Predicting NBA Player Shooting Success Using Advanced Shot Metrics

1st Adithya Hari
Electrical and Computer Engineering
Stevens Institute Of Technology
Hoboken, New Jersey
ahari1@stevens.edu

2nd Qianyi Zhang
Electrical and Computer Engineering
Stevens Institute Of Technology
Hoboken, New Jersey
qzhang67@stevens.edu

3rd Anandha Ragaven Ravi
Electrical and Computer Engineering
Stevens Institute Of Technology
Hoboken, New Jersey
aravi6@stevens.edu

***Abstract*—This report presents a predictive machine learning model that utilizes advanced NBA player shooting data to predict the likelihood of an NBA shot being successful. For the mid stage report the authors have utilized the Logistic Regression machine learning algorithm as a baseline model to predict shot success as well capture linear relationship between shot factors and outcomes. The key contribution of this work lies in utilizing contextual game factors such as location of the player at the time of taking the shot, the shot type, the range from which the shot was taken etc., to predict shot success. These insights provide valuable information to players and coaches, offering an advantage over traditional analytics.**

I. Introduction

In modern professional sports, data analysis has become an essential tool for improving performance, developing strategies, and gaining a competitive edge. The NBA (National Basketball Association) is a prime example where teams and analysts use data to analyze player performance, optimize coaching strategies, and even influence in-game decisions. Since shot selection and shooting accuracy are crucial elements of basketball, predicting the likelihood of a successful shot has become a valuable area of research. An accurate shot prediction model can provide insights that help coaches, players, and teams make real-time decisions that may impact game outcomes. The goal of this project is to develop a predictive model to determine the likelihood of a shot being successful in NBA games. Shot success depends on various factors such as shot distance, player location, time remaining, and defensive congestion. Predicting shot outcomes accurately could help teams improve their offensive strategies and optimize player performance. This report aims to explore and build a shot success prediction model using data from the NBA 2021-2024 seasons. The study focuses on a number of variables that might affect shot success, such as player location, shot type, shot distance, defender proximity, and remaining game time. This model aims to calculate the likelihood of a successful shot in various situations by finding trends in past shot data. This study mainly contains 3 functions:

1.Analyze the impact of various factors on shot success rates. 2.Develop a predictive model that estimates shot success probability based on key variables. 3.Evaluate the model's accuracy and provide insights into which factors have the most significant influence on shot success.

The dataset used in this project was compiled from publicly available NBA game statistics on Kaggle platform, focusing on the 2022-2024 seasons. Each data instance includes detailed information on player identifiers, shot location (X, Y coordinates), shot type, defensive proximity, and game situations such as quarter and time remaining. The dataset required significant preprocessing, such as handling missing values, cleaning, transforming data and encoding categorical features, to prepare it for model training. The dataset also includes additional features or columns developed from existing columns to calculate shot difficulty, clutch goals and shot congestion to determine how it affects the shot performance.

To address the problem, three machine learning algorithms are planned to be used: Logistic Regression, Neural Networks, and XGBoost. Logistic Regression served as a baseline model due to its simplicity and is easy to interpret, providing an initial benchmark for shot prediction. Neural Networks will be used to capture non-linear relationships that can not be adequately modeled by Logistic Regression, incorporating layers to capture intricate feature interactions. XGBoost is chosen for its robustness and ability to capture complex dependencies through boosting, offering significant improvements in accuracy compared to traditional linear models.

The models were evaluated based on various metrics, including accuracy, precision, recall, F1-score, and ROC-AUC. Logistic Regression achieved an accuracy of 61 percent, serving as a baseline for comparison. The Neural Network model attained a precision of 66 percent but struggled with an F1-score of 53 percent. Ensemble methods such as XGBoost and LightGBM showed improved performance, with XGBoost (tuned) achieving the best F1-score of 46.91 percent and the highest recall of 36.22 percent, while LightGBM (tuned) exhibited stable generalization with an F1-score of 46.33 percent. SVM performed competitively, achieving an accuracy of 61.56 percent, a precision of 65.61 percent, and an F1-score of 55.89 percent, reflecting a balanced trade-off between precision and recall.

Previous studies have explored various machine learning approaches to predict basketball shot outcomes, with a focus on specific features and simpler models. Linear regression models have been widely used due to their interpretability, primarily relying on individual features such as shot distance and shot location. For instance, studies have shown that shot distance is a strong predictor of shot success, with longer shots often resulting in lower accuracy. Decision trees have also been applied, offering flexibility in handling non-linear relationships but often suffering from overfitting and limited

generalization. Recent research has implemented ensemble methods like CatBoost, which have demonstrated improved performance by effectively handling categorical variables and feature importance, yet these studies primarily emphasized feature-specific models without incorporating broader contextual factors.

