**Abstract –**

I investigated League of Legends datasets hoping to find insight on match predictions. The game relies heavily on easily identifiable data. A person can tell from just the end of match summaries which team wins. Thought it would be interesting to attempt to predict matches before they are even halfway done. The data set I used [1] took a snapshot of the match at the ten minute mark. Using this information, I ran the data through multiple classification algorithms hoping to find something that reliably could predict a winner early in the match. In the end all algorithms performed similarly, and the simplest option was the best.

**Introduction –**

I was interested in this dataset for betting purposes. Betting on esports works just like betting on anything else. So, I could make a model that could accurately predict the winner at the ten minute mark it would be possible to make easier bets. This dataset isn’t ideal for what I set out to do. These are high level matches but not professional matches. People will play different at those two levels. Nonetheless, this data could still be used as a starting point for that future model.

I started by simply analyzing the data and making sure it was clean. The dataset is perfect from a quick overview of it. Nothing stands out at unnatural or even as outliers. This are all real matches pulled directly from Riots API and even when the data seems odd it should be kept because not everyone plays the same. Also, this being an online game with ten different people, outliers become common and should be expected. I used four different models in an attempt to find which was the most accurate at predicting matches. They all performed similarly, and none stand out as a clear winner.

Interestingly over the years I regularly read articles about how between the two teams, blue has always had a higher chance of winning [2]. Numerous articles have been written about this all approaching from a statistical standpoint. Even the company Riot Games has pointed out that blue had a higher chance of winning over the years. At the highest level of competition, the same anomaly was present. Over the years Riot Games has addressed this but it was always interesting to know during some years blue had a five percent higher chance of winning matches. Between my interest of making bets on high ranked games and always being wondering if the blue team had an ever present advantage I looked into this dataset.

**Dataset –**

The dataset is rather robust. There are 39 features to analyze. This is great because the data includes more information than I thought would be even necessary to predict matches. The data includes over 9000 matches. Enough matches that splitting for training and testing wouldn’t be a problem. They were also all pulled during the same season on the same patch which is important. This game gets updated regularly for balancing purposes and if the matches had been pulled over months a lot could have changed during the time.

To someone who doesn’t play this game the feature in the dataset could be a little confusing. GameId is useless for the classification model so it was dropped. The target feature was blueWins, this is because a value of 0 is red winning and 1 is blue winning. Wards are deployable items that reveal areas of the map. These are useful for map knowledge and control. First blood is also here because of important it is in matches. That starting jump in experience can be impactful on how the early part of the match goes. The next few columns are just NPC’s that either provide buffs for an individual or their whole team or just an extra minion to push one of the three lanes. Towers are in each lane and need to be destroyed before it is likely for a team to win. Though it not likely that in the first ten minutes to many towers will be destroyed. Gold is important in this game it allows players to buy items that make them stronger. Gold is earned via killing NPCs, other players, or towers. All the features are then repeated for the red team also which is nice if I wanted to do a deep dive comparing the blue and red team. Though that can be completed another time I was just interested in possible being able to predict matches from this dataset.

Rectangle

Description automatically generated with low confidenceAt the beginning of my EDA I immediately looked at the value counts of blueWins. This data has a fair split between blue and red wins. In the long run that helps me no have to imbalance the dataset. The next couple of feature I looked at were blue team kills and deaths and same for the red team. They looked almost identical to each other. So one was getting an early game advantage in that department that the model would be able to predict from kills and deaths alone. My favorite column to look at throughout my EDA was wards placed. It seems the most outlandish to me and also just seemed so common. Around 100 wards in the first ten minutes is understandable for a team to do. Five players each dropping twenty wards each is realistic in a serious high ranked match. All these matches above 150 wards placed in ten minutes just amazed me. I seriously contemplated removing them, but I want the model to be able to handle such outliers.

I looked into early game gold difference and that average close to 0 which was expected. I then went ahead and compared high gold in the first ten minutes to see if that affect who won in the end. Not surprisingly, if a team has a high amount of gold in the first ten minutes, they are heavily favored to win the match by the end. Though blue wins more often with high amount of gold in the beginning than red did. I wanted to look at graphs that I figured the models might pick up on having a factor into winning. I wanted to see what was likely to guide the models in the end. So next I looked at blueWins compared to multiple other features. Getting first blood seems significant in winning the match. It is an early game advantage that keeps the other team from being aggressive. If the team also kills there dragons which provide buffs is important. Heralds seemed the least important because in the end they are just stronger lane minions. The towers destroyed graph looks how I expected it to. In the first ten minutes it’s unlikely for a tower to be destroyed. But, I know if these snapshots of matches were taken at a later point in the match this feature would be extremely impactful. The last interesting graph I made in my EDA is comparing the experience difference and the gold difference to if blue wins or loses. It is as expected the more gold or experience they have the more likely they were to win. I just like seeing some loses even though they were ahead in both and some wins when they were behind. Just shows how matches can turn around quickly.

Graphical user interface, scatter chart

Description automatically generated

**Methods –**

Since this dataset would require classification, I needed to reduce multicollinearity. The original correlation matrix for the entire dataset was a mess but I immediately knew I needed to remove features that were too similar or didn’t really have an affect on the output of a match. I decided to remove features with gold and experience since every aspect of the game affects those two. I only wanted features that directly lead to a win. I then reran the correlation matrix on my new data frame. It still had some multicollinearity that I was worried about, for instance redFirstBlood, redKills, and redDeaths are all direct inverses of blueFirstBlood, blueKills, and blueDeaths. After they were removed, I looked at the correlation matrix again. I was satisfied with how the matrix looked there was still high correlation between a few features, but they directly impacted one another and were still important like kills and assists.

From here I was ready to being running the data through some models. This dataset is a classification problem. The target variable is blueWins and that determines which team wins. I started with logistic regression because it is the simplest classification model out there. Since we are just looking for a win/ loss scenario, logistic regression will label all outcomes as 1 or 0 if the probability is greater than 0.5. After training it takes all the features and tries to predict on the test set if the new games are blue wins or losses. I next used random forest which is another classification model that creates multiple decision trees. It then relies on majority vote to decide the final splits for classification. While deciding on if I should use random forest, I came across another model called extra trees and since I wasn’t sold with how my last two model came out, I decided to implement it also. Extra trees work exactly like how random forest does with the exception that the splits are random and not optimally calculated. Lastly, I used XGBoost which is extreme gradient boosting. The only difference between XGBoost and gradient boosting is that XGBoost is more regularized which improves its ability to generalize. It uses both lasso and ridge regression creating a complex model.

**Results –**

|  |  |
| --- | --- |
| **Model** | **Accuracy (%)** |
| **Logistic Regression** | **71** |
| **Random Forest** | **71** |
| **Extra Forest** | **71** |
| **XGBoost** | **69** |

After setting my y and x variables and splitting the data for testing and training I ran my data through logistic regression. The accuracy score was 71%, precision was 71% and recall was 71%. In my mind this was a good start especially for the simplest model I was going to use. I hoped the more complex models would have higher accuracy since they have more going on in the background. Random Forest produced the same accuracy of 71% that’s when I started looking into Extra Forest before implementing it. In the end Extra Forest produced the same accuracy. I kept XGBoost last thinking it would give me the best results and in the end is gave me the worse accuracy at 69%.

**Conclusion –**

This project was a great starting point for what I originally was interested in doing. It allowed me to analyze League of Legends games and get an insight into what really matters when determining who might win a match. Given data from professional matches and possibly at a later point in the game the models would be much better at predicting the winner. In the end I am happy that most of the models were at least over 70% accuracy. It can be difficult to predict matches with such early data and with so many factors that can change the outcome. This data doesn’t include features that only happen much later in the game that have heavy importance for example a NPC called baron spawns at 20 minutes. There’s a lot I could have done differently during data processing that might have given me better results. I could have done some PCA and actual feature analysis to determine which features to keep and drop. I could have also dropped the outliers past three standard deviations from the mean and reran all the models. I do believe it might be possible for a model to be close to 80% accurate but I don’t know if the tradeoff of removing most features or halving the dataset is worth it.

**Bibliography:**

[1] <https://www.kaggle.com/datasets/bobbyscience/league-of-legends-diamond-ranked-games-10-min>

[2] <https://www.unrankedsmurfs.com/blog/lol-blue-side-advantage#:~:text=In%20any%20League%20of%20Legends,only%20lasted%20for%20one%20patch>.