The Betlab Project

Tobsen1x

Tuesday, December 09, 2014

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 market prices are determined two times a year, so there is a need to integrate a short term form quantifier for each participating player. These data is provided by a german football magazin and is called kicker grade, which is a school grade evaluating every performance of the participating players.

Data

The data is collected from several websites, a Java application is written to grab them and write them into a relational database. These parser are not part of this report.

Datasets

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

```
source(file = 'loadData.R', echo = FALSE, encoding = 'UTF-8')
data <- loadData('BL1')
```

```
## MYSQL_HOME defined as C:\MySQL\MySQL Server 5.5
```

```
stats <- data$playerStats
matches <- data$matches
odds <- data$odds
```

The **stats** dataset contains 76816 observations with 22 variables. The most important variables are:

- matchId: Id of the match
- playerId: Id of the player
- kickerGrade: school grade evaluating the performance of the player
- transPos: position the player takes in the regarding match
- home: is the regarding player playing at home
- fitPrice: youngest marketprice of the regarding player before the match

```
library(dplyr)
summary(select(stats, kickerGrade, transPos, playerAssignment, fitPrice))
```

```
##   kickerGrade                    transPos          playerAssignment
##  Min.   :1.000   Innenverteidiger     : 9951   AUSGEWECHSELT:11846
##  1st Qu.:3.000   Mittelstürmer        : 8680   BENCH        :17788
##  Median :3.500   Defensives Mittelfeld: 7600   DURCHGESPIELT:35278
##  Mean   :3.624   Linker Verteidiger   : 4728   EINGEWECHSELT:11904
##  3rd Qu.:4.500   Rechter Verteidiger  : 4635
##  Max.   :6.000   (Other)              :23430
##  NA's   :27134   NA's                 :17792
##     fitPrice
##  Min.   :   25000
##  1st Qu.: 1000000
##  Median : 2500000
##  Mean   : 4111874
##  3rd Qu.: 5000000
##  Max.   :55000000
##  NA's   :28
```

The **matches** dataset contains 2142 matches from 1 leagues in the seasons 2007-2008 to 2013-2014.

The **odds** dataset contains booky odds for 2142 matches.

Enrich stats by adjusted grade

The function enrichAdjGrade in the script adjGradeModel.R enriches stats by an adjusted grade which represents the pure form of a player in a match. That means the kicker grade is purged by the the players position, the quality of the opponents players, the home advantage and the quality of the regarded player. The goal is to extract a player predictor which represents solely the player form, independent of the mentioned properties.

```
source(file = 'adjGradeModel.R', echo = FALSE, encoding = 'UTF-8')
# enriches stats with the opponent price
adjGradeData <- extractFeaturesForAdjGradeModel(stats)
library(magrittr)
# select just the variables necessary for modelling
modelData <- adjGradeData %>% select(kickerGrade, fitPrice,
                                     opponentPrice, home, transPos)
library(e1071)
# Checking for skewness in numeric predictors
modelData %>% select(fitPrice, opponentPrice) %>% apply(2, skewness)
```

