GLMboost with iRace tuning Summary

Table of Contents:

1. High-Level Summary

2. In-Depth Explanation

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. High Level Summary:

Part 1:

* Utilizing a GLMboost model with classification thresholds, we can correctly predict 86.67% of game outcomes (0.7 threshold)
* Utilizing a standard GLMboost model (with no thresholds) we can correctly predict the outcome of 68% of games.
  + The baseline was 66% (If positive instances were always predicted)
* The model was very good at predicting when the favorite was going to win.
  + Using the classification thresholds, it ONLY predicted instances where the favorite won and scored a 52/60.

This could be used to the advantage of the user if the goal is to parlay multiple bets together.

* Negative scenarios (underdog winning) would yield a higher return if they hit, but returns may even out if MORE bets were parlayed together. Even if you chose the favorite to win each game.

For example, the following 2 parlays MAY yield the same return

1. Betting 1 underdog to win & 1 favorite to win (2 games)

2. Betting 3 favorites to win (3 games)

Overall: Using classification thresholds, the model was good at predicting instances where the favorite won.

Part 2:

LogSpread\_Favorite is the most important indicator of the favorite winning.

* All else equal, once this feature exceeds 2, the probability of a “favorite winning” prediction increases to 0.70.
* It has a larger, positive impact on the chances of the favorite winning - meaning as the value increases, the chances of a favorite winning prediction increases.

Favorite Wide Receiver score was also an important feature which had a larger positive impact on the favorite winning.

* A higher fav wr score increased the chances of the favorite winning.
* As fav wr score evened out & exceed 0, the probability of a positive model prediction increased to > 0.6 and reached 0.70 once the value reached 4.

Favorite total points against was another important feature.

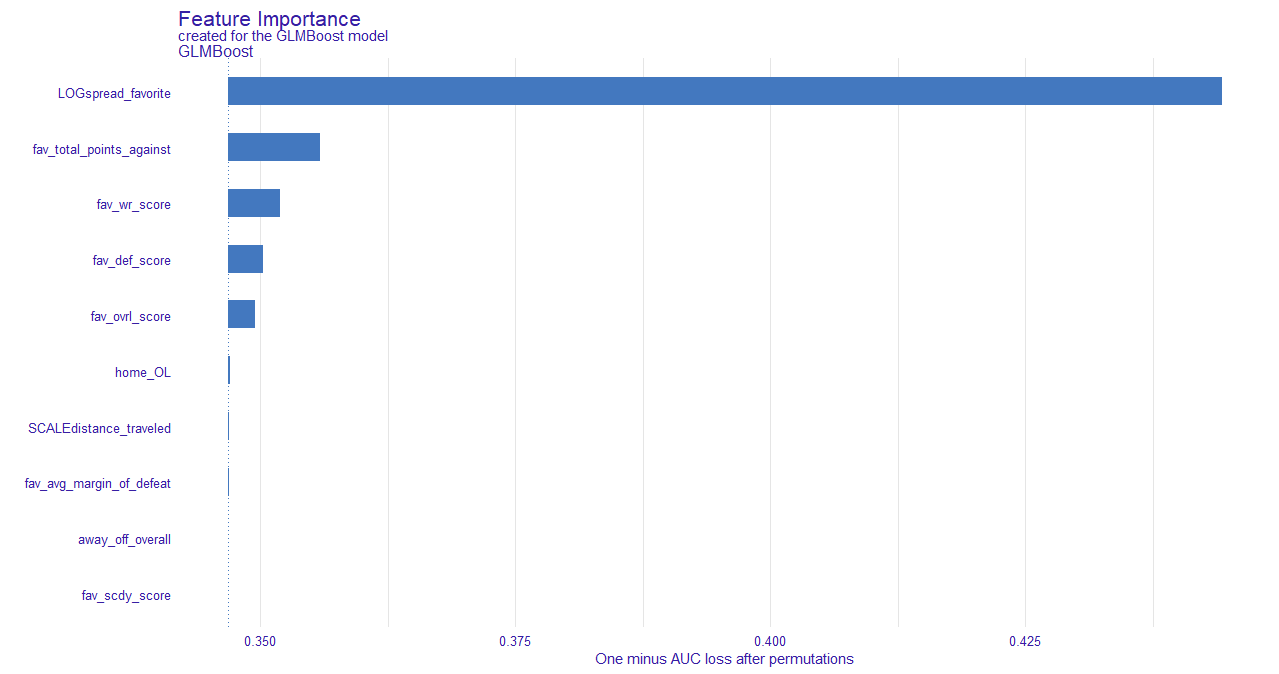
* As this value increased, the probability of a positive outcome increased. (PDPs)
* Consequently, lower values decreased the chances of the favorite winning. (Shap)
  + At first glance this seems counter intuitive. We would think that teams who have not given up many points would be more likely to win (lower points against indicates a higher chance of winning). But the shap plots & pdps suggest otherwise, so consider the following.
* This can be interpreted as - better teams require their opponents to score higher because the observed team is also scoring a larger amount.
* More points required to beat team X suggests that team X scores a larger number of points, thus team X is harder to beat.

