Project Document: NBA Over/Under Predictor
DROP TABLE "Assignments"

Tyler Frisinger
Riley Heimes
Reggie Schriner
Dylan Kramer

Contents

Abstract

In this report, we present an NBA Over/Under predictor that forecasts a player's next-game points for decision support in sports betting and basketball analytics. The learning problem is supervised regression, where inputs are built from a player's recent performance and game context, and the output is a single-number point prediction for the upcoming game. We use joined NBA Traditional and Advanced box-score data from 1997–2024 (over 690,000 player-game rows) and engineer features using only information available before each target game to prevent leakage. These features include rolling and cumulative player averages, efficiency and usage measures, minutes stability, home/away context, win-streak indicators, and opponent defensive signals.

Over the course of the project, we considered multiple modeling directions, including a graph-based approach to capture player relationships and a sequence-aware approach inspired by Transformer-style forecasting. In the final milestone, we focus on a compact deep neural network trained on the full engineered feature set and refine the training process through hyperparameter tuning and early stopping to reduce overfitting. We evaluate using chronological splits to preserve temporal integrity and report standard regression metrics. Our final model achieves MAE = 4.55 points and RMSE = 6.01 points on held-out test data, improving over a naive season-average baseline (RMSE $\approx$ 7.33). Based on these results, we emphasize using point forecasts to support over/under decisions (relative to short-term rolling baselines) rather than relying only on exact point totals, especially for high-variance, high-usage scorers. Future work includes integrating live betting lines through an external API and adding attention mechanics to the model through the use of transformers.

Chapter 1

Milestone 1: Project Ideas

1.1 Introduction

We propose and compare two applied machine learning projects that highlight different ends of the supervised learning spectrum. The first project proposed is an *NBA Over/Under Predictor* that forecasts player game-by-game statistics for decision support in sports betting and analytics. The second project proposed is an *Age, Race, and Gender Classifier* that performs real-time demographic classification recognition from a dataset full of facial images for research and benchmarking purposes. The goal for this project is for a model to estimate the age of a person as well as classify gender and ethnicity by live camera feed. Ultimately, the user will stand in front of the camera and the model will output as a prompt, the proposed information above. Both projects require a significant amount of data acquisition and preprocessing that ranges from model training, evaluation, to responsible-use considerations. The only differences are that both projects range in variability between modalities, target variables, evaluation metrics, and ethical risks. This chapter introduces each idea at a high level and motivates why it is worth studying in a semester project.

Included in this paper are various research papers and multiple datasets for both projects that backup our ideas and define our potential roadmap for either project. The following sections will dive into the fine details about each project, including cited sources, the models we propose to use, and other information necessary to fully understand our projects.

1.2 Project Idea 1: NBA Over/Under predictor

The problem we want to solve is to predict the next game statistics of an NBA player so that users can make more accurate decisions when betting on over/under lines. Applications include sports betting, fantasy basketball, and fan analytics. The learning problem is supervised regression, where the inputs are characteristics such as a player's recent game history, season averages and

contextual factors from the dataset, and the outputs are predicted values for key statistics such as points, rebounds and assists. We plan to try baseline methods like moving averages, more advanced models like gradient-boosted trees [4], and possibly sequence models for time-dependent data [18]. The primary data set is the publicly available 'NBA Player Data (1950–2021)' on Kaggle [3], which provides the historical player statistics needed to train and evaluate our models.

1.3 Project Idea 2: Age, Race, and Gender Classification

The goal of this project idea is to train a classification model in order to identify the age, race and gender. This model has the ability to be used in live human demonstration in order to classify humans by age, race and gender. The inputs that we will be using for images as well as labels to train and validate our model are the same that were used in this research paper: 'FairFace: Face Attribute Dataset for Balanced Race, Gender, and Age for Bias Measurement and Mitigation' [13]. However when doing a demonstration of our model we will be using a live camera feed in order to identify the persons age, race, and gender and then print that data on the bottom of the screen.