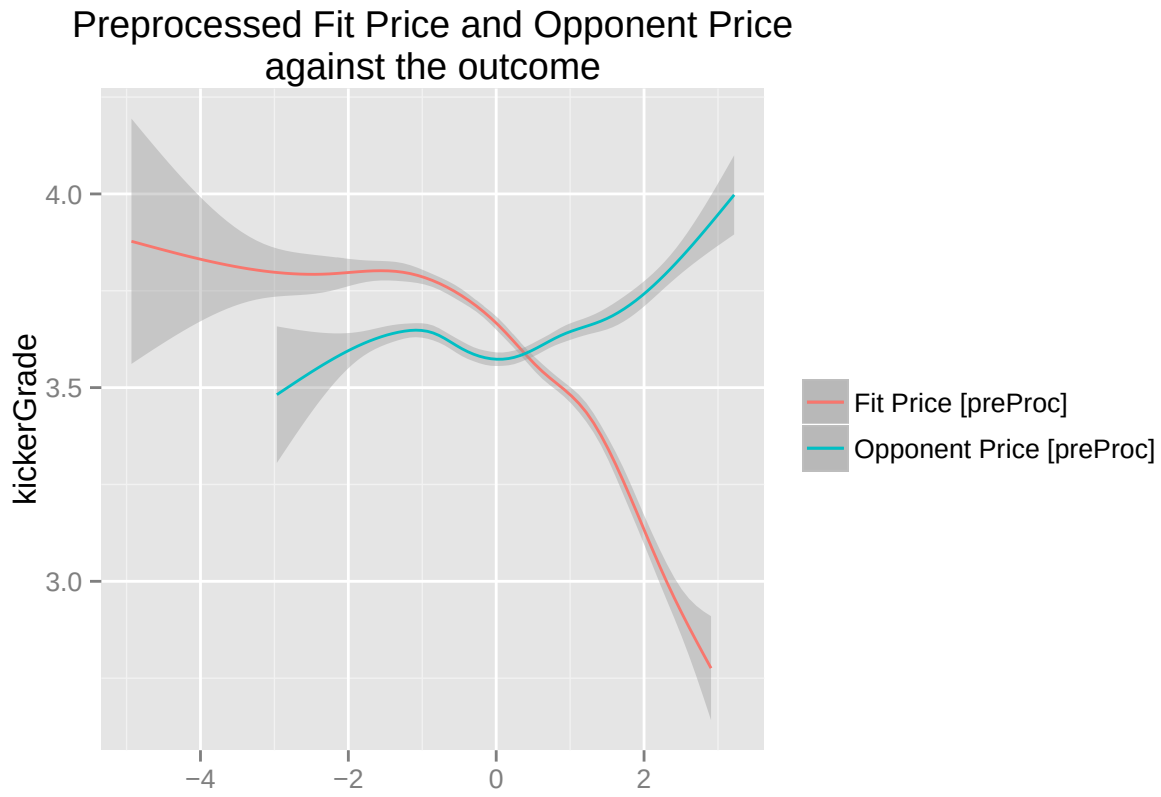
```
##      fitPrice opponentPrice
##      3.250798      2.895961
```

Both predictors are skewed (> 1), so a BoxCox transformation is used. Both predictors are centered and scaled, too.

```
library(caret)
# BoxCox Transformation to resolve skewness
preProc <- modelData %>% select(fitPrice, opponentPrice) %>%
    preProcess(method = c('center', 'scale', "BoxCox"))
preProcPredictors <-  preProc %>% predict(select(modelData, fitPrice, opponentPrice))
preProcPredictors <- preProcPredictors %>% rename(preProcFitPrice = fitPrice, preProcOpponentPrice = opp
```

```
# attach preprocessed predictors
modelData <- modelData %>% cbind(preProcPredictors)

library(ggplot2)
# Plot to visualize the relationship of the poredictors to the outcome
modelData %>% ggplot(aes(y = kickerGrade)) +
    geom_smooth(aes(x = preProcFitPrice, colour = 'preProcFitPrice')) +
    geom_smooth(aes(x = preProcOpponentPrice,
                    colour = 'preProcOpponentPrice')) +
    scale_x_continuous(name = element_blank()) +
    scale_colour_discrete(name  = element_blank(),
                          breaks=c("preProcFitPrice", "preProcOpponentPrice"),
                          labels=c("Fit Price [preProc]", "Opponent Price [preProc]")) +
    ggtitle(label = 'Preprocessed Fit Price and Opponent Price\nagainst the outcome')
```



This plot shows, that both preprocessed predictors have a nonlinear relationship to the outcome, so they are modelled as polynomials of grade 3.

```
repCVControl <- trainControl(method = 'repeatedcv', number = 10,
                             repeats = 3)
set.seed(1)
# ordinary linear regression
lmPreProcFit <- train(kickerGrade ~ poly(preProcFitPrice, 3) +
                          poly(preProcOpponentPrice, 3) +
                          home + transPos, data = modelData,
```

