Scoring with Few Shots: Applying Few-Shot Learning to Basketball Analytics

Josh Francis
Stanford University
josfran@stanford.edu

Abstract

One of the most important skills in basketball is shooting, and the quality of shooting form is crucial to success. This paper explores how computer vision can help to evaluate and improve basketball shooting form using a small dataset of stationary videos. We collected and labeled our own dataset due to the lack of existing suitable data, and applied VGG-16 for image feature extraction, YOLOv3 for shooter detection, and MoveNet for pose estimation. Several classification models were tested, including logistic regression, random forest, gradient boosting, MLPs, XGBoost, and a custom CNN + LSTM model. Our results indicate that XGBoost significantly outperforms other models, achieving the highest accuracy and providing meaningful interpretability using SHAP values. This study demonstrates the potential of using computer vision and machine learning to provide predictions and feedback on sports video, paving the way for future advancements in sports analytics.

1. Introduction

One of the most important skills in basketball is shooting, and how well you shoot is very dependent on your shooting form. "Shooting form" consists of how your body is posed as you shoot the ball, so your hip, knee, elbow, shoulder, and wrist angles, the distance between your feet, and the speed of your wrist, arms, and legs pushing the ball through its trajectory towards the hoop. All of these features are visible to an observer, so this paper seeks to evaluate shot form through computer vision techniques.

More formally, we evaluate shot form by using shot form to predict shot success, and then we use weight explainability techniques to determine which parts of the shot form are contributing positively or negatively to shot success. The goal of this is to create a model that lets players upload videos they record of themselves shooting a basketball and get feedback on how to improve their shot form. This means that all the training data is stationary videos of people shooting a basketball, with the hoop, ball, and shooter pose visible. The videos are clipped such that the ball isn't seen going into the basket, so the model can't cheat and learn to recognize the ball going in the basket. Instead, it must learn to predict shot success solely from the pixels of the person shooting the ball.

To do this, we had to collect and label our own dataset of videos, because there is no existing dataset of stationary videos of people shooting by themselves. Data collection and labeling is a hard task though, so in this paper we explore models that optimize shot prediction performance given a small dataset, a.k.a. few shot learning. To do so, we leverage VGG-16 to extract image features from video frames, YOLOv3 to extract shooter bounding boxes in each frame, and MoveNet to extract body keypoints for shooter pose analysis. After extracting these features from each frame, we explore multiple different classification models to see which ones are best able to predict shot success and produce meaningful weight explainability. These models include logistic regression, random forest, gradient boosting, MLPs, extreme gradient boosting, and a custom CNN + LSTM model. We find that extreme gradient boosting significantly outperforms all other methods on the dataset, and produces reasonable weight interpretability using SHAP.

In the next section we discuss the novelty of these results and how they differ from past work in the basketball analytics space.

2. Related Work

Many people have attempted to apply machine learning to solve the problem of basketball shot success prediction in the past. Murakami-Moses (2019) also used MLPs, logistic regression, and gradient boosting to try to predict shot success, but their data was NBA game data, such as player name, position on the court, and time left in the game[9]. Although all of these features correlate with shot success, there are many other important features that are left out, and they don't consider any image data in their approach. Thus, their approaches are capped at around 65% accuracy.

A similar project was done by Brett Meehan in CS229 in 2017[8]. Their dataset also didn't look at images, but it looked at more shot related features like touch time and defender distance. They also employ various statistical mod-

els and ultimately find that gradient boosting had the best performance with 68% accuracy. In their conclusion they noted the potential of using more spatial data in training a model, which is what our paper sets out to do.

Harmon et al. (2021) developed a novel approach that utilized SportsVU data and a combined CNN + FNN model to predict with up to 61% accuracy[4]. SportsVU data provides a lot of basketball related features extracted from a 6 camera system operating at 25 frames per second, such as 3d ball coordinates, all player coordinates, and basket make/miss timesteps. This provides valuable data for training a predictive model, but it still is missing a lot of causal data, such as the player's shot mechanics. Furthermore, although this approach uses a CNN, the CNN is being applied to a fabricated image that represents player and ball trajectories, so it doesn't apply to new videos of basketball shots unless they were recorded in the SportsVU system.

