DSC 190 Final Project: What is it like to be an analyst?

DENNIS WU[*] and TAT HEI TSIN[*], University of California, San Diego

Fig. 1. League of Legends, a widely popular MOBA game, 2020.

This final project aims to decipher the winning formula of a popular Esports game: League of Legends. We used data sets ranging from normal to competitive players in order to glimpse into the important features leading to winning through predictive modelling.

Additional Key Words and Phrases: datasets, data mining

ACM Reference Format:
Dennis Wu and Tat Hei Tsin. 2020. DSC 190 Final Project: What is it like to be an analyst?. 1, 1 (June 2020), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 DATASET

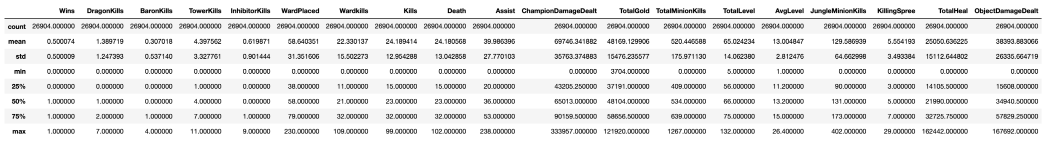
We have gathered our data from three different sites. The first option we did was scraping data from Riot API (Company that developed League of legends): https://developer.riotgames.com. However, we encountered a lot of problems in obtaining data due to strict regulation of number of requests per second, and we only gathered around 1000 instances till data. Therefore, we set eyes on other

[*]Both authors contributed equally to this research.

Authors' address: Dennis Wu, c8wu@ucsd.edu; Tat Hei Tsin, ttsin@ucsd.edu, University of California, San Diego, P.O. Box 1212, San Diego, California, 92093.

publicly available data sets. The second option we chose was a high elo players (High performance players in ranked games) data set from Kaggle: https://www.kaggle.com/gyejr95/league-of-legends-challenger-ranked-games2020. This data set consists over 200,000 games from the highest level of competition. The following Exploratory Data Analysis (EDA) would be done with this data set. For the third option, we found a data set consists of all the competitive matches recorded along the years: https://oracleselixir.com/matchdata/. There are different upsides and downsides regarding the three data sets. For option 1, we are able to get the richest data by far, but the distribution of the matches might be biased since we collected the player IDs according to their levels. For Option 2, that data set has the largest number of matches recorded among all the data sets, but the data set contains the least information per match among all the data sets. For the third option, the data set contains a decent number of matches and with more preprocessed details readily available. We decided to work on data set option 2.

We did our exploratory data analysis with the 26,000 challenger games in option 2. We implemented preprocessing steps to the data set since for one instance, the data set contains information for both teams. Therefore, we divided the instance into the red team/ blue teams, adding our instance to 52,000 instances in total. We listed the basic statistics of all the parameter in figure 2. All the parameters in the model are type int64, except Avglevel (float64). There are six categorical parameters which describe important objective in the game. After looking into the data, we are more interested in understanding what are the crucial factors that determines a win or a loss.



	Wins	DragonKills	BaronKills	TowerKills	InhibitorKills	WardPlaced	Wardkills	Kills	Death	Assist
count	26904.000000	26904.000000	26904.000000	26904.000000	26904.000000	26904.000000	26904.000000	26904.000000	26904.000000	26904.000000
mean	0.500074	1.389719	0.307018	4.397562	0.619871	58.640351	22.330137	24.189414	24.180568	39.986396
std	0.500009	1.247393	0.537140	3.327761	0.901444	31.351606	15.502273	12.954288	13.042858	27.770103
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	1.000000	0.000000	38.000000	11.000000	15.000000	15.000000	20.000000
50%	1.000000	1.000000	0.000000	4.000000	0.000000	58.000000	21.000000	23.000000	23.000000	36.000000
75%	1.000000	2.000000	1.000000	7.000000	1.000000	79.000000	32.000000	32.000000	32.000000	53.000000
max	1.000000	7.000000	4.000000	11.000000	9.000000	230.000000	109.000000	99.000000	102.000000	238.000000

	ChampionDamageDealt	TotalGold	TotalMinionKills	TotalLevel	AvgLevel	JungleMinionKills	KillingSpree	TotalHeal	ObjectDamageDealt
count	26904.000000	26904.000000	26904.000000	26904.000000	26904.000000	26904.000000	26904.000000	26904.000000	26904.000000
mean	69746.341882	48169.129906	520.446588	65.024234	13.004847	129.586939	5.554193	25050.636225	38393.883066
std	35763.374883	15476.235577	175.971130	14.062380	2.812476	64.662998	3.493384	15112.644802	26335.664719
min	0.000000	3704.000000	0.000000	5.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	43205.250000	37191.000000	409.000000	56.000000	11.200000	90.000000	3.000000	14105.500000	15608.000000
50%	65013.000000	48104.000000	534.000000	66.000000	13.200000	131.000000	5.000000	21990.000000	34940.500000
75%	90159.500000	58656.500000	639.000000	75.000000	15.000000	173.000000	7.000000	32725.750000	57829.250000
max	333957.000000	121920.000000	1267.000000	132.000000	26.400000	402.000000	29.000000	162442.000000	167692.000000

Fig. 2. Basic statistics for numerical values available in the datatset.