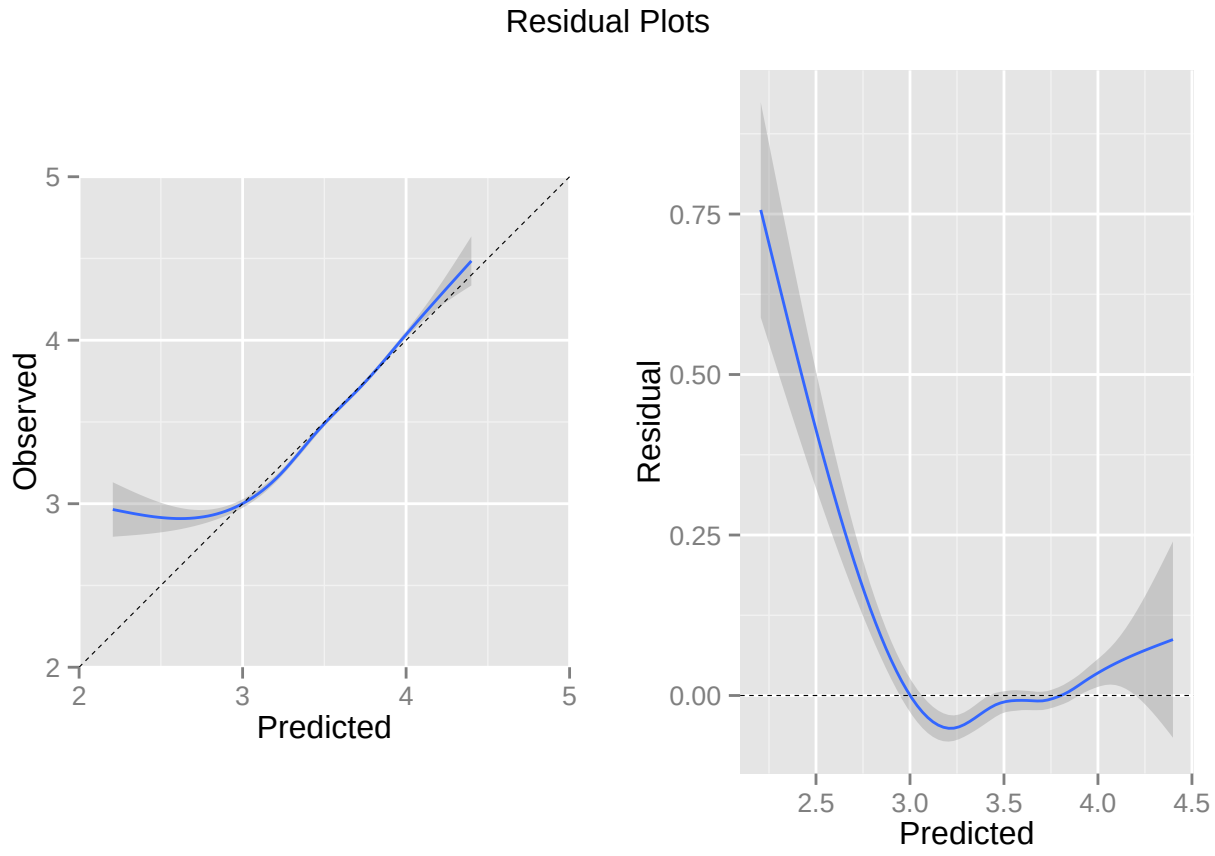
```
                      method = 'lm', trControl = repCVControl)
lmPreProcFit
```

```
## Linear Regression
##
## 49675 samples
##     6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
##
## Summary of sample sizes: 44707, 44708, 44708, 44706, 44708, 44707, ...
##
## Resampling results
##
##   RMSE       Rsquared    RMSE SD      Rsquared SD
##   0.9242919  0.08707081  0.007486358  0.005857198
##
##
```

```
residPlot1 <- modelData %>% ggplot(aes(y = kickerGrade, x = predict(lmPreProcFit, modelData))) +
    geom_smooth() +
    coord_fixed(ratio = 1, xlim = c(2, 5), ylim = c(2, 5)) +
    scale_x_continuous(name = 'Predicted') +
    scale_y_continuous(name = 'Observed') +
    geom_abline(intercept = 0, slope = 1, size = 0.2, linetype = 'dashed')

residPlot2 <- modelData %>% ggplot(aes(y = resid(lmPreProcFit), x = predict(lmPreProcFit, modelData))) +
    geom_smooth() +
    scale_x_continuous(name = 'Predicted') +
    scale_y_continuous(name = 'Residual') +
    geom_hline(size = 0.2, linetype = 'dashed')

library(grid)
library(gridExtra)
grid.arrange(residPlot1, residPlot2, ncol = 2, main = "Residual Plots")
```