1.4 Conclusions

Overall, we have found substantial material related to both projects. We have concluded that each project would provide a substantial service to each of their respective fields and we believe that each project would have a relatively low requirement of data collection, as we have found useful data sets for each use case. However, considering each project individually, they both have their challenges. The NBA Over/Under predictor faces challenges of perceived accuracy and overall results. This shows how this project may struggle to reach a functional conclusion with tangible results. As for the age, race, and gender classification project, this project will have challenges related to its dependence on hardware or other peripheral equipment like the camera used to capture the live video. This shows how this project may become blocked by limitations of the equipment, rather than the strength of the model. Despite these challenges, we feel passionate about both projects. In order to aid us in selecting a project, we would like to ask the instructor and TA about the challenges we have described. Will camera quality likely be a signifcant set back in the age, race, and gender classification project and will the NBA Over/Under predictor face unavoidable challenges that may hinder our ability to produce accurate and tangible results? Which project has less resistance to progress in these regards?

Table 1.1: Contributions by team member for Milestone 1.

Team Member	**Contribution**
Reggie Schriner	Section 1.4
Tyler Frisinger	Section 1.3
Riley Heimes	Section 1.1
Dylan Kramer	Section 1.2

Chapter 2

Milestone 2: Project Selection

2.1 Introduction

As a group, we have decided to move forward with the NBA Over/Under Predictor project, which forecasts game-by-game player statistics for decision support in sports betting and basketball analytics. We selected this project based on a discussion with Dr. Scott, who recommended it over our alternative idea, which was potentially more controversial. During that meeting, we agreed to begin the project with a simple model that predicts each player's stats based on their recent performance. However, as the project progresses, we plan to incorporate additional complexity, including team-level statistics against specific opponents and defensive metrics for likely individual matchups, both of which can significantly influence a player's output.

2.2 Problem Specification

The team has officially chosen to proceed with the NBA Over/Under Predictor. This specific model will project players stats for future games based on past games stats and player performance. The learning problem for this project is supervised regression, where the inputs are characteristics such as players previous box scores which consists of minutes played (MP), 3 point field goals, 2 point field goals, field goal attempts, total points, turnover %, free throws, player efficiency rating (PER) and more [20], as well as that players overall statistics such as average points per game (PPG), average rebounds per game (RPG), average assists per game (APG). The model will then output real-valued predictions for upcoming games, including points, assists, rebounds and more. As the model complexity grows and the team is getting accurate and reliable results from this base model. The team will introduce more factors that can

affect a players stats such as the opponents team defensive stats and particular players defensive stats.

Predicting player stats matters, lots of people make decisions from these numbers; fantasy managers deciding who to start, sports gamblers deciding if they want to bet, casual fans deciding whether to watch a matchup, and analysts comparing lineups. Even small improvements in accuracy can change those decisions. Our approach is interesting because we go beyond "last-game equals next-game" and build a context-aware predictor: we start simple (recent stats and minutes) and then add opponent context and a lightweight matchup proxy (the likely defender's position/minutes and team defensive rating, i.e., points allowed per 100 possessions). By defining these terms clearly and turning them into numeric features, we make the methodology simple to understand and simple to reproduce.

The resources needed for this project will not be expensive or overly complex. All the team needs in order to complete the creation of this model will be access to the UNL HCC. With access to the UNL HCC we plan on creating our own Conda environments which will be a clone of the CSCE479 environment as well as extra libraries such as XGBoost. The dataset that the team plans to start model creation with has accurate player and team data starting in 1997, up until the end of the 2024 season [10][11].

2.3 Related Work

Predicting basketball performance has been tackled using a wide range of machine learning techniques. For instance, Saladi's DeepShot project used deep learning to predict basketball success rates, demonstrating the ability of neural networks to model complex interactions in sports data [20]. However, a noted pitfall of such models is their high computational cost and risk of overfitting when trained on relatively small or noisy sports datasets.

