Open data analysis to predict the results of the League of Legends World Championship

Josep Castell, Sergi Laut & Daniel Salgado

Master's degree in Modelling for Science and Engineering
Research and Innovation, 2017/2018.

Abstract

League of Legends (LoL) is a MOBA which has become the online computer game with more active users in the world. In League of Legends, players assume the role of an unseen "summoner" that controls a "champion" with unique abilities and battle against a team of other players. Each team usually consists of five players and the main goal is to destroy the opposing team's "nexus", a structure which lies at the heart of a base protected by defensive structures ("turrets") situated along the three streets or lanes that constitute the map. There are currently 138 champions in League of Legends (by October 2017) which are classified in different types or classes which generally determine what part of the map the champions gravitates towards during the early game. There is the top-lane player, the mid-lane player, two bot-lane players which play together, and the player who heads to the "jungle", the areas of the map that lie between the three mentioned lanes.

This game also has the most solid competitive structure between all the e-sports (the League of Legends competitive scene moves more money than the European basketball League), and the sports betting sites are starting to offer services in these fields.

The aim of this project is to predict results of the 2017 World Championship, the biggest event in the e-sports, using open data resources from different APIs and websites. Using data about the different professional players and teams that compete in the World Championship, and the data of the different champions that they can play; a logistic regression model to predict the results of the knockout stage (quarterfinals, semifinals and final) using data from the group and pre-qualifier stages is developed.

Universitat Autònoma de Barcelona

CENTRE DE RECERCA MATEMÀTICA

Contents

1 Overview

Popularity of e-sports is on the up and up, what started as a hobby is becoming a big industry which is starting to move a huge amount of money. 2017 League of Legends World's Championship final was seen by 75 million people, and it filled the Olympic stadium of Beijing known as the "Bird Nest" (80,000 people). E-sports in Europe are already moving more money than the Euroleague of basketball. It already exists a solid base of supporters, teams and players and at the same time other industries are taking advantage of this new trend.

News paper Marca recently started a section on e-sports in its on-line version; the first TV channel devoted to e-sports has already been launched in some Nordic and Baltic countries (e-sports TV from the company ESL); lots of companies are already selling merchandise from the different e-sport's stars and finally, sports betting houses are starting to accept bets in e-sports.

The aim of this project is to build a simple model that can predict the result of a match in e-sports better than humans and using open data. We focus in the MOBA League of Legends, the game with more active players in the world, and specifically in the League of Legends World's Championship, the biggest e-sport event nowadays. We will try to predict the knockout stage of the tournament (quarterfinals, semifinals and the final) using the data from the previously played games (region-qualifier stage and group stage). Now we briefly explain how a match of League of Legends works and the major facts that must be taken into account (in our opinion) in order to build a model.

League of Legends is a game were two teams of 5 players face each other in a field called the "Summoner's Rift" using a champion [1]. Each team has a base and the team who breaks the enemy's base wins the game. Since there are 5 players playing in each team, there are more than a hundred champions to pick, there exist a lot of non-deterministic in-game variables that could determine the final result of a game, and the professional games data is quite scarce; it seems very difficult to make a model that only uses data available before a match starts given the complexity of the game.

However, each player plays a specific role: Top-laner, mid-laner, adc, support or jungle. Each role has its specific function which determines a subset of champions that can be played in that role. Then, despite there are more than a hundred different champions, each one can generally only play up to two roles, which reduces the amount of champions that each player can chose (professional players always play the same role).

Moreover, there is another factor that reduces the complexity of the game: the metagame. Metagame is a term widely used in e-sports and computer games in general, and it is the more effective strategy of playing a certain game. In the case of League of Legends, it means that some champions are more powerful than others. Every two weeks, Riot, the company that owns League of Legends, adds a patch (game update) that tries to balance the power between their champions. However, having more than a hundred champions balanced between them is almost impossible, and in every match some champions are more powerful than others, and some roles become more relevant than others. Since professional players use to only play champions that are in the metagame, this reduces even more the amount of different champions played.

These are the main reasons that motivate us to model professional League of Legends matches, because despite there is less data available than on the amateur games, the matches are less complex in some sense. Our purpose is to collect data to estimate the skill of each player individually, the

[1] When we say player we refer to the person who is physically playing, not the character that he is using. Players are physical persons, champions the virtual characters they use to play

power of each champion in the patch that the World Championship is held (patch 7.18), the team to which each player belongs to, and in which side of the field a team is playing. With all this variables we will do a logistic regression model, and we will try to predict the outcome of the games of the knockout stage, assuming that we already know which champions will be picked by each player just before the match starts [2]. Afterwards, our purpose is to compare the prediction accuracy of our model with the mean of all the humans that participated in the "pick'em", a League of Legend's World Championship pool where almost all the League of Legends players participated. We will see that the initial model is not satisfactory enough, and for this reason we will consider some improvements and modifications.

In short, Section 3 is devoted to explain the data sources and tools that have been used to develop, implement and analyse the base model that is explained in Section 2 and which improvements are presented and analysed in Section 4. Section 5 summarizes the main results of our study. Section 6 is devoted to summarize legal and ethical issues about the sources we have made use of, and finally in Section 7 we discuss limitations and possible extensions and alternatives of our models.

The GitHub repository containing all code and data files used can be found here: `https://github.com/dsalgador/LeagueOfLegends_Project`. During the report we are going to be citing the names of the files.

2 Problem statement and mathematical modelling

The League of Legends Championship is held by 24 teams and it is divided in 3 stages:

- **Pre-qualifier stage:** 12 teams are directly qualified for the following stage, depending on the performance on the regular season. The other 12 teams compete between them for the remaining 4 spots for the next stage.
- **Group stage:** 16 teams compete in this stage. They are split in 4 groups, and they play against each member of its group twice. The 2 teams with more wins in the group classify for the next stage
- **Knockout stage:** In this stage the quarterfinals, semifinals and the final take place. Teams face each other following the "best of 5 criteria", the loser is eliminated and the winner classifies for the next round.

Our goal is to develop a mathematical model able to predict the results of the quarterfinals, semifinals and the final stages of the 2017 World Championship by means of a model that requires information available strictly before the knockout stage of the tournament starts.

The model we propose consists of a binary logistic regression, since it is a technique that is well suited for examining the relationship between a categorical response variable Y, such as it is "winning" ($Y = 1$) or "losing" ($Y = 0$) a particular match, and one or more categorical or continuous predictor variables X_1,..., X_n, [1, 2, 3].

The magnitude of interest is the probability $p = P(Y = 1)$ (resp. $1 - p = P(Y = 0)$) that a team wins (resp. loses) a particular match, and the logistic regression model is written as

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n, \tag{2.1}$$

[2] Since it is possible to bet just after the players have picked and before the match starts, it should not be a problem to consider that this information is available to us.

where β_i are constant coefficients to be determined. Then, assuming that these coefficients have been determined numerically using predefined algorithms that we do not discuss here, and given some observational data $(x_1, ..., x_n)$ of the predictor variables of a particular match and a particular team of this match, the predicted probability that the selected team wins is given by:

$$p = \frac{1}{1 + e^{-z}}, \text{ where } z = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n. \tag{2.2}$$

The predictor variables we consider for our base model are the following:

<u>Categorical variables</u>

A k-levels categorical variable X_i leads to $k-1$ predictor variables $X_{i,1}, ..., X_{i,k-1}$ for the model equation (2.1) (see for instance [4]), and in our case we consider

a) A N-levels categorical variable X_1 representing **the team**, where N is the total number of teams that participate in the classification stage (N = 24 in 2017). One of the teams (lets call it T_0) is chosen as a kind of "origin" and then we have $N-1$ predictor Variables $X_{1,1}, X_{1,2}, ..., X_{1,N-1}$ associated to the other teams T_1,..., T_{N-1}, defined as follows:

$$X_{1,i} = \begin{cases} 1 & \text{if the team in consideration is } T_i \\ 0 & \text{otherwise} \end{cases}$$

These variables are supposed to account for the "power" of the team. For instance, it is well-known that Korean teams usually are pretty much better than European or American teams.

b) A two levels variable X_2: The **side of the map** where the team in consideration plays, denoted by BLUE or RED, and mathematically modelled as $X_2 = 0$ if BLUE and $X_2 = 1$ if RED. This variable is supposed to account for the possible advantage of playing in one part of the map or the other.

<u>Continuous variables</u>

Our base model starts by considering a total of 10 continuous variables:

c) Five continuous predictor variables $X_{3,j}$, $j = 1, ..., 5$ (referred to Top, Jungler, Mid, Adc and Support), whose values are between 0 and 1, and account for the **average matchup win rate of a champion** that is played in a given position by the team in consideration and against a given enemy champion from the same position [3]. These variables are supposed to account for the *metagame* of the patch where the tournament is being played, i.e. to account for the fact that there are champions that are more over-powerful than others.

d) Another five continuous predictor variables $X_{4,j}$, $j = 1, ..., 5$ (referred to Top, Jungler, Mid, Adc and Support), whose values are also between 0 and 1, and account for the **win rate that a given player has with the champion that is playing with** during the match in consideration. In this case, these variables are supposed to account for the skill of a player with a given champion.

[3] The average is done among all ranked matches that have been played for the best 5% players during the patch of interest.

With all these variables already defined, we can write our binary logistic regression model equation as follows:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \sum_{i=1}^{N-1} \beta_{1,i} X_{1,i} + \beta_2 X_2 + \sum_{j=3}^{4} \sum_{i=1}^{5} \beta_{j,i} X_{j,i}, \tag{2.3}$$

where p is the probability that the team in consideration wins the match. Similarly, we can consider an expression for finding p such as in equation (2.2).