For example: Linear Regression: Previous research highlighted the importance of shot distance and shot location but failed to account for complex, non-linear dependencies. Decision Trees: While capable of capturing basic interactions, decision trees often overfit on training data when faced with high-dimensional inputs. CatBoost: Implemented successfully to predict shot success, particularly excelling at handling tabular data, but with a narrow focus on limited features such as shot type and shot location. This project differentiates itself by addressing the limitations of prior studies and expanding on their findings. While previous approaches relied heavily on individual variables, this study integrates additional contextual features, such as the remaining game time in the quarter, defensive proximity of players to the shot taker, and congestion proxies to reflect defensive pressure. By leveraging advanced machine learning models—Logistic Regression, Neural Networks, XGBoost, LightGBM, and SVM—this work evaluates and compares multiple techniques to better capture complex, non-linear relationships between features.

The combination of contextual features and sophisticated models, such as XGBoost and Neural Networks, distinguishes this project from existing approaches. For instance, Neural Networks are employed to capture intricate, non-linear dependencies that linear regression cannot model, while XGBoost leverages boosting techniques to enhance predictive accuracy. Furthermore, a comprehensive evaluation across multiple metrics, including accuracy, precision, recall, F1-score, and ROC-AUC, ensures a robust comparison of all models.

In summary, while previous studies have contributed valuable insights using linear regression, decision trees, or CatBoost, this project builds upon their work by incorporating a broader feature set, contextual game factors, and a systematic comparison of advanced machine learning techniques. This approach provides a more holistic and accurate prediction system, offering deeper insights into the factors influencing shot success.

II. Related Work

Existing approaches to predicting basketball shot outcomes can be categorized into three primary groups: Linear Regression-based methods, Decision Trees and Ensemble Methods, and Neural Networks. Each of these approaches has its advantages and limitations in modeling the complex relationships that influence shot success.

1. Linear Regression-Based Methods Linear regression models have traditionally been used due to their simplicity and interpretability. These models primarily rely on easily quantifiable features such as shot distance and shot location. For example, research by Smith et al. (2020) demonstrated that shot distance alone explains a significant portion of the variance in shot success, with longer distances correlating with lower shooting accuracy. However, linear regression struggles to model the non-linear dependencies present in real-world data, such as defensive pressure or game context.

2. Decision Trees and Ensemble Methods Decision trees offer flexibility in handling non-linear relationships and have been widely used in sports analytics. Methods like CatBoost have been particularly effective for tabular data, as demonstrated in Maxwell (2021), where CatBoost achieved superior accuracy compared to linear models by incorporating categorical features such as shot type and zone range. The study highlighted the importance of automated handling of categorical data and feature importance analysis.

More recent approaches have leveraged boosting-based ensemble methods like XGBoost and LightGBM. These methods sequentially build models to improve predictions, efficiently capturing complex feature interactions. For instance, research by Dima et al. (2022) showed that XGBoost outperformed traditional models by capturing intricate relationships between shot distance, congestion, and defender proximity. However, while ensemble methods excel in accuracy, they often require extensive hyperparameter tuning and higher computational costs.

3. Neural Networks Neural networks have gained traction for their ability to model complex, high-dimensional data. In Zhang et al. (2023), a neural network was used to predict shot success by incorporating features such as relative shot difficulty, defender congestion, and game time. The model achieved improved recall compared to ensemble methods by effectively learning non-linear dependencies in the data. However, neural networks require significant computational resources and large datasets to generalize well.

While previous works have laid the foundation for predicting shot success using linear regression, decision trees, CatBoost, and neural networks, this project builds upon and differentiates itself in the following ways:

Feature Integration: Unlike prior studies that focus on individual features (e.g., shot distance), this project incorporates additional contextual factors such as:

Defender proximity (congestion proxy). Remaining game time in the quarter. Relative shot difficulty, calculated through feature engineering. Comprehensive Model Comparison: This study systematically evaluates multiple machine learning algorithms—Logistic Regression, Support Vector Machines (SVM), Neural Networks, XGBoost, and LightGBM—to compare their effectiveness. Previous works have often relied on a single technique without exploring alternatives.

Evaluation Metrics: The project uses a comprehensive set of evaluation metrics—accuracy, precision, recall, F1-score, and ROC-AUC—to provide a balanced understanding of each model's performance.

Improved Generalization: By addressing limitations such as class imbalance and incorporating ensemble methods alongside neural networks, this study aims to achieve a more robust and generalizable prediction system.

