

ISyE 7406 - Data Mining & Statistical Learning

League of Legends: Determining Success

Final Report

Instructor

Prof. Yajun Mei

Heurys Grullon - Last 3 digits of ID: 197

hgrullon3@gatech.edu

Abstract

League of Legends (LoL) is one of the most popular video games in the world today, possibly even the most popular game across our globe. There are over 150 million registered players, with an average of over 100 million monthly active players and over 10 million daily players in the U.S alone [1]. League of Legends is an E-Sport Multiplayer Online Battle Arena (MOBA) game developed by Riot Games in 2009. Within the game, two teams are pitted against each other with the objective of destroying the enemy's nexus (heart of their base). I attempted to gain insights into important predictors for team success and victory by employing statistical methods on over 50,000 ranked matches that were played throughout the course of one year. Exploratory Data Analysis, Logistic Regressions, Random Forests, Cross-Validation, and GBM were employed to determine factors that influence match outcomes and success, as well as match outcome prediction. Exploratory Data Analysis revealed that turrets and inhibitors are by far the most important features for win prediction with dragon and baron trailing close behind. The Gradient Boosting model performed the best in win prediction with a 97% game winner prediction accuracy and a .06 log loss mean. The GBM model showed that the most important features for winner prediction are the amount of tower kills for each respective team and the champions chosen for the match. Random Forest came in very close as well with a 96% game winner prediction accuracy and an f1-score of .97.

Introduction

This project aims to conduct a comprehensive analysis of player behavior, team dynamics, and match outcomes within the popular multiplayer online battle arena game, League of Legends (LoL). League of Legends is an E-Sport in which 10 players participate in 5 vs 5 matches where the goal is to destroy the enemy's base before they destroy yours. There are various objectives scattered across the map that aid teams in their journey to destroy the enemy's base. These objectives are highly contested by teams and lead to skirmishes and all out team fights due to how vital the power they award is.

Due to League of Legends being a virtual sport, this means it can provide an enormous dataset containing various metrics of gameplay at all times. From the beginning of the match to the end of the match, there is access to all player data and in-game events data. By leveraging this dataset, I seek to uncover insights into the factors that most influence match outcomes, the impact of blue side vs red side, and how meaningful certain conditions are for victory such as first Dragon secured, first Turret destroyed, etc. I will also predict the match outcomes using various statistical methods. My findings in this project may have implications for game balance, competitive strategy, and player meta optimization. By understanding which objective most

influences a win, teams may re-evaluate their playstyle and focus. Additionally, Riot Games may make balance adjustments to raise or lower the importance of certain objectives.

Problem Statement/Data Sources

The data sources used were straight from Riot's developer API <https://developer.riotgames.com/> and a league of legends kaggle dataset <https://www.kaggle.com/datasets/bobbyscience/league-of-legends-diamond-ranked-games-10-min> [2].

The total dataset contains 51490 rows and 61 columns. The data set contains no missing values. The rows are all data from Season 9 on League of Legends and the 61 columns are variables that describe first kill, first tower, first tower, first inhibitor, first dragon, amount of towers per team, amount of objectives per team, champions picked, bans done, amount of kills per player, winner, and more. One important caveat about this dataset is that Blue and Red side have a win rate of 50% each, which makes this an unbiased dataset.

Here, the total dataset uses the kaggle dataset as the base set, with the riot developer API used to pull more rows of data from the same season and server.

The columns for this dataset may be seen below. This entire dataset will be utilized in GBM to predict the winning team.

```
Index(['gameId', 'creationTime', 'gameDuration', 'seasonId', 'winner',
       'firstBlood', 'firstTower', 'firstInhibitor', 'firstBaron',
       'firstDragon', 'firstRiftHerald', 't1_champ1id', 't1_champ1_sum1',
       't1_champ1_sum2', 't1_champ2id', 't1_champ2_sum1', 't1_champ2_sum2',
       't1_champ3id', 't1_champ3_sum1', 't1_champ3_sum2', 't1_champ4id',
       't1_champ4_sum1', 't1_champ4_sum2', 't1_champ5id', 't1_champ5_sum1',
       't1_champ5_sum2', 't1_towerKills', 't1_inhibitorKills', 't1_baronKills',
       't1_dragonKills', 't1_riftHeraldKills', 't1_ban1', 't1_ban2', 't1_ban3',
       't1_ban4', 't1_ban5', 't2_champ1id', 't2_champ1_sum1', 't2_champ1_sum2',
       't2_champ2id', 't2_champ2_sum1', 't2_champ2_sum2', 't2_champ3id',
       't2_champ3_sum1', 't2_champ3_sum2', 't2_champ4id', 't2_champ4_sum1',
       't2_champ4_sum2', 't2_champ5id', 't2_champ5_sum1', 't2_champ5_sum2',
       't2_towerKills', 't2_inhibitorKills', 't2_baronKills', 't2_dragonKills',
       't2_riftHeraldKills', 't2_ban1', 't2_ban2', 't2_ban3', 't2_ban4',
       't2_ban5'],
      dtype='object')
```

The dataset used for the Random Forest method will have columns dropped that I myself find irrelevant due to my domain knowledge. This will be further explored below.