In order to compute values for the β coefficients we need *train data* coming from all the matches played before quarterfinals in a well structured format and containing values for all the predictor variables that appear in (2.3). The following table illustrates how the train data of a single match should look like:

```
 result        team  side topwr_champ jngwr_champ midwr_champ adcwr_champ
1      1     Team WE  Blue         0.5         0.5        0.49         0.5
2      0 Lyon Gaming  Red          0.5         0.5        0.51         0.5
  supwr_champ topwr_player jngwr_player midwr_player adcwr_player
1        0.53         0.58         0.60         0.66         0.62
2        0.47         0.55         0.78         0.50         0.00
  supwr_player
1          1.0
2          0.6
```

Figure 2.1

In the above example, the teams of the considered match are Team WE (winner) and Lyon Gaming (loser).

Therefore, for each match we need two rows of data, one for each team, where the fields (columns) are: the result of the match with respect to the team (1 if wins, 0 if loses), the name of the team, the matchup champion win rate for each position (top, mid, jungle, adc and support, respectively), and the player win rate with the specific champion that is playing (again for each position). In principle this model is open to be extended, in the sense that we could add more predictor variables; for instance, we would want to consider the KDA (which is the number of kills plus assistances divided by the number of deaths) of each player playing a particular champion, so that we would have five additional predictor variables, lets call them $X_{5,k}$, $k = 1, ..., 5$.

In the next Section 3 we explain how we have obtained and manipulated data from different sources so that we have been able to have a well-structured train data set containing the information from all matches played before the knockout stage of the tournament.

Similarly the *test data* is built using the necessary data coming from the quarterfinal, semifinal and final matches.

3 Data life-cycle

We will use data from 3 different sources, manipulate it to build our model, and deposit our project and edited data in a repository in GitHub, under a share-alike license in order to allow any other user to download it, re-edit it and redistribute it in the future.

Mainly, need the data about: the record of wins and loses with each champ for every professional player in competitive games of the 2016/2017 season, the win rates of each champion in the current metagame (patch 7.18), and the games played in every stage of the World Championship.

In Figure 3.1 we display a simple scheme to represent the data life-cycle and in the incoming subsections we are going to explain how we obtained, cleaned, manipulated and analysed the data in more detail.

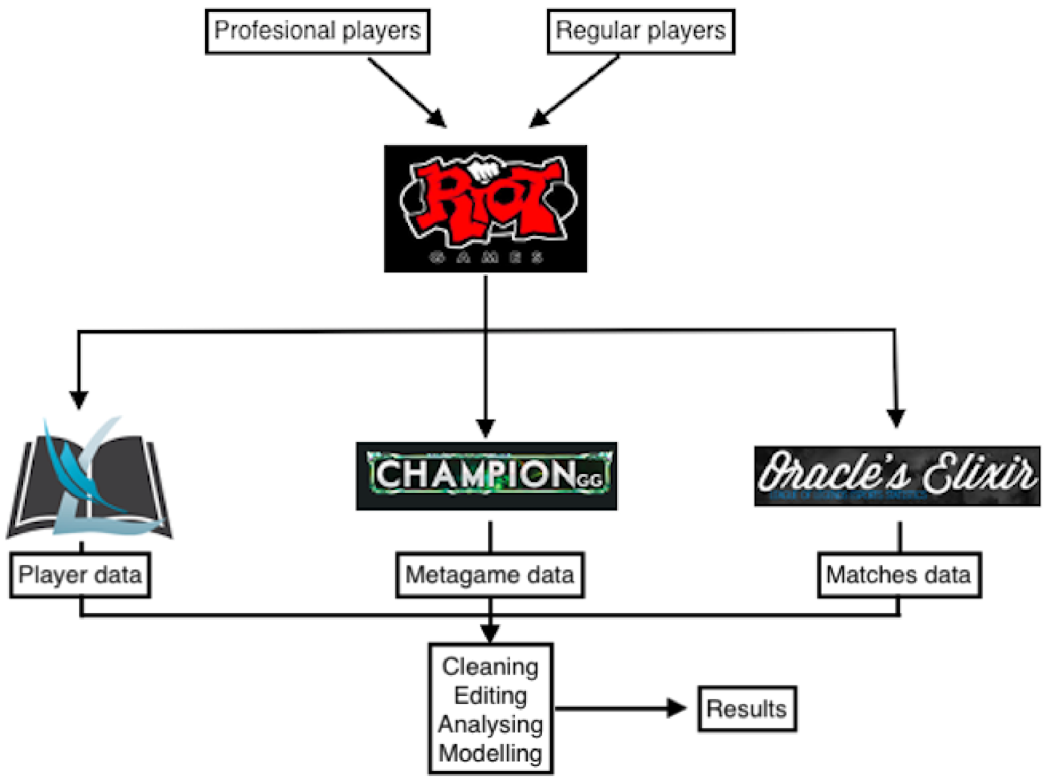


Figure 3.1: Professional and regular players generates data that is collected by Riot Games, the owner of League of Legends. Then using Riot's API, Champion.gg, Oracle's Elixir and Leaguepedia collect some of its data and they manipulate it. We use the data given by these places to develop our model

All the data that we will use is generated by players; hence, it is initially stored in Riot's servers. However, we do not need to access to all this data directly from Riot's API, since there are people that have already done that, and they have developed some data sources that are more suitable for our interests. In the next Section 3.1 we proceed to explain which data we extracted from each site, and how we have done it. Section 3.2 is devoted to explain how we have cleaned, manipulated and restructured the data sources that will allow us to implement the theoretical model presented in Section 2. Finally, in Section 3.3 we analyse and present the basic statistical tools and methods that will be useful to analyse variants of the initial model.

3.1 Data collection

3.1.1 Player data: scraping a web page with python

From lol.gamepedia.com [7] we have obtained players_info.json. This file contains data about the performance of the different players in their current season. For every player we extract from the web a list with the champions he has used, the games he has won and lost with them and their KDA performance (Kill, Death, Assist).

In order to do so, we have used Scrapy as the framework for extracting the information. We have developed spider_xpath.py, a simple spider that is run throughout all the websites of the players in the World Championship and gets the data from a table by accessing it through its Xpath. When running the spider through the terminal we choose to create a json file with all the extracted data. The Xpath for the different extracted elements is obtained by inspecting the webpage.

3.1.2 Champion data: querying an API

Static champion data From ddragon.leagueoflegends.com we have obtained a champ_info.json file containing all the static data from any of the champions released before patch 7.19 (see [5]). In this case the patch is not that important since we are using this data just to have a list of the champion names and ids (which are static, specific and non-consecutive integers that characterize a particular champion). Thus, there is a bijective relation between champion names and ids. This is important for when we manipulate the champion_matchups_patch7_21.json data set where champions are identified with the id number and not by the name (string).

Dynamic champion (and player related) data To obtain the average matchup win rates of each champion we have made use of the champion.gg API (see [6]). The particular query URL we have used to obtain a champion_matchups_patch7_21.json file containing a lot of champion matchup information is:

http://api.champion.gg/v2/champions?&champData=matchups&limit=200&api_key=<api_key>,

where the field <api_key> has to be replaced by a valid API key that one has to ask for by registering in the API web page.

This project was started when the current game patch was the 7.21. At champion.gg the data is updated to the current patch (almost) every two weeks, and the data from the previous patches is replaced by the current one. In order to be strictly according to our model, the data we would need would be that from the patch 7.18. Since it was already deleted, we will instead use that from patch 7.21 [4].

3.1.3 Match data: using already processed open data

A very important data source that we need to train and test our logistic regression model is that consisting of data from each match played in the World Championship. We have obtained this data set from [8] in .xlsx format and we have then converted it to .csv format via a online converter [9].

[4]We have made contact with the champion.gg owners and they have told us that one can only access to data from the current patch thorough their API.

The corresponding file containing this data is called 2017_WorldsMatchData_all.csv. The original data can be publicly accessed through lolesports.com, [8].

3.2 Data cleaning and manipulation

In this section we explain how we have manipulated the raw data files already mentioned in Section 3.1, in order to build the train and test datasets that will allow us to define and check our theoretical model numerically. All the code files used to do so are written in R language and can be found in the GitHub repository we already mentioned in Section 1.

3.2.1 Player data

Once the file players_info.json is read in R, we convert it to a data table format which at the end, after some simple data manipulation, consists of four columns and about 2700 rows. The full code can be found in the file PlayerChampionWinrates.R. The second column consists of the player names repeated as many times as the number of different champions for which there is information related to the player in consideration. The first column contains champion names, the third contains win rates of a given player with a given champion, and similarly in the fourth column there is the KDA.

Then using this data table we have coded a function winrate_player(champ_name, player_name) that given a champion name and a player name returns the win rate of this player with this champion. Similarly, using this auxiliary function we have built a function winrate_teamplayers(team_champs, team_players) that given an array team_champs containing the champion names (and respecting the position order: Top, Jungler, Mid, Adc and Support) and an array team_players containing the player names of that team (again respecting the position order), it returns an array of win rates associated to each player and the associated champion that he is playing.

```
#Example
Input:
team_champs = c("Trundle", "Sejuani", "Orianna", "Twitch",  "Janna")
team_players = c("Huni",  "Blank", "Faker", "Bang",  "Wolf")
round(winrate_teamplayers(team_champs, team_players),2)

Output:
[1] 0.67 0.60 0.66 0.79 0.50
```

If there is some combination of player name and champion for which there is no available data, we have decided to return a neutral win rate value, i.e. 0.5.