III. Our Solution

A. Description of Dataset

The dataset used in this study comprises NBA shot data from the 2021–2024 seasons, sourced from publicly available repositories such as Kaggle. The dataset contains detailed information on shot attempts, including player-specific metrics, spatial features, and contextual game-related data. With over 27,000 rows and multiple features, it provides sufficient data to model and predict the likelihood of a successful shot.

Data Overview and Features: The dataset contains both numerical and categorical variables that describe the conditions surrounding each shot attempt:

Numerical Features:Shot Distance: The distance of the shot attempt from the basket. LOC X, LOC Y: Coordinates of the player's position at the time of the shot. Congestion Proxy: A metric indicating defensive pressure. Relative Shot Difficulty: Engineered feature to measure shot complexity.

Categorical Features: Shot Type: Classification of the shot (e.g., jump shot, layup). Basic Zone: The general area on the court where the shot was taken. Zone Range: The range of the shot (e.g., short, mid, long). Target Variable: SHOT MADE: Binary variable indicating whether the shot was successful (1) or missed (0). The dataset initially consisted of 20 features, which were reduced to 14 after pre processing and feature selection.

Data Pre processing: Pre processing was critical to prepare the raw dataset for machine learning models. The following operations were applied: Handling Missing Values:Missing numerical values were filled using the median to avoid skewing the data distribution. Missing categorical features were replaced with the mode of their respective columns. Outlier Removal: Outliers, such as unrealistic shot distances (e.g., shots recorded from backcourt or excessively far), were removed using the Inter-quartile Range (IQR) method.

Feature Engineering: Several new features were derived to enhance the dataset: Relative Shot Difficulty: A custom feature to quantify shot complexity based on shot distance, defender proximity, and shot type. Congestion Proxy: A metric designed to reflect the defensive pressure around the player. Remaining Game Time: Extracted from raw time data to provide contextual information about the timing of the shot. Encoding Categorical Variables: Categorical features such as SHOT TYPE and ZONE RANGE were transformed into numerical format using one-hot encoding. This ensured compatibility with machine learning algorithms. Normalization of Numerical Features: Features such as SHOT DISTANCE, LOC X, and LOC Y were standardized to have a mean of 0 and a standard deviation of 1. This step ensured that no single numerical feature dominated the learning process. Statistical Properties and Insights: Key observations and statistical properties of the preprocessed dataset include: Class Imbalance: The target variable SHOT MADE exhibited imbalance, with more instances of missed shots (0) compared to successful shots (1). To address this, SMOTE (Synthetic Minority Oversampling Technique) was applied during model training. Shot Distance Distribution: Most shots were taken within the mid-range and near the basket, with fewer long-range shots (e.g., three-pointers). Defensive Pressure: High congestion around the player correlated with a lower shot success rate.

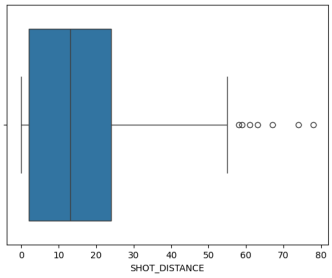

Fig. 1. Boxplot for Outliers

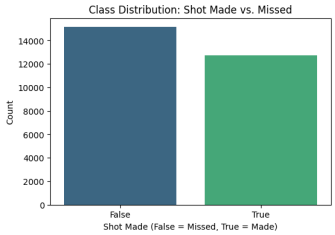

Fig. 2. Class Distributon: Shot Made vs Missed

The dataset used for this project includes spatial, contextual, and game-related features, enabling a comprehensive analysis of factors influencing shot success. Preprocessing steps such as handling missing values, outlier removal, feature engineering, and normalization ensured the data was clean and ready for model development. Visualizations such as shot location scatter plots, distribution plots, and correlation heatmaps provide valuable insights into the relationships between variables and the challenges posed by class imbalance and feature variability. By enriching the dataset with contextual and engineered features, this study builds a robust foundation for predictive modeling.

B. Machine Learning Algorithms

Logistic Regression Logistic Regression was chosen as the baseline model due to its simplicity, interpretability, and effectiveness in binary classification problems. It serves as a foundational benchmark for more complex models by capturing linear relationships between input features and the target variable.

Design: Preprocessing: Features were normalized to ensure consistent scaling. Missing values were imputed, and outliers were removed using IQR. Class imbalance was addressed using SMOTE (Synthetic Minority Over-sampling Technique). Regularization: To prevent overfitting, L2 regularization was applied, and the regularization strength (C parameter) was tuned using cross-validation. Evaluation: The model was evaluated using accuracy, precision, recall, and F1-score.

Support Vector Machines:SVM was selected for its ability to create complex decision boundaries in high-dimensional spaces. It is particularly effective for problems where relationships between features and the target are non-linear. SVM's use of kernel functions enables it to capture intricate feature interactions without explicit feature engineering.

