.

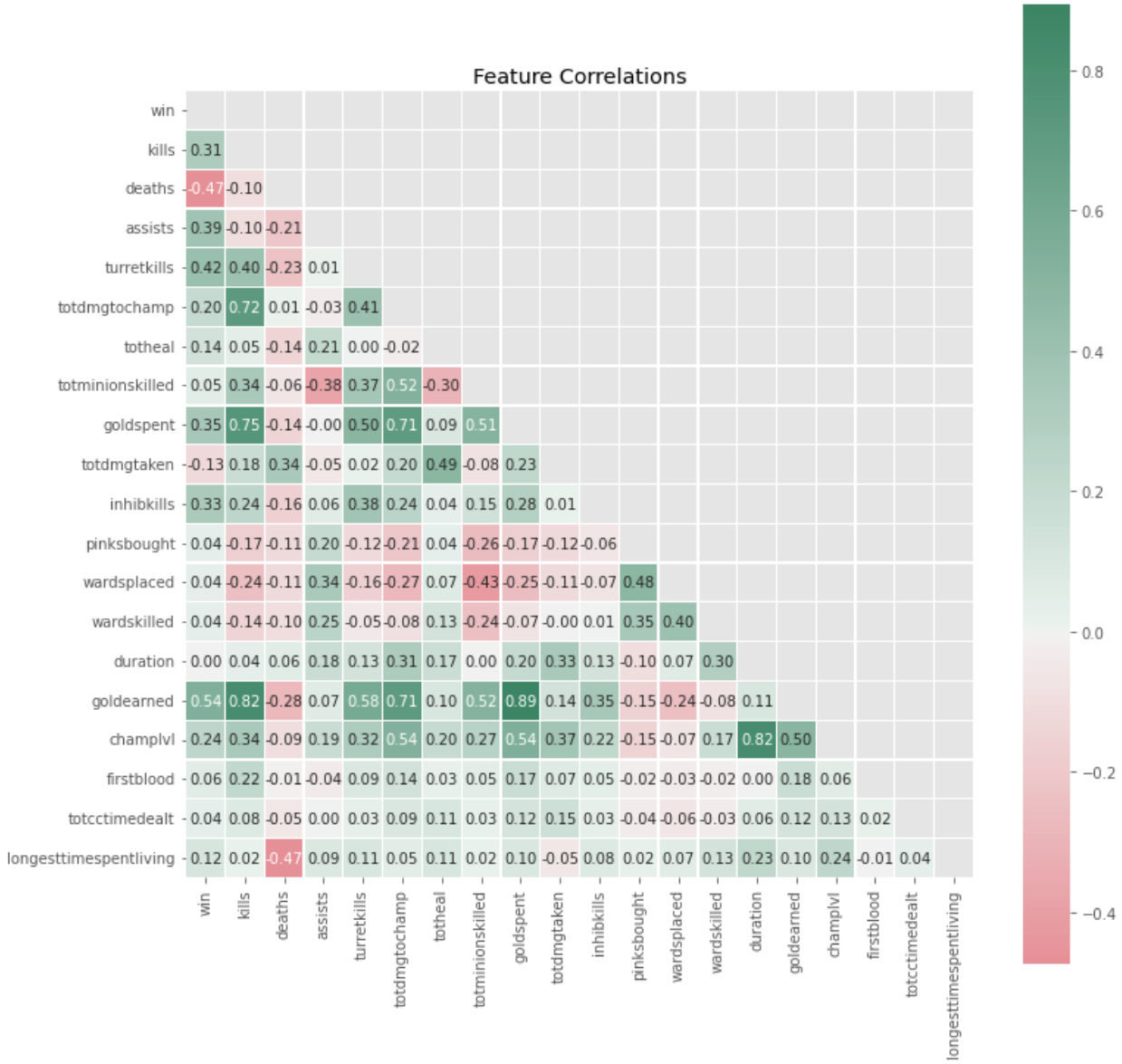



We used the f_regression to get scores for every one of the features. Using their scores to rank them gives the following:

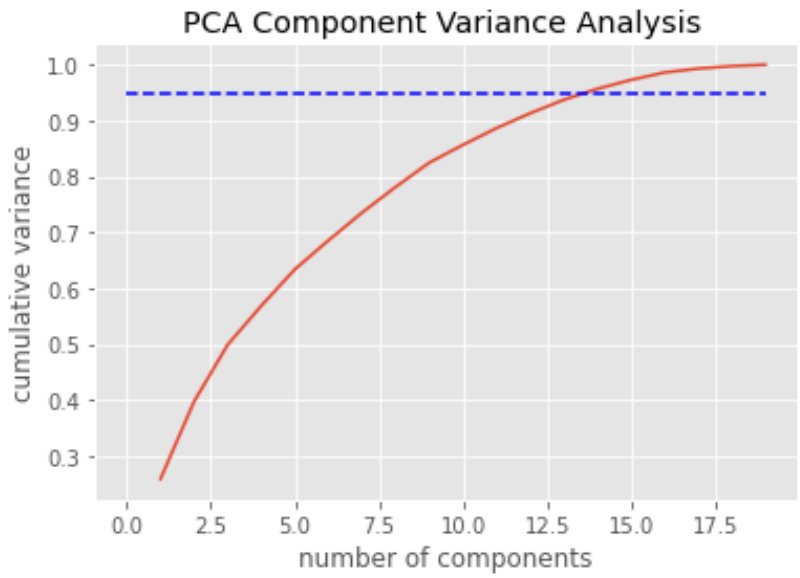
| Feature Name | f_regression score | Explanation / Notes |
|---|---|---|
| Gold Earned | 612876.07 | If a game goes on long enough, players will reach their item cap and gold will stop mattering. |
| Deaths | 423825.74 | Time needed to return after dying goes up during the game. While dead, you can fall behind as your opponent collects gold, xp, and damages or kills turrets. |
| Turret Kills | 318142.5 | Turrets stand between teams and the enemy base and must be destroyed in order to win. |
| Assists | 259543.0 | Overall, used as a metric to see if teams are fighting together or alone. |