With this code implementations we are now able to obtain values for the predictor variables $X_{3,j}$, $j = 1, ..., 5$, assuming that we previously know the champions that are played by each player in a given match.

3.2.2 Champion data

Using the champ_info.json and champion_matchups_patch7_21.json data files, we have coded a script in R (the MatchupChampionWinrates.R file) where for each position (Top, Jungler, Mid, Adc and Support) it is build a matrix $P = (p_{id_i,id_j})$ that has as column and row labels the ids of the champions that are played in that position, and such that the component P_{id_i,id_j} contains the average win rate of the champion with $id = id_i$ when it plays against the champion with $id = id_j$. In the cases where $id_i = id_j$ (for instance if the two adc players are playing the same champion), the win rate is set to 0.5. However, in official matches like the ones in the Worlds championship it is not possible that two players, regardless of the team, play the same champion.

For convenience we have built two functions that allow us to convert champion names to ids and ids to champion names (`id2name(ID,TABLE)`, `name2id(ID,TABLE)`, where `TABLE` is the matrix P of the corresponding position)

Once we have an "average matchup win rate" matrix for each of the five positions, we have built a function `winrate_pair(champion1, champion2, POSITION)` that returns the average matchup win rate of `champion1` when it plays against `champion2` in the position `POSITION`. Then, by using this auxiliary function, we have coded another function `winrate_match(team1, team2)` that given the champion names of the champions played by `team1` (an array of champion names, respecting the position order: Top, Jungler, Mid, Adc and Support) and the associated champion names to `team2` it returns an array of average matchup win rates for each of the five positions and in the order that we have been considering so far.

```
#Example:
Input:
team1= c("Tryndamere","Lee Sin", "Yasuo", "Miss Fortune", "Leona")
team2 = c("Dr. Mundo", "Xin Zhao", "Taliyah", "Lucian", "Janna" )
winrate_match(team1, team2)

Output:
[1] 0.4971 0.4331 0.4979 0.5519 0.4895
```

If there is some combination of champion names and positions for which there is no available data, we have decided to return a neutral win rate value, i.e. 0.5.

To sum up, with this code implementations we are now able to obtain values for the predictor variables $X_{3,j}$, $j = 1, ..., 5$, assuming that we previously know the champions that are played in each team and in a given match.

3.2.3 Match data: building the train and test datasets

Using the `2017_WorldsMatchData_all.csv` file and the functions and data sets explained in the last two sections we have implemented a script in R (`Match_TrainAndTest_data.R`) that, given a set of matches from `2017_WorldsMatchData_all.csv`, generates a table whose columns and rows gather information about each match as we showed in Figure 2.1. That is, for each match in the set of matches of interest, two rows (as seen in Figure 2.1 for one match) are generated and included in the final table. In this way we have created the train data set file `train_data.csv` containing information of matches played before the quarterfinals, and the test data set file `test_data.csv` containing the information of the matches that we want to predict using our model (i.e., those from quarterfinals, semifinals and the final).

3.3 Data analysis and statistical tools

In this section we familiarize with the base model we have developed formally in Section 2 and present the main statistical tools that will serve us to improve it. Otherwise noted, for statistical purposes (tests), we will work always with a significance level $\alpha = 0.05$ (i.e., with 95% confidence level).

3.3.1 Model fitting

First of all we have to use some `train` data (initially that from `train_data.csv`) to estimate the model coefficients β_i numerically. We can do this in `R` with the predefined `glm` function as follows:

`model <-glm(result ~ .,family=binomial(link='logit'),data=train),`

where `result` is the column of the data set `train` that contains 0 and 1 and represents wins and loses. At this point, `model` contains numerical estimations for each of the β_i coefficients.

It is interesting to consider confidence intervals for each of these coefficients. The following command allows us to create a table with a first column containing the estimated β_i values and two additional columns which are the left and the right bounds of a 95% confidence interval:

`round(cbind( coef(model), confint.default(model)) ,1)` (the output is found in Appendix A.1, listing 2).

We are dealing with a total of 30 β_i coefficients and it is quite probable that some of them are insignificant or inappropriate to obtain high prediction accuracies.

A useful way to rate the relevance of a predictor variable consists in performing a χ^2 ANOVA test:

`anova(model, test="Chisq")`

```
Model: binomial, link: logit
Response: result
Terms added sequentially (first to last)

             Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                           179     249.53
team         23   56.992       156     192.54 0.0001027 ***
side          1    1.053       155     191.49 0.3048675
topwr_champ   1    3.234       154     188.25 0.0721285 .
jngwr_champ   1    2.543       153     185.71 0.1107626
midwr_champ   1   10.664       152     175.05 0.0010923 **
adcwr_champ   1    1.028       151     174.02 0.3106769
supwr_champ   1    0.314       150     173.71 0.5750306
topwr_player  1   19.489       149     154.22 1.012e-05 ***
jngwr_player  1   13.724       148     140.49 0.0002117 ***
midwr_player  1   15.925       147     124.57 6.589e-05 ***
adcwr_player  1   13.828       146     110.74 0.0002003 ***
supwr_player  1   28.648       145      82.09 8.680e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Listing 1: Analysis of Deviance Table for the Base model (Model v1)

The difference between the null deviance and the residual deviance shows how our model is doing against the null model (a model with only the intercept). The wider this gap, the better. Analysing the table we can see the drop in deviance when adding each variable one at a time. Adding `team`, and the five "`positionwr_player`" predictors significantly reduces the residual deviance. The other variables seem to improve the model less, even though `midwr_champ` has a low p-value. A large p-value here indicates that the model without the variable explains more or less the same amount of variation, [3].

Another useful tool to asses the significance of predictor variables is the so called Wald test. "*A wald test is used to evaluate the statistical significance of each coefficient in the model and is calculated by taking the ratio of the square of the regression coefficient to the square of the standard error of the coefficient. The idea is to test the hypothesis that the coefficient of an independent variable in the model is significantly different from zero. If the test fails to reject the null hypothesis, this suggests that removing the variable from the model will not substantially harm the fit of that model*", [1].

If we test the overall effect of team names (i.e. the `team` predictor variable) executing the following code line `wald.test(b = coef(model), Sigma = vcov(model), Terms = 2:24)`, we obtain a p-value equal to $0.37 \gg 0.05$.

```
Wald test (teams):
----------
Chi-squared test:
X2 = 24.6, df = 23, P(> X2) = 0.37
```

Therefore, we do not reject the null hypothesis that the overall effect of the `team` predictor coefficients is not significant, so that we accept that the β coefficients of team names, as a whole, are statistically insignificant and equal to zero. Note that if we perform a Wald test for each team name individually, one can see that there are some teams that are statistically significant, but the majority are not[5] and, as a whole, we have seen that there is no statistical significance.

Similarly, by testing the `side` variable, `wald.test(b = coef(model), Sigma = vcov(model), Terms = 25)`, we conclude that the `side` of the map where a team is playing is not significant at all (note that the p-value is almost 1).

```
Wald test (side):
----------
Chi-squared test:
X2 = 1e-04, df = 1, P(> X2) = 0.99
```

About the the overall effect of the "`positionwr_champion`" predictors, we can say that it is quite significant, since `wald.test(b = coef(model), Sigma = vcov(model), Terms = 26:30)` gives

```
Wald test ("positionwr_champion"):
----------
Chi-squared test:
X2 = 22.2, df = 5, P(> X2) = 0.00049
```

but in fact the only significant predictor is `midwr_champion`, because `wald.test(b = coef(model), Sigma = vcov(model), Terms = c(26,27,29,30))` gives

[5] For the majority of team names 0 belongs to the 95% confidence interval of the associated β coefficient.

```
Wald test ("positionwr_champion" except midwr_champion):
----------
Chi-squared test:
X2 = 5.4, df = 4, P(> X2) = 0.25
```

having that $0.25 \gg 0.05 = \alpha$.

Finally if we test the overall effect of the "`positionwr_player`" predictors we see that it is statistically significant:

```
wald.test(b = coef(model), Sigma = vcov(model), Terms = 31:35)