Kuhlman et al. also developed a novel technique of classifying basketball shots based on wearable sensors[7]. Their shot classes were "good" and "bad" 2 pointers and 3 pointers, so there were 4 classes total. A "good" shot form is one that follows generally accepted shooting principles, like lining up your body with the basket and moving the ball up smoothly, and a "bad" shot form was a jerky movement and an unbalanced base. Using SVMs, they were able to classify shot form with a stunning 86.3% accuracy, showing great promise for predicting shot outcomes based off of form features and pose data.

In CS 231a, Chen et al. (2022) were able to reconstruct 3d poses from basketball shooting videos[2]. They didn't end up using these 3d poses for shot prediction, but they showed that their resulting 3d poses had low distance metrics from known reference poses. In this project we will use 2d pose estimation from MoveNet, but this work shows the potential to improve performance by looking at the full 3d structure, which is a lot more representative of the shot form than the 2d pose.

Chou (2020) demonstrated how you can apply OpenPose to extract pose features for use in basketball shot analysis[3]. They further referenced Cao et al. (2019), which proposed OpenPose and the bottom-up approach of pose estimation[1]. Chou first describes top-down as first getting a bounding box for a person before applying pose estimation to the bounding box, whereas bottom-up applies pose estimation to the whole image. Chou outlines the pros and cons of bottom-up versus top-down pose estimation, with top-down suffering from pitfalls of no recovery if person detection fails, but bottom-up being more computationally expensive. Since we only have one person per video in our dataset and they stand out from the background with high confidence, person detection works well enough that top-down methods work for our approach, but Chou and Cao's commentary suggest an extension to this paper in the future of applying OpenPose to the whole frame instead of MoveNet to just the bounding box.

Hauri et al. (2021) utilized a multi-modal LSTM on NBA trajectories to predict future trajectories with state-of-the-art performance[5]. While trajectory prediction is a fundamentally different task than shot classification, it shows that CNN + LSTM models can be applied with great performance on short time span sport prediction tasks. We take inspiration from this in developing a CNN + LSTM for shot prediction in this paper.

Ozkan (2020) also showed a novel concurrent neuro-fuzzy system, which utilized a fuzzy SVM model to achieve nearly 80% basketball play prediction accuracy[10]. While we don't employ these techniques in this paper, Ozkan illustrates how fuzziness can be used to improve model performance when optimizing under uncertainty, as we are in this project.

Finally, Hu et al. applied Extreme Gradient Boosting (XGBoost) to player performance prediction[6]. In their paper, they outline the math behind XGBoost and how it applies to NBA feature data and how to get interpretability out of it. In this paper we will also utilize XGBoost and find that it performs best out of all the models we apply.

Overall, related work gives lots of examples of applying machine learning methods for sports prediction tasks. However, none of these have tried leveraging image data for outcome prediction, and our dataset is unique. Therefore, although we can draw from the techniques in all these other works, how the techniques specifically apply to our problem space is unique and gives this project its novelty.

3. Methods

In this project we employed logistic regression, random forest, gradient boosting, MLPs, XGBoost, and a custom CNN + LSTM model. We describe our approach for each model in this section.

3.1. Logistic Regression

Logistic regression is a linear model for binary classification that predicts the probability that an instance belongs to a particular class. The probability is modeled using a logistic function:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p)}} \tag{1}$$

where $\beta_0, \beta_1, \ldots, \beta_p$ are the model parameters. The model is trained to minimize the binary cross-entropy loss:

$$L(\beta) = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(P(y = 1|x_i)) + (1 - y_i) \log(1 - P(y = 1|x_i))]$$

3.2. Random Forest

Random forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. It reduces overfitting by averaging multiple deep decision trees, trained on different parts of the same training set.

3.3. Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion and generalizes them by allowing optimization of an arbitrary differentiable loss function. The objective function for gradient boosting is:

$$L(\theta) = \sum_{i=1}^{n} l(y_i, F_m(x_i)) + \sum_{k=1}^{m} R(f_k) \tag{2}$$

where l is the loss function and R is the regularization term.

