Report on Rating Predictions for Movielens

Oliver Brueckner

2020-01-01

Introduction

This is a report on unsing machin learning methods to build better prediction strategies for soccer results. The data used in this report is from the first German soccer league "Bundesliga". The data is loaded from https://www.openligadb.de/ by using its JSON api.

This report is on the one hand somewhat familiar with the Lahman-report from XXX. On the other hand we try to predict an outcome based on the features of other similiar matches which is quiet familiar with the Netflix-Challange tasks from xxx. Because auf that I use the RMSE to demonstrate impovements of the predictions.

Soccer is a sport game where 11 players on each side try to score more goals than the opponent team. A match has a duration of 90 minutes. The result of a match can be: Team 1 wins, Draw, Team 2 wins. A Team gets 3 point for a win and 1 point for a draw.

Analysis: Loading and Inspecting the Data

I load the data from https://www.openligadb.de/ by using 2 endpoints:

- https://www.openligadb.de/api/getmatchdata gives back a list of all matches of a specific year. This data is nested so I unnest it and filter it so we only get the final result called "Enndergebnis" of each match.
- https://www.openligadb.de/api/getbltable gives back the table of a year including the points each team won in the year

```r
years_loaded<-2012:2018
years_inspected<-2014:2018

match_list<-lapply(years_loaded, function(y){
  match_url <- paste('https://www.openligadb.de/api/getmatchdata/bl1/', y, sep="")
  temp<-fromJSON(match_url,flatten=TRUE)%>%mutate(year=y)
  unnest(data = temp, cols = c(MatchResults))
})

ranking_list<-lapply(years_loaded, function(y){
  ranking_url<- paste('https://www.openligadb.de/api/getbltable/bl1/', y, sep="")
  temp<-fromJSON(ranking_url,flatten=TRUE)%>%mutate(year=y)%>%
    select(TeamName, Points_This_Year=Points, year)
  temp
})

match_data<-bind_rows(match_list)%>%filter(ResultName=="Endergebnis")
ranking_data<-bind_rows(ranking_list)
```

After loading the data is prepared:

- First I calculate the result of each match:
  - 1 stands for win of the home team

    – 0 stands for draw
    – -1 stands for win of the away team
- Then I create a tibble x for the features and calculate these features based on the historic matches of the two opponents

```
matches<-match_data%>%mutate(Begin=as.Date(MatchDateTime,"%Y-%m-%dT%H:%M:%S",tz="CET"),
                             result=ifelse(PointsTeam1>PointsTeam2,1,
                                           ifelse(PointsTeam1==PointsTeam2,0,-1)))%>%
  select(Begin, Team1.TeamName,Team2.TeamName,PointsTeam1,PointsTeam2,year,result)
x<-matches%>%select(Begin,Team1.TeamName,Team2.TeamName,year)
```

Some Teams have a higher probability of winning than others.

```
matches%>%group_by(Team1.TeamName)%>%summarize(r=mean(result))%>%top_n(3,r)
```

```
## # A tibble: 3 x 2
##   Team1.TeamName        r
##   <chr>             <dbl>
## 1 Borussia Dortmund 0.521
## 2 FC Bayern         0.765
## 3 RB Leipzig        0.451
```

```
matches%>%group_by(Team1.TeamName)%>%summarize(r=mean(result))%>%top_n(3,-r)
```

```
## # A tibble: 3 x 2
##   Team1.TeamName              r
##   <chr>                   <dbl>
## 1 Eintracht Braunschweig -0.235
## 2 SpVgg Greuther Fürth   -0.765
## 3 SV Darmstadt 98        -0.265
```

To take account into this I calculate some features for each match, based on the histroical data: * LongTermSignals: These Signals are calculated by subtracting the points, that the opponent gained in the last n years. * ShortTermSignals: These Signals are calculated by subtracting the points, that the opponent gained in the last n matches. * DirectSignals: These Signals are calculated by subtracting the points, that the opponent gained in the last n matches against each other.

```
# LongTermSignals
get_points_last_n_years<-function(n,team,y){
  temp<-ranking_data%>%filter(year<y & (TeamName==team))%>%top_n(n,year)
  sum(temp$Points_This_Year)/max(ranking_data$Points_This_Year)
}
get_long_term_signals<-function(n,team1,team2,y){
  (get_points_last_n_years(n,team1,y)-get_points_last_n_years(n,team2,y))
}

x<-x%>%rowwise()%>%
  mutate(LongTermSignal1=get_long_term_signals(1,Team1.TeamName,Team2.TeamName,year))
x<-x%>%rowwise()%>%
  mutate(LongTermSignal3=get_long_term_signals(3,Team1.TeamName,Team2.TeamName,year))
x<-x%>%rowwise()%>%
  mutate(LongTermSignal5=get_long_term_signals(5,Team1.TeamName,Team2.TeamName,year))
x[is.na(x)] <- 0



# ShortTermSignals
```

```r
get_last_n_games<-function(n,team, begin){
  matches%>%
    filter(Begin<begin & (Team1.TeamName==team | Team2.TeamName==team))%>%top_n(n,Begin)
}

get_points_from_last_n_games<-function(n,team,begin){
   a<-get_last_n_games(n,team,begin)%>%
     mutate(
       points=ifelse(result==0,1,ifelse(Team1.TeamName==team,
                                        ifelse(result==1,3,0),ifelse(result==-1,3,0))))
   sum(a$points)
}

get_short_term_signals<-function(n,team1, team2, begin){
  (get_points_from_last_n_games(n,team1,begin)-
     get_points_from_last_n_games(n,team2,begin))/(n*3)
}

x<-x%>%rowwise()%>%
  mutate(ShortTermSignal10=get_short_term_signals(10,Team1.TeamName,Team2.TeamName,Begin))
x<-x%>%rowwise()%>%
  mutate(ShortTermSignal5=get_short_term_signals(5,Team1.TeamName,Team2.TeamName,Begin))
x<-x%>%rowwise()%>%
  mutate(ShortTermSignal3=get_short_term_signals(3,Team1.TeamName,Team2.TeamName,Begin))
matches$PointsTeam1<-NULL
matches$PointsTeam2<-NULL

## DirectSignals.

get_last_n_direct_games<-function(n,team1,team2,begin){
  matches%>%
    filter(Begin<begin & Team1.TeamName %in% c(team1,team2) &
             Team2.TeamName %in% c(team1,team2))%>%
    top_n(n,Begin)
}
get_direct_signals<-function(n,team1, team2, begin){
  a<-get_last_n_direct_games(n,team1,team2,begin)
  points_team1<-a%>%
    mutate(points_team1=ifelse(result==0,1,
                               ifelse(Team1.TeamName==team1,
                                      ifelse(result==1,3,0),ifelse(result==-1,3,0))))%>%
    .$points_team1
  points_team2<-a%>%
    mutate(points_team2=ifelse(result==0,1,
                               ifelse(Team1.TeamName==team2,
                                      ifelse(result==1,3,0),ifelse(result==-1,3,0))))%>%
    .$points_team2
  (sum(points_team1)-sum(points_team2))/(n*3)
}
x<-x%>%rowwise()%>%
  mutate(DirectSignal10=get_direct_signals(10,Team1.TeamName,Team2.TeamName,Begin))
x<-x%>%rowwise()%>%
  mutate(DirectSignal5=get_direct_signals(5,Team1.TeamName,Team2.TeamName,Begin))
```

```
x<-x%>%rowwise()%>%
  mutate(DirectSignal3=get_direct_signals(3,Team1.TeamName,Team2.TeamName,Begin))
```

Now I can delete duplicated columns in x and matches and filter the data by years that will be inspected. Then I add the x-Tibble to the matches data frame.

```
x$Begin<-NULL
x$Team1.TeamName<-NULL
x$Team2.TeamName<-NULL

x$Team1.PointsLastYear<-NULL
x$Team2.PointsLastYear<-NULL

matches<-matches%>%filter(year %in% years_inspected)
x<-x%>%filter(year %in% years_inspected)
matches$year<-NULL
x$year<-NULL

matches<-add_column(matches,x=as_tibble(x))
```

Now I can split the matches data frame into a traing set and a test set.

```
test_index <- createDataPartition(y=matches$result,times=1,p=0.2,list=FALSE)
test_set<-matches[test_index,]
train_set<-matches[-test_index,]
```

Method

Now we can start building our model. First I try some naive approaches on the trainig set:

- Guessing allways the most common result
- Guessing allways the mean of the three possible results
- Geussing allway the average result of the traing set

Then I try to improve these approaches by using different training models. These models will later been ensembled to build the final model. This ensembled model later is used on the test set.

```
add_row_to_rmse_table<-function(method_name,rmse_value){
  if(!exists("rmse_results")){
    data.frame(method = method_name, rmse_value)
  }
  else {
    bind_rows(rmse_results, data.frame(method = method_name, rmse_value))
  }
}
rmse_results<-NULL

# guessing always the most comon outcome
which.max(train_set$result)
```

```
## [1] 1
```

```
RMSE(which.max(train_set$result),test_set$result)
```

```
## [1] 1.157049
```

```
rmse_results<-add_row_to_rmse_table("Most common result",
                          RMSE(which.max(train_set$result),train_set$result))

# guessing the mean of the three possible outcomes
mean(c(1,0,-1))
```

```
## [1] 0
```

```
RMSE(mean(c(1,0,-1)),test_set$result)
```

```
## [1] 0.8541907
```

```
rmse_results<-add_row_to_rmse_table("Mean of the possible results",
                          RMSE(mean(c(1,0,-1)),train_set$result))

# guessing allways the mean of the results
mean(train_set$result)
```

```
## [1] 0.1692559
```

```
RMSE(mean(train_set$result),test_set$result)
```

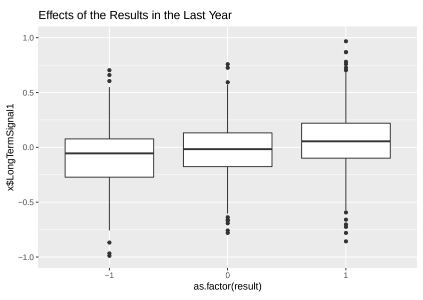
```
## [1] 0.8319439
```

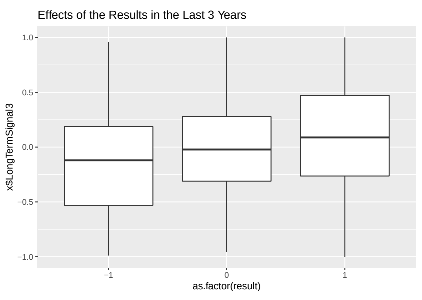
```
rmse_results<-add_row_to_rmse_table("Average of the training results",
                          RMSE(mean(train_set$result),train_set$result))
rmse_results
```

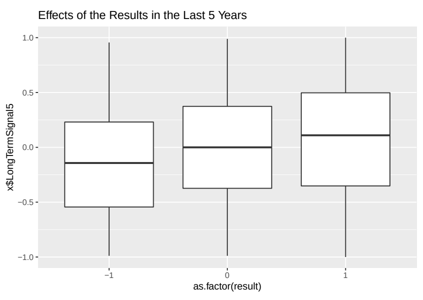
```
##                            method rmse_value
## 1              Most common result  1.1900377
## 2    Mean of the possible results  0.8687356
## 3 Average of the training results  0.8520880
```

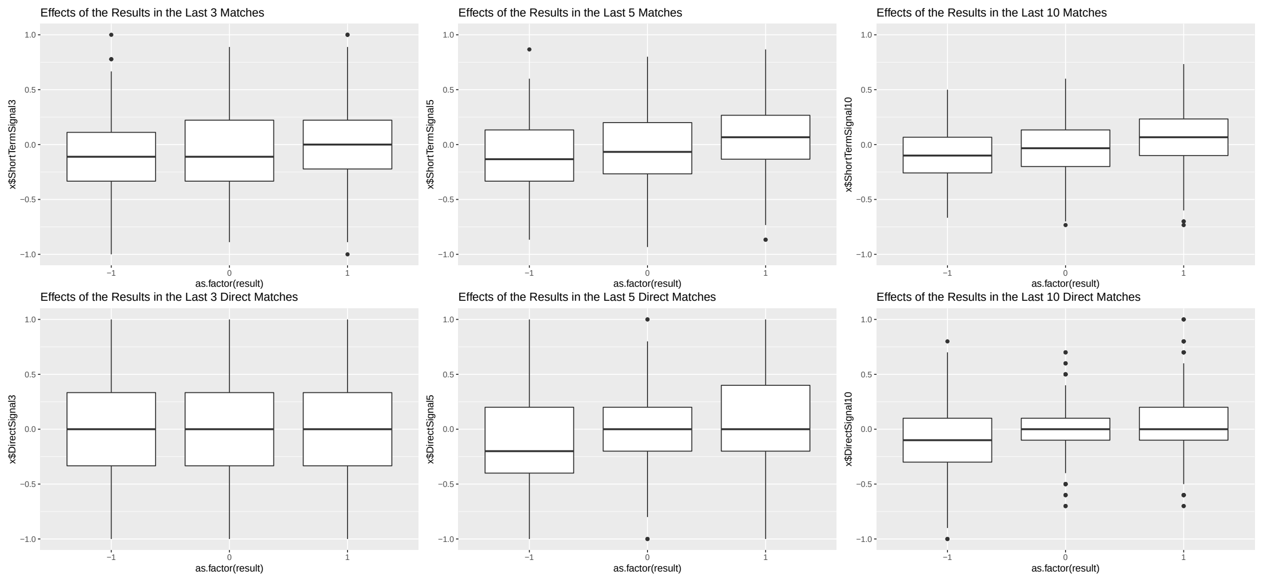
These naive approaches can be impoved by using the earlier calculated features. The following figures show the effects of the features on the results by using a boxplot:

```
train_set%>%ggplot(aes(as.factor(result),x$LongTermSignal1))+geom_boxplot()+ylim(-1,1)+ggtitle("Effects
train_set%>%ggplot(aes(as.factor(result),x$LongTermSignal3))+geom_boxplot()+ylim(-1,1)+ggtitle("Effects
train_set%>%ggplot(aes(as.factor(result),x$LongTermSignal5))+geom_boxplot()+ylim(-1,1)+ggtitle("Effects
train_set%>%ggplot(aes(as.factor(result),x$ShortTermSignal3))+geom_boxplot()+ylim(-1,1)+ggtitle("Effects
train_set%>%ggplot(aes(as.factor(result),x$ShortTermSignal5))+geom_boxplot()+ylim(-1,1)+ggtitle("Effects
train_set%>%ggplot(aes(as.factor(result),x$ShortTermSignal10))+geom_boxplot()+ylim(-1,1)+ggtitle("Effec
train_set%>%ggplot(aes(as.factor(result),x$DirectSignal3))+geom_boxplot()+ylim(-1,1)+ggtitle("Effects o
train_set%>%ggplot(aes(as.factor(result),x$DirectSignal5))+geom_boxplot()+ylim(-1,1)+ggtitle("Effects o
train_set%>%ggplot(aes(as.factor(result),x$DirectSignal10))+geom_boxplot()+ylim(-1,1)+ggtitle("Effects
```









These figures show, that the features have an impact. They also show, that there is some overlapping of the boxes, that needs to be fuguered out by combining these features. There are some models that can do this for us. So in the next step I use 7 machine learning models to use the features for prediction.

```r
control <- trainControl(method = "cv", number = 5)
models <- c( "lm","glm","knn","gamLoess","svmLinear","rpart","rf")
fits <- lapply(models, function(model){
  train(train_set$x,train_set$result,model,trControl = control)
  })
```

```
## Loading required package: gam

## Loading required package: splines

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loaded gam 1.16.1
```

```r
model_rsme<-sapply(fits, function(o){
       min(o[["results"]][["RMSE"]])
   })
rmse_results<-add_row_to_rmse_table(models,model_rsme)
```

Now I use these models to build my ensembled model.

```r
pred_train <- sapply(fits, function(object)
  predict(object, newdata = train_set$x))

RMSE(rowMeans(pred_train),train_set$result)
```

```
## [1] 0.7330527
```

```r
rmse_results<-add_row_to_rmse_table("Model-Ensemble",
                                    RMSE(rowMeans(pred_train),train_set$result))
rmse_results
```

```
##                              method rmse_value
## 1                Most common result  1.1900377
## 2      Mean of the possible results  0.8687356
## 3  Average of the training results  0.8520880
## 4                                lm  0.8129584
## 5                               glm  0.8161781
## 6                               knn  0.8538149
## 7                          gamLoess  0.8168110
## 8                         svmLinear  0.8313525
## 9                             rpart  0.8430714
## 10                               rf  0.8313380
## 11                   Model-Ensemble  0.7330527
```

Evaluation

The ensebled model I built will now predict the results of the test set.

```r
pred_test <- sapply(fits, function(object)
  predict(object, newdata = test_set$x))
RMSE(rowMeans(pred_test),test_set$result)
```

```
## [1] 0.8008138
```

These pedictions can be further improved by adding more data to the models:

- detailled match stats like running distance, shoots, shoots on target, passes, passes completed
- stats for each player, so a team performance can be calculated as the sum of the players perfomances
- external data, such as attendance, coaching experience or weather conditions