Wald test ("positionwr_player"):
----------
Chi-squared test:
X2 = 22.2, df = 5, P(> X2) = 0.00049
```

Moreover, if we test each of these predictors individually, the p-values obtained are always smaller than $0.02 \leq 0.05$, which mean that each of these predictors are statistically significant.

3.3.2 Assessing the predictive ability of the model

Now we would like to see the behaviour of the model when predicting y_i (win or lose of a given match i) on a new set of `test` data. By setting the parameter type='response', R will output the estimated probabilities in the form of $P(y_i = 1|x)$, where $x = (x_1, ..., x_n)$ are the observed values of the predictor variables of the match. In fact, in our case, each two consecutive rows of the `test` data correspond to one match. Thus, for each match i, we have two output predicted probabilities $P\left(y_i^{(1)} = 1|x\right)$ and $P\left(y_i^{(2)} = 1|x\right)$, where the superscript indicates the team.

Our decision boundary must satisfy that $y_i^{(1)} = 1$ (resp. $y_i^{(2)} = 1$) if and only if $y_i^{(2)} = 0$ (resp. $y_i^{(1)} = 0$). Thus, for each pair of rows that correspond to the same match, we have to decide which team should have won according to the model predictions. To do so we consider the following criteria:

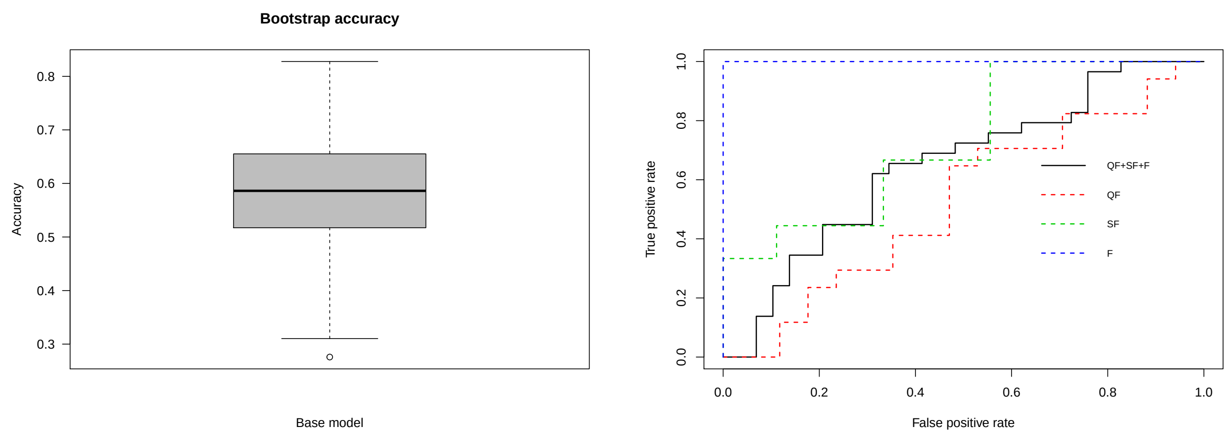
- If $P\left(y_i^{(1)} = 1|x\right) < P\left(y_i^{(2)} = 1|x\right)$, then $y_i^{(2)} = 1$, $y_i^{(1)} = 0$,
- otherwise $y_i^{(2)} = 0$, $y_i^{(1)} = 1$.

```
fitted.results <- predict(model,newdata=test,type='response')
num_results <- length(fitted.results)
for(i in seq(1,num_results, 2) ){
      fitted.results[i] <- ifelse(fitted.results[i]>fitted.results[i+1],1,0)
      fitted.results[i+1] <- abs(fitted.results[i]-1)
    }
```

Then we can compute the accuracy of the model by comparing the predicted results with the real results:

```
misClasificError <- mean(fitted.results != test$result) #proportion of failures
print(paste('Accuracy',1-misClasificError)) #proportion of right predictions
```

If we use as `test` data set the whole file `test_data.csv`, the model accuracy is: 0.586. In order to obtain some confidence interval for the accuracy, we can perform a bootstrap procedure (see `Bootstraping_accuracy.R`) where one takes a different `test` data at each bootstrap iteration obtained by sampling the original `test` data with replacement. In our case we have to be careful when sampling, since the data observations (rows) are paired.



(a) (b)

Figure 3.2: (a) Box plot of the bootstrap sample of model accuracies (1000 iterations). (b) ROC curves for the base model with respect to the whole knockout stage (black line), quarterfinals (red dashed line), semifinals (green dashed line) and the finals (blue dashed line).

After bootstrapping 1000 times, using the quantile method we obtain the following 95% confidence interval: $[0.41, 0.76]$. In Figure 3.2a we have a box plot of the bootstrap sample of accuracies.

Now we would like to compare the accuracy of our model with the average accuracy of the Worlds pick'em performance made by real League of Legends players. The pick'em data set has been obtained by copy-pasting the table that is found in the knockout stage window from [11]. By doing some simple calculations (which can be found at the beginning of the file `BinaryLogitRegression.R`) we obtain that the mean accuracy of all the people that have taken the knockout stage pick'em is: 0.558. Since $0.558 \in [0.41, 0.76]$, we cannot reject the null hypothesis that the model and the mean pick'em accuracies are equal. In other words, there are statistical evidences of the fact that the average prediction ability of humans and that of our model are equal. For this reason, in Section 4 we try to improve our base model by eliminating insignificant or nonsensed predictor variables.

As a last step, we are going to plot the ROC (receiving operating characteristic) curve and calculate the AUC (area under that curve) which are typical performance measurements for a binary classifier, [3]. The ROC is a curve is constructed by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. That is, *using the proportion of positive data points that are correctly considered as positive and the proportion of negative data points that are mistakenly considered as positive, we generate a graphic that shows the trade off between the rate at which you can correctly predict something with the rate of incorrectly predicting something*, [1].

The AUC metric ranges from 0.5 to 1. As a rule of thumb, a model with good predictive ability should have an AUC closer to 1 (1 is ideal) than to 0.5, [3]. Values above 0.8 indicate that the model does a good job in discriminating between two categories which comprise our target variable, [1]. In Figure 3.2b we have plotted the ROC curves for the whole knockout stage pick'em (black line), together with the ROC curves for the separate phases QF (quarterfinals), SF (semifinals) and F (finals). We can see that the ROC curve corresponding to the finals is the unity (the ideal case), which means that the model behaves perfectly when predicting the results of the final's matches.

4 Model variations and results

In the last section 3.3 we analysed the numerical results obtained for our base model and introduced the basic tools to asses the goodness of a logistic regression model and its predictor variables. We saw that the base model is quite unsatisfactory, and for this reason we will now consider some variations of the current model (we call it Model v1).

The first change we consider is to remove the `team` predictor variable from all computations (i.e., from the `train` data used to build the model), since we saw that the overall effect of it is not significant by means of a Wald test. We call this variation Model v2. The fact that the overall effect of the `team` variable is not significant, does not mean that if we eliminate it from he model, then the predictability (for example accuracy) will not change at all. In fact, there were some team names whose β coefficients were quite significant. However, since the number of teams is quite large (24) compared with the total amount of match data available (90 matches, which is quite small), it is probable that the effect of adding the `team` variable to the model harms its predictability. We will see that this is what in fact happens.

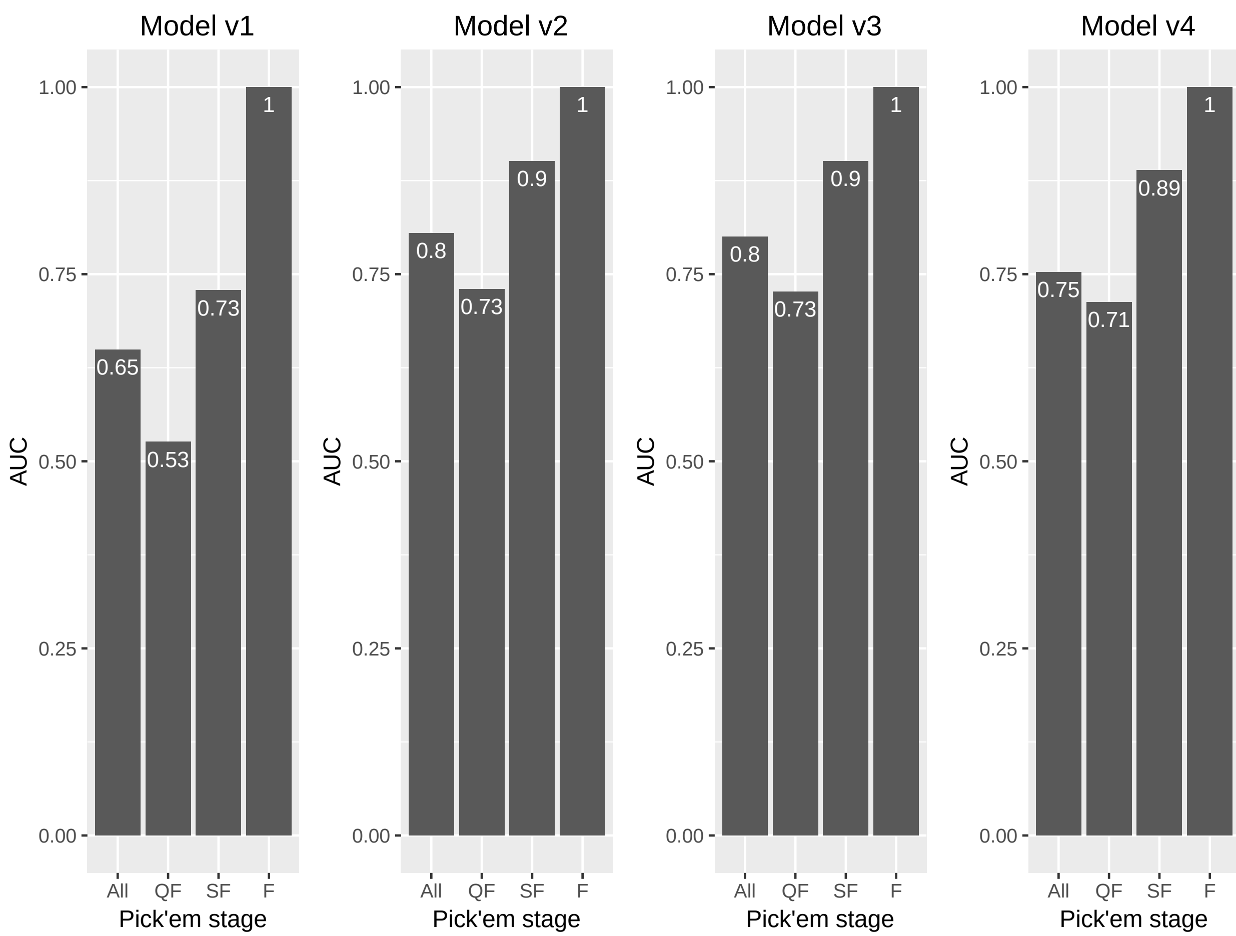


Figure 4.1: AUC values for the four model variations (or versions). Overall performance of the model (All), quarterfinals (QF), semifinals (SF) and final (F).

The second change we consider is to remove the `side` predictor variable from the model. The corresponding Wald test we discussed in section 3.3.1 told us that this variable was very insignificant (the p-value was almost 1). We call this variation Model v3. We will quantitatively check the fact that the importance of the `side` variable in Model v2 is very low (later in Figure 4.5).

Finally, we have considered to remove the champion matchup win rates (i.e., the "`positionwr_champion`" predictor variables) for two main reasons:

1) First, we can see (see the outputs from Appendix A.1) that there are statistical evidences of the fact that the β coefficients associated to some of the champion matchup win rate variables (in our case `jngwr_champion` and `topwr_champion`) are negative. If we remember equation (2.2), this would imply that the higher the matchup champion win rate for the top or jungle position, the smaller the probability of winning the match. This clearly makes no sense for our model so we decide to remove at least these two variables from the model.

2) Second, since the champion matchup win rate dataset corresponds to the game patch 7.21, with a delay of about 6 weeks with respect to the Worlds patch (7.18), and we are aware of the fact that the metagame have changed quite since then, we consider a model where we also remove the other three variables (`midwr_champion`, `adcwr_champion`, `supwr_champion`), although they are (statistically) positive.

We call it Model v4.

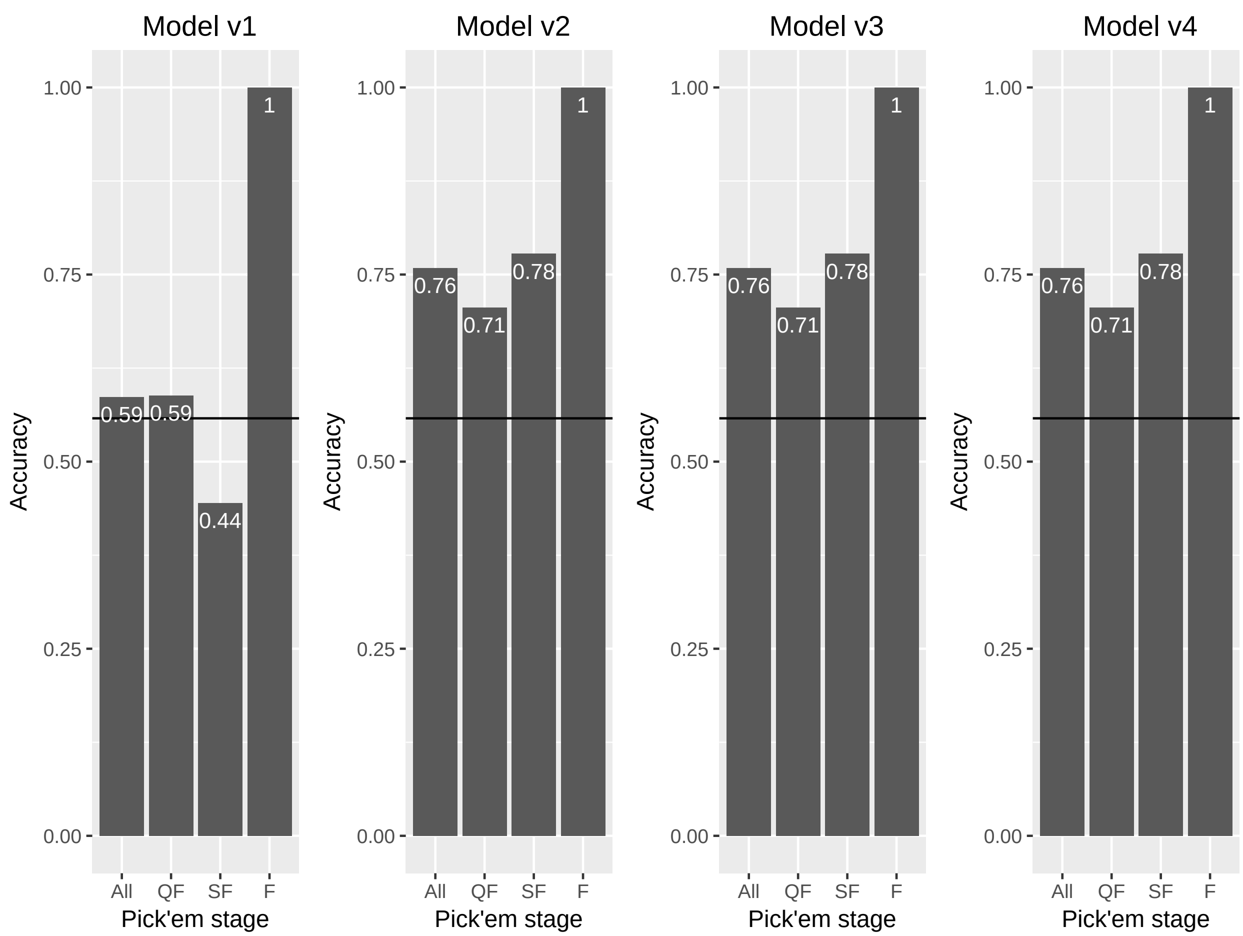


Figure 4.2: Model accuracies for the four model variations (or versions). Overall accuracy of the model (All), quarterfinals (QF), semifinals (SF) and final (F). The horizontal black line represents the mean accuracy threshold given by the mean pick'em performance of all participants.

In Figure 4.1 we have displayed a bar chart for each model to represent the AUC values corresponding to the overall knockout stage (All = QF+SF+F), the quarterfinals (QF) stage, the semifinals (SF) stage and the final (F). Similarly, in Figure 4.2 there are the corresponding bar charts for the accuracy of the model when predicting the results of every stage we already mentioned.

The most significant result we can extract from these bar charts is that when we remove the `team` variable from the base model (to obtain Model v2), the AUC of every single stage and the overall stage increases between 0.15 and 0.2 units. Similarly, the prediction accuracy of the base model for the overall stage, the quarterfinals and the semifinals, increases significantly so that they surpass the "mean pick'em accuracy" (black horizontal line) threshold in about 0.2 units.

When the `side` variable is removed (Model v3), the change in the AUC values is negligible if we only consider the first two significant digits; this is consistent with the fact that a Wald test for this predictor had a p-value very close to 1. The same applies for the accuracy, where no changes are observed.

Similarly, if we remove from the model the predictor variables "`positionwr_champion`" (Model v4), the accuracy of the model with respect to each stage remains unchanged again, but the AUC for the overall stage, the quarterfinals and the semifinals decreases between 0.01 and 0.05 units.

In Figure 4.3 we show a table that contains means, variances and 95% confidence intervals for the overall accuracies of each model, obtained via a bootstrap procedure (1000 iterations) as we commented in Section 3.3.2. The information of this table can be understood graphically by looking at the box plot that is on the right. Again we confirm the fact that the important change that makes the base model improve significantly is removing the `team` predictor variable. The other changes that lead to Model v3 and Model v4 are almost equivalent to Model v2 with regard to accuracy.

Analysing the data from the table below, one can conclude that for models v2, v3 and v4, there are statistical evidences of the fact that their accuracies are strictly greater than the average accuracy of the players that participated in the "pick'em" for the knockout stage[6]. This is because 0.558 do not belong to the 95% confidence intervals associated to models v2, v3 and v4.

Bootstrap (mean and variance) accuracies and 95%
confidence intervals (quantile and standard methods)

	Mean	Variance	2.5%(q)	97.5%(q)	2.5%(s)	97.5%(s)
v1	0.590	0.00898	0.414	0.759	0.404	0.776
v2	0.762	0.00646	0.586	0.897	0.604	0.919
v3	0.759	0.00628	0.586	0.897	0.604	0.915
v4	0.756	0.00634	0.586	0.897	0.600	0.912

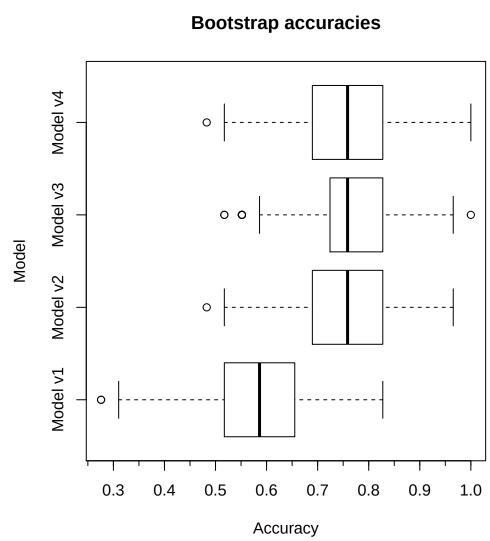


Figure 4.3: On the left, table of means, variances and confidence intervals for the bootstrap accuracy samples for each model. Two kind of confidence intervals are considered: using the quantile method (q) and using the standard method (s). On the right, we have a box plot associated to the bootstrap accuracy samples

[6] Remember that the value of this average accuracy was ≈ 0.558

As one may have already noticed, both the AUC and the accuracy with respect to the final stage (F) is equal to 1 regardless of the model variation we consider. Thus, from the very beginning, each of our model variants predict that Samsung Galaxy (SSG) won the three games played (3-0) against SK Telecom T1 (SKT). This must be considered a quite surprising and positive result, because since 2013, SKT has won three World Championships including the last two back to back.

"*They have never lost a best of five at Worlds and have far and away the most experienced and proven player of all time anchoring their mid lane. Until proven otherwise, it's likely best to assume that SKT will win whatever series they are playing. But this one won't be easy.*

The only other team to have won worlds in that time span is Samsung Galaxy, who have played better than SKT in the Knockout Stage of this tournament. Not only that, but Samsung are likely the best team in the world right now at reading their opponent's strengths and countering them, something SKT has already proven vulnerable to at Worlds 2017.

In the quarterfinals, Samsung took down Longzhu Gaming, a team widely considered to be the best in the world at the time, in a quick 3-0 [...]" [13].

Therefore, in principle both teams seemed to be equally stronger and candidates to win the 2017 Worlds Championship, but our model has been able to realize that SSG was, in fact, stronger. However, lets analyse in a bit more detail how are our models are distinguishing SKT and SSG.

In Figure 4.4 we have plotted, for each match played in the final stage, the difference between the predicted probability that SSG wins the match and the predicted probability that SKT wins the match, i.e., P(SSG wins) - P(SKT wins), as a function of the model variation (v1, v2, v3 and v4). This probability increment is a measure directly proportional to "how well is the model distinguishing the winner of the match".

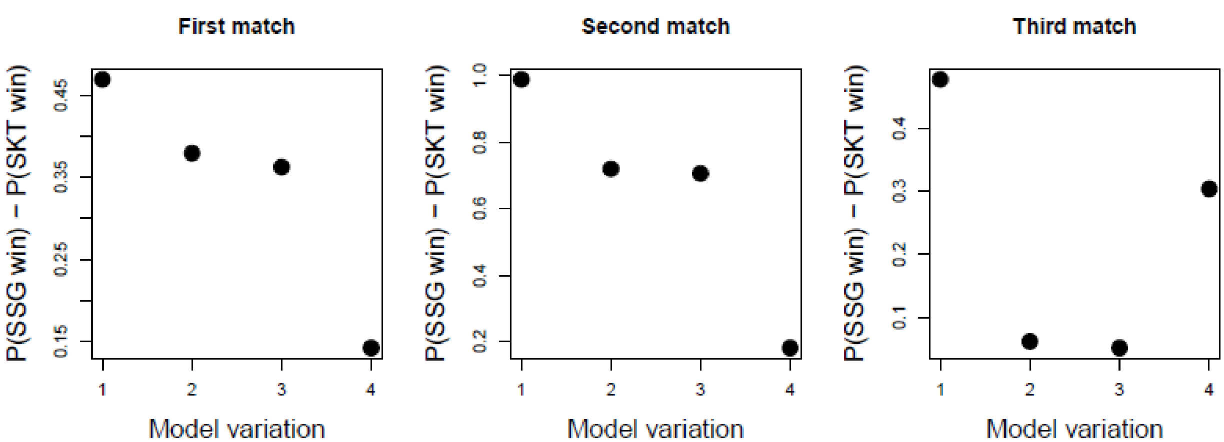


Figure 4.4

We can see that as the model version increase, or equivalently, as the number of model predictor variables decrease, the distinguishability of the winner significantly decreases. We could analyse this for all the matches of the knockout stage, but our purpose is just to show that although the AUC values and accuracies from models v2, v3 and v4 are very similar, in general this similarity is not observed for *distinguishability*.

Thus, we should do a balance between the AUC values, the accuracy and the *distinguishablity* (among other possible quantities) to decide which model is better.

From figures 4.1, 4.2 and 4.4 it is clear that the version that simultaneously maximizes AUC, accuracy and *distinguishablity* (for the final stage) is Model v2.

Taking into account all the results and observations we have done so far, and the fact that the matchup champion data we use (patch 7.21) corresponds to a quite different metagame from the one that was played during the tournament (patch 7.18), we believe that **the best model version would include both the champion matchup win rates and the player's champion win rates**. We expect that the model coefficients for the champion matchups would have been all positive if we could have accessed to data from patch 7.18. Moreover, we do not discard the possibility of including the `side` variable predictor, since it could be an important predictor during the next tournament patches. Thus, we would choose Model v2 as a final model if the metagame data of the tournament's patch was available, and we choose Model v4 for the current project since we are using metagame data from a more recent patch.

