Unlocking Rise-Fall-Peak: Evaluating XGBoost, TabNet and other models in Predicting the post bounce peak ball height

This paper is a Preprint

Example Citation for manuscript : *Liyanage, S. (2022). Unlocking Rise-Fall-Peak: Evaluating XGBoost, TabNet and other models in Predicting the post bounce peak ball height.*

Abstract

Tennis is an open skill sport where players often have a range of choices on shot selection (Wang, Chang, 2013). One potential choice relates to how early or late a player decides to (or is forced to) hit the ball, often these decisions are referred to as 'taking the ball on the rise', 'hitting the ball at the peak of the bounce' or 'letting the ball drop'. Tennis coaches and commentators often speculate about the advantage or disadvantage of each option for a player from a strategic perspective.

An inhibition to validating these claims is due to data not readily being available on how early or late a player takes the ball in relation to the post bounce peak ball height. Hawk-eye datasets provide the data point for post bounce peak ball height, but when the player makes contact with the ball early, the value of the post bounce peak ball height is the same as where the player makes contact with the racquet, hence inhibiting the accurate calculation of how early a player makes contact with the ball.

In this paper, various models were trained on Hawk-eye data and evaluated in order to determine the best model to predict post bounce ball location. Two separate approaches were examined, the first a multi-output regressor prediction (MORP) where the models would predict the location (x,y,z) as one prediction, the second approach a separate regression prediction predicted the post bounce peak horizonal, vertical and lateral (x,y,z) separately. It is hoped that one or both approaches with sufficient model performance can unlock the ability to calculate balls taken on the rise, at the peak or on the drop.

A range of models starting from simple linear regression models to more complex Neural Networks like TabNet and Advanced Tree based models like XGBooost were examined in this paper.

We found that the XG Boot model performed the best under the MORP approach when considering the RMSE, R^2 and Training time for the model, whilst XG Boost also performed better under the SRP approach in predicting the x and y axis, and was equal in performance with TabNet in predicting the z axis.

The MORP approach is recommended if you want to calculate both the vertical and horizonal difference from post bounce peak height, where as the SRP approach may be preferred to calculate only the vertical height difference.

Introduction

The model will predict the x, y, z coordinates of the post bounce peak height (peak height)- hit_peak_x, hit_peak_y and hit_peak_z using variables directly from and derived from the Hawk-eye dataset.

From the Hawk-eye dataset you can attain the peak height of the ball before contact is made with the racquet however you can only use it to determine shots hit after the peak height, because the peak provided will always match the contact when the ball is taken early (e.g., hit_peak_z = hit_z if the ball is hit on the rise). Hence, you will not be able to determine *how early* the player took the ball directly with the Hawk-eye dataset. To get around this dataset shortfall, predicting the peak height is required.

We will take two different prediction approaches (figure 1):

- a Multi Output Regressor prediction – e.g. treat x,y,z coordinates as one prediction (MORP)
- 3 separate regression predictions – e.g. predict x, y, z separately (SRP)

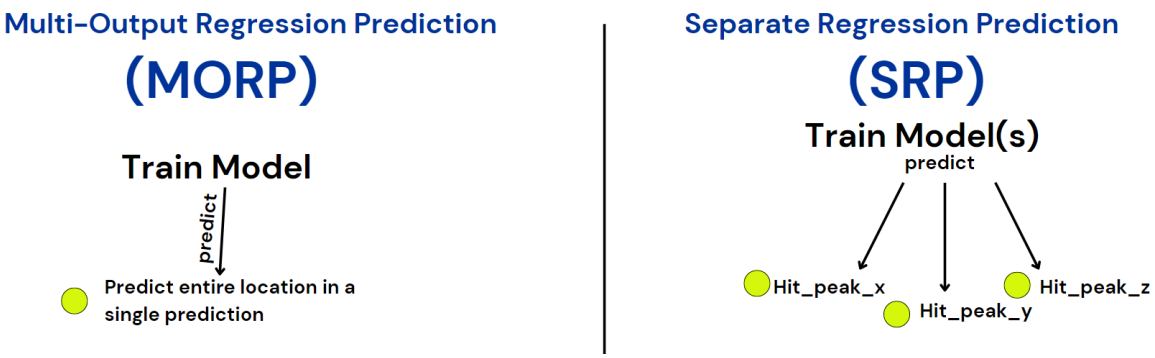


Figure 1 MORP and SRP prediction approaches

Why predicting post bounce peak height is valuable and its potential application

Being able to tell if player is contacting the ball earlier, around the peak or later than the ball reaching its peak bounce height (on the rise vs at the peak vs on the fall) can help players understand their game better as well as help understand the game style of opponents they will face.