3.4. Multi-Layer Perceptron (MLP)

A multi-layer perceptron (MLP) is a class of feedforward artificial neural network (ANN). An MLP consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Each node (except for the input nodes) is a neuron that uses a nonlinear activation function:

$$y = \sigma(\sum_{i=1}^{n} w_i x_i + b) \tag{3}$$

where σ is the activation function, w_i are the weights, x_i are the inputs, and b is the bias. The MLP is trained to minimize the loss function:

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{4}$$

3.5. Extreme Gradient Boosting (XGBoost)

XGBoost is an implementation of gradient-boosted decision trees designed for speed and performance. It provides parallel tree boosting (GBDT/GBM) which gives it significant speedups in comparison to alternative methods. The objective function for XGBoost is:

$$L(\theta) = \sum_{i=1}^{n} l(y_i, F_m(x_i)) + \sum_{k=1}^{m} R(f_k) \tag{5}$$

where l is the loss function and R is the regularization term to penalize the complexity of the model.

In the context of our basketball shot success prediction project, XGBoost proved to be the most effective method for several reasons:

Handling Small Datasets: XGBoost is highly effective at handling small datasets, which is essential for our project focused on few-shot learning. Its robust regularization techniques (both L1 and L2) prevent overfitting, ensuring that the model generalizes well even with limited data. The regularization term $R(f_k)$ controls the complexity of the model by penalizing large weights, thus preventing overfitting.

Feature Importance: XGBoost inherently provides feature importance scores, which are crucial for understanding which features contribute most to the prediction of shot success. This capability aligns well with our goal of using weight explainability techniques to determine which parts of the shot form contribute positively or negatively to shot success.

Speed and Efficiency: XGBoost is optimized for speed and performance, making it suitable for our dataset, which includes a large number of video frames and pose features. The parallel tree boosting mechanism allows for efficient training and prediction, which is beneficial given the computational demands of processing video data.

The XGBoost model was trained using the following hyperparameters:

- **n_estimators:** 100
- **learning_rate:** 0.02
- **max_depth:** 6
- **subsample:** 0.8
- **colsample_bytree:** 0.8

These parameters were chosen to balance model complexity and training efficiency, ensuring optimal performance on our dataset.

3.6. CNN + LSTM Model

Our custom CNN + LSTM model is designed to leverage the strengths of CNNs for spatial feature extraction and LSTMs for temporal sequence modeling. The architecture consists of two parallel branches: one for processing the image features of the shooter and the other for processing their pose features. The outputs of these branches are then concatenated and fed into fully connected layers for final classification.

Image Feature Extraction: To start, we use a CNN to extract spatial features from the video frames of the shooter. The input to this branch is a sequence of video frames, each resized to 224×224 pixels. The CNN consists of several layers:

- **Convolutional Layers:** We use three TimeDistributed convolutional layers with increasing filter sizes (32, 64, 128) and a kernel size of 3×3, each followed by a MaxPooling layer with a pool size of 2×2. These layers help in capturing spatial features and reducing the dimensionality of the input.
- **Flatten Layer:** The output from the convolutional layers is flattened to create a 1D feature vector for each frame.
- **LSTM Layer:** An LSTM layer with 128 units processes the sequence of feature vectors, capturing the temporal dependencies between frames.
- **Fully Connected Layer:** The LSTM output is passed through a Dense layer with 128 units and ReLU activation, followed by a Dropout layer with a rate of 0.5 to prevent overfitting.

Pose Feature Extraction: We then use an LSTM to process the pose features of the shooter. The input to this branch is a sequence of pose keypoints for each frame, represented by 34 features (17 keypoints each with x and y coordinates). The LSTM branch consists of:

- **LSTM Layer:** An LSTM layer with 64 units processes the sequence of pose features, capturing the temporal dynamics of the shooter's movement.
- **Fully Connected Layer:** The LSTM output is passed through a Dense layer with 64 units and ReLU activation, followed by a Dropout layer with a rate of 0.5.

Combining Features: The outputs from the image and pose branches are concatenated to form a combined feature vector. This vector is then passed through additional fully connected layers to perform the final classification:

- **Fully Connected Layer:** A Dense layer with 128 units and ReLU activation processes the combined feature vector.
- **Dropout Layer:** We once again use a Dropout layer with a rate of 0.5.
- **Output Layer:** The final output layer is a Dense layer with a single unit and sigmoid activation, providing the probability of shot success.