Although the residual plots show very poor performance for predicted values below a value of 3, we stick to this model for now. A nonlinear modelling approach should be tried here, this task is on the TODO list.

```
enrichedStats <- enrichAdjGrade(adjGradeData)

# Merges adjusted grade in stats
mergedStats <- merge(stats, select(
    enrichedStats, matchId, playerId, adjGrade),
    by = c('matchId', 'playerId'),  all.x = TRUE)
summary(mergedStats$adjGrade)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -3.213  -0.632   0.011   0.000   0.642   3.122   27141
```

The adjusted grade is calculated by subtracting the achieved kicker grade by the estimated grade predicted by the model. The bigger the value for adjGrade, the better the performance of the regarded player in the associated match.

Enrich stats by an estimated form value for the upcoming match

In the next step the before calculated adjusted grades are used to estimate a form parameter for every player in each upcoming match. To acomplish a valid form parameter, an arima time series analysis is performed, the past performances, represented in the adjusted grade, are passed to the time series analysis to predict the performance for the upcoming match. There have to be at least 5 past matches to make a form calculation

possible. Matches in which the regarded player did not participate, or in which he sit on the bench are imputed with a static value.

```
source(file = 'formModel.R', echo = FALSE, encoding = 'UTF-8')
formEnrichedPlayerStats <- enrichForm(mergedStats, matches, 'arima', minMatchdays = 5, imputeBenchBy =
summary(formEnrichedPlayerStats$playerForm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -3.464  -0.182   0.000  -0.029   0.000   6.585   11313
```

Feature extraction for the final models

The player prices and form parameters are teamwise averaged. The resulting dataset contains matches as observations with the features homePrice, visitorsPrice, homeForm and visitorsForm.

```
source(file = 'simple/simpleResultFeatureExtraction.R', echo = FALSE, encoding = 'UTF-8')
featuredMatches <- simpleMatchFeatureExtract(
    formEnrichedPlayerStats, matches)
# Replace NANs with NAs
featuredMatches[is.nan(featuredMatches$homeForm), 'homeForm'] <- NA
featuredMatches[is.nan(featuredMatches$visitorsForm), 'visitorsForm'] <- NA
```

These predictors by themselfs are not very meaningful, so they are transformed to reflect the relation of the home team to the visitors team.