One significant contribution is DeepShot, which introduced a deep learning framework for predicting basketball success rates [20]. This work demonstrated the value of neural networks in capturing nonlinear interactions between player actions and game outcomes, setting a precedent for applying deep learning to player-level forecasting. More recently, Papageorgiou et al. conducted a study of machine learning models for basketball performance prediction [18]. They found that ensemble methods and gradient boosting achieved stronger performance than simple linear models, especially when incorporating advanced features like pace, efficiency, and contextual metrics. Yet, their work also highlighted that model interpretability can suffer as complexity increases, making it difficult to explain predictions to end users such as coaches or analysts.

XGBoost, in particular, has become a widely adopted library for sports analytics because of its scalability and predictive strength [4]. Prior projects such as Hu et al.'s NBA Player Performance Prediction leveraged XGBoost and synergy features to predict individual player performance [9]. Similarly, other implementations have used random forests to estimate player scoring output,

highlighting how tree-based methods are effective at handling tabular, feature-rich basketball data [8].

Among deep-learning approaches for forecasting, Transformer-based models have emerged as a strong choice for sports analytics. They can look back over recent games, decide which past moments and statistics matter most, and combine that with information about the upcoming matchup. The Temporal Fusion Transformer (TFT) is a great example, it was designed to mix different kinds of inputs (player/team IDs, recent box-score stats, and game context) and to highlight the parts of the history that drive the forecast, this fits our goal of predicting a player's next-game points/rebounds/assists/etc. while accounting for opponent strength and pace [14]. Independent evaluations comparing Transformers (including TFT) to classic baselines also find competitive performance in practice on real forecasting problems [17]. Finally, sports-specific research shows that who you play matters for individual player performance, reinforcing our choice to include opponent information in the Transformer, rather than relying solely on recent game stats that ignore the opponent's defensive strength [15].

Beyond published research, open-source datasets have driven much of this modeling progress. The per-player, per-game box-score corpora from NBA Traditional Boxscores (1997–2024)[11] and NBA Advanced Boxscores (1997–2024)[10] provide exactly the granularity needed for next-game forecasting. These collections span 1996–97 through 2023–24 and include both player and team logs; the advanced set alone accounts for hundreds of thousands of player box scores across 28 seasons. Together they supply rich per-game features (points, minutes, usage, efficiency) and opponent context (team defensive rating, pace, etc.). Using these at scale lets us compare approaches across eras, stress-test robustness, and refine predictors for modern play styles while remaining reproducible.

In summary, prior work shows that advanced machine learning methods, particularly ensembles and deep learning—can achieve strong predictive performance. The pitfalls, however, lie in overfitting on sparse or biased data, loss of interpretability with complex models, difficulty in incorporating contextual variables, and dataset limitations. Our project aims to mitigate these challenges by starting with simple baselines, carefully expanding feature sets, and validating models on recent seasons to account for modern play-styles.

2.4 Proposed Method 1: Graphical Neural Network combined with random forest model

Research shows that combining a Graph-Based Neural Network (GNN) and a Random Forest decision making method performs well at estimating outcomes of NBA games [12]. Reading this brought up the question: Can we utilize this type of architecture to predict certain stats in NBA games? The plan is to use a rolling historical window of games (e.g., the prior X amount of games v before the current date t to construct feature and a graph at each prediction time t.

The idea is we have nodes identified as players and teams. Each player node v in current time has a feature vector $\mathbf{x}_{v,t} \in \mathbb{R}^d$ built from a curated set of 20 stats emphasizing offensive and defensive stats. An example of this would be:

- **Offense:** usage rate, TS%, eFG%, AST%, TOV%, ORB%, points/75, on/off offensive rating, 3PA rate, FTr.
- **Defense:** DRB%, STL%, BLK%, opponent FG%.

All features are standardized within season and computed as *rolling aggregates* up to t to avoid leakage. In our GNN architecture, the above would be gained from our nodes, however what stood out between GNNs and CNNs are the relationships gained from the information held in our nodes. The relationships gained are:

1. **Teammate relationships** (undirected): connect players who share minutes in the last N games;
2. **Matchup relationships** (directed): from offensive player to primary defender.

These relationships will be our edges for this project to capture the chemistry with a certain team, as well as the matchup relationships proposed. The weights for teammate relationships would be the amount of *shared* minutes, and the Matchup relationship edge will be the amount of *matchup* minutes.