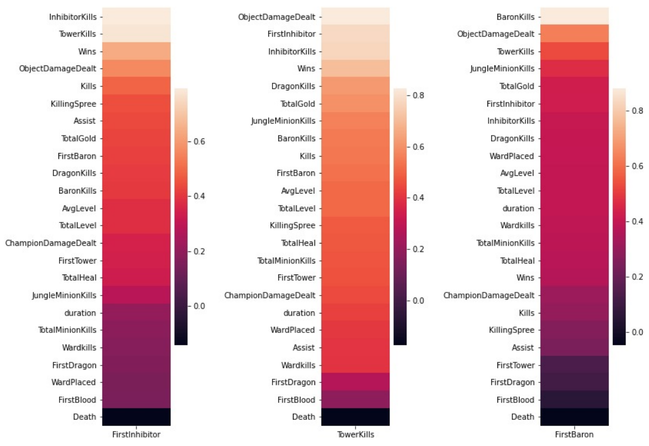


Fig. 3. Correlation of First Inhibitor, Tower kills and first baron with other parameters in the dataset, ranked by importance.

2 PREDICTIVE TASK

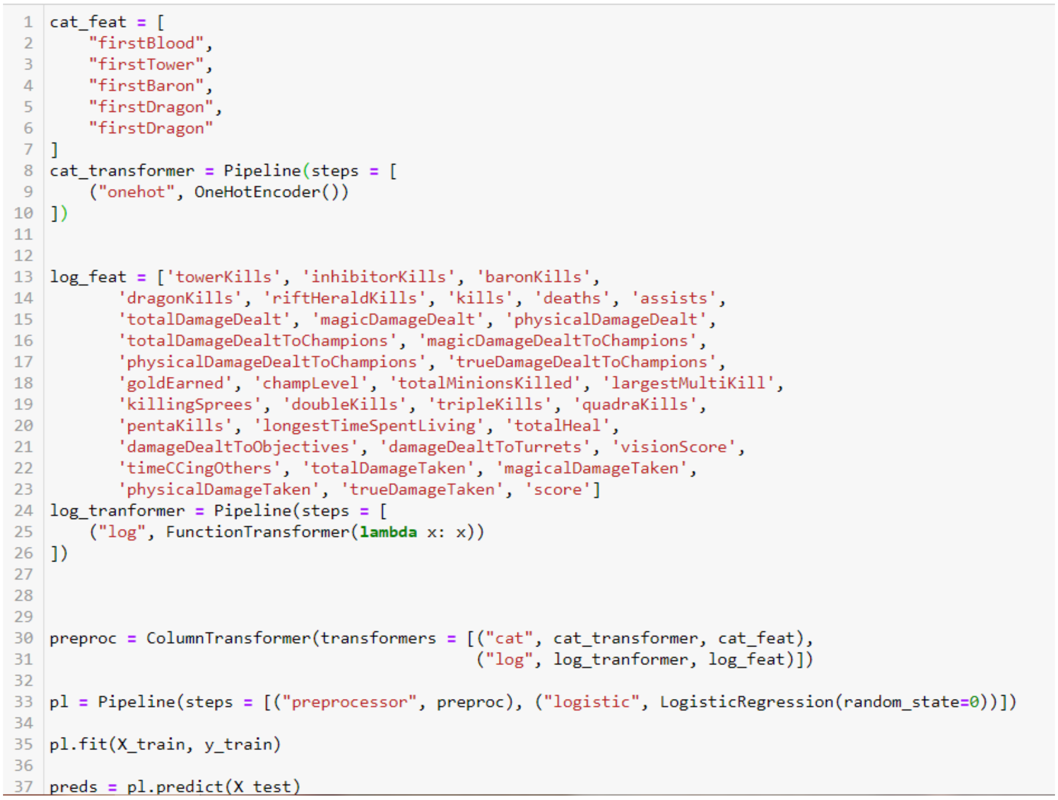
In this predictive task, we hope to find out more about the weight off different factors in predicting a win / loss to a game. We plan to use a variety of models to tackle this classification problem: logistic regression and XGBoost. We have all our end-game statistics provided in the data set, our baseline model would be logistic regression since we have a binary outcome: win/ lose. Moreover, and we make the assumption that all the statistics are a linear combination towards the end result. However, to further amplify our model, we utilize the XGBoost classifier in the refined model to increase the accuracy of our model, because boosted trees can largely increase the prediction power with gradient boosted decision trees that could iteratively refine the model. To evaluate the model(s), we would be using two metrics: Accuracy and F1 score. We choose accuracy since we would like to know how well the model predicts wins/ loses. F1 score would also be another comprehensive evaluation metric since it would give us an insight to the recall and precision of the model(s).