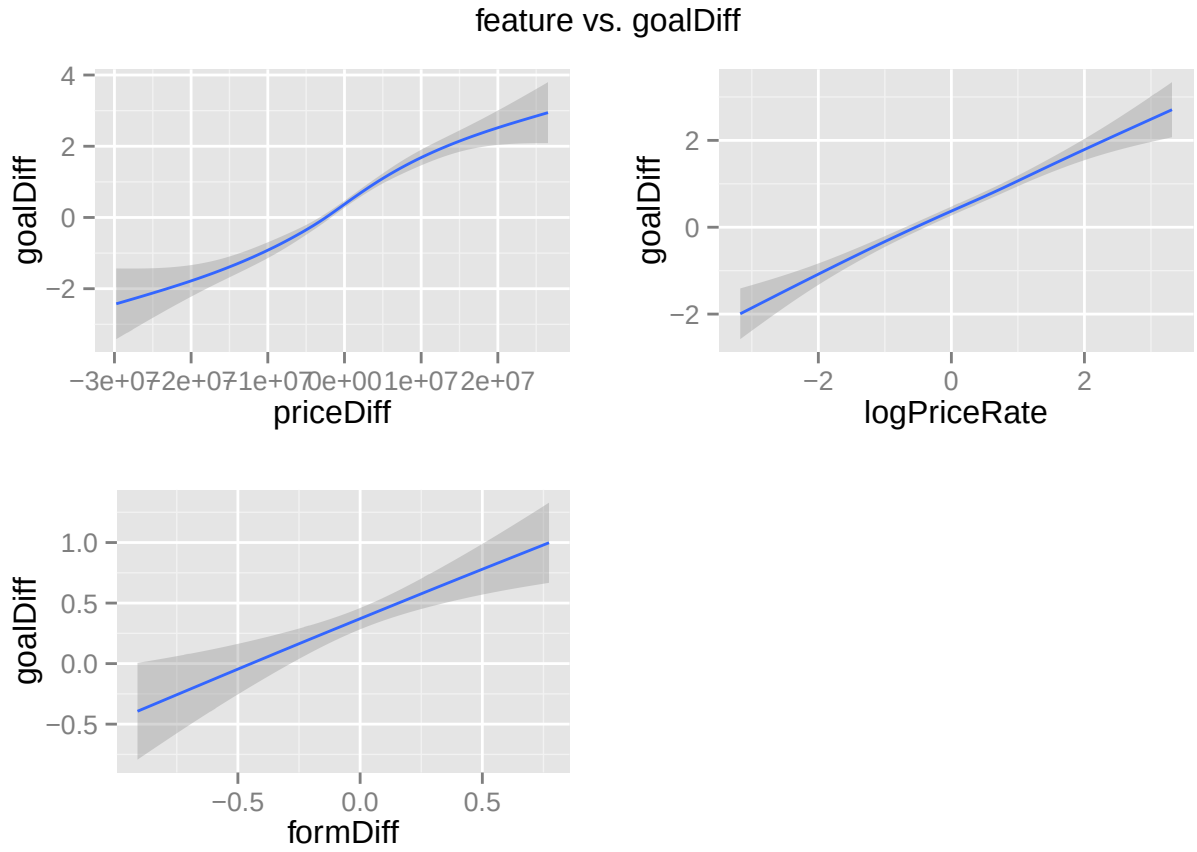
```
modelData <- featuredMatches %>% filter(!is.na(homePrice), !is.na(visitorsPrice),
                                 !is.na(homeForm), !is.na(visitorsForm))
modelData <- modelData %>% mutate(priceDiff = homePrice - visitorsPrice,
                                 logPriceRate = log(homePrice / visitorsPrice),
                                 formDiff = homeForm - visitorsForm)

library(psych)
describe(select(modelData, homePrice, visitorsPrice, homeForm, visitorsForm,
                priceDiff, logPriceRate, formDiff))
```

```
##               vars    n       mean         sd     median    trimmed
## homePrice        1 1820 4795353.36 4085294.72 3717857.14 4021675.70
## visitorsPrice    2 1820 4820339.22 4187590.85 3715178.57 4025657.65
## homeForm         3 1820       0.00       0.16      -0.02      -0.01
## visitorsForm     4 1820       0.00       0.16      -0.02      -0.01
## priceDiff        5 1820  -24985.86 5918435.39  -15384.62  -12153.14
## logPriceRate     6 1820       0.00       0.98      -0.01       0.00
## formDiff         7 1820       0.00       0.23       0.00       0.00
##                     mad          min         max       range  skew
## homePrice     2483355.00    610714.29 31296428.57 30685714.29  2.74
## visitorsPrice 2488650.00    532142.86 32357142.86 31825000.00  2.79
## homeForm            0.14        -0.87        0.72        1.59  0.28
## visitorsForm        0.14        -0.59        0.69        1.28  0.51
## priceDiff     3658845.00 -29764285.71 26542857.14 56307142.86 -0.12
## logPriceRate        1.02        -3.17        3.32        6.49  0.01
## formDiff            0.19        -0.91        0.77        1.68 -0.11
##               kurtosis        se
```

```
## homePrice          9.56  95760.79
## visitorsPrice     10.06  98158.64
## homeForm           1.27      0.00
## visitorsForm       1.36      0.00
## priceDiff          4.56 138730.27
## logPriceRate      -0.16      0.02
## formDiff           0.89      0.01
```

```
p1 <- modelData %>% ggplot(aes(y = goalDiff, x = priceDiff)) +
    geom_smooth()
p2 <- modelData %>% ggplot(aes(y = goalDiff, x = logPriceRate)) +
    geom_smooth()
p3 <- modelData %>% ggplot(aes(y = goalDiff, x = formDiff)) +
    geom_smooth()
grid.arrange(p1, p2, p3, ncol = 2, main = "feature vs. goalDiff")
```