A message passing GNN produces player embeddings that summarize each player's current form and relationships. We pool active players to form team embeddings for both sides of the game (offense and defense). For a target offensive player, we combine the player's embeddings with the opponent's primary defender's embeddings. The Random Forest will later take these embeddings and forecast a single-number prediction for points, assists, and rebounds.

The importance of this fusion is to utilize the GNN to capture who plays whom and categorize relationships into embedding vectors. Then use Random Forest to produce accurate estimations based on these embeddings.

For the RF, we input the concatenated vector of the target player embedding, the opponent team embedding, and the likely defender embedding(s). The RF learns non-linear rules from these signals and outputs single-number forecasts for points, assists, and rebounds.Over/under decisions are obtained by comparing the forecast to a provided line, or by training a parallel RF classifier on over/under labels when calibrated probabilities are desired.

2.5 Proposed Method 2: Transformer Neural Network

We forecast an NBA player's next-game points, rebounds, and assists using a compact, TFT-style deep-learning model that combines five pieces: (1) a small Variable Selection Network (VSN) built from Gated Residual Networks (GRNs) that learns which input stats matter most right now, (2) an LSTM that reads a short sequence of each player's recent games in order to capture form and minutes trends, (3) a light self-attention layer that lets the model focus on the most relevant past games (for example, those with similar minutes or usage), (4) a Static Covariate Encoder that turns player/team/opponent/season IDs and opponent metrics (defensive rating, pace) into a compact context vector, and (5) single-number forecast heads that output point estimates for points, rebounds, and assists (plus a helper head for minutes, since time on court drives totals). The VSN cleans and re-weights each game's input features (e.g., minutes, usage, efficiency, shot volume), the LSTM summarizes the recent run of games, self-attention highlights standout moments, and the static encoder injects "who you play" and "when/where" context so the final predictions are matchup-aware. This design follows Temporal Fusion Transformer components [14] while keeping the implementation straightforward and focused on single-value forecasts.

We will train and evaluate on joined per-player, per-game logs from the NBA Traditional and NBA Advanced box-score collections. Preprocessing merges traditional+advanced by (GAME_ID, PLAYER_ID), parses minutes from "MM:SS" to a number, attaches opponent defensive rating and pace by team-season (or a recent rolling average), standardizes identity fields (player, team, opponent, season/era, position if available), and keeps clean per-game inputs such as totals (PTS, REB, AST, 3P, FGA/3PA/FTA, TOV, STL, BLK), efficiencies (FG%, 3P%, TS%, eFG%), usage measures, and minutes. For each target game, we construct a short recent-games window (e.g., last 10, padded and masked when those games are not available) as the sequence for the VSN→LSTM→attention path, and we scale numeric features using statistics fit only on the current training portion to avoid leakage.

To ensure a fair, time-aware evaluation, we use rolling, season-by-season time-series cross-validation rather than a single split. In Fold 1 we train on Seasons 1–2, validate on Season 3, and test on Season 4. In Fold 2 we train on Seasons 1–3, validate on Season 4, and test on Season 5. We continue this pattern, always training on the earliest block, validating on the next season, and testing on the following season; this procedure mimics real deployment and reduces the chance that results hinge on any single year [17].

The end-to-end pipeline is trained with simple losses and regularization, each head (PTS/REB/AST) uses MAE to produce single-number forecasts. The minutes head uses a smaller-weight MAE so the model learns playing-time dynamics without overpowering the main targets. We optimize with AdamW, use dropout in the GRNs and small mixers, clip gradients, and apply early stopping based on validation MAE within each fold. We'll judge the model mainly by how

far off its predictions are on average—that's the Mean Absolute Error (MAE) on the test season in each fold. As a secondary check, we'll also look at Root Mean Square Error (RMSE), which penalizes bigger mistakes more strongly. To make sure the model behaves correctly, we'll break results into simple groups, games where a player plays few minutes (≤ 20), mid minutes (20–30), or heavy minutes (30), and games against strong, average, or weak defenses. Finally, we'll compare our model to straightforward baseline: last game's value, average of the last five games and a simple linear model (Ridge) built from the same inputs. If our approach can consistently beat these, we'll know it's adding real value. We will also run tests by removing opponent features, removing the minutes head, and varying the history length (e.g., 5/10/15 games) to show which choices actually improve accuracy. Sports-specific evidence that "who you play matters" [15] justifies the static-context path and TFT-style work [14] supports variable selection, context encoding, and attention over short histories.