Additionally knowing the relationship between where the player took the ball and where the ball was going to bounce up to, used alongside other metrics/statistics like winners, forced errors, unforced errors, spin rate off the racquet, speed of shot off the racquet and time pressure can help provide context for tv audiences and performance coaching.

Once the post bounce peak height is predicted, the difference of the of z axis values with racquet contact can determine vertical height difference between contact and post bounce peak height.

The distance between the contact location and where the ball is predicted to have peaked post bounce can be determined using Pythagorean theorem to find distance between two coordinate points.

Rise-Peak-Fall

Shots taken on the rise (rise), at the peak of the bounce (peak) or on the fall (fall) can be worked out by determining contact location, one example could be to split rise, peak and fall based on 5% horizonal distance from hit_peak_x like figure 2.

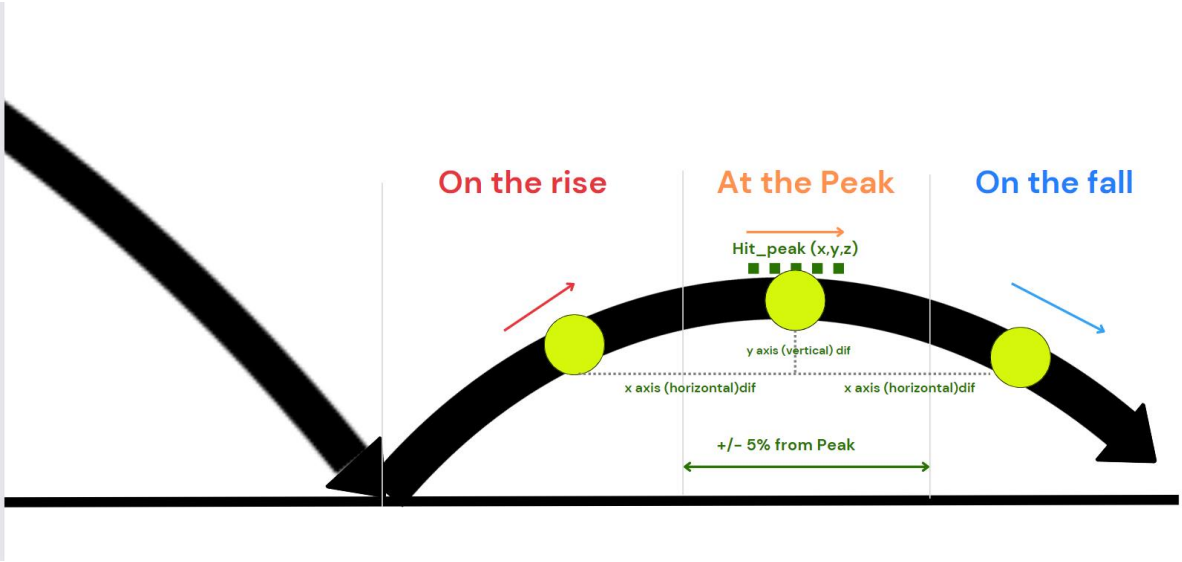


Figure 2 Calculate shots taken on the rise, at the peak and on the fall once post bounce peak (hit_peak) is predicted

Method

From the Hawk-eye Json files extract relevant variables relating to bounce, speed, spin, ball contact and peak bounce height, net clearance, whether the ball clipped the net and velocity off the court for each event.

Manipulate the data to bring the previous shot info for bounce, speed, spin, ball contact and net clearance to the current event. Derive the distance to the net and the distance to the bounce of the ball from the previous shot. Create a Boolean variable (hit_peak_match) looking at if contact point is equal to peak bounce height in the Hawk-eye dataset (hit_z = hit_peak_z).