Fitting final models

To predict the probabilities of the three possible outcomes (home victory, draw, visitors victory), a two step approach is implemented. First, the goal difference (home - visitors) is estimated by the three predictors *priceDiff*, *logPriceRate* and *formDiff*. In the second step, the categorical match result is estimated by the estimated goal difference, predicted by the first model.

First the data is split in three parts: 1. Train set for the goal difference model (40% of the data) 2. Train set for the result model (40% of the data) 3. Test set to evaluate the predictions against the booky predictions (20% of the data)

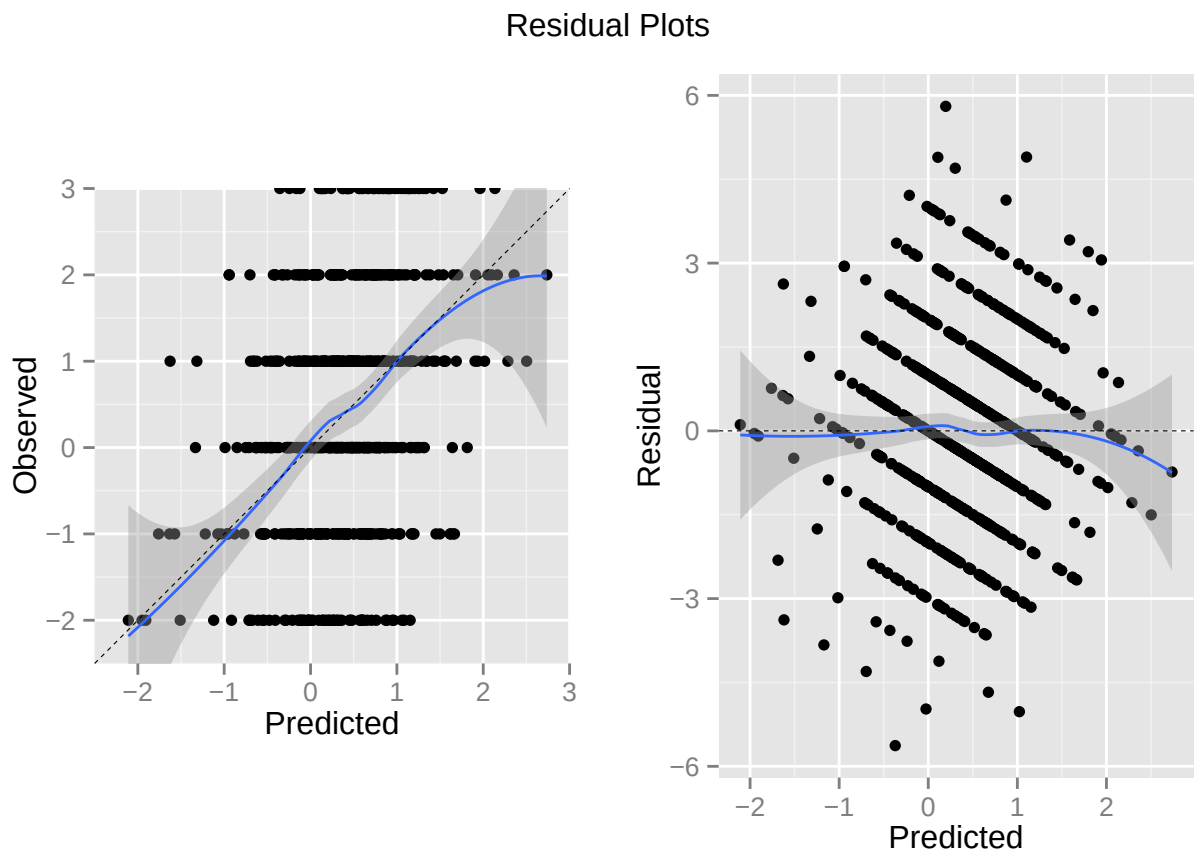
```
set.seed(1)
testIndex <- createDataPartition(modelData$goalDiff, p = 0.2,
                                 list = FALSE,
                                 times = 1)
test <- modelData[ testIndex, ]
train <- modelData[ -testIndex, ]
set.seed(1)
goalDiffTrainIndex <- createDataPartition(train$goalDiff, p = 0.5,
                                          list = FALSE,
                                          times = 1)
goalDiffTrain <- train[ goalDiffTrainIndex, ]
resultTrain <- train[ -goalDiffTrainIndex, ]
```

