The Betlab Project

Tobias Diederich

Saturday, October 31, 2015

Abstract

The Betlab Project aims to predict the outcome of football matches. The predictions are compared with booky odds to place value bets. To predict the outcome, the market prices of the participating players are analysed because they are assumed to represent the base quality of a player. The main metric to evaluate model performance is the simulated earning with respect to the booky odds.

I show the high potential of my approach, even if it is not yet practicable.

Data

The match, team and player data are collected from Transfermarkt. Booky odds are collected from Sfstats. The parsers are written in Java and are not part of this paper.

Relevant data will be of germanies 1. Bundesliga from season 2005-2006 to 2014-2015.

```
source(file = 'production/loadData.R', echo = FALSE, encoding = 'UTF-8')
toMatchday <- 34
seasons <- c('2005-2006', '2006-2007', '2007-2008', '2008-2009', '2009-2010',
             '2010-2011', '2011-2012', '2012-2013', '2013-2014', '2014-2015')
leagues <- c('BL1')
trainingRaw <- loadTrainingData(toMatchday = toMatchday, seasons = seasons, leagues = leagues)
matches <- trainingRaw$matches
odds <- trainingRaw$odds
stats <- trainingRaw$stats
```

Datasets

matches -> contains all matches
odds -> contains booky odds and probabilies for all matches
stats -> an observation contains information for one player in one match

```
describe(select(matches, goalsHome, goalsVisitors, matchResult))
```

```
## select(matches, goalsHome, goalsVisitors, matchResult)
##
##  3  Variables      3060  Observations
## ---------------------------------------------------------------------------
## goalsHome
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##    3060       0      10    0.94   1.619       0       0       1       1
##     .75     .90     .95
##       2       3       4
##
##             0   1   2   3   4  5  6 7 8 9
```

```
## Frequency 647 957 779 403 182 63 20 6 2 1
## %          21  31  25  13   6  2  1 0 0 0
## ---------------------------------------------------------------------------
## goalsVisitors
##       n missing  unique    Info    Mean
##    3060       0       9    0.92   1.255
##
##             0    1   2   3   4  5  6 7 8
## Frequency 928 1055 647 285 104 28 11 1 1
## %          30   34  21   9   3  1  0 0 0
## ---------------------------------------------------------------------------
## matchResult
##       n missing  unique
##    3060       0       3
##
## VisitorsVictory (901, 29%), Draw (779, 25%)
## HomeVictory (1380, 45%)
## ---------------------------------------------------------------------------
```

```
describe(select(odds, HomeVictory, VisitorsVictory, Draw))
```

```
## select(odds, HomeVictory, VisitorsVictory, Draw)
##
##  3  Variables      6859  Observations
## ---------------------------------------------------------------------------
## HomeVictory
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##    6859       0     623       1  0.4829  0.1681  0.2257  0.3663  0.4762
##     .75     .90     .95
##  0.6024  0.7407  0.8065
##
## lowest : 0.03774 0.05882 0.06329 0.06557 0.06835
## highest: 0.90090 0.90909 0.91743 0.92593 0.94340
## ---------------------------------------------------------------------------
## VisitorsVictory
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##    6859       0     863       1  0.3157 0.08764 0.11587 0.19608 0.29499
##     .75     .90     .95
## 0.40000 0.56497 0.64103
##
## lowest : 0.03150 0.03968 0.04000 0.04274 0.04310
## highest: 0.82645 0.83333 0.84034 0.85470 0.86957
## ---------------------------------------------------------------------------
## Draw
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##    6859       0     389       1  0.2766  0.1757  0.2078  0.2604  0.2941
##     .75     .90     .95
##  0.3077  0.3125  0.3155
##
## lowest : 0.08598 0.09174 0.09434 0.10060 0.10194
## highest: 0.32895 0.33003 0.33113 0.33223 0.33445
## ---------------------------------------------------------------------------
```

```
describe(select(stats, fitPrice, position, playerAssignment, formation))
```

```
## select(stats, fitPrice, position, playerAssignment, formation)
##
##  4  Variables      109552  Observations
## ---------------------------------------------------------------------------
## fitPrice
##        n  missing   unique     Info     Mean      .05      .10      .25
##   109219      333      137        1  4021380   275000   500000  1000000
##      .50      .75      .90      .95
##  2400000  4500000  9000000 13000000
##
## lowest :        0    25000    40000    50000    75000
## highest: 42000000 45000000 48000000 50000000 55000000
## ---------------------------------------------------------------------------
## position
##       n missing  unique
##  109552       0      13
##
## Torwart (12209, 11%), Innenverteidiger (18817, 17%)
## Linker Verteidiger (8595, 8%)
## Rechter Verteidiger (8155, 7%)
## Defensives Mittelfeld (12953, 12%)
## Zentrales Mittelfeld (5907, 5%)
## Linkes Mittelfeld (3522, 3%)
## Rechtes Mittelfeld (3379, 3%)
## Offensives Mittelfeld (7366, 7%)
## Haengende Spitze (2421, 2%)
## Mittelstuermer (15189, 14%)
## Linksaussen (5562, 5%), Rechtsaussen (5477, 5%)
## ---------------------------------------------------------------------------
## playerAssignment
##       n missing  unique
##  109552       0       4
##
## AUSGEWECHSELT (16932, 15%), BENCH (25235, 23%)
## DURCHGESPIELT (50385, 46%)
## EINGEWECHSELT (17000, 16%)
## ---------------------------------------------------------------------------
## formation
##       n missing  unique
##  108379    1173      25
##
## lowest : 3-1-4-2     3-3-3-1     3-4-2-1     3-4-3       3-4-3 flach
## highest: 4-5-1       4-5-1 flach 5-3-2       5-4-1       5-4-1 flach
## ---------------------------------------------------------------------------
```