3 MODEL

The original data set has comprehensive features on what we are looking for (the important factors that can determine the result of the games), we basically normalize the data with the duration of each game. This feature of engineering is proven to be effective. To find the best features that have a high correlation to the result of the game, we devised EDA to analyze the features and tried to find the correlation between the features. For example, damage dealt with champions can be a strong factor determining the result of the games, but other features like first blood, first baron, and a number of dragons killed may have a high correlation to total damage. In terms of optimization, our model has done a great improvement from the baseline model to the improved model, with accuracy increasing from around 70% to nearly 98%. We had to admit that the task is simple given that the statistics after the match can easily show the result of the match, but the significance of the optimization is that some of the features being engineered to do play an important role in determining the result of the game. For example, after finishing the normalization, ObjectDamageDealt doesn't affect the accuracy of the model that much.

Before successfully setting up the model, we encountered some problems with implementation. The first one we encountered is dataset availability. Riot API has some strict rules against the individual developers so that at first we were only able to develop the baseline model based upon a limited amount of instances. By comprehensive EDA on the dataset, we figured that there is no need for us to parse instances every time we opened the notebook, so we decided to broaden the number of features by utilizing the data set available on Kaggle. The other problem we met is how to evaluate the champion combination for each side. The huge challenge of League of Legends champion combinations is that the strength is not static with time. Some may pose a great threat against the enemy team at the very early stage of the game, while others dominate the game later. In other words, the scale is astonishingly different. Times series analyses of champion strength can be a good approach to address this problem, but it's hard to quantify the champion skills. Hence, we cited metasrc.com's [1] champion tier list, which contains the subjective score for each champion based upon "win rate, ban rate, pick rate, and KDA." In other words, we avoided this problem by going the other way around.

In our demo, we tried to present an interactive demo of how we can calculate the result of the game based on the objectives being taken down by each team. This can a prospective application of the model we develop. Choose the statistics in the drop down bar, can the result can be calculated and presented.



```python
cat_feat = [
    "firstBlood",
    "firstTower",
    "firstBaron",
    "firstDragon",
    "firstDragon"
]
cat_transformer = Pipeline(steps = [
    ("onehot", OneHotEncoder())
])


log_feat = ['towerKills', 'inhibitorKills', 'baronKills',
        'dragonKills', 'riftHeraldKills', 'kills', 'deaths', 'assists',
        'totalDamageDealt', 'magicDamageDealt', 'physicalDamageDealt',
        'totalDamageDealtToChampions', 'magicDamageDealtToChampions',
        'physicalDamageDealtToChampions', 'trueDamageDealtToChampions',
        'goldEarned', 'champLevel', 'totalMinionsKilled', 'largestMultiKill',
        'killingSprees', 'doubleKills', 'tripleKills', 'quadraKills',
        'pentaKills', 'longestTimeSpentLiving', 'totalHeal',
        'damageDealtToObjectives', 'damageDealtToTurrets', 'visionScore',
        'timeCCingOthers', 'totalDamageTaken', 'magicalDamageTaken',
        'physicalDamageTaken', 'trueDamageTaken', 'score']
log_tranformer = Pipeline(steps = [
    ("log", FunctionTransformer(lambda x: x))
])



preproc = ColumnTransformer(transformers = [("cat", cat_transformer, cat_feat),
                                            ("log", log_tranformer, log_feat)])

pl = Pipeline(steps = [("preprocessor", preproc), ("logistic", LogisticRegression(random_state=0))])

pl.fit(X_train, y_train)

preds = pl.predict(X_test)
```

Fig. 4. baseline model code

4 RESULTS

The results shows the relevance and importance of the game statistics to the result of each match. From the baseline model to the refined model, the accuracy improves from 0.66 to over 0.98. The gap is significant in that the model basically produces the accurate results for near every instances.

Here are the interesting findings we delve out of the model. With the refined model being able to predict the final result, we conditionally eliminate some of the features to see if there are some features that can outperform others. For example, the most important objective, for the players should be the first inhibitor, which is a late game objective in LoL. At the same time, the most accurate single numeric feature is the total gold difference for each match. One recommendation I can give for the professional LoL teams is try as hard as they can to compete the other team on the golden.