Favorite margin of defeat provides a valuable insight that was not anticipated.

* A lower value decreased the chances of the favorite winning.
  + This also seems somewhat counter intuitive. We would think that teams with a smaller margin of defeat would be “better” (more likely to win) since they play in close games.
* This can be interpreted as the team participated in closer games & was not getting "blown out"
* This suggests that the team might have trouble closing out games & performing under pressure
* Since they are losing by smaller margins, they are less clutch - which explains why a lower value decreases the chances of a win.

\*\*Variable Importance, PDPs, & SHAP plots are on the following pages.

Variable Importance:



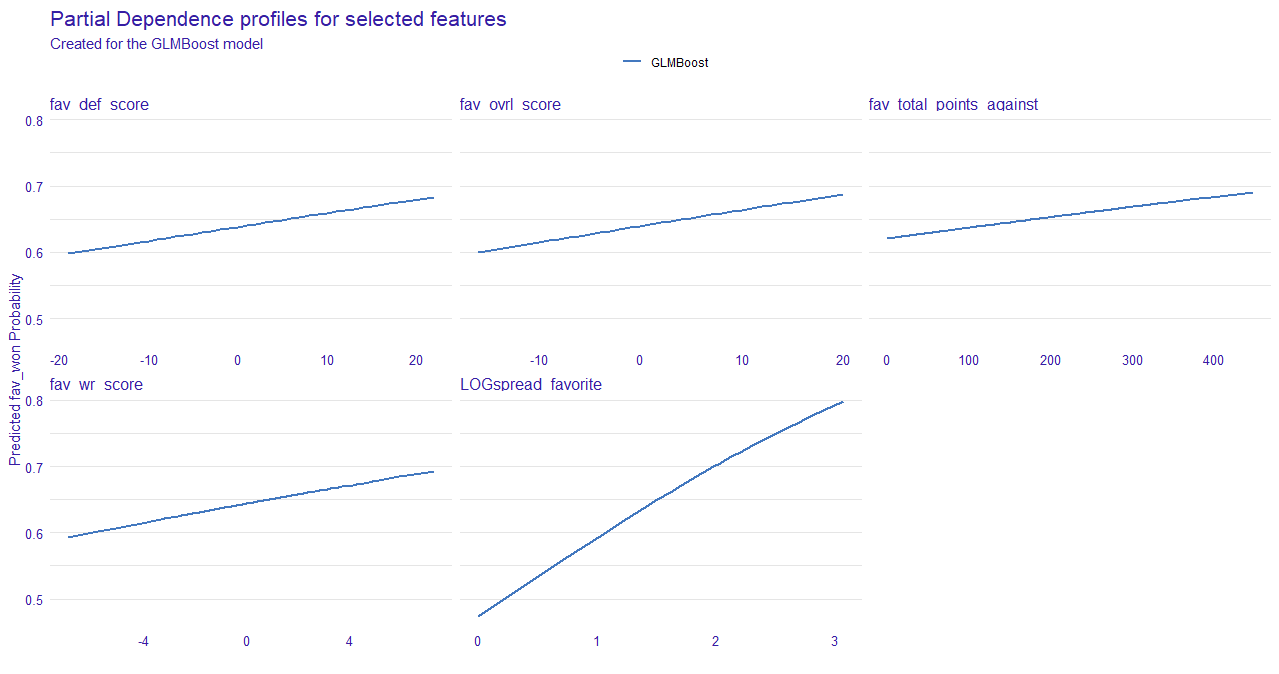
***Explanation:***

* Variable importance based on 1-AUC after permutations
  + This indirectly gives us the variable importance by assessing the AUC after said feature is permutated
    - For instance, if LOGspread\_fav is permutated, 1-AUC = approximately 0.43
      * So as a result, the AUC = 0.57
    - If fav\_total\_points\_agnst is permutated, 1-AUC = approximately 0.36
      * So as a result, the AUC = 0.65
  + Therefore, we can conclude that LOGspread\_fav has more importance since the AUC was lower when this variable was permutated