Filter the dataset to be used for training to only have hit_peak_match = FALSE. This is because we only want situations where the peak bounce height is known and does not equal the contact point of the ball. We will use this data to train and validate

the prediction model and then apply the model to the rows of data where hit_peak_match = TRUE (see figure 3).

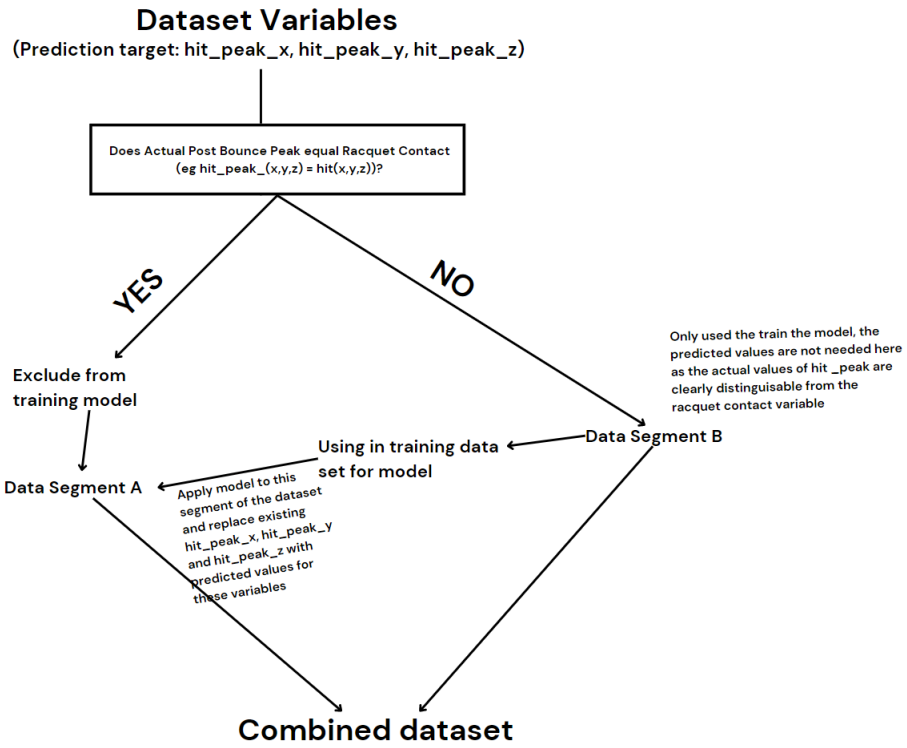
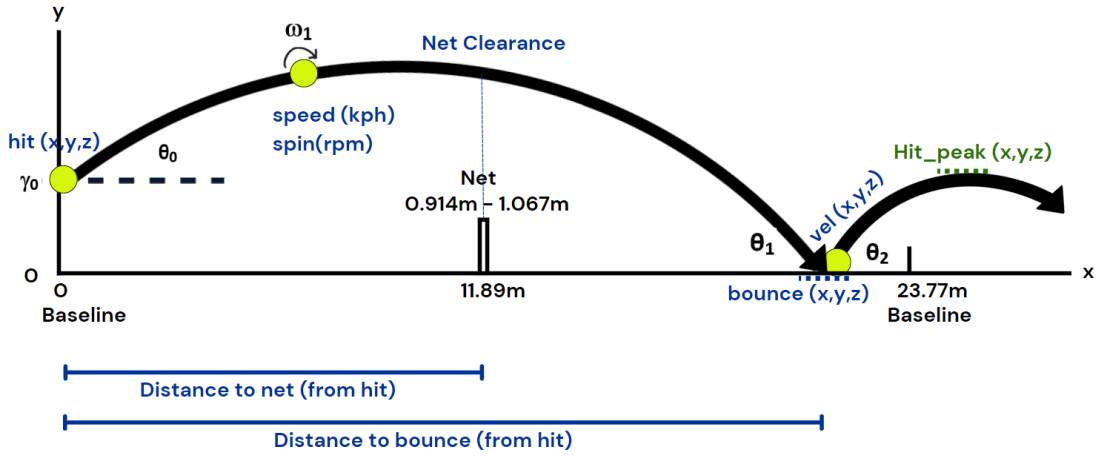


Figure 3 Preparing the dataset – Training on data where contact and post bounce peak is not the same, then applying on data where pounce bounce peak and contact is the same