To end with this section we would like to discuss about the relative importance of individual predictors in the models v2 and v4. This can be done by looking at the absolute value of the t-statistic for each model parameter [1]. In R we can do it by executing the command `varImp(model)`.

The results for Model v2 and Model v4 predictors are shown in Figures 4.5 and 4.6, respectively.

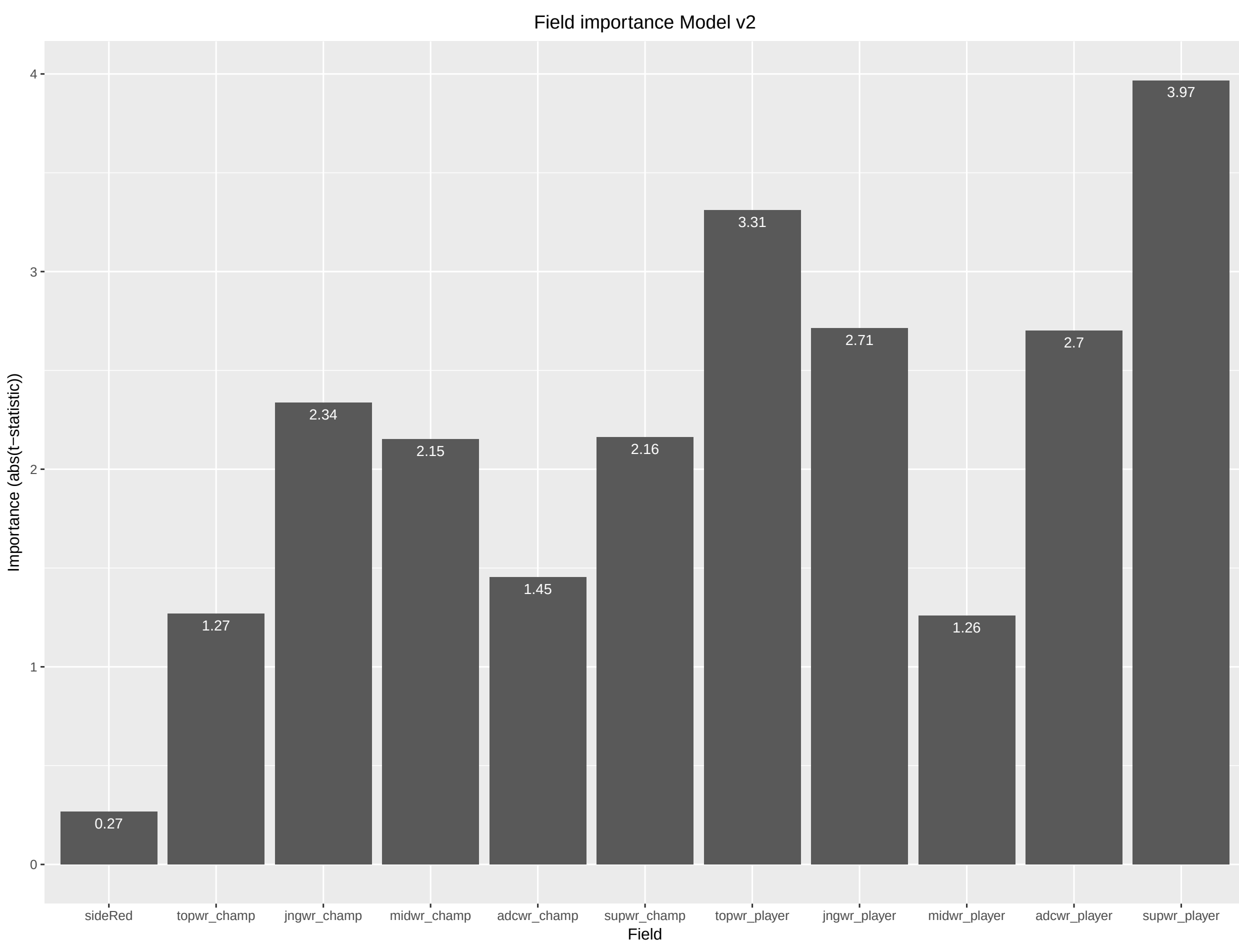


Figure 4.5: Importance of each predictor variable from Model v2

For Model v2, we don't take care about the `topwr_champion` and `jngwr_champion` predictors since they are negative and do not make sense for us.

Now we point out the following observations regarding Model v2:

- As we expected, the importance of the `side` variable is insignificant (0.27) with respect to the others.
- For each `position` (top, jng, mid, adc and sup), the importance of the predictor variables "`positionwr_player`" is larger than the importance of the "`positionwr_champion`" predictor variables. This means that, according to our model, it is more important "how good is a player with a particular champion" than "how good is a particular champion in the current metagame". However, we have to be careful with this last affirmation, since we are working with data from a different patch and the results could vary.
- The outstanding position (both for the metagame and the goodness of the player) is the `Support` and the worst is the `Midlaner`. We can say that the other three positions are almost equally important. The fact that the mid position seems to be the less important one is an argument in favour of the fact that SKT was less likely to win. As we commented before in the text, SKT has probably the current best League of Legends player in the World (Faker) whose main position is mid.

Finally we discuss about the importance of the predictors from Model v4 (Figure 4.6).

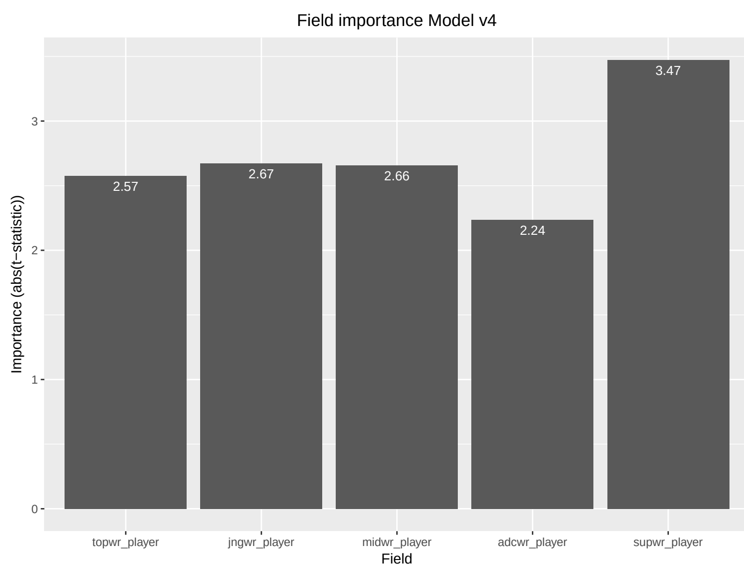


Figure 4.6: Importance of each predictor variable from Model v4

Whereas the top, jungle, mid and adc positions have approximately the same importance; as it happened in Model v2, the importance of the support position outstands quite significantly with respect to the other positions. This result is quite in agreement with the fact that the metagame from the 2017 Worlds Championship was dominated by the role of `Adcs` and `Supports`.

5 Conclusions

After gathering the data, analysing it, and predicting the results of the League of Legends World Championship via a simple logistic regression model, we can conclude that the our model without the `team` predictor variable is more accurate than the average accuracy coming from the knockout stage "pick'em", although we did not have exactly the same metagame data. While the mean "pick'em' accuracy was about 55.8%, our improved model versions have a mean accuracy of about $74.15\% \pm 15.55$ [7], in a knockout stage full of surprises and unexpected winners. We guess that if we have had the tournament metagame data, we would have had even better results.

Moreover, there are more facts that make our model realistic. The first one is the irrelevance of the variable `side`. In past metagames, the side where the team played was very important due to the fact that some neutral objectives were more important than others. Riot fixed that in the current metagame, and it is reflected in our model.

Another fact is the importance of the support and the toplaner. Professional casters said numerous times during the broadcasting of the World Championship that this meta was about tanks (which are usually played at the top lane), and supports, since one supporting item was pretty overpowered in the tournament metagame (the ardent censer). Our model also predicted that midlaner was the less important role, one fact that was also widely commented during this metagame, and one of the motives of the SKT lost.

We think that taking into account that we did not have the exact data of the Worlds metagame, and the fact that our model is pretty simple compared with the huge complexity of the game, we can be satisfied enough with our results. Specially, we are proud of the prediction of the final stage result, since it has been a huge surprise in the world of e-sports.

To conclude, we would like to say that we have learnt a lot while doing this project, since we had never worked with open data before. We have learnt to gather data from different kinds of open sources such as APIs and webpages, and about the legal and ethical issues of these sources. In particular, we have learnt to do queries to an API and to scrape a webpage, and we have improved and expanded our knowledge in R programming and statistics. We are very satisfied both with the results of the project and the personal outcome that we have obtained by doing it.

6 Legal and ethical issues

The data we have used for the prediction of the results of this World Championship comes from several sources. In order to use it we must know if such data is open, under the acceptance of terms of use, its user privacy policy, among others.

The first used source is `lol.gamepedia.com` [7]. The content we can find in this site is under the license CC BY-SA 3.0. This license allows us to **Share**, copy and redistribute the material in the site in any medium or format, and to **Adapt**, transform and build upon the data for any purpose. However, in order to be able to do so, we must give the corresponding credit, a link to the license and indicate the performed changes. Also, the newly created material must be under the same license as the original, son one may not restrict others from doing anything allowed by the license.

The `champion.gg` API [6] is provided as a free service. The amount of data extracted per unit of

[7] This comes from the 95% confidence intervals obtained via the quantile method. See the table from Figure 4.3.

time is set to a fixed value but it has no license or conditions that limit what one can do with its data whatsoever. The same happens with the sets of data obtained from `ddragon.leagueoflegends.com` and `oracleselixir.com`, [5, 8]. However, in both cases the obtained data is processed information, which is firstly extracted from the Riot API [18]. The data from `lolesports.com` also belongs to Riot, so it is convenient to carefully read the information the developer provides.

6.1 Riot policies

The API requires of a key in order to be used. Such a key is to be used, only by the user and may not be sold or transfered, in a developer account on the Riot Developer Site [18] which in the registration process asks to accept the API's Terms and Specifications. The compliance with the Terms grants the user "*a limited, non-exclusive, non-assignable, non-sublicensable, non-transferable, revocable right to access and use the Materials owned by Riot in order to develop and distribute Applications that incorporate, access or use any Game Information, for the purpose of displaying such Game Information to end users of such Applications*" [14]. But the use of the access to the information has some prohibited behaviours. We will briefly review some of the most important points:

- It is not possible to resell, charge or require any payment through a developed application with the game information extracted from the API unless it is notified and accepted by Riot. However, it is possible to advertise such application.
- The API must not be used to develop applications that promote illegal activities.
- The API shall not display any kind of political bias.
- Any application using the API must respect everyone's privacy rights.
- You cannot use the API to disrupt any part of the game by, for example, creating spyware, malware or promoting cheating in the game.
- You cannot modify the API in any way.
- You cannot try to reverse engineer, decompile or try to discover in any way the source code of the API or Riot games.

Riot also provides Privacy Policies when creating an account and visiting its sites or playing its games, since it collects some user information. All the provided information will not be shared with any third parties without the user's consent, but can be used anonymously to analyze site traffic or to deliver content according to specific interests by using cookies. The information might be disclosed if required by law.

6.2 Ethical issues

There are no ethical implications with the analysis of the data we have performed. Our goal is to predict some results without any commercial objective by the use of several sources of data. This project respects all Riot Terms and Specifications and complies with the CC BY-SA 3.0 license in the first data source. This project is to be understood as a contribution to the open data world and in this way we consider it is important to deliver it under a CC BY-SA 3.0 license.

7 Limitations and future work

- In order to use our model to make predictions in future World Championships we need to know, a priori, which champions will select each team. As it is the model right now, it would be useful to bet in real time and just before each match starts.

- We have been discussing about the hypothetical negative effect of using "metagame-related" data from a different game patch on the predictability and the model interpretations (importance of predictor variables and so on). Thus, it would be good, whenever possible, to avoid using data from a patch that does not correspond to that in the tournament.

- Instead of using the `team` predictor variables which is a categorical variable of dimension 24, it could be better to use a similar categorical variable but with smaller dimension. An option would be to consider a predictor variable that says from which region comes a particular team (Europe, America, Korea,...).

- Since we have seen that the most important predictor variables are those related with "how good is a player", a viable extension of the model that would probably improve the accuracy, would be to add new predictor variables related to players. To begin with, we could include the average KDA ratio of each player with a given champion. Another option would be to consider one predictor variable associated to each player and related to the "ranking position" as professional players.

- If we wanted to make predictions just before a knockout stage starts, we would not know which champions would play every single player in every single match, so all the "win rate - related" predictor variables of our current model would be useless. A solution to deal with this problem would be, for instance, to consider the average win rate of each player with its most played champions; to account for the metagame, we could use an average champion win rate obtained by averaging the win rate of the most played champions by a given player.

  An alternative that would maintain the model as it is right now would be, for example, to consider the top five champions of each player and then simulate all possible permutations of matchups (or a part of them, since this number of permutations will be very large) between one team and the other, and estimate which of the two teams in consideration is more likely to win (i.e., which team wins a higher number of matches from the sample obtained by doing the mentioned permutation procedure).

References

[1] Binary logistic regression in R, https://www.r-bloggers.com/evaluating-logistic-regression-models/

[2] Binary logistic regression example (Titanic survivors) https://www.r-bloggers.com/how-to-perform-a-logistic-regression-in-r/.

[3] Binary logistic regression with a single categorical predictor https://onlinecourses.science.psu.edu/stat504/node/150

[4] Binary logistic regression with a categorical predictor of multiple levels https://onlinecourses.science.psu.edu/stat504/node/152

[5] Static champion data (in english) from patch 7.19 http://ddragon.leagueoflegends.com/cdn/7.19.1/data/en_US/champion.json

[6] Champion.gg API http://api.champion.gg/.

[7] Leaguepedia: https://lol.gamepedia.com/<player_name>/Champion_Statistics, where <player_name> must be substituted by the professional player of interest.

[8] Tim Sevenhuysen, OraclesElixir.com, 2017. http://oracleselixir.com/gamedata/2017-worlds/.

[9] xlsx to csv online converter: https://convertio.co/es/xlsx-csv/.

[10] Visualization and statistics of logistic regressions in R https://stats.idre.ucla.edu/r/dae/logit-regression/.

[11] Knock out stage Worlds 2017 pick'em results. http://pickem.euw.lolesports.com/en-GB#leaderboards/everyone

[12] Bootstrap and permutation tests lectures, Pere Puig (Department of Mathematics from the UAB), Data Visualization and Modelling (subject from the Master's).

[13] *The 2017 Worlds finals between SKT and Samsung Galaxy will be all about adapting*, by Austen Goslin.https://www.riftherald.com/lol-worlds/2017/11/3/16603216/lol-worlds-finals-2017-skt-ssg-preview

[14] Terms and limitations of the Riot API. https://developer.riotgames.com/terms-and-conditions.html#limitations

[15] Scrapy software and documentation. https://scrapy.org/

[16] Example of data extracting from a table using Scrapy. http://www.olafdietsche.de/2015/04/05/extracting-table-data-with-scrapy

[17] Creative commons information. https://creativecommons.org/

[18] Riot API webpage, https://developer.riotgames.com/.

A Code outputs

A.1 Models' β coefficients and confidence intervals

Listing 2: Model β coefficients of the base model (v1)