	Master	Grand Master	Challenger
Accuracy	0.993	0.988	0.986
F1 Score	0.99295	0.98820	0.98607

Table 1. Evaluation metric for the models in three different tiers. Each tier consists of 5000 games. Trained with XGBoost classifier

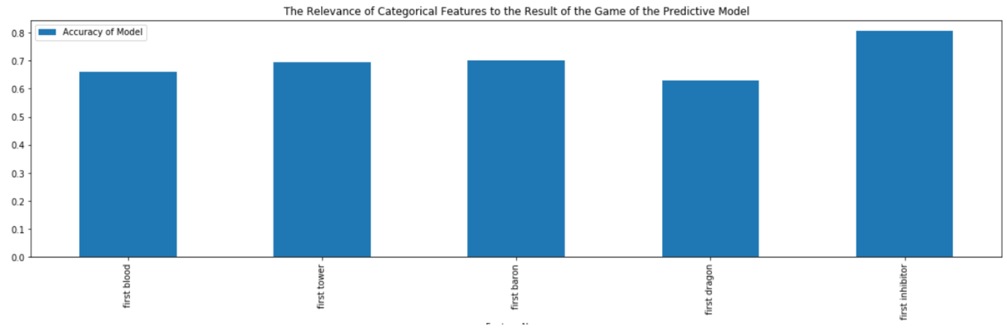


Fig. 5. categorical features relevance to the result

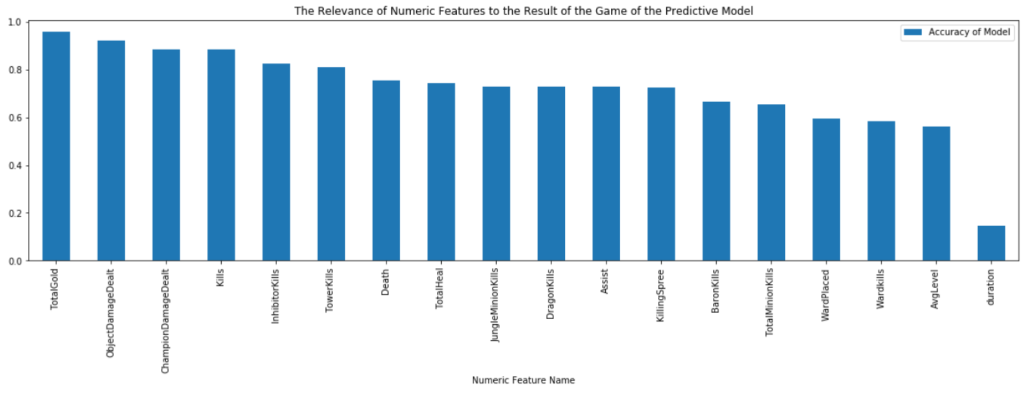


Fig. 6. numeric features relevance to the result

However, there are some potential dangers of our model. The champion scores calculated based on matasrc.com [1] may vary from patch to patch. Therefore, the model should constantly refresh its database of champion scores.

Given the consistency of our model based on the available dataset, we deem it to be useful in analyze the games in professional LoL arena. With no special attention to the hyperparameter, the model is trained for finding the special feature that can should be prioritized for the team.

5 LITERATURE

There have been some literature that study win prediction using machine learning algorithms. Different methods have been implemented for feature extraction. There are papers that used simpler metric [2] that records the presence of champions on both teams or comprehensive metrics [3] that further dissected a match into features like champion attributes and post-game statistics for further analysis. In both papers mentioned above, different machine learning models are also Utilised. For [2], they have used an augmented logistic regression that obtained a F1 Score pf 0.781. In another paper [3], they used classifiers like random forest and Support Vector Machine (SVM) for the model, obtaining a F1 score of 0.641 with their feature extractions techniques. Comparing across existing literature, we can see that our model has a more accurate performance, this might be due to the more mature machine learning we used and we did not utilise specific champion statistics in our data set. The major novelty of our work is that we studied this data set using a state-of-the-art machine learning classifier and analyses importance of respective factors towards the success of the model.

REFERENCES

[1] [n.d.]. *Patch 10.11 5v5 NA Tier List.* Retrieved June 9, 2020 from https://www.metasrc.com/5v5/na/tierlist

[2] Kaushik Kalyanaraman. 2015. To win or not to win ? A prediction model to determine the outcome of a DotA 2 match.

[3] Nanzhi Wang, Lin Li, Linlong Xiao, Guocai Yang, and Yue Zhou. 2018. Outcome prediction of DOTA2 using machine learning methods. In *ICMAI '18*.