***Summary:***

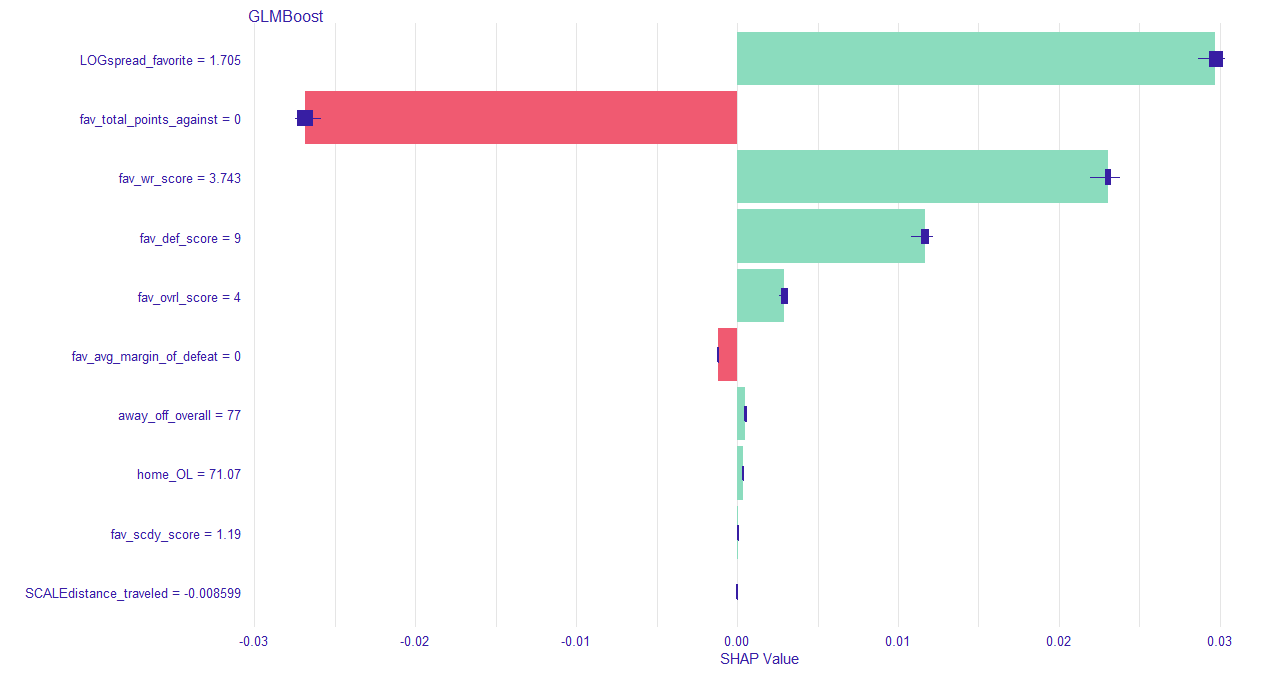
* Log Spread was by far the most important variable, this was based off the Las Vegas Spread & transformed to capture absolute versus relative differences
* Fav\_total\_points\_agnst was the second most important variable, but one of the drawbacks of a simple importance plot is that there is no directionality.
  + Intuition tells us that the greater the log spread, the more likely the favorite is to win. And the more points the favorite gives up in a season, the less likely they are to win. But we will not know until we do further investigation (below).

Partial Dependency Plots



* Above are the partial dependency plots for the 5 most important variables outlined in the previous section.
  + LogSpread\_Favorite
    - We can see that as the value increases, the probability of a positive prediction increases as well. This supports the intuition behind the idea of a “spread”
    - We see that once the value reaches 1.5-2, the probability creeps toward 0.70 & anything above a 2 strongly indicates there is a high probability that the favorite will win the game
  + Fav Total Points Agnst
    - This was an interesting feature
    - It is the cumulative sum of the points scored against the favorite team (keeping each season independent)
    - We would think that as the value increased, the probability of the favorite winning would decrease. Since the team is giving up more points.
      * But in each season, it is inevitable that points will be given up. So, the amount will increase throughout the season
        + Because of this, we should refrain from making any assumptions & refer to the SHAP values before a final interpretation is made
  + Fav wr score
    - As the value approaches 0, the favorite winning probability steadily increases. And once the value surpasses 0 & approaches 4, the probability closes in on 0.70
      * We can conclude that when the favorite wr rating is greater than the underdogs scdy rating, the probability of the favorite winning increases.
  + Fav def score & fave ovrl score
    - Both are also supportive of the idea that a higher score will increase the probability of the favorite winning
      * I.E. a team that has better players should have a better chance of winning the game.