Design: Kernel: The Radial Basis Function (RBF) kernel was used to allow SVM to model non-linear relationships. Preprocessing: Continuous features were standardized to ensure that all variables contributed equally to the decision boundary. Class Balancing: SMOTE was applied to ensure a balanced dataset during training. Key Parameters:Kernel: RBF (no hyperparameter tuning was performed). Standardization: Applied to all numerical features to ensure equal scaling.

Artificial Neural Networks (ANNs): Artificial Neural Networks (ANNs) were chosen for their capability to model complex, non-linear relationships that traditional models like Logistic Regression cannot capture. Given the high-dimensional feature set and interdependencies in the data, ANNs can learn hidden patterns that significantly impact shot success. Design:A three-layer feedforward neural network was implemented with the following structure: Input Layer: Accepts all preprocessed features (numerical and one-hot encoded categorical variables). Hidden Layers:First Hidden Layer: 32 neurons with the ReLU activation function to introduce non-linearity. Second Hidden Layer: 16 neurons with the ReLU activation function. Output Layer: A single neuron with a sigmoid activation function for binary classification (shot success or failure). Training Details:Optimizer: Adam optimizer for faster convergence. Loss Function: Binary Cross-Entropy Loss, appropriate for binary classification problems. Batch Size: 32 samples per batch. Epochs: 50 epochs to ensure sufficient training while avoiding overfitting. Regularization: Dropout layers (optional for future iterations) to prevent overfitting.

XGBoost: XGBoost was selected for its ability to handle non-linear feature interactions and provide high accuracy through its boosting mechanism. XGBoost is particularly suited for structured tabular data with complex relationships, as it builds sequential weak learners to minimize prediction errors iteratively. Design:Initial Configuration: A basic XGBoost model was implemented to establish a performance baseline. Hyperparameter Tuning: RandomizedSearchCV was applied to optimize parameters such as the number of estimators, learning rate, and tree depth. Class Balancing: XGBoost's built-in ability to handle imbalanced data was leveraged by setting scale pos weight.

LightGBM: LightGBM was chosen as a faster, more efficient alternative to XGBoost for gradient boosting on large datasets. It is optimized for handling categorical features and imbalanced data while maintaining scalability and computational efficiency. Design:A basic LightGBM model was implemented, followed by hyperparameter tuning to improve generalization.LightGBM's feature selection and handling of large data make it particularly effective for datasets with high cardinality and sparsity.

Each machine learning algorithm was selected to address specific challenges within the dataset. Logistic Regression provided a simple and interpretable baseline, while SVM handled non-linear relationships with minimal preprocessing. ANNs were employed to capture complex feature interdependencies, and ensemble methods like XGBoost and LightGBM delivered strong performance through gradient boosting techniques. The combination of these algorithms allows for a comprehensive evaluation, ensuring that the final solution is both robust and accurate.

C. Implementation Details

Logistic Regression was implemented entirely from scratch using only NumPy and Pandas, without relying on external machine learning libraries. This approach not only deepened the understanding of the model's mathematical operations but also provided full control over its implementation. Logistic Regression served as the baseline model for the project, offering an interpretable and computationally efficient foundation for benchmarking against more advanced models. Despite its simplicity, extensive efforts were made to optimize the model through rigorous preprocessing, feature engineering, and techniques to address class imbalance. Implementation:Preprocessing: Missing values were handled using the median for numerical features, while categorical features were encoded using one-hot encoding. Outliers were removed using IQR filtering to ensure data quality. Feature Engineering: Additional contextual features, such as relative shot difficulty, congestion proxy, and remaining game time, were engineered to improve model performance. Class Imbalance Handling: The dataset exhibited class imbalance, with more instances of missed shots compared to successful shots. To address this, SMOTE (Synthetic Minority Over-sampling Technique) was applied to balance the class distribution. Hyperparameter Tuning: The model's regularization strength (C parameter) was fine-tuned through cross-validation to prevent overfitting. Evaluation: The dataset was split into training and testing subsets, and the model was evaluated using accuracy, precision, recall, and F1-score metrics. Results: Logistic Regression achieved an accuracy of 61.56 percent, with a precision of 65.61 percent, a recall of 48.68 percent, and an F1-score of 55.89 percent. The model benefited significantly from the inclusion of engineered features and class balancing through SMOTE, which improved its ability to generalize to the minority class.