Dataset

Variable selection

The variables (Figure 1) were selected in consultation with a subject matter expert (High Performance Coach) and evaluation of feature importance to predicting hit_peak_x, hit_peak_y and hit_peak_z (the point where the bounce is at the peak).



eventId, prevshot.bounce_x, prevshot.bounce_y, prevshot.bounce_z, 'hit_peak_x', 'hit_peak_z', hit_peak_match', 'prevshot_distance_to_net', prevshot_distance_to_bounce, prevshot.speed.kph, prevshot.spin.rpm, prevshot.net.clearance, prevshot.clipped.net, prevshot.hit.x. prevshot.hit.y, prevshot.hit.z, prevshot.bounce.vel.x, prevshot.bounce.vel.y, prevshot.bounce.vel.z

Figure 4: Variables used for predicting peak bounce height

Over 500,000 rows contacting bounce peak events were used in the data sample for the prediction. The distribution of each variable presented in Figure 5, with the correlation presented in Figure 6.

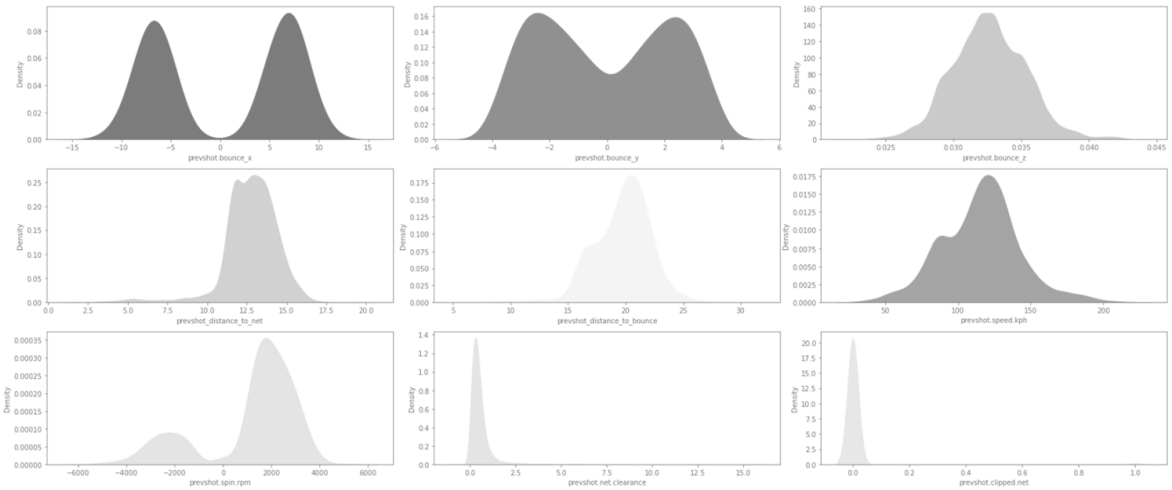


Figure 5 Variable Distributions: Input Data Size: (504506, 17)

