Win Predictions For League of Legends Game Using Early Phase Data

Individual report

Introduction

Multiplayer Online Battle Arena (MOBA) is popular and becomes a professional field in recent years. Win prediction in online video games has become an import application of machine learning (Lincoln,). League of Legends (LOL),the biggest Eports game in the world, is one of the main MOBA games. League of Legends World Championship Series is the annual professional tournament hosted by Riot Games since 2011 (League of Legends WiKi). The results of LOL marches attract public attention. Many researches have been done to predict the results of LOL game. For example, Gradient boosted tree (GBT) and logistic regression (LR) was used to predict the Gold tier game (Lin 2017). In Lincoln's research, the accuracy of model (AUC value) achieved 97%. Historical performance of the teams, characters data, match features and other 11 variables, had been use with Random Forest and Logistic Regression algorithms to build models (Lincoln, 2021).

However, previous researches focused on using historical records to predict march results. Some of them were small-scale study. Few researches used performance data in the first 15 minutes of a certain march to predict a final win result.

The purpose of this project is to build an effective prediction model which uses in-game data to predict the final result. By using the data within the first 15 minutes and supervised machine learning algorithms, like Random forest, Decision tree, Logistic regression, a binary result of two teams (Blue and Red) would be predicted. 1 for blue win and 0 for red win.

Individual work

The aim of our team is every team member has a deeper understanding of data mining process. As the arrangement of our team, team members did what they can do first no matter whether these steps are repeated or not. I did pre-processing, feature selection, modeling and GUI.

Results

Pre-processing

I plotted distributions of 19 variables(Figure 1& Figure 2) and the target (Figure 3). Four columns were dropped. First, 'Unnamed: 0' and 'matchId' were dropped. No dragon appears in the first 15 minutes. Therefore, other two columns, 'blueDragonKills' and 'redDragonKills', were dropped because the value of them was zero. The number of 0 and 1 in target variable is almost same so this dataset is balanced.

```
#distribution
plt.figure(figsize=(20,18))
for i, col in enumerate(df):
    plt.subplot(5,4,i+1); sns.distplot(df[col])
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()

df.drop(labels=["Unnamed: 0", "matchId", "blueDragonKills", "redDragonKills"]
        , axis=1, inplace=True)
pd.set_option("display.max_columns", len(df.columns))

df.info()   # no missing value
# data and target
df_data = df.drop(labels="blue_win", axis=1)
target = df['blue_win']
```