Model Training: We trained our model with the Adam optimizer and the binary cross-entropy loss function, and it is trained with class weights to handle class imbalance. The data is split into training and testing sets using an 80-20 split, and is then trained for 50 epochs with a batch size of 4. In the end we evaluate performance using accuracy and classification report metrics.

4. Dataset and Features

The raw dataset contains 81 videos of singular basketball shots. Each clip starts when the shot motion appears to begin, and it ends right before the ball gets close to the rim, so that the model can't train on seeing the ball go into the basket or not. This means that each video is about 30-50 frames, and they're all in 480p. Videos were initially recorded in 1080p, and we considered extracting just the person bounding box in 1080p and then using 480p for the rest of the image, but this was too computationally expensive for the scope of this project, so we ended up only using 480p videos. These videos are taken using a stationary camera, such as an iPhone propped up against something. This was intentionally done so that if a new person is trying to use the model to improve their form, they can go out and record themselves by just putting their phone on the ground near them, and the format of the video will be what the model expects.

We split the dataset using a 80/20 split such that 20% of the data was used for testing and 80% was used for training. This meant that we had 17 videos in the testing data, and with the fixed seed we used, 10 of these were misses and 7 were makes. In the training data there were 22 makes and 42 misses. Instead of doing data sampling or augmentation techniques to handle these imbalances, we used class weights to balance out the disparity.

We preprocessed this dataset in 3 ways:

4.1. Image Features

The first preprocessing we did was to resize each frame to 224 by 224 so that it could be fed into the VGG-16 model at a constant size. We first applied VGG-16 to the whole frame and extracted the entire image's features.

4.2. YOLO + Image Features

The second approach we used was to first apply YOLO to the whole image to get the bounding box of the shooter, and then to apply the first preprocessing step to just the bounding box. This was much more effective because often times the shooter was quite small relative to the whole frame, and so resizing could lose valuable data about the shooter due to the lack of granularity. This often times even had the effect of upsizing the shooter due to them being smaller than 224 by 224 in the whole frame, and the image features were also only relevant to the person's pose as desired.

4.3. YOLO + Image Features + MoveNet

Our final and best approach was to first apply YOLO to get the bounding box of the shooter, and then to use VGG-16 and MoveNet to extract both image features of the shooter as well as their pose keypoints. MoveNet extracts

17 tuples of x, y coordinates in addition to a confidence score. We then concatenated the features for each frame so that models could take advantage of both the image features as well as the pose estimates.

5. Experiments

In this section, we present the results of each of the models we used to predict shot success.

5.1. Hyperparameter Selection

For our experiments, we chose specific hyperparameters for each model to optimize performance on our dataset. The learning rate, optimizer, and batch size were selected based on initial experiments and cross-validation results. Specifically, we used the Adam optimizer with a learning rate of 1×10^{-3} for the CNN + LSTM model, and a learning rate of 0.02 for the XGBoost model. The mini-batch size was set to 4 for the CNN + LSTM model to ensure stable training with our small dataset. Cross-validation was performed with 5 folds to ensure robust evaluation of model performance.

5.2. Evaluation Metrics

We used accuracy as our primary evaluation metric, which is defined as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{6}$$

Additionally, we report precision, recall, and F1-score to provide a more comprehensive evaluation of model performance. These metrics are defined as follows:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{7}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{8}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

5.3. Results

Table 1 summarizes the performance of different models on the test set.

The XGBoost model achieved the highest accuracy (82.4%), outperforming other models significantly. The CNN + LSTM model, despite its complexity, did not perform well due to the small dataset, highlighting the challenges of training deep learning models with limited data.

5.4. Model Interpretability

SHAP Weight Visualizations To understand why the XGBoost model performed best, we used SHAP (SHapley Additive exPlanations) values to interpret the model's predictions. SHAP values provide insights into the contribution of each feature to the model's output. Figure 1 shows the SHAP summary plot for the test set, highlighting the importance of various features.