```
                                2.5 % 97.5 %
(Intercept)              -57.4   -95.6  -19.1
teamCloud9                 5.9     2.1    9.8
teamDire Wolves            0.7    -4.1    5.4
teamEDward Gaming          0.5    -3.9    4.9
teamFenerbahce Esports     2.5    -1.3    6.4
teamFlash Wolves          -2.4    -9.9    5.2
teamFnatic                 3.0    -0.6    6.6
teamG2 Esports             6.8     2.0   11.6
teamGambit Esports       -17.3 -4533.7 4499.1
teamGigabyte Marines       9.0     2.2   15.8
teamHong Kong Attitude     5.2     0.7    9.7
teamImmortals              1.5    -3.1    6.1
teamKaos Latin Gamers      3.0    -2.8    8.8
teamLongzhu Gaming        21.8 -4044.3 4088.0
teamLyon Gaming            5.0    -0.7   10.7
teamMisfits                9.1     3.8   14.3
teamRampage                0.4 -5721.8 5722.6
teamRoyal Never Give Up    4.5     0.4    8.7
teamSamsung Galaxy         1.1    -3.0    5.3
teamSK Telecom T1          2.2    -2.6    7.0
teamTeam oNe Esports       3.3    -2.7    9.4
teamTeam SoloMid           4.8     0.3    9.3
teamTeam WE                8.6     3.4   13.9
teamYoung Generation      12.4     5.8   18.9
sideRed                    0.0    -1.3    1.3
topwr_champ              -13.3   -35.7    9.1
jngwr_champ              -16.5   -43.6   10.5
midwr_champ               24.3     5.4   43.2
adcwr_champ               32.9   -20.6   86.4
supwr_champ               18.0    -9.8   45.9
topwr_player              11.8     2.7   20.8
jngwr_player               5.5     1.7    9.3
midwr_player               6.8     0.8   12.9
adcwr_player              15.5     5.9   25.1
supwr_player              12.9     6.1   19.6
```

Listing 3: Model β coefficients of Model v2