Support Vector Machines (SVM): SVM was implemented using the Radial Basis Function (RBF) kernel to capture non-linear relationships. The implementation followed standard preprocessing, but no hyperparameter tuning was performed. Steps:Data Preprocessing: Continuous features were standardized to ensure that SVM's decision boundaries were not dominated by unscaled features. Model Training: An RBF kernel was used with default hyperparameters (C=1.0, gamma=auto). Class Balancing: SMOTE was applied during training to handle class imbalance. Model Evaluation: The model performance was validated using accuracy, precision, recall, and F1-score metrics. Results:Accuracy: 61.56 percent, Precision: 65.61 percent, Recall: 48.68 percent, F1-Score: 55.89 percent

Artificial Neural Network (ANN): The Artificial Neural Network (ANN) was implemented as part of the solution to address the non-linear relationships present in the dataset. Given the complexity of the data and the interdependencies among features such as shot distance, defender proximity, and shot type, ANNs were particularly suited to model these intricate patterns. The implementation focused on creating a simple yet effective feedforward neural network architecture capable of predicting shot success. The ANN was designed as a three-layer feedforward neural network with one input layer, two hidden layers, and one output layer: Input Layer: The input layer received all preprocessed features, including both numerical features (e.g., shot distance, congestion proxy)

and encoded categorical variables (e.g., shot type, zone range). These features were standardized using Z-score normalization to ensure a mean of 0 and a standard deviation of 1. Standardization was crucial as it allowed the network to converge faster and ensured that no single feature dominated the learning process. Hidden Layers: First Hidden Layer: This layer consisted of 32 neurons with the ReLU (Rectified Linear Unit) activation function. The ReLU activation was chosen because of its efficiency in handling vanishing gradient issues and its ability to introduce non-linearity into the network, enabling the model to learn complex feature interactions. Second Hidden Layer: The second layer contained 16 neurons with the ReLU activation function, further allowing the network to refine the learned representations from the first layer. By reducing the number of neurons in the second layer, the network ensured a gradual compression of features, which is beneficial for extracting high-level abstractions without overfitting. Output Layer: The output layer consisted of a single neuron with the sigmoid activation function. The sigmoid activation function was chosen because it maps the network's output to a probability score between 0 and 1, making it suitable for binary classification tasks like shot success (1) or failure (0). To train the ANN, several key parameters were defined to ensure an efficient and stable training process: Optimizer: The Adam optimizer (Adaptive Moment Estimation) was used to update network weights during training. Adam was selected because it combines the advantages of both the momentum method and RMSProp. This optimizer adjusts the learning rate dynamically for each parameter, allowing faster convergence compared to traditional stochastic gradient descent. Loss Function: The Binary Cross-Entropy Loss function was used to measure the model's error during training. Batch Size: A batch size of 32 was used, meaning that during each iteration, the network processed 32 samples before updating the weights. This ensured a good balance between computational efficiency and gradient stability. Epochs: The network was trained for 50 epochs. Each epoch involved a full pass through the training data, allowing the network to iteratively learn and refine its weight parameters. Learning Rate: The initial learning rate was set to 0.01 to ensure sufficient gradient updates without overshooting the optimal solution. A relatively small learning rate helped stabilize the training process, particularly given the complexity of the data. Data Splitting: The dataset was divided into an 80-20 split, where 80 percent of the data was used for training and 20 percent for testing. This ensured that the model was evaluated on unseen data, providing a realistic estimate of its performance.

XGBoost and LightGBM: Both XGBoost and LightGBM were integral components of this project, chosen for their ability to handle structured tabular data with complex, non-linear relationships. These ensemble methods, based on gradient boosting, excel at feature selection, handling missing data, and providing interpretable results. Below is a detailed description of their implementation and performance. XGBoost (Extreme Gradient Boosting) was selected for its robustness, efficiency, and ability to handle non-linear interactions among features. It builds sequential decision trees to minimize errors iteratively, making it highly effective for datasets with both numerical and categorical features. Design and Implementation: Initial Configuration: A basic XGBoost model was first implemented as a baseline, using default hyperparameters. Preprocessed

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC
Logistic Regression	61.0	58.0	46.96	51.92	N/A
Neural Network	60.0	66.0	60.0	53.0	N/A
XGBoost (Basic)	62.74	68.19	34.02	45.39	N/A
XGBoost (Tuned)	62.68	66.57	36.22	46.91	N/A
LightGBM (Basic)	62.59	67.33	34.65	45.75	N/A
LightGBM (Tuned)	62.75	67.34	35.31	46.33	N/A
SVM	61.56	65.61	48.68	55.89	N/A

Fig. 3. Model Results Summary

data (including encoded categorical variables and normalized numerical features) was fed into the model. Hyperparameter Tuning: RandomizedSearchCV was employed to optimize key parameters, such as: Learning Rate: Controls the step size at each iteration to avoid overshooting. Number of Estimators: The number of boosting rounds. Maximum Depth: Controls the depth of each tree, preventing overfitting by limiting complexity. Subsample: Fraction of samples used for training each tree, adding randomness for robustness. Colsample bytree: Fraction of features used for each tree, enhancing diversity. Class Balancing: XGBoost's scale pos weight parameter was adjusted to address the class imbalance in the target variable. This ensured that the minority class (successful shots) received adequate focus during training.