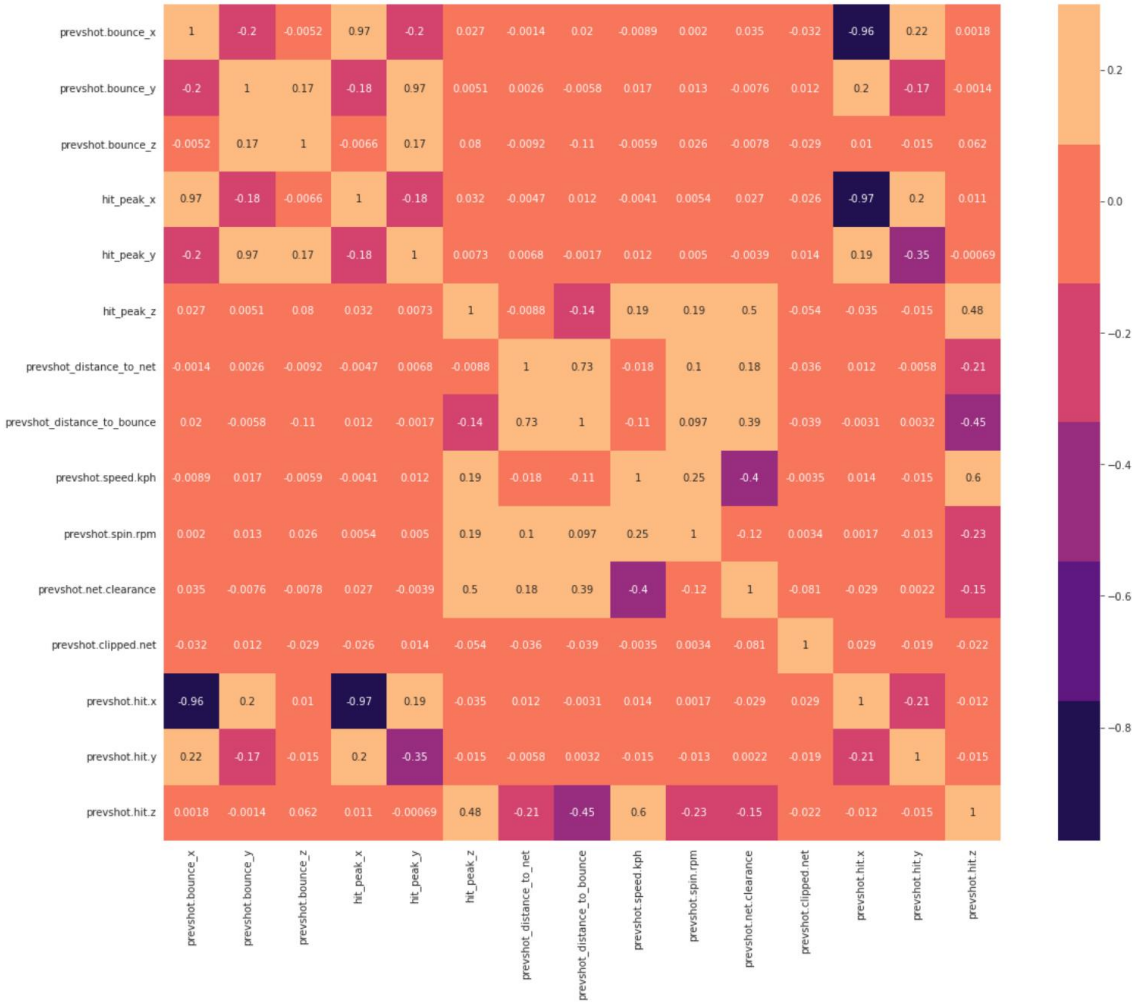


Figure 6 Correlation of variables plot

Under MORP approach a single prediction of the location is sought using a multi-output regressor, while under the SRP approach hit_peak_x, hit_peak_y, hit_peak_z are our (x,y,z) coordinates target that require separate predictions.

Data Transformation and Cleaning

In the dataset the rows which have null values were removed and data types were unified for input into the various machine learning models.

Machine Learning Models and their Evaluation

We evaluate in this paper some simple prediction models and some complex machine learning models and compare their performance for both the MORP and SRP approach.

Evaluation Metric

Four evaluation metrics will be primarily used for our regression analysis

1) Mean Squared Error (MSE)
   MSE is the average of the square of the errors. The larger the number the larger the error. Error in this case means the difference between the observed values and the predicted ones.

2) Root mean squared error (RMSE): is the square root of the average of squared differences between prediction and actual observation.
   The lower the MSE and RMSE the better the model.

3) R^2 Score
   R^2 Score is a statistical measure of fit that indicates how much variation of a dependent variable is explained by the independent variable(s) in a regression model. The higher the R^2 the better the model.

4) Training time
   Training time is also used to compare the models give some additional context to the effort required to train the model.

Results

MORP Approach

Machine Learning Models

The machine learning models that we will be using are as follows with the MORP evaluation result outputs presented.

1) L1 Linear regression (Lasso Regularization)

   A simple linear regression model (Schneider, Hommel, Blettner, 2010) is developed using L1 (Lasso) regularization. L1 regularization is a technique for reducing the complexity of a model by adding a penalty term to the loss function that is proportional to the sum of the absolute values of the weights. This can help to reduce overfitting and improve the interpretability of the model (Vidaurre, Bielza, Larrañaga,2013).