| | | |
|---|---|---|
| Gold Spent | 210293.67 | Gold is used to buy items that improve stats and can provide new abilities. |
| Inhibitor Kills | 184150.42 | At least one inhibitor being destroyed is required for a win. |
| Kills | 160946.57 | When a player gets a kill, they earn gold, experience, and most importantly an edge over their lane opponent. This can lead to snowballs in games if a player gets many kills in the opening minutes. |
| Champion Level | 89209.08 | Champions level up from experience and gain more powerful abilities with each level. |
| Total Damage To Champions | 64660.83 | For more defensive champions, the goal is usually to survive damage. For offensive, it is dealing damage. |
| Total Healing | 27992.51 | Not all champions can heal themselves or others, but all can buy healing potions and build lifesteal. |
| Total Damage Taken | 23635.04 | A higher amount of damage taken can correspond to more deaths. |
| Time Spent Living | 20226.38 | Living Players have the potential to actively participate towards their team's success instead of being out of play. |
| First Blood | 5524.02 | Refers to the first kill of the game. This kill rewards bonus gold. |
| Total Minions Killed | 3775.25 | Killing minions is the most consistent way to earn gold. During early phases of the game, this will be the focus of many players, even over getting kills. |
| Wards Placed | 2271.04 | Wards provide vision at the location they are placed and are essential for high-level play. |
| Wards Killed | 2258.64 | Killing wards removes the vision of the enemy team around a certain area. Teams without vision in certain areas may be forced to abandon objectives or risk getting ambushed. |
| Pink Wards Bought | 2111.36 | Pink wards disable normal wards and cost gold each time they are used. |
| Total CC Time Dealt | 2055.69 | CC (Crowd Control) refers to things like being stunned, slowed, or put to sleep. Many champions are designed to CC the enemy team to enable damage dealers to move in for the kill. |
| Duration | 0.12 | Some champions get stronger over time, whereas some peak early in the match. Longer games benefit the former and shorter the latter. |

Looking at the graph, the obvious most important feature seems to be gold earned. This makes the most sense, as gold is what is needed to purchase items that allow champions to become more powerful. A player who earns lots of gold will see a gap develop between their champion and their lane opponent's champion. It is also worth noting that many other of the top features reward players with gold, such as getting kills, assists, killing minions, and destroying turrets and inhibitors. The next most important feature is deaths, which is the most negatively impacting of the top features. Dying rewards your opponents with gold, experience, and it removes you from the game for anywhere from five seconds to over a minute in long games. During that time, the opponents have one less player to keep track of and a massive advantage in being able to kill minions and damage turrets uninterrupted. Turrets and inhibitors are a genuine requirement to win, as a team needs to destroy at minimum five turrets and one inhibitor to destroy the enemy's nexus and win the game. We can also see the further results of this impact via the next most important pieces of data all being kill-related: All of these individual factors contribute towards increasing the opponent's death count. Some of the lesser important of the top features include things like wards. Wards provide information to players and while they do not explicitly cause kills or deaths, good players know to use them to keep tabs on the position of the enemy players.




This graph provides us with a useful look at which features correlate to each other. Worth noting in particular is the use of “win” as a feature, which grants us direct insight into which of the features are most important to a winning team. Unsurprisingly, earning gold is still the most positively impactful feature, and deaths is the most negative. Since wins require things like turret kills and inhibitor kills, these are also positively correlated. While the rest of the chart plays out as expected- such as longer games correlating with higher champion level - some may point out an odd negative correlation between minions killed and wards placed. In League of Legends, one of the three lanes traditionally features a two-player combo of Bottom Laner and Support. It is the support’s job to buy and place wards, and they have special items that generate gold for them over time. This is because the Bottom Laner is the person expected to get all the minions (and associated gold) in that lane. Support players, as they typically have less damage than their non-support teammates, get more assists than kills. This explains why assists correlate strongly with buying pink wards, killing wards, and placing wards.




Having chosen the important features though f_regression, we then used PCA to do the further feature selection. As shown in the graph above, we chose the top 14 components to account for 95% of variance.