One of my intents in this project is to discover what factors within the game are most important for winning a league of legends game. To explore that, I first dropped columns related to champID, champion banned by player, summoner spells chosen, etc. Though it may be interesting to explore how champion picks, champion bans, summoner spells chosen, etc influence a win, it would require a vastly different approach and much more time. The data used

for Random Forest classifier has not been normalized nor standardized as this method relies on data partitioning rather than data point distances.

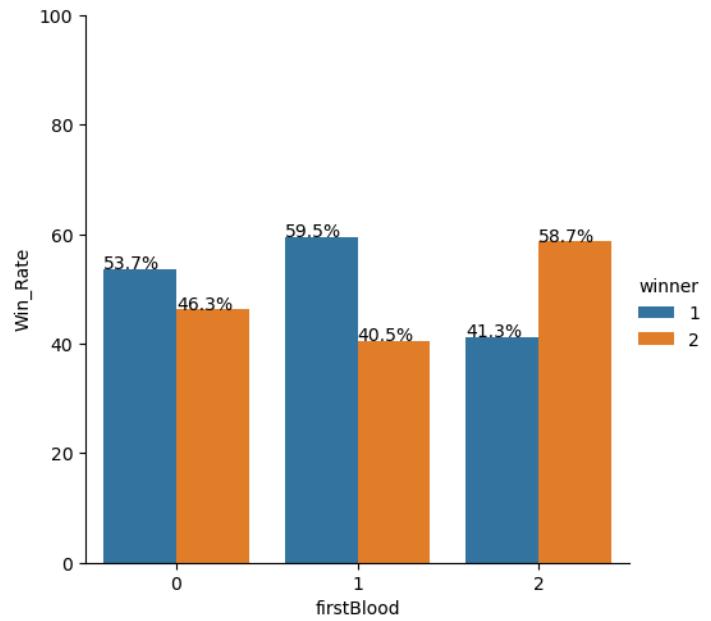
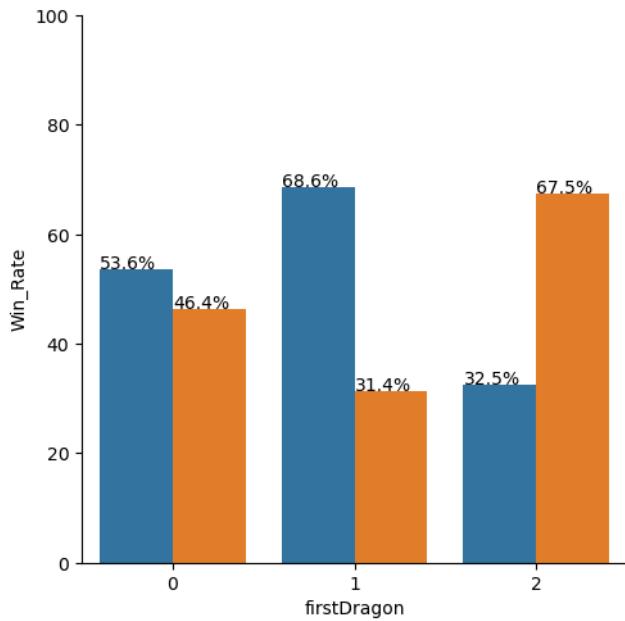
Below are the variables of dataset that shall be used for the Random Forest classifier.