Shap Plot:



* The SHAP plot allows us to interpret the directionality of variable impact.
* As discussed in the XGBoost summary, the SHAP values are the change in log-odds.
  + Log Spread Favorite
    - A high positive impact on the log-odds.
      * This indicates a high value increases the chances of the favorite winning
  + Fav total points angst
    - A high negative impact on the log odds
      * This means a low value decreases the chances of the favorite winning
        + This aligns with/supports this features PDP
        + This could partially be explained because of the following

As the season goes on teams have more and more points scored against them

Teams who tally a larger point total against tend to give up points

But in our cases, said team is favored.

So, this could be explained by teams participating in high scoring games, which tend to have high stakes

If a team has a lower value, it either means they are shutting out the opponent, or the other team is not being forced to score a larger number of points to win the game – meaning the favored team is not putting up many points

If the value is large, teams are having to score more points to be competitive, meaning that the favored team is also scoring a larger number of points

* + Fav wr score & fav def score
    - A high positive impact on log odds
      * A high value increases the chances of the favorite winning
  + Fav average margin of defeat
    - A lower value decreases the chances of the favorite winning
      * A lower margin of defeat means the team participated in closer games, and they were not being “blown out” by the other team
      * A lower value means they have lost close games
        + This could be interpreted as the favorite has trouble “closing out” or performing under pressure, when they are in a tight game
        + Since a lower value decreases the chances of them winning, this may suggest that the favorite has issues winning close ball games – performing under pressure

Which would explain why a lower value decreases the probability of them winning (the favorite is not clutch)

\*\*\*In depth explanations are on the following pages

1. In-Depth Explanation:

***Preprocessing***:

* Packages are loaded
* Data is loaded, shuffled, & split into training & testing

***Variable Selection:***

* Used a random forest model on the training data to reduce the feature
  + OOB error for each data point is recorded
    - Prediction error via bagging
    - Bagging: Subsamples with replacement creates the training samples
    - The model is trained on the bagged observations
    - The OOB error is calculated on the observations OOB
    - The selected feature values are permuted for the training data (noise/randomness is added) then the OOB error is computed again
    - The difference between the OOB error before & after the permutation is calculated & this is how importance is determined
      * Mean Decrease Accuracy (MDA)
        + How much the accuracy decreases after the process above
      * Mean Decrease Gini/Impurity (MDG)
        + How much the node purity decreases after the process above
    - Kept variables with an MDA > 0 & an MDG > 5

***Preprocessing 2:***

* Created the learning tasks for the mlr3 package
* Created the learner for mlr3 (classif.glmboost) with prediction probabilities returned
* Set the parameter search spaces
* Set the evaluation metric (Logloss)
* Set the resampling method to be used (Repeated cross validation (3 repeats on 10 folds))
* Defined a training budget (200 evals (max trials))
* Set the tuning instance
* Set the tuner (irace) – Could be random search, grid search, etc.
* Tuned the model

***Explanations:***

***Learner***: GLMBOOST

* A generalized linear model fitted using a boosting algorithm based on component-wise univariate linear models (rdocumentation.org/packages/mboost)
  + Gradient boosting for optimizing arbitrary loss functions where component-wise linear models are used as base learners (mboost CRAN documentation)

Explanations:

* Generalized Linear Models (GLMs)
  + Used when there is a nonlinear relationship between x & y.
  + Generalizes a linear model by relating the model to the response through a link function.
* There are 3 parts of a GLM model
  + Linear Predictor (η)
    - Linear combination of an unknown parameter & feature variables
    - η = Xβ
    - Unknown parameter β, independent variables, X.
    - Goal is to integrate information about the independent variables in the model
  + Link Function (Xβ=g(μ)): Logit
    - Provides the relationship between the linear predictor (η) & the mean of the distribution function (Binomial distribution in this case)
    - Link functions differ depending on the distribution used.
    - For a Binomial distribution, a logit link function is used.
      * Where the linear predictor (η) = Xβ = ln(μ/(n-μ))
        + And where the mean of the distribution function

μ = (1/(1+exp(-Xβ)))