   This model can be a comparison to more complex models.

   The MSE for Linear Regression Lasso is: 1.9074, the RMSE is: 1.3811 and r2 score is: 63.8385%

2) L2 Linear Regression (Ridge Regularization)

   A linear regression model with L2 Regularization (Ridge Regularization). L2 regularization adds a penalty term to the loss function that is proportional to the square of the weights whereas L1 regularization adds a penalty term to

the loss function that is proportional to the sum of the absolute values of the weights (Cortes, Mohri, Rostamizadeh, 2009)

The MSE for Linear Regression Ridge is: 1.8029, the RMSE is: 1.3427 and r2 score is: 82.3219%

3) Simple Random Forest

A random forest is an ensemble learning method that can be used for regression predictions. It is called a random forest because it is made up of a collection of decision trees, each of which is trained on a random subset of the data (Brieman, 2001). For this model we will set a max depth of 2. This means that each decision tree in the forest is limited to having a maximum of 2 levels, or 2 "splits" on the input data. It is considered relatively shallow, and therefore each tree will have a simpler structure. This can be advantageous because it can help to prevent overfitting and improve the interpretability of the model (Nadi, Moradi, 2019).

The MSE for Simple Random Forest is: 1.6980, the RMSE is: 1.3031 and r2 score is: 72.4230%

4) Multi-Layer Perceptron (MLP) Neural Network

An MLP is a neural network made up of multiple layers of interconnected neurons. An MLP has an input layer, one or more hidden layers, and an output layer. The input layer receives the input data, and each subsequent layer transforms the data using an activation function, until the output layer produces the final output of the model. MLPs are trained using a variant of the backpropagation algorithm, which adjusts the weights of the connections between the nodes in order to minimize the error between the predicted output and the actual output (Ramchoun, Idrissi, Ghanou, Ettaouil, 2016)

In this model we used drop out regularization to ensure the model learned a more general representation of the data and therefore generalises better to unseen data (i.e. prevent overfitting). RELU activation is also used to improve the performance of the model and deal with vanishing gradients problem (Byrd, Lipton,2019).

The model has 500 iterations with early stopping enabled.

The MSE for MLP is: 0.8253, the RMSE is: 0.9084 and r2 score is: 82.8666%

5) Random Forest with Hyperparameter optimization

The random forest is a collection of trees so when ensembling such model there is a need to optimize the hyper parameter.

Hyperparameter tuning refers to the process of selecting the optimal values for the hyperparameters of a model. Hyperparameters are parameters that are not learned during training but are instead set by the practitioner before training begins. Examples of hyperparameters include the learning rate, the number of trees in the random forest, and the maximum depth of each tree (Probst, Wright, Boulesteix,2019)

Tuning the hyperparameters of a random forest can help to improve the performance of the model and make it more effective at fitting the data. This can be done by using a method such as grid search, which involves training the model with a range of different hyperparameter values and selecting the combination that performs the best (Probst, Wright, Boulesteix,2019).

The MSE for RF Tuned Hyperparameter is: 0.4670 the RMSE is: 0.6834 and r2 score is: 84.3428%

6) Deep Learning Neural Network – TabNet

TabNet is a deep learning architecture for tabular data, which is data that is organized into rows and columns.

TabNet is designed to be an efficient and interpretable model for tabular data. It combines several different techniques, including attention mechanisms, batch normalization, and sparsity-inducing regularization, to achieve good performance on a variety of tasks. It also includes a feature called virtual batch normalization that allows it to adapt to the distribution of the input data in a way that is similar to batch normalization, but without the need to compute statistics over the entire dataset (Arik, Pfister, 2021).

Training on 500 Epochs with early stopping, the following results were achieved

Early stopping occurred at epoch 394 with best epoch = 198 and best_rmse 0.1876

7) LightGBM

LightGBM is a tree based gradient boosting framework model. It uses a histogram-based algorithm to approximate the value of the gradient, rather than computing it exactly. This has the effect of reducing the amount of computation required to train the model and can make it possible to train large-scale models that would be impractical to train using traditional random forest models (Meng, 2017).