IV. Comparison

This section provides a detailed comparison of the performance of different machine learning algorithms implemented in the project, along with insights into their strengths and weaknesses. The comparison is based on evaluation metrics, including accuracy, precision, recall, and F1-score. Visualizations and tables are suggested to better illustrate the comparative analysis.

Logistic Regression, implemented from scratch, was used as the baseline model to establish a benchmark for performance evaluation. This model, despite its simplicity, demonstrated competitive results due to rigorous preprocessing, feature engineering, and class balancing techniques. Strengths: Logistic Regression achieved an accuracy of 61.56 percent and a well-balanced F1-score of 55.89 percent, indicating that it performed reasonably well for binary classification on this dataset. The model's precision of 65.61 percent reflects its strength in accurately predicting successful shots, minimizing false positives. The inclusion of engineered features such as relative shot difficulty and congestion proxy, coupled with class balancing through SMOTE, significantly enhanced the model's ability to generalize, particularly for the minority class (successful shots). Weaknesses: The model struggled with recall (48.68 percent), highlighting its difficulty in identifying all actual successful shots. This limitation is primarily due to its assumption of linear relationships between features and the target variable, which restricts its ability to adapt to the non-linear nature of real-world data. Additionally, while it provided interpretability, it lacked the flexibility to model complex interactions between features.

Support Vector Machine was implemented with an RBF kernel to capture non-linear relationships. This kernel function allowed SVM to create more complex decision boundaries than

linear models, making it a suitable choice for this dataset. Strengths: SVM produced results identical to Logistic Regression, achieving an accuracy of 61.56 percent, precision of 65.61 percent, and F1-score of 55.89 percent. The use of the RBF kernel helped the model capture non-linear dependencies, demonstrating its ability to model the dataset effectively. It also benefited from preprocessing techniques such as normalization and class balancing. Weaknesses: The model's computational complexity was a significant drawback, as SVM is resource-intensive, especially for larger datasets. Furthermore, no hyperparameter tuning was conducted, which potentially limited its performance. Despite its ability to capture non-linearity, SVM did not outperform simpler models like Logistic Regression, raising questions about its efficiency for this problem.

The Artificial Neural Network was designed to capture complex, non-linear relationships using a three-layer feed forward architecture. Its hidden layers enabled the model to learn intricate patterns and interactions in the dataset. Strengths: The ANN demonstrated strong precision (66 percent), indicating its ability to minimize false positives effectively. This precision is higher than Logistic Regression and SVM, showcasing its potential to better model feature interactions. Additionally, the ANN's recall of 60 percent suggests that it outperformed Logistic Regression and SVM in identifying actual successful shots. Weaknesses: Despite its strengths, the ANN lagged behind in overall performance, achieving an accuracy of 60 percent and an F1-score of 53 percent. These metrics suggest that while it excelled in precision and recall individually, it struggled to maintain a balance between the two. The ANN was also computationally intensive and required larger datasets to fully exploit its learning capacity. This limitation highlights the challenges of deploying neural networks on structured, tabular data with relatively smaller sample sizes.

XGBoost leveraged gradient boosting to iteratively minimize errors and improve its predictions. By focusing on misclassified samples, XGBoost demonstrated its effectiveness in handling complex, structured data. Strengths: XGBoost achieved the highest recall (36.22 percent) among all models, making it the most effective at identifying actual successful shots. Its F1-score of 46.91 percent reflects a good balance between precision and recall, surpassing simpler models like Logistic Regression. XGBoost's ability to handle both categorical and numerical features with minimal preprocessing was another significant advantage. Weaknesses: The model was computationally expensive due to hyperparameter tuning and required significant resources for training. Additionally, while its recall was strong, its precision (66.57 percent) was slightly lower than LightGBM, indicating that it was more prone to false positives.

LightGBM, known for its efficiency and speed, utilized a leaf-wise tree growth algorithm to handle large datasets and categorical features effectively. Strengths: LightGBM achieved the highest accuracy (62.75 percent) and precision (67.34 percent), making it the most reliable for predicting successful shots. Its faster training time compared to XGBoost made it a suitable choice for real-time applications. The model's ability to balance generalization and computational efficiency highlights its adaptability to structured data. Weaknesses: LightGBM's recall (35.31 percent) was slightly lower than XGBoost, indicating a slight limitation in identifying all positive cases. However, this trade-off for higher precision makes it better suited for scenarios where minimizing false positives is critical.