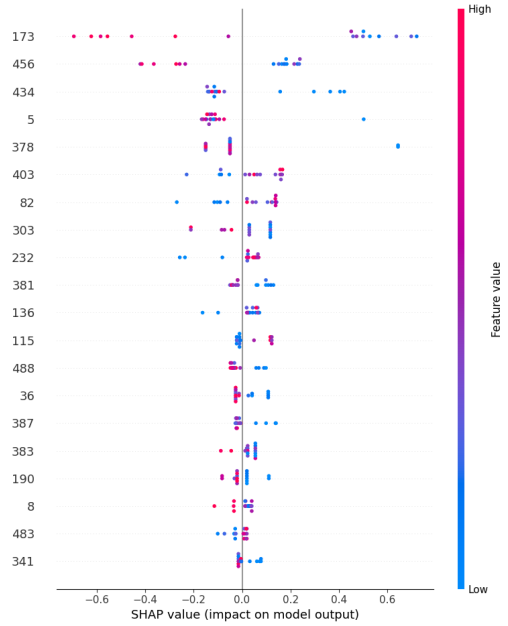

Figure 1. SHAP summary plot for the XGBoost model.

The SHAP visualizations indicated that key joints, such as the elbow and hips, significantly impacted the model's predictions. To illustrate this, Figure 2 shows frames from correctly classified videos with feature visualizations overlaid.

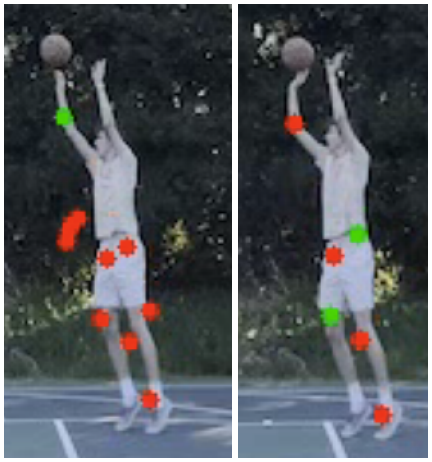

Figure 2. Left shows correctly classified make, right shows correctly classified miss, both show elbow joint correlating with the model decision.

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression (Whole Image)	0.529	0.47	0.53	0.47
Logistic Regression (Shooter Only)	0.588	0.56	0.59	0.52
Logistic Regression (Shooter + Pose)	0.588	0.56	0.59	0.52
Gradient Boosting (Shooter + Pose)	0.706	0.72	0.71	0.68
MLP	0.647	0.65	0.65	0.61
XGBoost	0.824	0.86	0.82	0.81
CNN + LSTM	0.529	0.47	0.53	0.47

Table 1. Performance of different models on the test set.

t-SNE Visualization We also used t-SNE to visualize the distribution of combined features (image and pose) in the dataset. Figure 3 shows the t-SNE plot with makes and misses labeled. No clear pattern is visible from the plot, suggesting that more data would be necessary to train a deep learning model, and that estimator based classifiers will perform better in the few shot learning scenario.

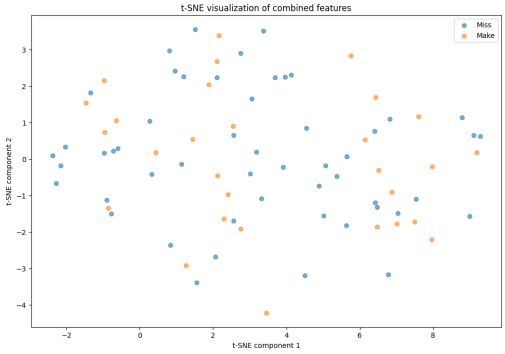

Figure 3. t-SNE visualization of combined features.

Training and Validation Loss Figure 4 shows the training and validation loss for the CNN + LSTM model. The increasing gap between training and validation loss over time indicates overfitting, likely due to the small dataset size. Regardless of hyperparameter choices, the model always ended up either overfitting or performing as well as random guessing.

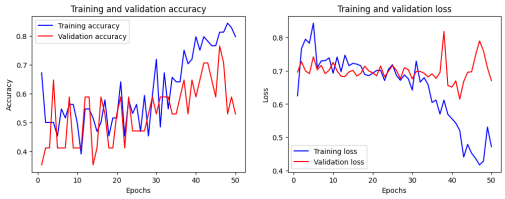

Figure 4. Training and validation loss for the CNN + LSTM model.