Fitting Goal Difference Model

We first fit a linear model:

```
set.seed(1)
lmGoalDiffFit <- train(goalDiff ~ priceDiff + logPriceRate + formDiff, method = 'lm',
               data = goalDiffTrain, trControl = repCVControl)
summary(lmGoalDiffFit)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6294 -1.0738 -0.0499  1.0635  5.8039
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.867e-01  6.359e-02   6.082 1.93e-09 ***
## priceDiff    5.348e-08  2.344e-08   2.282  0.02278 *
## logPriceRate 3.380e-01  1.411e-01   2.396  0.01683 *
## formDiff     7.160e-01  2.735e-01   2.618  0.00904 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.714 on 724 degrees of freedom
## Multiple R-squared:  0.1314, Adjusted R-squared:  0.1278
## F-statistic: 36.51 on 3 and 724 DF,  p-value: < 2.2e-16
```



Although the residual plots show, that a linear model is suitable for the data, we additionally fit a neural network and a multivariate adaptive regression spline model (not shown here). But in terms of R^2 and RMSE the linear model outperforms the other two models.

Fitting Result Model

The second stop fits a categorical model to predict the match outcome by the estimated goal difference. I chose a ordered logistic regression (polr), because it is the only model I know which takes an ordered categorical outcome into consideration.

```
resultModelInput <- resultTrain %>% mutate(goalDiffPred = predict(lmGoalDiffFit, resultTrain))
set.seed(1)
polrFit <- train(matchResult ~ goalDiffPred, method = 'polr', data = resultModelInput, trControl = repC
polrFit
```

```
## Ordered Logistic or Probit Regression
##
## 727 samples
##  24 predictor
##   3 classes: 'VisitorsVictory', 'Draw', 'HomeVictory'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
##
## Summary of sample sizes: 654, 654, 654, 654, 655, 654, ...
```

```
##
## Resampling results
##
##   Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.5066654  0.1674958  0.04317136   0.07434475
##
##
```

Results

To evaluate the predictions, the propabilities for the outcomes of the testset are calculated. These propabilities are passed to the fuction evaluatePrediction which calculates statistics to compare the pridictions with real booky odds.

```
# Predicting testset
testSet <- data.frame(goalDiffPred = predict(lmGoalDiffFit, test))
testSet$matchResult <- test$matchResult
testSet$matchId <- test$matchId
testResult <- predict(polrFit, testSet, type = 'prob')
testResult$matchResult <- testSet$matchResult
testResult$matchId <- testSet$matchId

# Evaluation
source(file = 'evaluatePrediction.R', echo = FALSE, encoding = 'UTF-8')
evaluations <- evaluatePrediction(testResult)
printEvaluation(evaluations)
```

```
## [1] "Stake: 136"
## [1] "Gain: -13.73"
## [1] "Gain [%]: -10.0955882352941"
## [1] "Value Diff [%]: -3.51477149623352"
## [1] "Accuracy [%]: 51.7808219178082"
## [1] "Booky Accuracy [%]: 51.7808219178082"
```