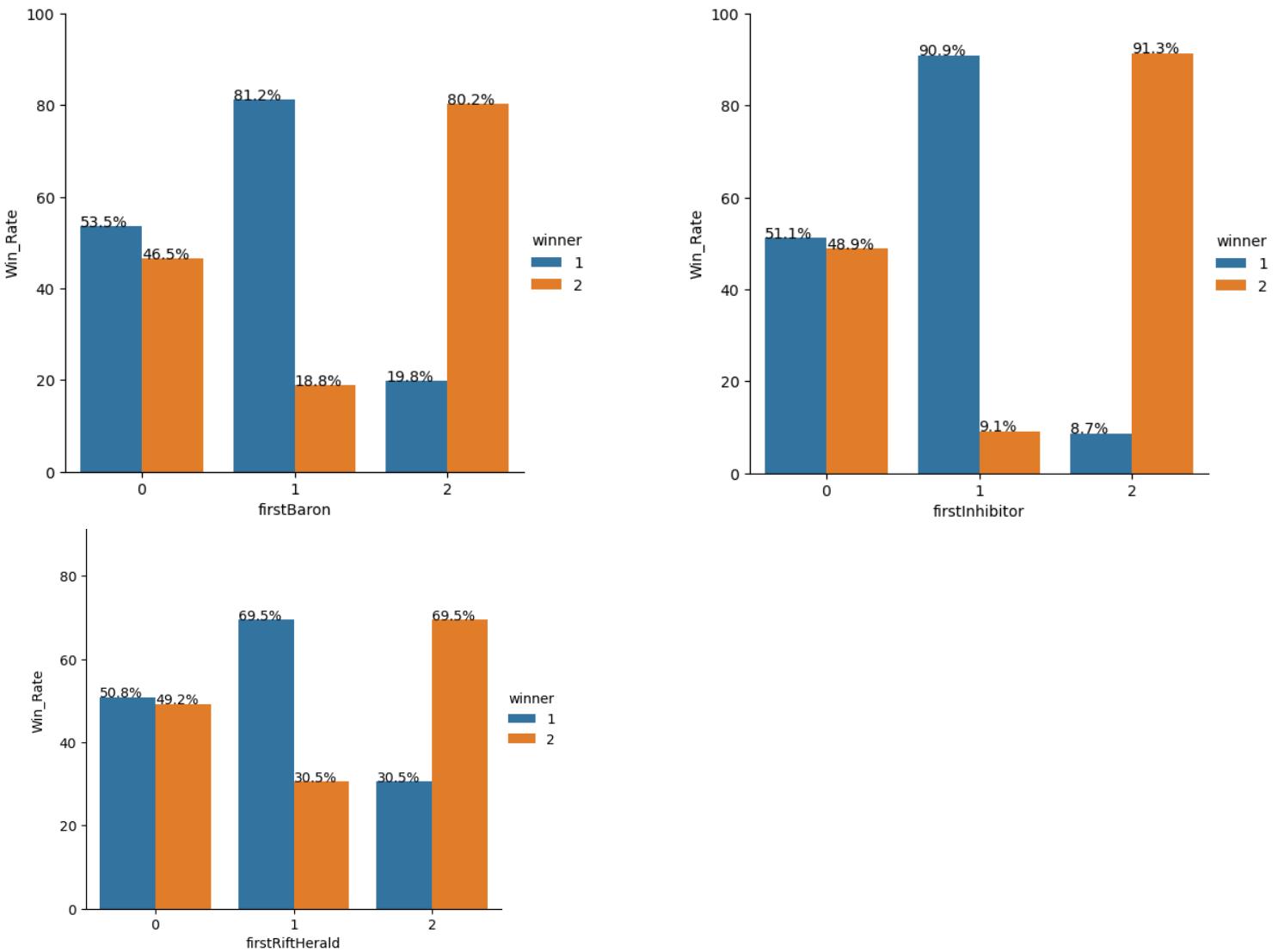
```
Index(['winner', 'firstBlood', 'firstTower', 'firstInhibitor', 'firstBaron',
       'firstDragon', 'firstRiftHerald', 't1_towerKills', 't1_inhibitorKills',
       't1_baronKills', 't1_dragonKills', 't1_riftHeraldKills',
       't2_towerKills', 't2_inhibitorKills', 't2_baronKills', 't2_dragonKills',
       't2_riftHeraldKills'],
      dtype='object')
```

Data Exploration

I first wanted to explore the dataset to be used with the Random Forest classifier. I wanted to first determine what percentage of games are won by the team that first claims a respective objective. So essentially, when first blood (first kill) happens by team 1, what's the win rate?

When the first tower is claimed by team 2, what's their win rate? These questions are applied to any column that begins with "first". The bar charts seen below are an attempt to explore the data.





Through the charts seen, it appears then that securing the first inhibitor is the most important feature for winning a match. Whenever a first inhibitor is secured, that team wins 90% of the time.