5.5. Discussion

The XGBoost model outperformed other models in this study, achieving the highest accuracy and providing meaningful interpretability through SHAP values. The importance of key joints in the predictions suggests that the model effectively learned to identify critical aspects of shot form that contribute to success, and the significantly better results from the combined features versus just image features corroborates this claim.

Other models, such as the CNN + LSTM, struggled due to the limited dataset size. Deep learning models typically require large amounts of data to generalize well, and our small dataset likely led to overfitting, as evidenced by the training and validation loss plot.

In future work, increasing the dataset size and incorporating more advanced techniques, such as transfer learning, could improve the performance of deep learning models. Additionally, exploring 3D pose estimation could provide more detailed insights into shot form and further enhance prediction accuracy.

6. Conclusion and Future Work

In this study, we explored various machine learning models to predict basketball shot success based on video data of shooters. Among the models tested, XGBoost emerged as the highest-performing algorithm, achieving an accuracy of 82.4%. The success of XGBoost can be attributed to its robust regularization techniques and its ability to handle small datasets effectively. The use of SHAP values provided valuable insights into the importance of key joints in the prediction, aligning with our goal of understanding shot form dynamics.

On the other hand, our custom CNN + LSTM model struggled with overfitting due to the limited size of our dataset. Deep learning models generally require large amounts of data to generalize well, and the small dataset size was a significant constraint in this study. Despite the complexity of the CNN + LSTM model, it did not outperform simpler models like XGBoost, highlighting the challenges of training deep learning models with limited data and the power of simpler estimator based classifiers when used in conjunction with pretrained models for few shot learning.

For future work, increasing the dataset size would be a primary focus to improve model performance and general-

izability. Additionally, incorporating transfer learning techniques could help mitigate the data scarcity issue. Exploring 3D pose estimation could also provide more detailed insights into shot mechanics and further enhance prediction accuracy. Another promising direction would be to investigate the application of advanced ensemble methods or neuro-fuzzy systems to leverage the strengths of multiple models.

Overall, this study demonstrates the potential of using computer vision and machine learning to evaluate and improve basketball shooting form, offering valuable feedback to players and coaches. With further advancements in data collection and model optimization, this approach could significantly impact sports analytics and training methodologies.

References

[1] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2019. 2

[2] Y. Chen and H. Lu. Basketball shooting analysis via 3d pose estimation. https://web.stanford.edu/class/cs231a/prev_projects_2022/lCS231a_Project_Final_Report.pdf, 2022. Department of Computer Science, Stanford University. 2

[3] T. Chou. Openpose research paper summary: Multi-person 2d pose estimation with deep learning. https://towardsdatascience.com/openpose-research-paper-summary-realtime-multi-person-2d-pose-estimation-3563a4d7e66, 2020. 2

[4] M. Harmon, A. Ebrahimi, P. Lucey, and D. Klabjan. Predicting shot making in basketball learnt from adversarial multiagent trajectories, 2021. 2

[5] S. Hauri, N. Djuric, V. Radosavljevic, and S. Vucetic. Multi-modal trajectory prediction of nba players. https://djurikom.github.io/pdfs/hauri2021wacv.pdf, 2021. Temple University, Uber ATG, Spotify. 2

[6] H. Hu, G. Dimitrov, D. Menn, and S. Wu. Nba player performance prediction based on xgboost and synergies. https://courses.cs.washington.edu/courses/cse547/23wi/old_projects/23wi/NBA_Performance.pdf, 2023. University of Washington. 2

[7] N. Kuhlman and C.-H. Min. Analysis and classification of basketball shooting form using wearable sensor systems. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1478–1482, 2021. 2

[8] B. Meehan. Predicting nba shots. https://cs229.stanford.edu/proj2017/final-reports/5132133.pdf, 2017. Stanford University. 1

[9] M. Murakami-Moses. Analysis of machine learning models predicting basketball shot success. The American School in Japan; Tokyo, Japan, 2024. Email: 22murakami-mosesm@asij.ac.jp. 1

[10] I. A. Ozkan. A novel basketball result prediction model using a concurrent neuro-fuzzy system. *Applied Artificial Intelligence*, 34(13):1038–1054, 2020. 2