1. **Previous steps**

So first when I started with the data I found that this was probably one of the better models for the whole data:

Reg = glm(yt\_mcurrent ~ 0 + ratingdiff + ratingdiffCurrentSurface + DummyBo5TimesAvgRatingdiff + RetiredOrWalkoverDiff +FatigueDiff + HeadtoHead, data = xt\_mcurrent, family = binomial)

Yet problems started arising when I took the uncertainty parameter in account, since eventually I will probably want to take a value in the 20-40% uncertainty quantile of the train\_model test. Yet for these values the HeadtoHead score and the RetiredOrWalkoverDiff appear to be less relevant.

Then I tried creating models per surface, which found that the use of the variable fatigue is very dependent on the surface, for Grass it is very irrelevant, while for both clay and hard-court it is very relevant. The models I decided to use were:

RegGrass = glm(yt\_mcurrentGrass ~ 0 + ratingClaydiff + ratingHarddiff + ratingGrassdiff + DummyBo5TimesAvgRatingdiff + RetiredOrWalkoverDiff, data = xt\_mcurrentGrass, family = binomial)

RegHard = glm(yt\_mcurrentHard ~ 0 + ratingClaydiff + ratingHarddiff + ratingGrassdiff + DummyBo5TimesAvgRatingdiff + RetiredOrWalkoverDiff  
+FatigueDiff, data = xt\_mcurrentHard, family = binomial)

RegClay = glm(yt\_mcurrentClay ~ 0 + ratingClaydiff + ratingHarddiff + ratingGrassdiff +  
DummyBo5TimesAvgRatingdiff + RetiredOrWalkoverDiff  
+FatigueDiff, data = xt\_mcurrentClay, family = binomial)

For both Grass and Hard it was unclear whether the values of the cross-validated log loss were significantly lower (on some runs they were). Yet the improved winnings on Bankroll seemed very consistent.

Another thing to note is that clay seems difficult to predict even making losses on predictions, so probably these matches will not be used to make forecasts. Yet these ratings might still be useful for making predictions.

1. **Validation Rating parameters**

So now the question arises, what will be important to compare to decide which parameters for rating are best.

Probably I can save the values for the LogLoss out of sample of the 3 individual models and the combined model. I could also save BR, yet that might be too much work let’s see what happens when I try programming.

K <- function(numberOfGames) {

return (250 / (numberOfGames + 5) ^ 0.4)

}

In this function the 3 hyperparameters are the factor, 250. This value is, I believe irrelevant, it decided the differences between the ratings so it only gives the ratings a different scale, but since I use a logistic regression on it afterwards it will not change the quality of the model therefore I will not look for this one in the grid search.

Might be a problem with the expected value is used to update ratings, oh well.

1. **Results grid-search on power 0.22-0.6 and offset 1-20**

**(p.s. I also fucked up and didn’t obtain results of the total model, yet this one is probably irrelevant for my analysis so suck it)**

First thing to notice is that the sample size of Grass is generally small and therefore the results are much less reliable than the ones from FinalHard, which is caused by the small samples I assume.

|  |  |  |
| --- | --- | --- |
| Model | Sample at Uncertainty <= 20% | Variance Logloss 400 trials file |
| Total | 1812 | 0.010207842 |
| Grass | 1078 | 0.012287409 |
| Clay | 180 | 0.035368129 |

This means that the results from clay will be taken a lot more serious, yet the are still quite some big differences in the total set.

To reduce the variance, I will calculated the means of the logloss off blocks of 5x5 to get a better sense of the approximate logloss which is more robust to both variance and slight changes of the rating parameter.

I have decided not to care about the winnings for the rating parameters since these might reward overestimations of chances (since this results in a higher Kelly bet), yet purely look at the Logloss.

Hardcourt:

The lowest value for the logloss in the blocks is obtained at an average offset of 12 and an average power of 0.44, yet there are a few close contenders at the edge of the matrix, for the highest powers allowed. Therefore I feel obliged to expend my gridsearch, so I can look into whether higher powers will reduce the logloss.

Grass:

Grass optimizes somewhere else, at offset 13 and power 0.3 and does not do very well at the edges. This makes sense, there are less games of grass so the optimization of the grass rating needs to do well for people with less games, which means that the K-factor should be relatively bigger I believe. Anyways, we will still look into the results for the expanded gridsearch of Hardcourt since the timedifference in doing this is small

1. **Results grid-search on power 0.22-0.94 and offset 1-20**

In the end I have chosen to go with the parameters: **offset = 12, power = 0.44.**

This choice is made by looking at the hard results, and especially the blocks. Hard is chosen since it has a bigger sample, therefore the results are more reliable than the results of grass. There is more data for the general model, yet the problem is that this will also look at the results of improving forecasts for Hard, yet this is unimportant since there are no plans on batting on clay matches, since these appear to be hard to bet on and are not beating the bookies.

One problem with persists is the variance in forecast results making the results less reliable of a parameter. That is why I have used blocks. The assumption I have made is that slight changes in parameters should give similar results, so this gives a more robust result. **Yet even so I am not convinced that this is the optimal parameter settings and therefore I should consider researching this later again. Possible looking into fixing the viewpoints of winner and loser and maybe bootstrapping it and or cross-validation, unfortunately this could increase searching time by a lot.**

**K =**

|  |  |
| --- | --- |
| **Offset** | **Power** |
| **12** | **0.44** |