* + - * + And n = +/- occurrences
  + Probability Distribution: Binomial
    - Binomial is the preferred choice for binary classification (Buehlmann and Hothorn (2007))
* These GLMs are fitted using a boosting algorithm based on component wise univariate linear models
  + Component wise boosting just means “model-based” boosting
  + Univariate linear models mean one feature
    - * Each “model” only has one feature
    - So, the boosting is done on the univariate model-based level
    - Just like XGBoost boosts trees, model-based boosting uses a selection of important “base learners”
      * Base Learners are basically just the effect that certain features have on the target variable (Univariate – one at a time)
    - During each boosting iteration, all base learners are fitted & new base learners are selected by selecting the learner with the smallest error

(See: “About component-wise Boosting” compboost)

***Resampling***: Repeated Cross Validation

* Folds are defined (10)
* Each fold is used as the held-back testing data
* All other (9) folds are used as the training data
* A total of 10 models are fit & evaluated on the held back testing data
  + The mean performance is reported
* In repeated CV, this is done for each repeat (3)

***Tuner***: iRace (Iterated Racing)

* Could have used other methods such as random search/grid search
* Chose Iterated Racing

Background:

References:

* *“The irace package: Iterated racing for automatic algorithm configuration”*
  + *(López-Ibáñez, Dubois-Lacoste, et al. 2016)*
* *“A Racing Algorithm for Configuring Metaheuristics”*
  + (*Birattari, Stutzle, et al. 2002)*
* Tuning used to be done in an ad-hoc fashion
  + This had many drawbacks
* Automatic algorithm configuration was developed
  + The goal was to find configurations that minimize some cost measure
  + The goal was also to find the configurations that generalize to similar, but unseen instances
  + There are many methods for automatic algorithm configuration (tuning)
  + One of these methods was Iterated Racing
    - Racing: Selects one configuration among several candidates using sequential statistics
      * + Candidates being parameters.
      * I.E. quickly reducing the number of candidates & focusing on more promising candidates through statistically guided experiments, while minimizing the number of experiments
      * Dropping inferior candidates speeds up the procedures
    - iRace implements the I/F Race Algorithm (Second Reference)
    - The I/F Race Algorithm is based on the Freidman Test as the statistical method for hypothesis testing & ranking of the candidates
      * With respect to the Friedman Test, in the case of I/F Race testing…
        + The null hypothesis is that all possible rankings of each candidate within each block are equally likely for a given iteration

If the null is not rejected (the rankings are equally likely at some significance level[[1]](#footnote-1)), all the candidates tested will be passed to the next iteration

If the null is rejected & the candidate ranks are not equally likely, pairwise comparisons are executed between the best candidate & all the other candidates in that iteration – all candidates that are significantly worse than the best is dropped & will not appear in the next iteration.

* + - * This is essentially how irace tuning works. It gets the name since the candidates are “racing” against each other & the winners are the only candidates passed on to the next iteration.

***Tuning Visualizatoin***Chart, line chart

Description automatically generated

* We can see the evaluation metric (log loss) decreased along with more tuning
  + Here “batch” refers to the resampling (3 repeats with 10 folds)

Chart, scatter chart

Description automatically generated

* We can see the different tuning parameters & their respective batch’s log loss value.
  + Logit, GLM, Binomial(only one) outperformed their counterparts
  + An mstop around 275 was ideal – stopping parameter, optimal number of boosting iterations
  + An offset around 0.0025 was ideal. This is used as a covariate that is added to the predictions.

***Final Parts of Explanation:***

* The tuning parameters were set on the model
* The model was then used to predict on the testing data.
* Evaluated the standard predictions
* Then incorporated classification thresholds
* Model visualization/interpretation tools were deployed

As mentioned on the code, there were multiple references for this project:

* <https://cran.r-project.org/web/packages/mboost/index.html>
* mlr3extralearners.mlr-org.com
* StackExchange
* GitHub
* *“The irace package: Iterated racing for automatic algorithm configuration”*

*(López-Ibáñez, Dubois-Lacoste, et al. 2016)*

* *“A Racing Algorithm for Configuring Metaheuristics”*

(*Birattari, Stutzle, et al. 2002)*

* (Buehlmann and Hothorn (2007))
* “About component-wise Boosting”

1. * Along with the rankings being equally likely, T has to be approximately X2 distributed with n-1 degrees of freedom. Defining T & X, along with the equation required to evaluate the pairwise comparisons of the candidates would have dramatically increased the complexity & scope of this summary. Both equations are in the paper referenced. *“A Racing Algorithm for Configuring Metaheuristics”*

   [↑](#footnote-ref-1)