The MSE for LightGBM is: 0.4129 the RMSE is: 0.6426 and r2 score is: 86.8931%

8) CatBoost

CatBoost is another tree based gradient boosting framework model, one of the key features of CatBoost is its ability to handle categorical features. CatBoost can automatically encode these features in a way that is appropriate for gradient boosting, which can improve the performance of the model compared to other methods that require the categorical features to be pre-processed (Dorogush, Ershov, Gulin, 2018).

The MSE for CatBoost is: 0.2811 the RMSE is: 0.5302 and r2 score is: 86.2662%

9) XGBoost (eXtreme Gradient Boosting)

XG Boost is another tree-based gradient boosting model. It is an ensemble learning method that involves combining the predictions of multiple models to make a final prediction. Like other decision trees-based models it is well-suited to structured/ tabular data because it can easily handle complex interactions between multiple features. XGBoost also uses regularization techniques like L1 and L2 regularization, which help prevent overfitting and improve the generalizability of the model (Chen, Guestrin, 2016).

The MSE for XGBoost is: 0.0108 the RMSE is: 0.1039 and r2 score is: 99.3612%

Algorithm	MSE	RMSE	R2 Score	Training Time
Linear Regression Lasso	1.91	1.38	63%	0.14 sec
Linear Regression Ridge	1.80	1.34	82%	0.05 sec
Random Forest - Simple	1.70	1.30	72%	63 sec
Neural Network - MLP	0.83	0.91	83%	51 sec
Random Forest Hyperparameter Optimization	0.47	0.68	84%	1 hour
Neural Network - TabNet	0.03	0.18	88%	33 min per epoch
LightGBM	0.41	0.64	87%	10 seconds
CatBoost	0.28	0.53	86%	55 Seconds
XG Boost	0.01	0.10	99%	3 minutes

Table 1 MORP Model Results trained with Google Colabs GPU

SRP Approach

Under the SRP approach the models will be predicting hit_peak_x, hit_peak_y, hit_peak_z independently.

Similar to the MORP approach a linear regression model was used as a baseline comparison. A quick feature importance analysis was also conducted to get an understanding of the importance of each of the variables in predicting the three target variables.

Tuning for better predictions of each hit peak axis

In order to achieve better predictions for each axis, the grid search for hyper tuning was applied to observe which set of hyper parameters the Random Forest shows better metrics for the given axis.

Result Tables

Predicting the X axis of the post bounce peak location

Algorithm	MSE	RMSE	R2 Score	Training Time
Linear Regression Lasso	2.41	1.55	90%	0.11 sec
Linear Regression Ridge	1.89	1.37	93%	0.05 sec
Random Forest - Simple	1.69	1.30	94%	51 sec
Neural Network - MLP	1.09	1.04	90%	54 sec
Random Forest Hyperparameter Optimization	0.67	0.82	91%	55 min
Neural Network - TabNet	0.15	0.38	96%	27 min per epoch
LightGBM	0.60	0.77	94%	7 seconds
CatBoost	0.59	0.76	91%	50 Seconds
XG Boost	0.02	0.13	98%	2 minutes

Table 2 SRP Model Results predicting hit_peak_x trained with Google Colabs GPU

Predicting the Y axis of the post bounce peak location

Algorithm	MSE	RMSE	R2 Score	Training Time
Linear Regression Lasso	1.13	1.06	96%	0.12 sec
Linear Regression Ridge	1.09	1.04	95%	0.04 sec
Random Forest - Simple	0.22	0.47	92%	50 sec
Neural Network - MLP	0.20	0.44	95%	51 sec
Random Forest Hyperparameter Optimization	0.11	0.33	94%	55 min
Neural Network - TabNet	0.02	0.13	96%	27 min per epoch
LightGBM	0.02	0.14	94%	7 seconds
CatBoost	0.02	0.14	97%	47 Seconds
XG Boost	0.01	0.11	99%	2 minutes

Table 3 SRP Model Results predicting hit_peak_y trained with Google Colabs GPU

Predicting the Z axis of the post bounce peak location

Algorithm	MSE	RMSE	R2 Score	Training Time
Linear Regression Lasso	0.08	0.28	54%	1.2 sec
Linear Regression Ridge	0.08	0.28	63%	1.3 sec
Random Forest - Simple	0.07	0.26	60%	48 sec
Neural Network - MLP	0.05	0.22	78%	2 min
Random Forest Hyperparameter Optimization	0.03	0.17	84%	64 min
Neural Network - TabNet	0.002	0.04	98%	33 min per epoch
LightGBM	0.03	0.14	94%	7 seconds
CatBoost	0.02	0.09	97%	47 Seconds
XG Boost	0.002	0.05	98%	2 minutes