```
                   2.5 % 97.5 %
(Intercept)  -21.3 -44.0    0.1
sideRed       -0.1  -0.9    0.7
topwr_champ   -8.8 -23.0    4.4
```

```
jngwr_champ   -20.5  -38.5    -3.9
midwr_champ    10.3    1.3    20.2
adcwr_champ    24.2   -8.1    57.6
supwr_champ    15.8    1.8    30.6
topwr_player    5.4    2.4     8.8
jngwr_player    2.8    0.9     4.9
midwr_player    1.7   -0.9     4.5
adcwr_player    4.0    1.3     7.2
supwr_player    5.0    2.6     7.6
```

Listing 4: Model β coefficients of Model v3

```
                        2.5 %  97.5 %
(Intercept)   -20.9  -43.3     0.3
topwr_champ    -8.8  -23.0     4.3
jngwr_champ   -20.7  -38.7    -4.2
midwr_champ    10.4    1.4    20.3
adcwr_champ    23.4   -8.4    56.2
supwr_champ    15.7    1.7    30.5
topwr_player    5.4    2.4     8.8
jngwr_player    2.8    0.9     4.9
midwr_player    1.8   -0.8     4.6
adcwr_player    4.0    1.3     7.2
supwr_player    5.0    2.7     7.6
```

Listing 5: Model β coefficients of Model v4

```
                      2.5 %  97.5 %
(Intercept)   -9.4  -12.9    -6.5
topwr_player   3.8    1.1     6.8
jngwr_player   2.5    0.7     4.4
midwr_player   3.3    1.0     5.9
adcwr_player   2.9    0.5     5.7
supwr_player   3.9    1.8     6.2
```

A.2 ANOVA tests for each model

Listing 6: ANOVA test for Model v2

```
Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                          179      249.53
side            1     1.424       178      248.11  0.2327296
topwr_champ     1     0.535       177      247.57  0.4643226
jngwr_champ     1     4.457       176      243.12  0.0347515 *
midwr_champ     1     4.727       175      238.39  0.0296887 *
adcwr_champ     1     1.077       174      237.31  0.2994692
supwr_champ     1     1.477       173      235.84  0.2243128
topwr_player    1    33.530       172      202.31  7.019e-09 ***
```

```
jngwr_player  1   14.276        171      188.03 0.0001578 ***
midwr_player  1   11.671        170      176.36 0.0006348 ***
adcwr_player  1    8.534        169      167.82 0.0034857 **
supwr_player  1   19.434        168      148.39 1.041e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Listing 7: ANOVA test for Model v3

```
             Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                           179     249.53
topwr_champ   1    0.494       178     249.04 0.4820141
jngwr_champ   1    4.724       177     244.31 0.0297513 *
midwr_champ   1    4.942       176     239.37 0.0262176 *
adcwr_champ   1    0.715       175     238.66 0.3979253
supwr_champ   1    1.516       174     237.14 0.2182490
topwr_player  1   33.501       173     203.64 7.121e-09 ***
jngwr_player  1   14.493       172     189.15 0.0001407 ***
midwr_player  1   12.084       171     177.06 0.0005085 ***
adcwr_player  1    8.982       170     168.08 0.0027259 **
supwr_player  1   19.621       169     148.46 9.445e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Listing 8: ANOVA test for Model v4

```
             Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                           179     249.53
topwr_player  1   26.319       178     223.22 2.895e-07 ***
jngwr_player  1   14.093       177     209.12 0.0001740 ***
midwr_player  1   19.938       176     189.18 8.000e-06 ***
adcwr_player  1    5.722       175     183.46 0.0167535 *
supwr_player  1   14.282       174     169.18 0.0001574 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

B Additional plots

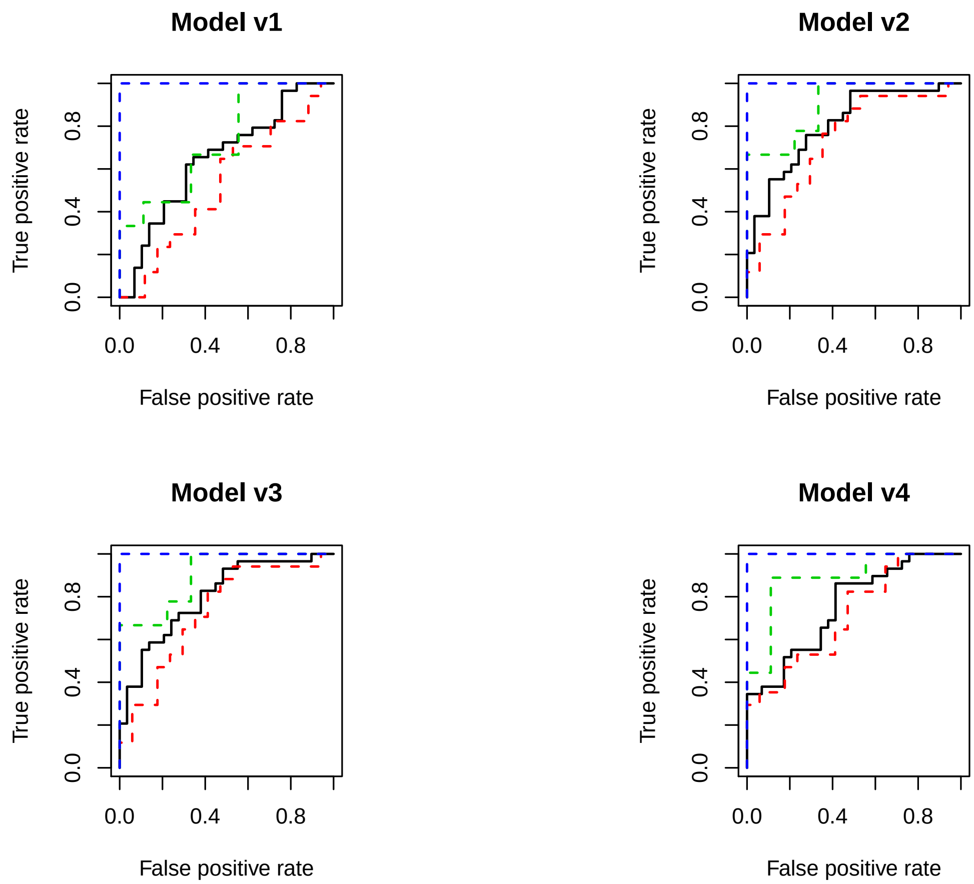


Figure B.1: ROC curves for the four models considered in the text, with respect to the whole knockout stage (black line), quarterfinals (red dashed line), semifinals (green dashed line) and the finals (blue dashed line).