Feature Engineering

The features I extract are the marketprices of participating players aggregated by team, grouped position and aggregation method. Participating players are those who played the whole match, came from the bench or got substituted. Grouped positions are goaly, defense, midfield and offense. Aggregation methods are: min, max, mean and sum.

```
source('production/positionFeatureExtraction.R',
       echo = FALSE, encoding = 'UTF-8')
### Preparation
#[1] "Torwart"               "Innenverteidiger"      "Linker Verteidiger"    "Rechter Verteidiger"   "D
#[6] "Zentrales Mittelfeld"  "Linkes Mittelfeld"     "Rechtes Mittelfeld"    "Offensives Mittelfeld" "H
#[11] "Mittelstuermer"        "Linksaussen"           "Rechtsaussen"
positions <- c('tw', 'def', 'def', 'def', 'mid', 'mid', 'mid', 'mid', 'off', 'off', 'off', 'off', 'off'
relNormalAssignments <- c('DURCHGESPIELT', 'EINGEWECHSELT', 'AUSGEWECHSELT')
normalMatches <- extractMatchResultFeatures(playerStats = stats,
                                            matches = matches,
                                            priceAssignedPositions = positions,
                                            functs = c('min', 'max', 'avg', 'sum'),
                                            relNormalAssignments)

filteredNormalMatches <- filterFeaturedMatches(normalMatches)
```

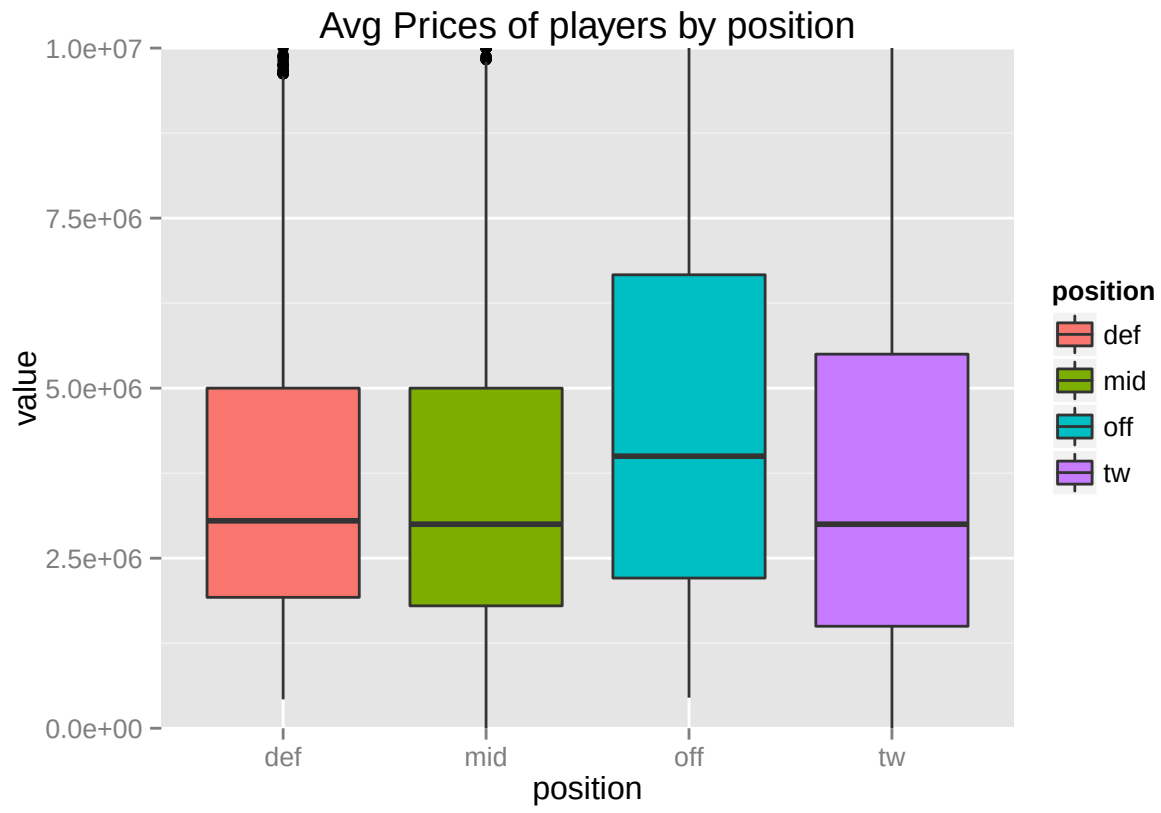
```
explMatches <- select(filteredNormalMatches, -matchId, -matchResult, -goalDiff)
# Features:
colnames(explMatches)
```

```
##  [1] "tw_Price_Home_avg"      "def_Price_Home_min"
##  [3] "def_Price_Home_max"     "def_Price_Home_avg"
##  [5] "def_Price_Home_sum"     "mid_Price_Home_min"
##  [7] "mid_Price_Home_max"     "mid_Price_Home_avg"
##  [9] "mid_Price_Home_sum"     "off_Price_Home_min"
## [11] "off_Price_Home_max"     "off_Price_Home_avg"
## [13] "off_Price_Home_sum"     "tw_Price_Visitors_avg"
## [15] "def_Price_Visitors_min" "def_Price_Visitors_max"
## [17] "def_Price_Visitors_avg" "def_Price_Visitors_sum"
## [19] "mid_Price_Visitors_min" "mid_Price_Visitors_max"
## [21] "mid_Price_Visitors_avg" "mid_Price_Visitors_sum"
## [23] "off_Price_Visitors_min" "off_Price_Visitors_max"
## [25] "off_Price_Visitors_avg" "off_Price_Visitors_sum"
```

```
library(magrittr)
library(tidyr)
explGathered <- explMatches %>% gather(feature, value)

getGroupStr <- function(feature, group) {
    charList <- strsplit(as.character(feature), '_')
    charFrame <- data.frame(do.call(rbind, charList))
    if(group == 'func') {
        return(charFrame[, 4])
    } else if(group == 'pos') {
        return(charFrame[, 1])
    } else {
        return(NA)
    }
}
groupedMatches <- mutate(explGathered, funct = factor(getGroupStr(feature, 'func')),
                         position = factor(getGroupStr(feature, 'pos')))
avgPlot <- ggplot(filter(groupedMatches, funct == 'avg'), aes(x = position, y = value, fill = position))
    geom_boxplot() +
```

```
    ggtitle('Avg Prices of players by position') +
    coord_cartesian(ylim = c(0, 10000000))
avgPlot
```



Model fitting

```
seed <- 1834
tcontr = trainControl(method = 'cv', number = 20, classProbs = TRUE)
train <- dplyr:::select(filteredNormalMatches, -goalDiff)
resultFormula <- as.Formula('matchResult ~ . - matchId')
set.seed(seed)
polrModel <- train(resultFormula, data = train, method = 'polr',
                   preProcess = c('center', 'scale'),
                   trControl = tcontr)
polrModel
```

```
## Ordered Logistic or Probit Regression
##
## 3060 samples
##   27 predictor
##    3 classes: 'VisitorsVictory', 'Draw', 'HomeVictory'
##
## Pre-processing: centered, scaled
```

```
## Resampling: Cross-Validated (20 fold)
## Summary of sample sizes: 2908, 2907, 2907, 2907, 2907, 2907, ...
## Resampling results
##
##   Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.5189287  0.1854046  0.02781615   0.0501598
##
##
```

Evaluate model in comparison to booky odds

To evaluate the predictions, matches have to be predicted iteratively. For this all matches are split in 10 disjunctive validation sets. In every of the 10 iterations, a model is trained and used to predict the observations in the corresponding validation set.

```
source(file = 'production/models.R',
       echo = FALSE, encoding = 'UTF-8')
test <- dplyr:::select(filteredNormalMatches, -goalDiff)
noneContr = trainControl(method = 'none')
folds <- 10
splits <- splitMatches(matchesToSplit = test,
                       testingMatches = test, folds = folds, seed = seed)

allPredictions <- iterativelyPredict(splits, folds,
                                     resultFormula, meth = 'polr',
                                     prePr = c('center', 'scale'),
                                     tControl = noneContr, seed = seed)

source(file = 'evaluatePrediction.R',
       echo = FALSE, encoding = 'UTF-8')

evaluations <- evaluatePrediction(prediction = allPredictions,
                                  comparison = odds,
                                  probRatioToBet = 1.1, stake = 1)
printEvaluation(evaluations)
```

```
## [1] "Stake: 1650"
## [1] "Gain: 364.01"
## [1] "Gain [%]: 22.0612121212121"
## [1] "Value Diff [%]: -1.94123385162278"
## [1] "Accuracy [%]: 52.0915032679739"
## [1] "Booky Accuracy [%]: 50.6535947712418"
```

22 % earning is very good, the problem is, that for this calculation it is necessary to know all playing players. This is not possible in practice, you just know the starting lineup.

Prediction with just the starting lineup

This time only the players in the starting lineup are considered.

```
relPredAufstellungAssignments <- c('DURCHGESPIELT', 'AUSGEWECHSELT')
predAufstellungMatches <- extractMatchResultFeatures(playerStats = stats,
                                                     matches = matches,
                                                     priceAssignedPositions = positions,
                                                     functs = c('min', 'max', 'avg', 'sum'),
                                                     relPredAufstellungAssignments)
filteredPredAufstellungMatches <- filterFeaturedMatches(predAufstellungMatches)

aufstTest <- dplyr:::select(filteredPredAufstellungMatches, -goalDiff)
aufstSplits <- splitMatches(matchesToSplit = aufstTest,
                        testingMatches = aufstTest, folds = folds, seed = seed)

allPredictions <- iterativelyPredict(aufstSplits, folds,
                                     resultFormula, meth = 'polr',
                                     prePr = c('center', 'scale'),
                                     tControl = noneContr, seed = seed)

aufstEvaluations <- evaluatePrediction(prediction = allPredictions,
                                   comparison = odds,
                                   probRatioToBet = 1.1, stake = 1)
printEvaluation(aufstEvaluations)
```

```
## [1] "Stake: 1361"
## [1] "Gain: -16.62"
## [1] "Gain [%]: -1.22116091109478"
## [1] "Value Diff [%]: -3.79593469651173"
## [1] "Accuracy [%]: 49.281045751634"
## [1] "Booky Accuracy [%]: 50.6535947712418"
```