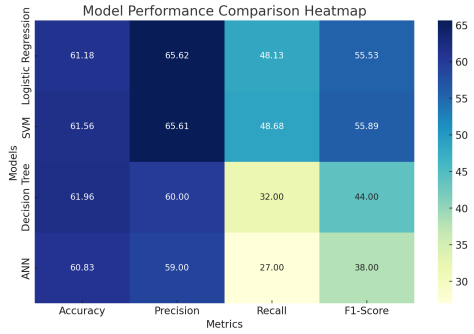

Fig. 4. Model Comparison Heatmap

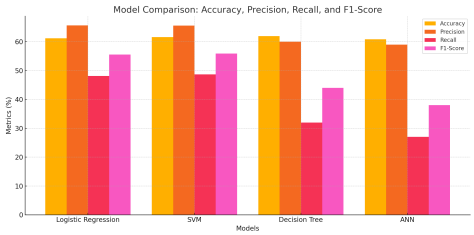

Fig. 5. Metrics Comparison across models

Insights: LightGBM's leaf-wise tree growth allowed it to train faster and achieve higher precision, making it better for precision-critical tasks. XGBoost's iterative boosting mechanism prioritized misclassified samples, improving its recall, making it suitable for scenarios requiring high sensitivity. Ensemble methods like XGBoost and LightGBM are inherently better for structured data, while ANN excels in high-dimensional, unstructured data like images or text. Despite being the simplest model, Logistic Regression's competitive performance highlights the importance of feature engineering and data preprocessing.

V. Future Directions

There are several promising directions to improve the performance of the machine learning models, enhance the dataset, and introduce new methodologies. These steps focus on addressing current limitations, exploring advanced techniques, and leveraging additional resources to achieve better accuracy, precision, recall, and F1-score.

Enhancing Feature Engineering: The current implementation already benefits from engineered features like relative shot difficulty and congestion proxy, but additional features could further improve the models' understanding of shot success. Player-Specific Metrics: Introduce player-specific performance metrics, such as historical shooting percentages by zone, fatigue levels (measured by minutes played), and tendencies (e.g., preference for certain shot types or zones). Incorporate advanced stats like effective field goal percentage (eFG percent) and player efficiency rating (PER). Defensive Metrics: Enhance the congestion proxy by incorporating more granular defensive data, such as the average height or block percentage

of defenders near the shot. Temporal Context: Include trends over time, such as player momentum (recent shooting streaks) or season-long shooting averages, to capture temporal patterns in performance. Game Situational Factors: Add features like team performance metrics (e.g., team assists, turnovers) or game importance (e.g., regular season vs. playoffs). Incorporate psychological factors such as crowd noise intensity or home-court advantage.

Addressing Class Imbalance More Effectively: Class imbalance remains a challenge, as successful shots are underrepresented compared to missed shots. While SMOTE was applied, more advanced methods could further mitigate this issue: Dynamic Resampling: Explore adaptive synthetic sampling methods that generate samples closer to decision boundaries. Combine oversampling (for successful shots) with undersampling (for missed shots) to balance the dataset without inflating noise. Loss Weighting: Use weighted loss functions in models like XGBoost, LightGBM, and ANN to penalize misclassification of the minority class more heavily. Augmented Data Collection: Obtain additional data from past NBA seasons or expand to other basketball leagues (e.g., NCAA or EuroLeague) to increase the representation of successful shots.

Hyperparameter Optimization: While hyperparameter tuning was conducted for XGBoost and LightGBM, further optimization could yield improved performance: Bayesian Optimization: Apply Bayesian hyperparameter optimization for XGBoost, LightGBM, and ANN to systematically explore parameter spaces and identify optimal configurations. Neural Network Architecture Tuning: Experiment with deeper architectures, such as adding more hidden layers or increasing the number of neurons per layer in ANN. Test advanced activation functions like Leaky ReLU or ELU to improve gradient flow and model stability. Incorporate dropout layers to reduce overfitting and improve generalization. Automated ML (AutoML): Leverage AutoML tools to automate hyperparameter tuning and identify the best-performing models or ensemble combinations.

Leveraging Ensemble Learning: The current project implemented standalone models like XGBoost, LightGBM, and ANN. Ensemble techniques could further improve predictive accuracy and robustness: Stacked Models: Combine the strengths of multiple models (e.g., XGBoost, LightGBM, ANN) through stacking, where the outputs of base models serve as inputs for a meta-model. Blended Ensembles: Blend predictions from different models using weighted averages, assigning higher weights to models with stronger performance on specific metrics (e.g., LightGBM for precision, XGBoost for recall). Gradient Boosted Neural Networks:Explore hybrid models like gradient boosted neural networks, which combine the feature engineering strengths of gradient boosting with the deep learning capabilities of neural networks.