Risks include leakage from improper time handling, missing or inconsistent minutes, role changes (injuries/trades) that make recent history a poor guide, overfitting if the model is too large, and era drift across seasons. We mitigate these by strict time-series cross-validation (train→validate→test in chronological order), computing all features "as of" the day before the target game, standardizing and validating minutes parsing, emphasizing short recent sequences while conditioning on opponent context, keeping the network compact with dropout and early stopping, and focusing on modern seasons while including a season/era indicator. The project timeline spans ten weeks across folds: Weeks 1–2 for data audit, joins, minutes parsing, and opponent context; Week 3–4 for sequence windowing, ID vocabularies, scaling, and coding baselines; Weeks 5–7 to train the full VSN→LSTM→attention→static-context→ mixer→single-number heads model and run the rolling cross-validation folds (logging per-fold validation curves, saving best checkpoints); Week 8 to run ablations and finalize the simplest winning configuration; and Week 9–10 to aggregate cross-validation test results, generate figures, package code/checkpoints, and write the report. Mid-term and final "exams" are concrete: by the midpoint (after the first two folds) the model should beat a last-five average on validation MAE for points (ideally also rebounds and assists) and show improved stability from the minutes head; by the end, averaged across test folds, it should beat all baselines on MAE for points, produce sensible matchup-conditioned behavior, and be fully reproducible from a clean run. References to TFT-style forecasting [14], transformer evaluations under realistic splits [17], and opponent-effect studies in sports [15] support these design and evaluation choices.

2.6 Conclusions

In summary, both proposed approaches offer promising but distinct methods to create an NBA Over/Under predictor, reflecting comparative findings on basketball forecasting methods in the literature [18]. The GNN+Random Forest method provides a hybrid design that naturally incorporates relational informa-

tion between players and teams, using graph connectivity to model teammate chemistry, matchups, and other hidden qualities of the model [12]. Its primary strengths are interpretability and modularity, since graph embeddings and random forest decisions can be analyzed directly. However, this method may face challenges related to graph construction complexity, computational scalability when updating edges for each game, and potential overfitting when player relationships change frequently due to trades or lineup adjustments—issues that are amplified because opponent and matchup context measurably influence player performance [15]. The Transformer Neural Network, on the other hand, offers a modern sequence-based approach that excels at learning temporal dependencies and contextual dynamics [14]. By combining recent game sequences, player and team identifiers, and opponent defensive context, this model can adapt to evolving player performance trends and matchup effects [15]. Its major advantages are flexibility and predictive power, though it comes with a trade-off in interpretability and requires careful handling of temporal data to prevent information leakage and evaluation bias [17]. For example, we may never know what the weights in the model correspond to in the real world. Nevertheless, the transformer framework is a great tool for player-level prediction [14, 18].

Based on current understanding, we anticipate the Transformer approach will outperform the hybrid GNN+Random Forest model in predictive accuracy, particularly for continuous regression targets such as points, rebounds, and assists [17, 18]. However, the GNN approach remains an attractive baseline for exploring relational factors and could complement the transformer model through feature enrichment or ensemble learning [12]. Despite this, we expect to use the Transformer approach for our main predictive model during this project.

Our immediate next steps include building the initial data preprocessing pipeline, implementing time-series cross-validation, and defining standardized evaluation metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). As we progress, we will seek feedback on the strategies best suited for deep models such as the transformer, leveraging techniques introduced with TFT for variable selection and attribution [14]. These discussions will help us with the final production model and ensure the project stays on track, using modern, comprehensive datasets that include traditional and advanced box scores [10, 11].

Table 2.1: Contributions by team member for Milestone 2.

Team Member	**Contribution**
Tyler Frisinger	2.1, 2.2, 2.5, Abstract
Dylan Kramer	2.3, 2.4, 2.6
Reggie Schriner	2.4, 2.5
Riley Heimes	2.4, 2.6

Chapter 3

Milestone 3: Progress Report 1

3.1 Introduction

As a group, our team is working on creating an NBA Over/Under Predictor, which predicts game-by-game player statistics for decision support in sports betting and basketball analytics. After discussing Milestone 2 with Dr. Scott, the team decided to go with a deep neural network as our first approach. By using a neural network, we leverage fast training times, straightforward implementation, and reliable performance for regression tasks. In this milestone, we present our experimental setup, quantitative results, and analysis of the model's strengths and limitations. We also discuss potential improvements for future iterations as we work toward Milestones 4 and 5.

3.2 Related Work

After creating our basic deep neural network, we started researching some ways that this approach could be adapted and modified to follow some more complex or better suited designs in regards to the model itself as well as the data included in the training or testing sets.

3.2.1 Data Points

When researching other we found two other data points or methodologies that could be used to improve our set up. First, we found that some proposed solutions used data with a higher level of detail. According to Don Ha in 2025, great results were achieved when using data that is compartmentalized. For example, they not only used 3-point percentage and attempts, but also used 2-point percentage and attempts which can be deduced from the 3-point percentages and attempts subtracted from the total percentages and attempts

[8]. This higher granularity of data can help the model to better predict patterns since it will have better insight on the player and how they score points. Also, other papers highlighted the benefit of using overall team data as well as player data. For example, team defensive stats could help the model guess how a player will perform against them. This also can help with the player's team stats. For example, the player could have a team that depends on him specifically to score since the other players are of a lower caliber. Using the player's team stats gives the model a better insight on how the player relates to his own teammates. In our research we saw that this could yield promising results. Kai Zhao, and his research in team in 2023, found that using GNNs and other related convolutional networks to map relationships between teams and the team stats gave better predictive results [12].

3.2.2 Relationships Between Players

Our current implementation uses averages for the opposing team to determine how the given player will perform. After seeing our results, we suspect that using data for each specific player on the opposing team and relating them to the given player based on who they face off and how their stats interact with the rest of the team with may give better predictions. As stated by Ruiqi Luo and Vimal Krishnamurthy in 2023, research shows that using methods such as graph attention and convolution can lead to higher prediction accuracy for NBA stats. This research found that using GCNs and similar methods to create an interaction model between the individual players can help the model understand how each individual player affects the performance of the others, yielding a much better prediction model as it now understands the relationships on a player basis, rather than the player relating just to the general opposing team stats [15]. This relationship is also very similar to the team relationships described in section 3.2.1, and we suspect that these two methods put together in tandem will allow the model to think about the prediction wholistically, looking at how the player interacts with the teams, how the player interacts with the other individual players, and how the teams interact with each other as a whole. This could be done through clustering, which, according to Haochen Hu in 2023, can be used to create representations of players to draw relationships by creating archetypes or patterns. This allows certain styles of play to be represented numerically and let the model draw relationships between players based on play style. This would further the player to player relationship effectiveness by giving a latent representation of how that player performs to use to connect players and draw conclusions about certain stats.

3.2.3 Temporal Methods and Player Representation

As stated in Section 3.2.2, through our research we are finding that having a numerical representation of the categorical behavior for players can help predictive models. As stated by Haochen Hu in 2023, clustering can be used to group players into certain types based on their style of play. This representation of