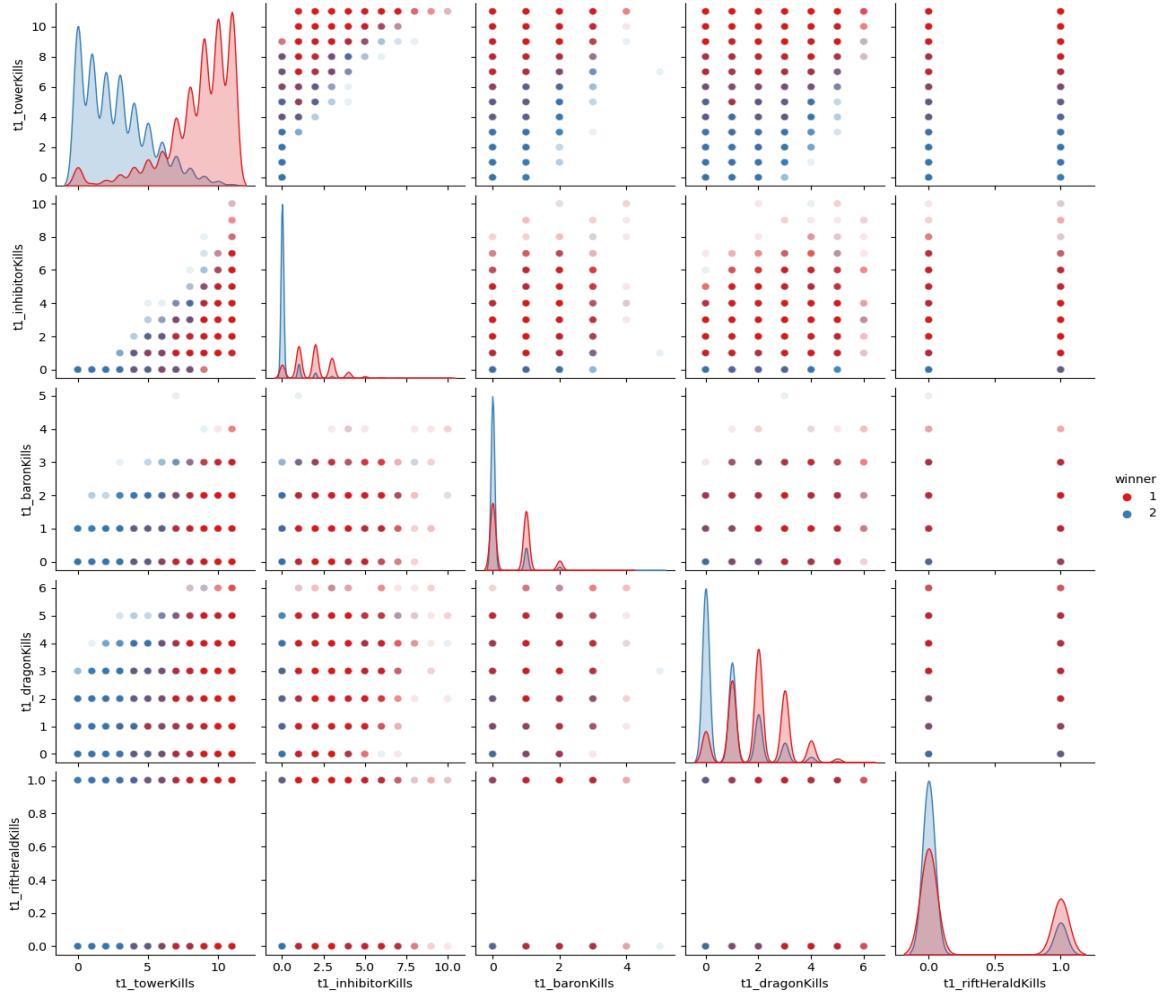
Comparing this to first Baron taken, where the first baron taken corresponds to a win 80% of the time, the first dragon corresponds to a win 68% of the time, and the first rift herald corresponds to a win 69.5% of the time.

First blood corresponds to a win only around 58% of the time, which is a meaningful impact, but nowhere near as impactful as securing a first objective. This makes sense as acquiring kills can place a bounty on you. This bounty increases in value the more gold you acquire above your opponent. Once you're killed, the opponent can receive a large sum of gold. This is done as an anti-snowball mechanic and can flat out erase any leads you have. This makes first blood the more volatile first claim to go for, as a player can always negate that early

advantage, but a first inhibitor or first dragon advantage can never be negated.

Interestingly enough, the first claim statistics are incredibly similar for Red and Blue side across the board, no matter what the feature is. This displays a symmetrical relationship between first claim win rates and the side of the map you play on, which is interesting since the map is asymmetrical and players have always claimed the blue side has an advantage.

Further exploring through the pair plot seen below.