# Method

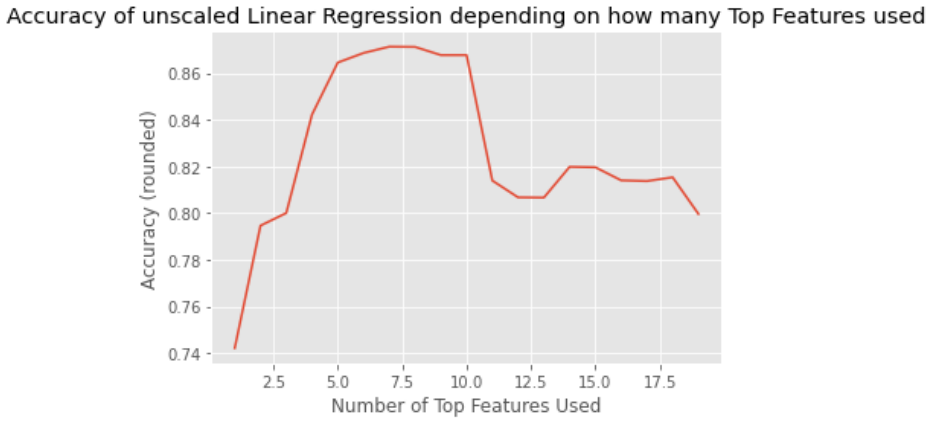
## Linear Regression:

Our first attempt at creating an outcome prediction model is to fit the training dataset with linear regression. Linear regressions provide us with a simple way to infer the binary outcome (win or lose) from a set of features. The linear regression also allows us to gain some insights into the direct correlations and connections between our features.

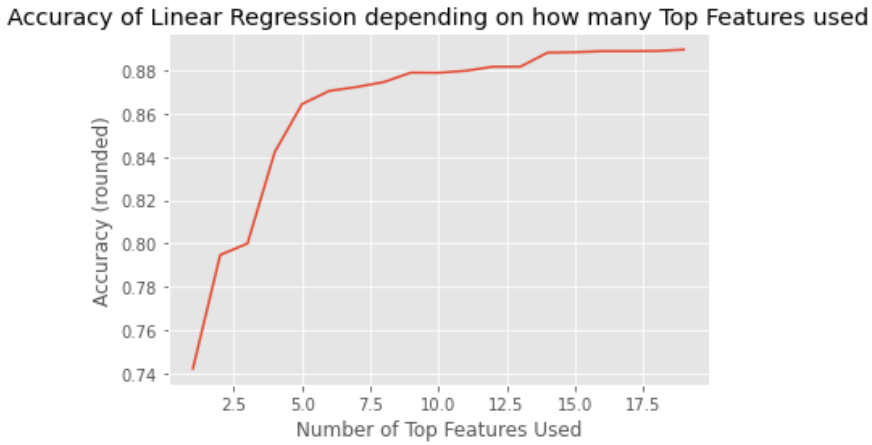
# Potential Results and Discussion

## Linear Regression:

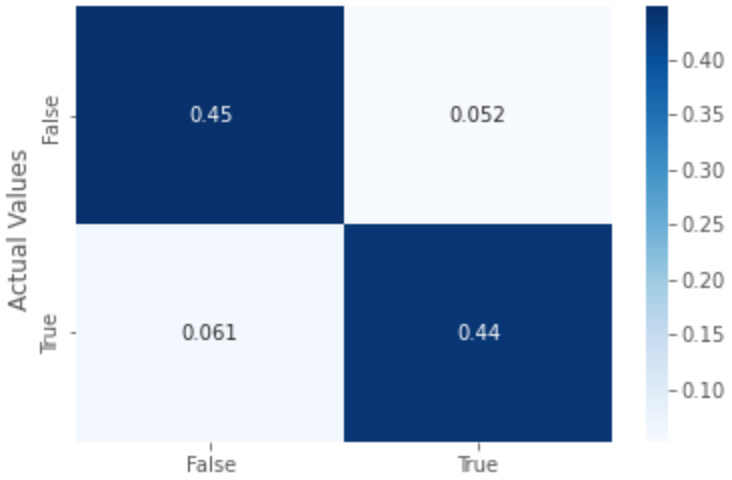
Linear Regression performed well on this dataset, after scaling.  Using KBest as shown above to select the top features, we demonstrated below how increasing the amount of features changes the accuracy.




However, we did not find this result to be satisfactory, because it did not seem to make sense that the less important features would have enough of an effect to decrease the accuracy if they are included. To fix this, we performed standard scaling on the data. This scaled the data to unit variance, and allowed it to give more expected and higher accuracies.
Below is the accuracy after scaling:




And as also stated before, 14 components is the spot where the accuracy begins to give diminishing returns. The exact accuracy of the final linear regression we used, which is the one using the top 14 components, is 88.8%.




As the proportional confusion matrix shows, the true positives and about equal to the true negatives, as are the false positives to false negatives. This shows the model is relatively balanced, not too eager to show something as a win or as a loss.