each player can be used in many ways. For one, it can be used to give a better prediction on events that have previous historical data as it allows the model to understand the play style for each player on the team and how these interact. Furthermore, it can be used to give more accurate predictions on situations that have not been seen before, as the model will have an understanding of how the player interacts with other players of a similar type, represented in some kind of latent space where one section could mean better and defense or another could show players who are good facilitators. This would allow the predictions to still be accurate by relating players to how they perform against certain player types, not just certain players [9]. Another approach that we saw was using temporal methods, or using time based convolution or transformers. According to Ruiqi Luo and Vimal Krishnamurthy in 2023, the temporal convolution layer gives the model temporal predictive power, allowing the model to make predictions rooted in factors that revolve around the time period in which they take place like, for example, time of year, time respective to a player's career, and even time related to the basketball season's factors [15]. This research shows how temporal methods could help deduce patterns in how things play out time-wise over the course of a season, the year, or the player's career. Also, we found that time series methods can be used to help predict sequential data as shown by George Papageorgiou in 2024. This research showed that Time Series Transformers significantly outperformed baseline models in predicting future statistics in retail and we believe a similar approach may be beneficial to our model [17]. We may even propose taking one step further, pursuing a Temporal Fusion Transformer because these methods are designed to understand time related dependencies, many independent statistical inputs, and interpretation needs.

Overall, our research has greatly improved our understanding of what implementations and paths have already been taken. We will take these related works and aim to incorporate them into our future processes and use them to shape the methodologies that we pursue to create a powerful predictive NBA model.

3.3 Experimental Setup

The experiments were conducted using a comprehensive dataset of NBA player statistics spanning multiple seasons. Data was sourced from two primary files: `traditional.csv` [11] containing basic box score statistics (minutes, points, rebounds, assists, etc.) and `advanced.csv` [10] containing advanced metrics (player efficiency rating, usage rate, net rating, etc.). The dataset covers NBA games from 1997 through 2024, comprising over 690,000 individual player-game observations.

Data pre-processing involved several key steps. First, the traditional and advanced datasets were merged on player name, date, and minutes played using an inner join to ensure data consistency. To prevent data leakage, all features were engineered using only historical data available before each game. This included

calculating rolling averages over the previous 5 games and cumulative averages across all prior games for 12 core statistics: minutes, field goal attempts, assists, rebounds, turnovers, true shooting percentage, usage rate, net rating, assist percentage, rebound percentage, pace, and player impact estimate. Additionally, we computed the cumulative standard deviation of points to capture scoring volatility since that can play a large roll in predicting accurate point estimates.

The feature set was expanded with several contextual factors, shooting efficiency trends (field goal, three-point, and free throw percentages), defensive metrics (steals, blocks, defensive rating), rebounding percentages, home/away status, win streaks, and minutes stability. Player-specific baselines included career scoring averages and home/away scoring differentials. Opponent defensive context was captured through league-wide opponent points allowed averages and recent defensive performance. After pre-processing, the final feature set contained 42 dimensions. All features were standardized using Standard Scaler to have zero mean and unit variance, which improves neural network training stability and convergence. The target variable (points) was similarly scaled and later inverse-transformed for interpretability of results.

The model architecture consisted of a deep neural network with 4 hidden layers (128-64-32-16 neurons) using ReLU activation functions. Batch normalization was applied after the first two layers, with dropout rates of 0.3, and 0.2 to prevent overfitting. The model was compiled with the Adam optimizer (learning rate=0.001, β_1=0.9, β_2=0.999) and used Huber loss for robustness to outliers. Training employed early stopping (patience=10, min delta=0.001), learning rate reduction (patience=7, factor=0.5), and model check pointing to preserve the best weights.

The dataset was split chronologically with an 80/20 train-test split to maintain temporal integrity. Model performance was evaluated using mean absolute error (MAE), root mean squared error (RMSE), and accuracy within 3, 5, and 8-point margins. As a baseline comparison, we note that a naive model using only player season averages, which predicts the same points value for every game based on historical performance achieves an RMSE of 7.333 in similar NBA scoring prediction tasks [9]. Our model aims to significantly improve upon this baseline by incorporating game-to-game contextual factors and opponent-specific information.