Figure 1. Feature distribution code

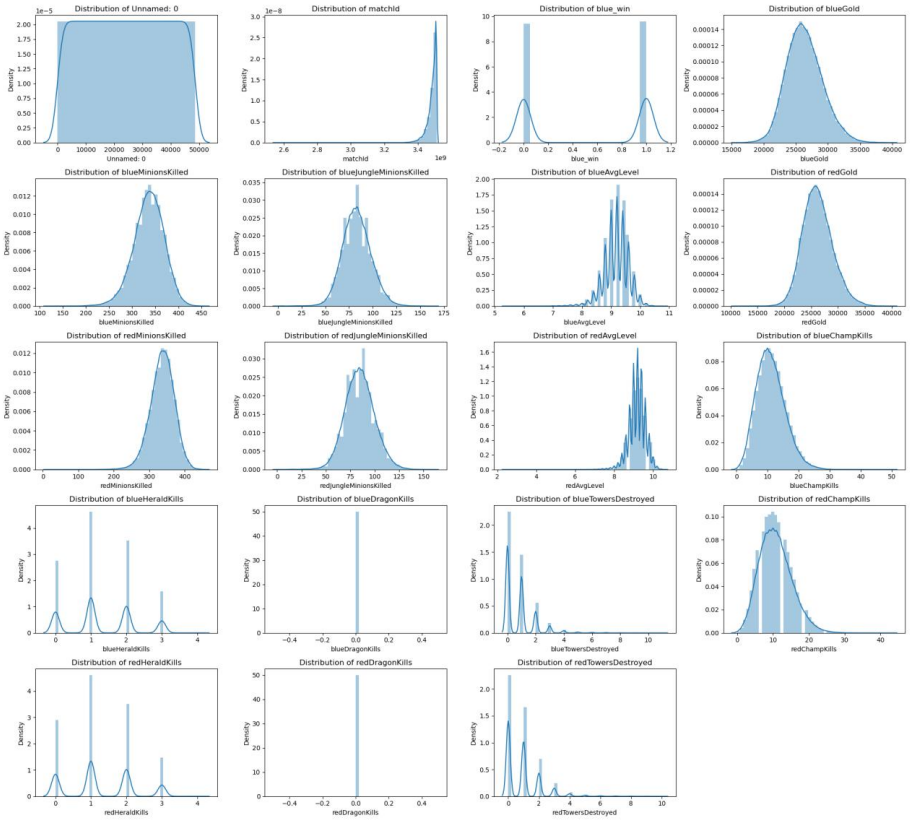

Figure 2. Feature distribution plot

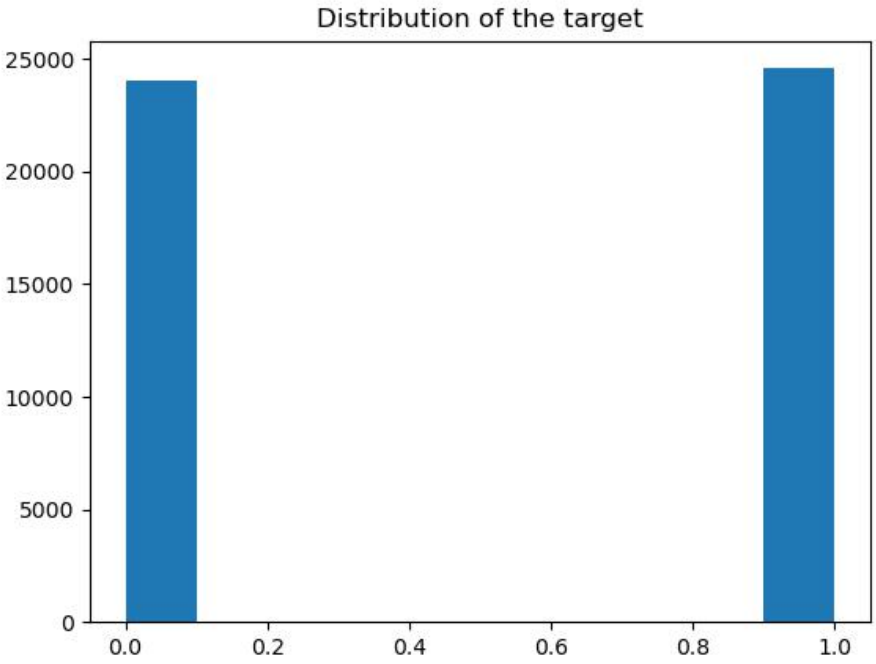

Figure 3. Target distribution

Boxplots of 14 features were plotted to find ouliers (Figure 4). One low point was found in redMinionskilled. Also other single point with low performance in red team (Figure 5). This instance was dropped. There is no duplicate in the data set. For the normalization part, MinMaxScaler was used to scale data. However, this step didn't influence the accuracy of models finally.

```
 # outliners
plt.figure(figsize=(15,18))
for i, col in enumerate(df_data):
    plt.subplot(5,3,i+1); sns.boxplot(df_data[col])
    plt.title(f'Boxplot of {col}')
plt.tight_layout()
plt.show()

redMinionsKilled_min = df.loc[df['redMinionsKilled']==14]
df1 = df.drop(labels=26207,axis=0,inplace=False)
df1.info()
```

Figure 4. Outliers Check code

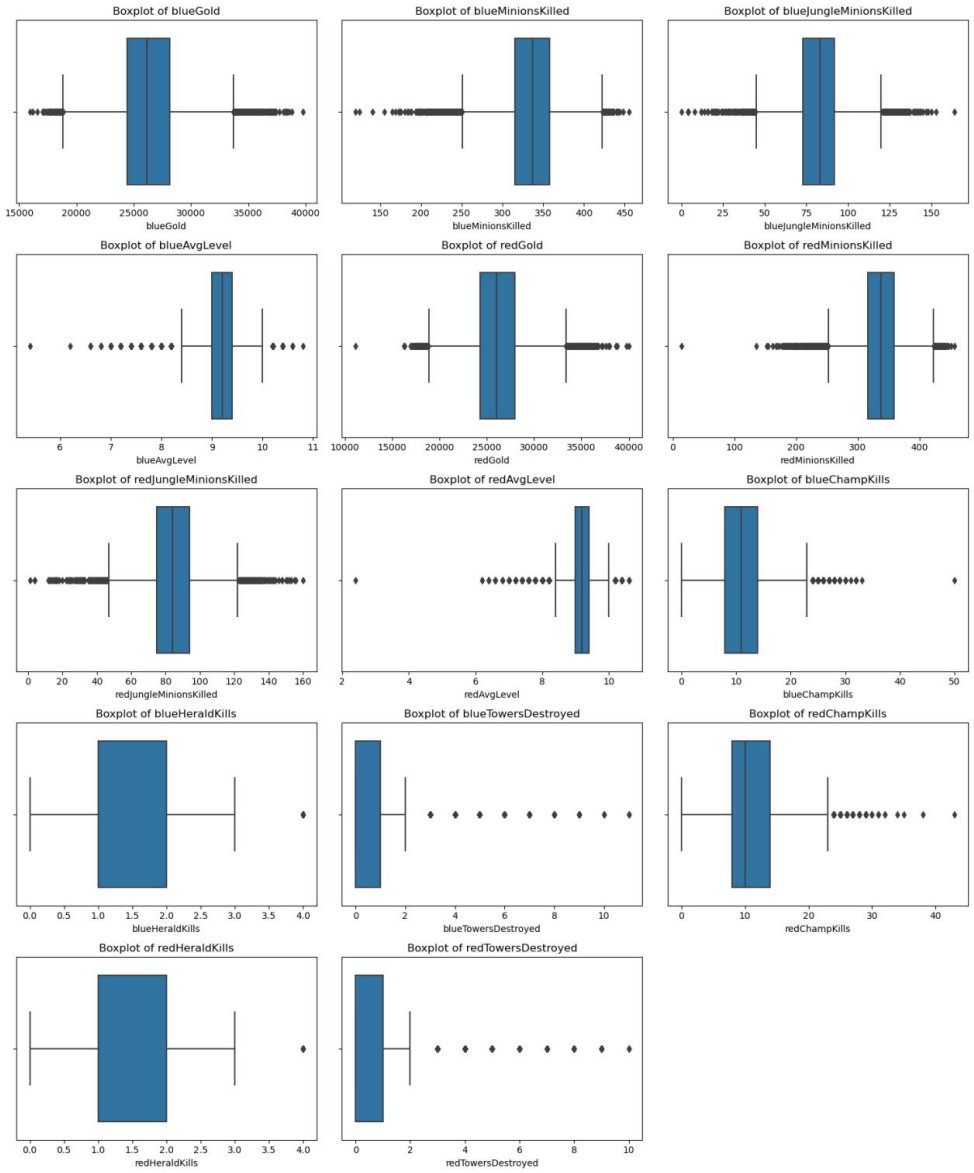

Figure 5. The boxplots of Outliers

Feature selection

PCA was used to do feature selection, reducing 15 variables to 10 variables. Parameter n_components was set to 0.95, which means 95% of the information was reserved. However, this step didn't influence the accuracy of the models finally either. The explanation might be 15 variables are not that much and every feature is significant to predict the target.

Modeling

This project is a classification problem. Many models can be used, like Decision tree, Random Forest, Logistic regression, SVM, KNN, Naive Bayes. Two models were built. One is Decision tree and another is Logistic regression. Attribute score and cross validation score were used to evaluate models. Train set is 70% and test set is 30%.

For the first model, three parameters were set, max_depth=8, min_sample_leaf=40, min_samples_split=40. A parameter curve was plotted to find the best value of max_depth (Figure 6 & Figure 7). A Decision tree was plotted (Figure 8 & Figure 9). The score of decision tree model was around 0.78 on train dataset and around 0.77 on test dataset. Therefore, the model is not overfitting or underfitting.

```
import matplotlib. pyplot as plt
from sklearn import tree
test = []
for i in range(10):
    clf = tree.DecisionTreeClassifier(criterion='entropy'
                                      ,random_state=20
                                      ,splitter='random'
                                      ,max_depth=i+1
                                    #  ,min_samples_leaf=10
                                      )
    clf = clf.fit(xtrain,ytrain)
    score = clf.score(xtest,ytest)
    test.append(score)
plt.plot(range(1,11),test,color="red",label="max_depth")
plt.title("Parameter Curve for Max_depth")
plt.legend()
plt.show()
```

Figure 6. Parameter Curve for max_depth

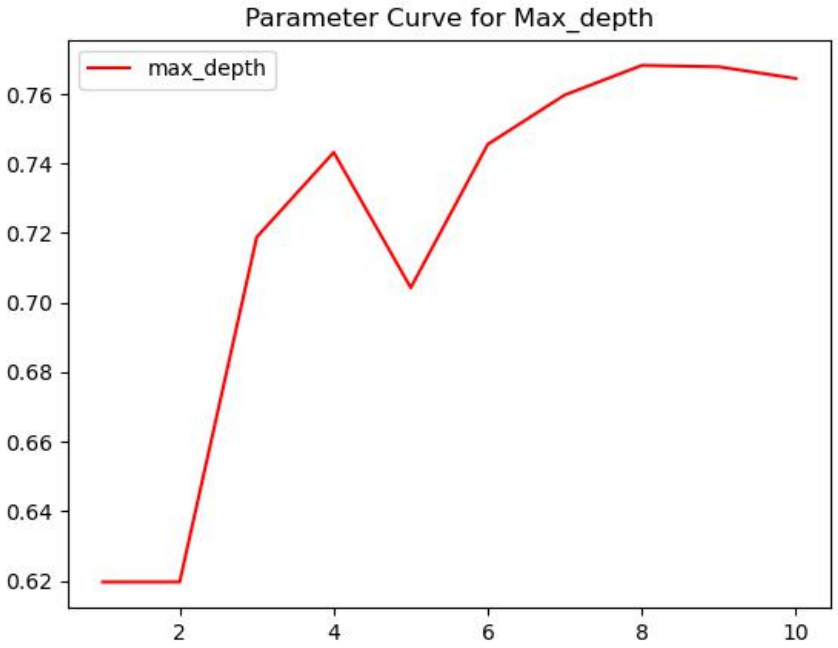

Figure 7. Parameter Curve of max_depth

```
from pydotplus import graph_from_dot_data
dot_data = tree.export_graphviz(clf
                                ,feature_names=['blueGold', 'blueMinionsKilled', 'blueJungleMinionsKilled',
    'blueAvgLevel', 'redGold', 'redMinionsKilled', 'redJungleMinionsKilled',
      'redAvgLevel', 'blueChampKills', 'blueHeraldKills',
   'blueTowersDestroyed', 'redChampKills', 'redHeraldKills',
        'redTowersDestroyed']
                                ,class_names=['blue win','red win']
                                ,filled=True
                                ,rounded=True
                               )
graph = graph_from_dot_data(dot_data)
graph.write_pdf("decision_tree_entropy.pdf")
```

Figure 8. Decision Tree code

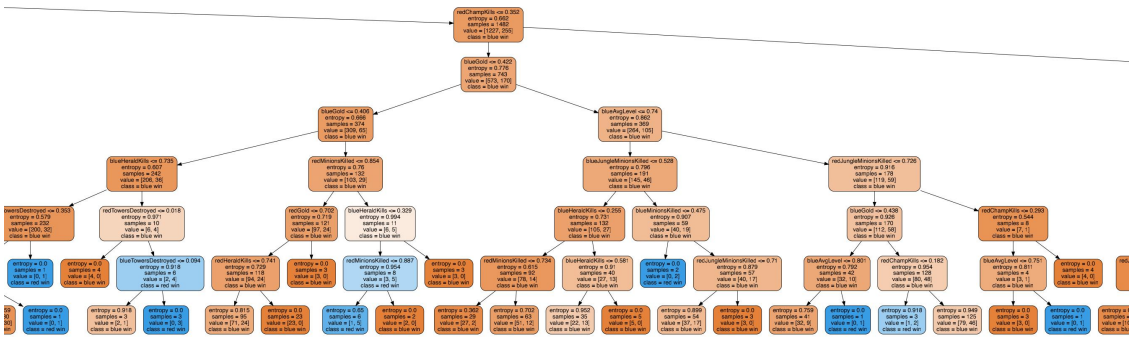

Figure 9. Decision Tree Plot

For the second model, two parameters were set, C=0.9 and max_iter = 5. Penalty L2 is default. I plotted a parameter curve to find the best value of C (Figure 10 and Figure 11). The score of Logistic Regression model was around 0.79 on train dataset and around 0.78 on test dataset (Figure 12). Therefore, the model is not overfitting or underfitting.

```
# C Curve
for i in np.linspace(0.05,1,19):
    lrl1 = LR(penalty='l1', solver='liblinear', C=i, max_iter=1000)
    lrl1 = lrl1.fit(Xtrain, Ytrain)
    l1.append(accuracy_score(lrl1.predict(Xtrain),Ytrain))
    l1test.append(accuracy_score(lrl1.predict(Xtest),Ytest))
graph = [l1,l1test]
color = ['green','lightgreen']
label = ['L1','L1test']

plt.figure(figsize=(6,6))
for i in range(len(graph)):
    plt.plot(np.linspace(0.05,1,19),graph[i],color[i],label=label[i])
plt.legend(loc=4)
plt.show()
```

Figure 10. Code of Parameter Curve for C

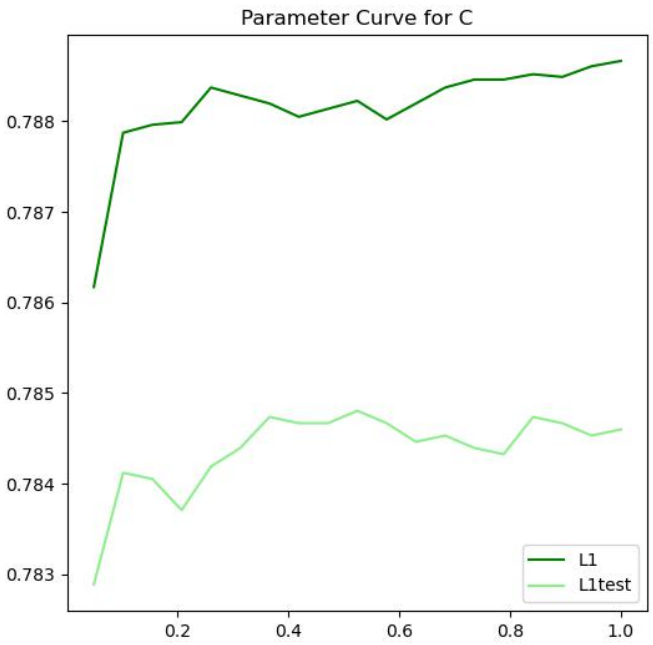

Figure 11. Parameter Curve for C

```
The Decision Tree model score on test: 0.7737736366127707
The Decision Tree model score on train: 0.7920129202760241
The accuracy of Logistic regression train model is:
0.7877257377771253
The accuracy of Logistic regression test dataset is:
0.7856947108796931
```

Figure 12. Two Models Results

GUI

Basic GUI construction was built with two button on the top. One is EDA and another is ML model (Machine learning model). EDA included three parts which were target distribution, feature vs target, and correlation plot. ML model included four models built by team members. Different layout, like grid layout and vertical layout were used. Inside of these lists, check boxes, figures, buttons, text line were designed to show the result of models (Figure 13).

```
def MatchTimelinesFirst15():
    #::-------------------------------------------------
    # Loads the dataset MatchTimelinesFirst15.csv
    # Loads the dataset MatchTimelinesFirst15
    # Populates X,y that are used in the classes above
    #::-------------------------------------------------
    global df_f
    global X
    global y
    global features_list
    global class_names
    df_f = pd.read_csv('data_team06.csv')
    X= df_f.drop(labels="blue_win", axis=1)
    y= df_f['blue_win']
    features_list = ['blueGold', 'blueMinionsKilled', 'blueJungleMinionsKilled',
       'blueAvgLevel', 'redGold', 'redMinionsKilled', 'redJungleMinionsKilled',
       'redAvgLevel', 'blueChampKills', 'blueHeraldKills',
       'blueTowersDestroyed', 'redChampKills', 'redHeraldKills',
       'redTowersDestroyed']
    class_names = ['Blue Win', 'Red Win']


if __name__ == '__main__':
    MatchTimelinesFirst15()
    main()
```

Figure 13. Part Code of GUI

Summary and Conclusions

The scores of two models are around 0.78. At first, I tired a lot of way to scale features or set parameters in order to improve the performance of models. However, the scores were around 0.78. First, I was so confused. The data was so good, no noise, no much useless features, but the score couldn't be over 0.9 just as what the sample data set like the wine data set and the iris data set could be done. Then, team member Annie told me that maybe because this data set is only the first 15 minutes of the whole march and something would happen in the another 15 minutes, so the score of 0.78 was reasonable. Before tiring to improve the score of models, understanding what the data set can bring it also important. A realistic and reasonable expectation are really important. From this project, I am more familiar with the whole process of building a model. Data pro-processing and feature selection are more important step, which will decide the top level of final models. Parameters in each model have their logic based on the basic mathematics principles. Different models have their advantages and disadvantages to be used for different data sets. In the future, I will more focus on the principles of feature selection and each model by actual practices. The percentage of the code that I found from internet is 57.3%.

Reference

Lin, L.(2017). *League of Legends Match Outcome Prediction*. Retrieved from https://cs229.stanford.edu/proj2016/report/Lin-LeagueOfLegendsMatchOutcomePrediction-report.pdf.

Lincoln Costa Magalh˜aes. (2021). *Feature Analysis to League of Legends Victory Prediction on the Picks and Bans Phase*. Retrieved from https://ieee-cog.org/2021/assets/papers/paper_247.pdf.