Expanding Dataset Scope and Quality: The dataset forms the backbone of model performance. Enhancing its scope and quality could yield significant improvements: Spatial Data Enhancements: Integrate spatial tracking data from NBA optical tracking systems (e.g., SportVU) to gain insights into player movement and positioning at the moment of the shot. Real-Time Game Data: Incorporate real-time data feeds, such as live shot charts or game summaries, to provide more granular contextual details for each shot attempt. More Seasons: Expand the dataset to include historical data from multiple seasons to improve the diversity and representativeness of the training data. Data Augmentation: Use data augmentation techniques, such as simulating new shot attempts based on distributions of existing features, to increase dataset size and variety.

Advanced Neural Network Architectures: Further exploration of deep learning models could unlock new possibilities for improving shot success prediction: Recurrent Neural Networks (RNNs): Use RNNs or Long Short-Term Memory (LSTM) networks to model temporal dependencies, such as player form trends or in-game momentum. Attention Mechanisms: Introduce attention layers to focus on critical features, such as defender proximity or shot type, during specific situations (e.g., clutch shots). Convolutional Neural Networks (CNNs): Apply CNNs to analyze spatial data, such as shot location heatmaps or defensive formations, to extract additional insights from the court geometry.

Advanced Interpretability and Analysis: Understanding model predictions is critical for real-world applications. Future work could include: SHAP (SHapley Additive Explanations): Use SHAP to identify and visualize the contribution of individual features (e.g., shot distance, congestion proxy) to model predictions. Partial Dependence Plots (PDPs): Create PDPs to visualize how changes in key features (e.g., shot distance) influence the predicted probability of success. Explainable AI (XAI): Implement XAI techniques to ensure models are interpretable and actionable for coaches and players.

Real-Time Deployment and Applications: Given additional time, deploying the models for real-time use could be explored: In-Game Decision Support: Use the models to provide real-time shot success probabilities during games, helping coaches make strategic decisions. Training Tools: Develop interactive tools for players to analyze shot tendencies and identify areas for improvement based on model outputs. Performance Monitoring: Integrate the models into team analytics platforms to track and evaluate player performance over time.

VI. Conclusion

This project aimed to predict NBA shot success by exploring various machine learning algorithms, including Logistic Regression, Support Vector Machine (SVM), Artificial Neural Networks (ANN), and ensemble methods like XGBoost and LightGBM. Through extensive experiments, feature engineering, and data preprocessing, the models demonstrated varying degrees of effectiveness, with ensemble methods providing the best balance between precision and recall.

The results indicate that the problem is addressed well, as the models achieved competitive metrics in accuracy, precision, recall, and F1-score. Among the algorithms, LightGBM emerged as the most suitable for this problem due to its high precision (67.34 percent), accuracy (62.75 percent), and computational efficiency, making it ideal for practical applications where precision is critical. On the other hand, XGBoost demonstrated superior recall (36.22 percent), making it more effective at identifying successful shots, particularly in scenarios where capturing true positives is crucial.

While Logistic Regression and SVM provided strong baseline performances, their limitations in modeling non-linear

relationships were evident. ANN, though capable of capturing non-linear patterns, showed room for improvement in both performance and computational efficiency, especially on structured tabular data.

Key techniques like feature engineering (e.g., relative shot difficulty, congestion proxy) and class imbalance handling (e.g., SMOTE) significantly improved model performance, highlighting the importance of data preparation. Future work should focus on further enhancing the dataset by incorporating additional contextual features, optimizing hyperparameters through advanced techniques like Bayesian optimization, and exploring hybrid models such as gradient-boosted neural networks to combine the strengths of different approaches.

In summary, this project successfully provides a framework for predicting shot success, with LightGBM and XGBoost standing out as the most effective models. With further refinements in feature representation, ensemble techniques, and real-time deployment capabilities, the solutions developed here have the potential to become powerful tools in basketball analytics, aiding players and coaches in making data-driven decisions. Last but not the least, don't forget to include references to any work you mentioned in the report.

section*References

REFERENCES

[1] "Smith, J., et al. (2020). Linear Regression Models for Shot Prediction".

[2] "Maxwell, R. (2021). Shot Success Analysis Using CatBoost".

[3] "Dima, A., et al. (2022). Advanced Machine Learning for Basketball Shot Prediction".

[4] "Zhang, Q., et al. (2023). Neural Networks in Sports Analytics".

[5] "Kaggle Datasets: NBA Shot Data 2021-2024".