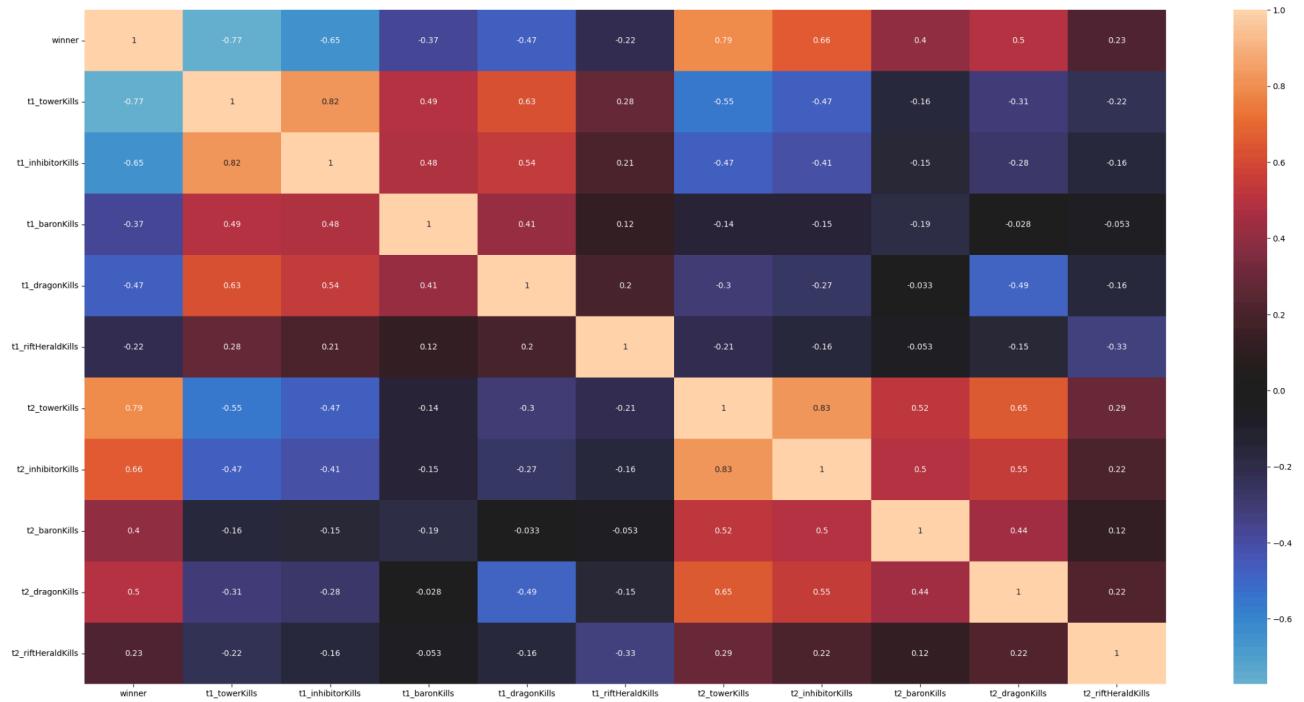


The plot above examines total objective kills against winning. Red represents wins for Team 1 and losses for Team 2. The same is true in reverse, blue represents wins for Team 2 and losses for Team 1.

It appears the most important objective kills for securing a win are tower kills. The more towers a team kills, the more likely they are to win. This finding makes complete sense, as to win a match (unless the enemy surrenders), you must destroy at least 5 towers and 1 inhibitor.

We also find that monster objective kills are important to winning, but not as much as tower kills. The more dragons a team takes, the more likely they are to win, by a longshot.

The correlation heatmap seen below shows Pearson correlation coefficients for each pairwise comparison of features. As I am focused on determining which factors are most important to predicting a win, I am only focused on the first row. Here then, we may find that tower kills and inhibitor kills are the most strongly correlated with a win, followed by dragon kills, then baron kills, then rift herald kills.



The weak correlation between winning and rift herald kills is pretty surprising to me as rift herald is an objective that is used to directly destroy turrets. A single rift herald can end up taking two to three turrets at a time.

Another surprising correlation is baron kills. To find it less correlated with a win than dragon is surprising, since Baron is supposed to be the objective a team takes to secure a win and end the game. It's a highly contested objective and not very easy to take at all. It usually requires at least three team members to take the objective and a meaningful time investment. It's such a time investment that the enemy will notice you're missing on the map and will assume you must be on baron. Meanwhile dragons can be taken by one team member and are much safer to take.

Random Forest and GBM

As I am focused on predicting the winning outcome with these statistical models based on various features, I need to use a classification model. Classification models are designed to predict categorical outcomes like win or lose (1 or 0) and are well-suited for this type of binary prediction task.

Random Forests are good for classification tasks. Random forests use a bunch of decision-makers (trees) to look at features and make a suggestion (prediction). An important piece is that each decision-maker (tree) doesn't see all the features. They only see a random subset of them. After all the decision-makers (trees) make their suggestions, the best one is voted on.

Random Forests can capture complex interactions between features and are robust to overfitting. It's much less likely to overfit compared to a single decision tree, which makes it a reliable pick. It also can provide insights into feature importance which was important to me for this project.

GBM is another statistical method great for classification tasks. Crudely, GBM is a statistical method that makes a prediction and then focuses on fixing the prediction through each new tree it generates. So if the first tree guesses wrong, the second tree tries to fix it. Then the third tree sees the improvement of the second tree and improves upon that. This continues until the final tree is close to the correct solution.

GBM was chosen due to how well renowned it is in model performance through increasingly refined approximations and its high efficiency in reducing bias and variance. Additionally at how well it handles non-linear relationships and its flexibility.

Analysis and Results

To set up the Random Forest Classifier, the data was split into a training set, which accounts for 75% of the total dataset, and a testing set, making up the remaining 25%. This split ensures that our model is trained on a substantial portion of the data while retaining a meaningful quantity for performance validation. The training set is using all of the dataset features pertaining to in-game objectives and kills.

As seen below, the model has demonstrated robust performance, achieving high scores in accuracy and the F1-score, which balances precision and recall. These metrics are crucial for assessing the effectiveness of the predictive model in real-world settings. Additionally, K-Fold Cross-Validation was employed to estimate the average performance of the model.

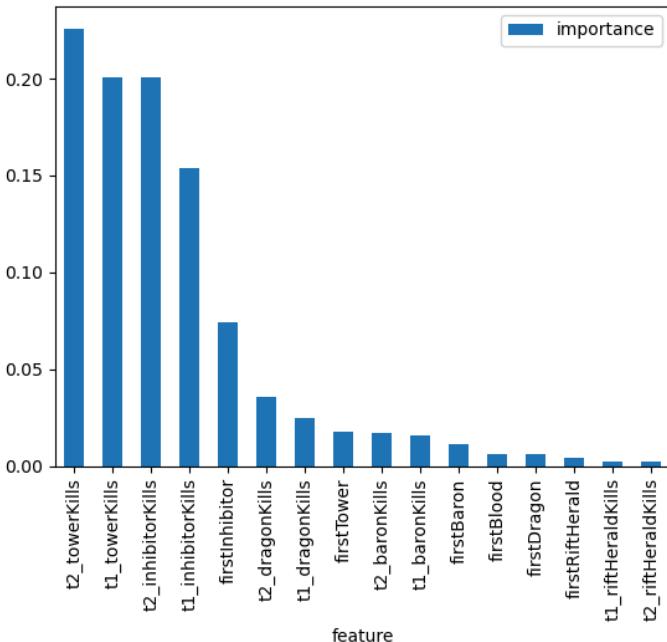
The first ran Random Forest accuracy is found to be 96.9% with the confusion matrix shown right below.

Accuracy = 0.9693933038141848				
[[6441 163]				
[231 6038]]				
	precision	recall	f1-score	support
1	0.97	0.98	0.97	6604
2	0.97	0.96	0.97	6269
accuracy			0.97	12873
macro avg	0.97	0.97	0.97	12873
weighted avg	0.97	0.97	0.97	12873

The confusion matrix, which helps us visualize the accuracy of predictions, showed a small number of misclassifications. However, these errors are minimal and do not significantly impact the overall accuracy of the model.

Using the Random Forest's built-in capability to assess feature importance, we have identified that tower kills and inhibitor kills are the most influential factors in predicting match outcomes. This insight aligns with our prior data exploration and validates the relevance of our model in strategic game planning.

As we can see below, tower kills and inhibitor kills for each team are found to be the most important features by the Random Forest classifier.



Performing 5-fold CV on the Random Forest retrieves a mean Random Forest prediction accuracy of 96%, as shown below by the output.

```
Cross-validation scores: [0.97115945 0.97154787 0.96921732 0.96941154 0.9751408 ]
Mean CV accuracy: 0.961295397164498
```

Moving forward towards using a GBM classifier, I first decided to feed all the data columns into this statistical method. I first ran a GBM with mostly default parameters and received a very high accuracy rate, very close to the random forest classifier. However, I was still concerned with tuning the model overall to find the optimal parameters for the model, so I employed a randomized search CV. Randomized search CV essentially picks random different combinations of parameters to use with my model and then evaluates the model performance using cross-validation. After trying out a number of random combinations, it identifies which one gave the best results. Using the parameters which were found to be optimal by randomized search CV, I employed the GBM classifier on the data and attempted to predict the game winner.

The first GBM model gave me a prediction with accuracy very close to Random Forest.

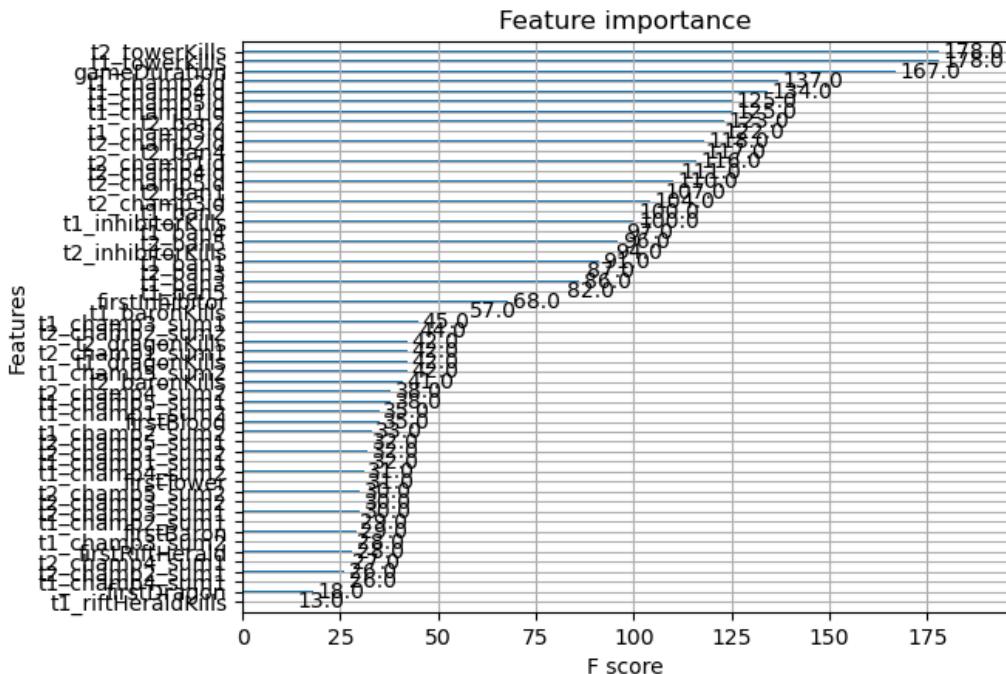
```
0.96931562184417
```

Using the optimized parameters given through randomized search CV, I used K-Fold CV to find the average accuracy of the GBM classifier.

```
Accuracy per fold: [0.97028549 0.97154787 0.96979996 0.96989707 0.97232472]
Average accuracy: 0.9707710234997087
```

The average accuracy is found to be 97%. This is very close to Random Forest's accuracy, only above it by just a tiny bit.

It's possible to extract the features GBM finds to be most important. The plot of feature importance may be seen below.



The tower kills being the most important feature aligns with what was found earlier through the exploratory data analysis and the random forest feature importance graph. Afterwards, champID and inhibitor kills are found to be the next important features.

ChampionIDs here being ranked highly in the feature importance for the GBM is not surprising at all and it's something I would love to dive into further. Ranked games specifically are heavily influenced by the existence of one-tricks which are known to have a very high win-rate on their respective champion. In the future, with enough time and data, I would love to look at the amount of games played per champion per player. I would be able to derive meaningful mastery curve graphs in which we may see how win rates per champion are influenced by amount of games.

For example, the champion Riven is known to be a champion that is difficult to master and perform with. Riven initially has an average of a 48% win rate, but as a player plays more games, it incrementally goes higher. On average, a player with 50 total Riven games played will have a slightly higher win rate than a player with one total Riven game played. A Riven player with 500 total games played on Riven will have a much higher win rate than both those players.

I'd like to explore the impact of champion mastery and how common it is for players to play their mastered champions in Ranked.

Conclusions

This analysis of over 50,000 ranked matches in the competitive E-Sport MOBA, League of Legends, has provided significant insights into the factors that influence match outcomes. By employing statistical methods such as Logistic Regression, Random Forest, Cross-Validation, and Gradient Boosting Machines (GBM), I was able to pinpoint key strategies and in-game objectives that teams should prioritize to increase their chances of victory.

The Random Forest and GBM models have both demonstrated exceptional performance, with accuracies around 97%. These models have shown that in-game objectives such as tower and inhibitor kills are crucial for winning matches, underscoring the strategic importance of these objectives. The Random Forest model highlighted tower kills and inhibitor kills as the most influential features in predicting match outcomes, which was corroborated by the exploratory data analysis. Similarly, the GBM model confirmed the significance of these features and also revealed the impact of specific champions (ChampionIDs) on game outcomes.

The insights gained from this analysis not only affirm the balanced nature of the game, as reflected in the equal win rates per first claim for the blue and red sides, but also highlight the nuanced strategies that teams can adopt to enhance their gameplay. These findings are invaluable for players looking to optimize their strategies, for coaches in developing team tactics, and even for the game developers at Riot Games who may consider these insights for future game balance updates. As an example, Riot may look at Dragon and Baron as objectives that influence a win outcome too weakly. A buff to these monster objectives may be in order and can definitely cause the meta to shift.

Furthermore, the project has opened avenues for deeper exploration into individual champion mastery and its impact on game outcomes. Understanding the relationship between a player's experience with specific champions and their performance in ranked matches could lead to more personalized coaching and training strategies, enhancing player engagement and competitive performance.

In conclusion, this project has not only advanced my understanding of what drives success in League of Legends but has also set the stage for further research into player behavior and game dynamics. By continuing to leverage data-driven insights, we can better understand the complexities of this global game and contribute to the evolving landscape of competitive E-Sports.

Lessons I have learned

An important lesson that I came across while making this project is how invaluable domain knowledge is. With proper understanding of the data set you're working with, you have an innate understanding of what the data should look like and what should truly matter to your models.

Additionally, an important lesson I learned from both this course and project is the importance of model selection, model evaluation, and model performance. Selecting the appropriate models, model techniques, and more, based on the nature of the data is incredibly important. Cross validation used to be something I didn't really care about too much in my first courses, I just wanted to do the math and I always felt like it was an intermediary step that got in the way. But as I was doing this project, when I first ran my first Random Forest model and received an accuracy of 97%, I was taken aback and thought I was overfitting quite hard. I knew I had to seriously employ some version of cross-validation to check on how much randomness was being over captured by the model. This is when I realized the power of cross validation in both parameter tuning selection and evaluating model performance.

I learned exactly when to use linear regression models, linear classification, ensemble methods, and more. I learned to appreciate the little intricacies and deviation from one class of model to another, just to deal with specific problems.

Appendix

Using the Random Forest's built-in feature `feature_importances_attribute`, which computes the mean and standard deviation of accumulation of the impurity decrease within each tree, I ranked the features that are most influential in the model's predictions. Given the data exploration done previously, the ranking of the feature importance done by the Random Forest makes sense.



For GBM, I trained and tested the dataset on over 1,000 boating rounds, including both the mean and the standard deviation of log loss across the folds for each round. The model's performance consistently improved over the 1,000 rounds, as indicated by the decreasing mean log loss values for both the training and test data sets seen below.

```
In [36]: dtrain = xgboost.DMatrix(X, label=y_transformed)
```

```
params = {
    'objective': 'binary:logistic',
    'eval_metric': 'logloss',
    'max_depth': 4,
    'eta': 0.01,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
}

cv_results = xgboost.cv(
    params,
    dtrain,
    num_boost_round=1000,
    nfold=5,
    metrics=['logloss'],
    early_stopping_rounds=50,
    stratified=True,
    seed=42
)

print(cv_results)
```

	train-logloss-mean	train-logloss-std	test-logloss-mean	test-logloss-std
0	0.684312	0.00021	0.684323	0.00013
1	0.675726	0.00048	0.675744	0.00091
2	0.667361	0.000138	0.667387	0.000194
3	0.659239	0.000272	0.659281	0.000292
4	0.651216	0.000313	0.651272	0.000244
..
995	0.054264	0.000340	0.062806	0.001571
996	0.054246	0.000340	0.062804	0.001571
997	0.054230	0.000341	0.062801	0.001570
998	0.054212	0.000342	0.062797	0.001572
999	0.054198	0.000342	0.062795	0.001573

```
[1000 rows x 4 columns]
```

During the Randomized Search CV used on GBM, I found the subsample to be 0.22, n_estimators as 200, max_depth as 9, and learning rate as 0.02. Lowest RMSE found was 0.14. Each tree built is only using 22% of the training data, randomly sampled without replacement. The number of trees built within the ensemble model is 200. More trees leads to better performance indeed, but I was risking overfitting.

The max depth of each tree is 9, I didn't want to overfit so I also tried different hyperparameters and it seemed okay to leave it at 9. A smaller learning rate of 0.02 requires more trees but I did have a more generalized model due to a more gradual learning process. A RMSE of 0.14 is quite low and satisfying indeed, which led me to believe the model performs quite well in terms of minimizing RMSE and overall.

Bibliography

[1] Priori Data. "League of Legends Data." Priori Data,
<https://www.prioridata.com/data/league-of-legends/>

[2] BobbyScience. "League of Legends Diamond Ranked Games - 10 Min." Kaggle,
<https://www.kaggle.com/datasets/bobbyscience/league-of-legends-diamond-ranked-games-10-min>