Table 4 SRP Model Results predicting hit_peak_z trained with Google Colabs GPU

Discussion

Under the MORP approach the XG Boost model performed significantly better than the other models. It had the lowest RMSE (0.10), meaning the model is making post bounce height predictions relatively close (average difference of 0.1m) to the actual post bounce height. The XG Boost model also had the best R^2 score of 99%, meaning the model can almost perfectly explain the variance in the outcome variable making it a stronger model. While some of the other models had a faster training time, the 3 minutes to train the model on over 500,000 rows of data is not too onerous, and the benefits of the superior RMSE and R^2 outweigh the slightly slower training time.

Under the SRP approach, the models were evaluated separately for performance in predicting the post peak bounce for the x, y, z axis. The XG Boost model was clearly the superior model in predicting the post bounce peak x axis with a much lower RMSE (0.13) and the best R^2 Score of 98%, and the training time of only 2 minutes. With predicting the post peak bounce peak y axis, the results between the models were a lot more converged, with the XG Boost model having the lowest RMSE (0.11) and best R^2 score (99%) with a relatively fast training time. When predicting the post peak bounce peak z axis the TabNet neural network slightly outperformed the XGBoost RMSE 0.04 to 0.05 respectively. Both models had an R^2 score of 98%, but the XG Boost model only had a 2 min training time compared to 33 min per epoch for TabNet.

While the evaluation of whether the MORP or SRP approach is better was not conducted in this paper, either approach provides sufficient accuracy to calculate post bounce peak and to subsequently calculate shots taken on the rise, at the peak or on the fall.

References

Wang C-H, Chang C-C, Liang Y-M, Shih C-M, Chiu W-S, Tseng P, et al. (2013) Open vs. Closed Skill Sports and the Modulation of Inhibitory Control. PLoS ONE 8(2): e55773. https://doi.org/10.1371/journal.pone.0055773

Schneider, A., Hommel, G., & Blettner, M. (2010). Linear regression analysis: part 14 of a series on evaluation of scientific publications. Deutsches Arzteblatt international, 107(44), 776–782. https://doi.org/10.3238/arztebl.2010.0776

Vidaurre, D., Bielza, C., & Larrañaga, P. (2013). A Survey of L_1 Regression. International Statistical Review / Revue Internationale de Statistique, 81(3), 361–387. http://www.jstor.org/stable/43299642

Cortes, C., Mohri, M., & Rostamizadeh, A. (2009). L2 Regularization for Learning Kernels. Conference on Uncertainty in Artificial Intelligence.

Breiman, L. (2001) Random Forests. Machine Learning, 45, 5-32. http://dx.doi.org/10.1023/A:1010933404324

Abolfazl Nadi, Hadi Moradi, Increasing the views and reducing the depth in random forest (2019), Expert Systems with Applications, Volume 138,2019, 12801,ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2019.07.018. (https://www.sciencedirect.com/science/article/pii/S095741741930497X)

Ramchoun, H., Idrissi, M.A., Ghanou, Y., & Ettaouil, M. (2016). Multilayer Perceptron: Architecture Optimization and Training. Int. J. Interact. Multim. Artif. Intell., 4, 26-30.

Byrd, J., & Lipton, Z. (2019, May). What is the effect of importance weighting in deep learning?. In International Conference on Machine Learning (pp. 872-881). PMLR.

Probst, Philipp & Wright, Marvin & Boulesteix, Anne-Laure. (2019). Hyperparameters and Tuning Strategies for Random Forest. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 9. 10.1002/widm.1301.

Arik, S. Ö., & Pfister, T. (2021). TabNet: Attentive Interpretable Tabular Learning. Proceedings of the AAAI Conference on Artificial Intelligence, 35(8), 6679-6687. https://doi.org/10.1609/aaai.v35i8.16826

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. NIPS.

Dorogush, A.V., Ershov, V., & Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. ArXiv, abs/1810.11363.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM