

Machine intelligence CMP402B

Faculty of engineering

Cairo University

Machine Intelligence

Project Proposal

Team 15

Team Members

|  |  |  |
| --- | --- | --- |
| Name | Sec | BN |
| Islam Ahmed Mohamed | 1 | 12 |
| Mohamed Abobakr Gad | 2 | 14 |
| Mohamed Ibrahim Gab Allah | 2 | 13 |
| Omar Tarek Mohamed | 2 | 8 |

Team Member Contributions

1- Islam Ahmed

2- Mohammed Ibrahim

3- Omar Tarek

4- Mohamed AbuBakr

Problem

League of legends

League of Legends is a MOBA (multiplayer online battle arena) where 2 teams (blue and red) face off. There are 3 lanes, a jungle, and 5 roles. The goal is to take down the enemy Nexus to win the game.

The game has a competitive mode (ranked solo queue) where players are ranked according to their skill levels (iron, silver, ...). The average time for a league of legends game is 25 to 30 min (depends on the rank).

The dataset contains the first 10min. stats of approx. 10k ranked games (SOLO QUEUE) from a high ELO (DIAMOND I to MASTER). Players have roughly the same account level.

This data can help gain insights about the effect of the early game (first 10 min) on the final winner and develop strategies for winning.

[Dataset](https://www.kaggle.com/bobbyscience/league-of-legends-diamond-ranked-games-10-min)

League of legends

League of Legends is a MOBA (multiplayer online battle arena)

Evaluation Metrics

In each of the problems stated above data will be split as follows 80% for training and 20%

the test.

The evaluation metrics used will be accuracy and loss. we might also include f1 score or confusion matrix based on the type of problem chosen.

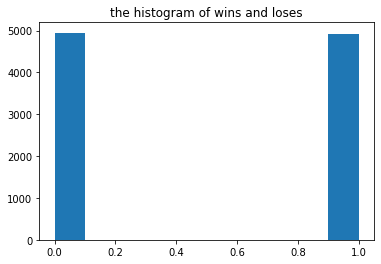
Experiment Setup

Preparing the Data

we began by splitting the data into 80% for training and validation and 20% for final testing of the models and each of the train.csv and test.csv files were placed in separate folders named train and test consecutively.

Preprocessing and Analysing the data

**1. Debiasing of data**

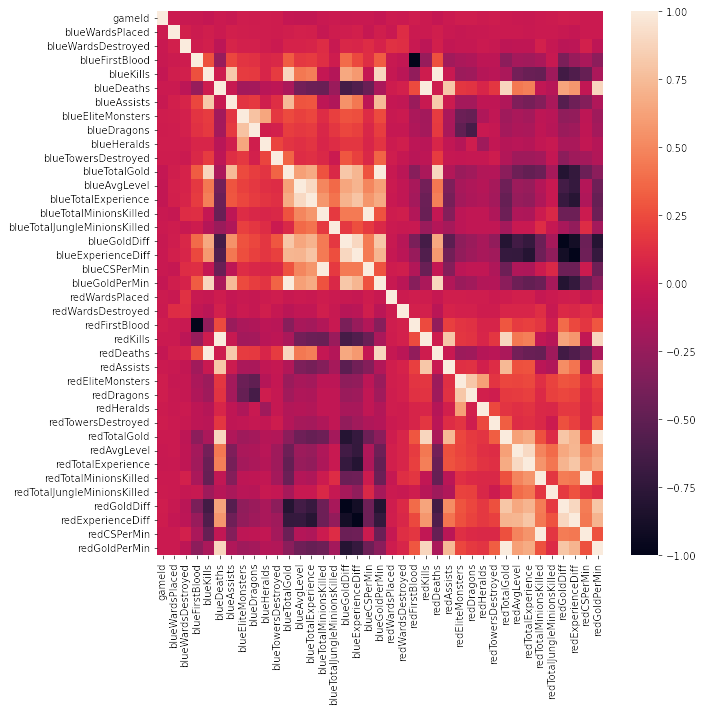


as show in the histogram above data seem to be balanced interms of the number of observations in each class so no debiasing techniques are needed.

**2. removing correlated features**

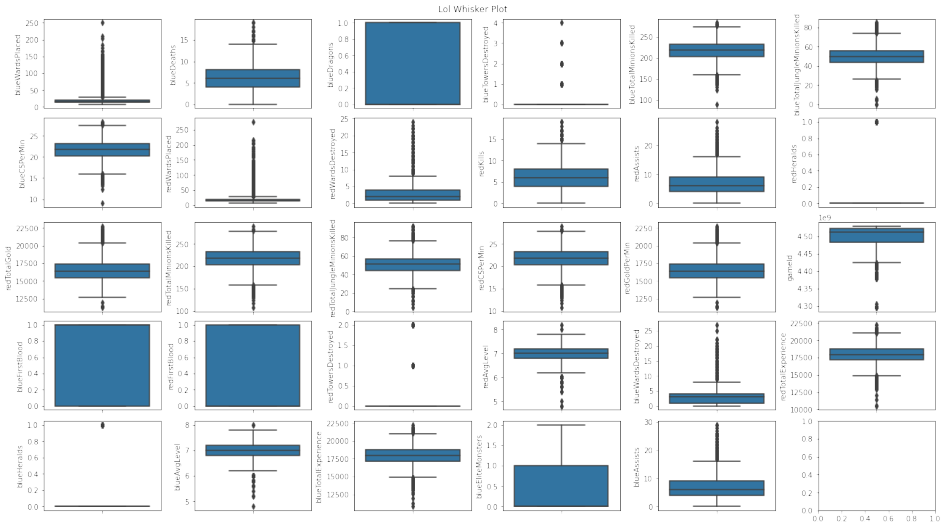
I started by drawing the correlation matrix among the features of the data which are 40 features.

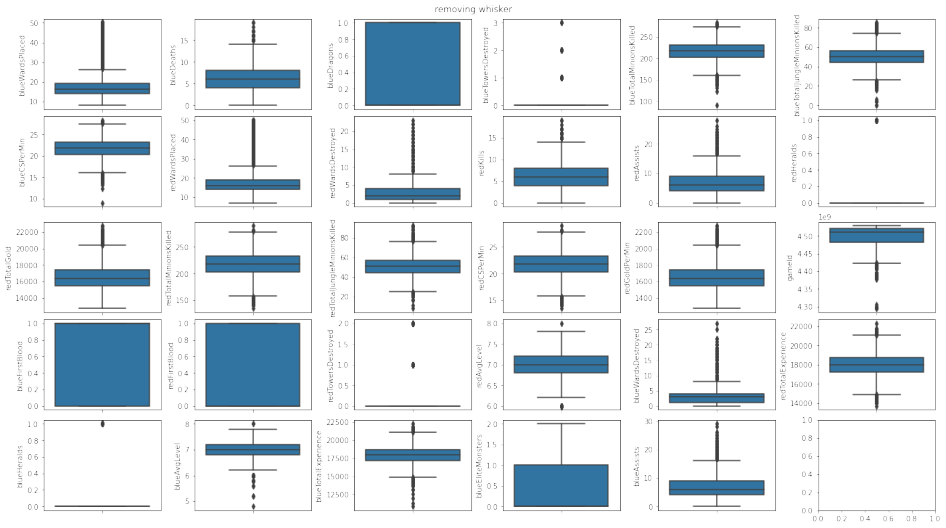
From the correlation matrix I could see that some feature had very high correlation with almost all of the other features these I decided to remove to enhance the accuracy of the model by decreasing its complexity.

Figure 1: correlation matrix

**3. removing Outliers**

Shown below is our box and whisker plots which shows the distribution of the features in our dataset before and after removing some outliers that were marked by the box and whisker plot. Actually removing outliers didn’t increase the accuracy a lot that’s why we only removed the outliers for only blueWardsPlaced, redWardsPlaced, and redAvgLevel features of our dataset.

Figure 2: *box and whisker plot of data after removing correlated features*

Figure 3: box and whisker plot after removing outliers

Models and Hyper Parameter Tuning

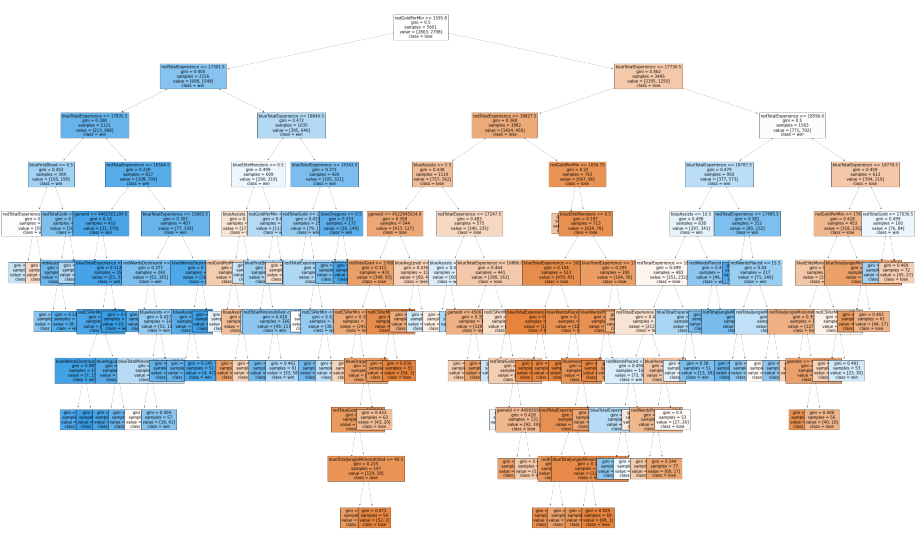
before testing on the models the 80% training data we had were further divided into 20% for validation and 80% for training.

**1. Decision Tree**

for the decision tree model show in the figure below we didn’t need to scale the input features since the decision don’t require feature normalization unlike the other models, so we passed the data as is from the output of the preprocessing stage

we tried to do hyper parameter tuning on the decision tree using an automatic method using the RandomizedSearchCV which iterates on the proposed parameters which we proposed after studying different parameters used by different people implementing similar algorithm.

After iterating on a number of parameters for the decision tree we found that the best parameters for the decision tree are {'min\_samples\_split': 50, 'min\_samples\_leaf': 50, 'max\_depth': 12, 'criterion': 'gini'}

Figure 4: decision tree generatted from the data

**2. Logistic Regression**

for the logistic regression we tried a lot of tuning to it and found that the best parameters for logistic regression are {'C': 2.0, 'penalty': 'l2'} where C here is the inverse of lambda (the regularization strength inverse

**3. Adaboost Classifier**

TODO

**4. SVM**

TODO

Results

accuarcy results before removing outliers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metric \ Model | Decision Tree | Logistic Regression | Adaboost | SVM |
| Validation Accuracy | 0.7046173308 | 0.7273877292 | 0.7272267206477733 | 0.7223276407 |
| Test Accuracy | 0.69838056680 | 0.7338056680 | 0.71157495256 | 0.7186234817 |

accuarcy results after removing outliers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metric \ Model | Decision Tree | Logistic Regression | Adaboost | SVM |
| Validation Accuracy | 0.686652391 | 0.7323340471 | 0.729478943611706 | 0.7323340471 |
| Test Accuracy | 0.6842105263 | 0.7343117408 | 0.7211538461538461 | 0.7277327935 |

as seen from the accuracy result removing of outliers has improved a little our accuracy.

Also as seen from the different accuracies we had with different classifiers we can say that the best classifier in terms of test accuracy is the logistic regression one which seems to be the best fit for this problem.

Also as seen from the accuracy the difference between the test and validation accuracy for each classifier isn’t too much which shows that we succeeded in generalizing our models and this actually the product of tuning their parameters and adding regularization.

Conclusion

after reviewing the accuaracies and taking into consideration the generalization and