3.4 Experimental Results

For the adjusted results from further alterations, we focused on bettering our results from Milestone 3 when it came to predicting PLayer point totals. After implementing changes we brought up our orginal 40% of predictions that were within 3 points of the actual point totals to a total of 41%. This shows an increase of a substantial 1 percent. for the 5 point differential, the team had a 64% chance of being within 5 points for milestone 3. This went up to 65.3% total classification rate, which shows an upward trend by 1 total percent as well. Finally, for our 8 point differential, our milestone 3 had a 84% accuracy

for a point total with 8 points difference. For our Milestone 4 this number went up to 84.3& which is a magnificent increase of .3% being within 8 points, with an overall Root Mean Squared Error (RMSE) of 6.08 points which is an increase of .08 from milestone 3 (6.16). These updated findings indicate that the model increased in accuracy from Milestone 3 in all three phases of classification for predipredicting player performance. These findings soldify that our model can indeed increase in performance when introducing hyper parameter tuning. These findings also solidy our hypothesis from Milestone 3 where we concluded that it would not be a good idea to predict actual scoring outputs, rather analyze if it is higher or lower the actual point total.

Table 3.1: Comparison of Actual vs. Predicted Points for Selected Players

Player	**Date**	**Actual PTS**	**Predicted PTS**	**Diff**
Terrence Ross	2019-01-19	16.0	13.9	2.1
Aron Baynes	2019-01-19	6.0	4.6	1.4
Darren Collison	2019-01-20	19.0	10.8	8.2
Sindarius Thornwell	2019-01-20	4.0	2.4	1.6
Taj Gibson	2019-01-20	17.0	10.0	7.0
Josh Okogie	2019-01-20	4.0	10.4	6.4
Myles turner	2019-01-20	9.0	13.1	4.1
Davis Bertans	2019-01-20	8.0	19.0	11.0
Avery Bradley	2019-01-20	6.0	7.3	1.3
Demar DeRozen	2019-01-20	15.0	8.1	6.9

As shown in Table 4.1,only three of the ten test players achieved predictions within one point of their actual values, which can be considered highly accurate. This outcome is expected, as players with lower PPG averages generally exhibit smaller variations in performance. In contrast, players with higher scoring averages tend to show greater fluctuations due to the natural variability of “hot” and “cold” shooting nights.

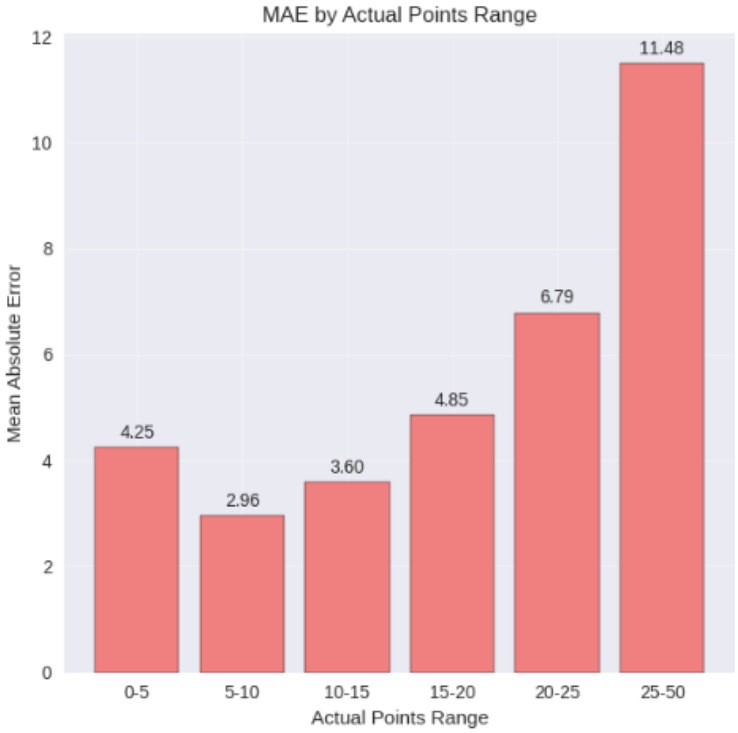

Figure 3.1: Average Prediction Error Across Player PPG Ranges

Figure 4.1 further highlights this variability. Players averaging between 25 and 50 points per game show prediction errors of roughly ± 11 points, which is nearly four times higher than those averaging only 5 to 10 points per game. This demonstrates the model's difficulty in accurately capturing performance for high-scoring players, whose outcomes tend to fluctuate more widely. 16

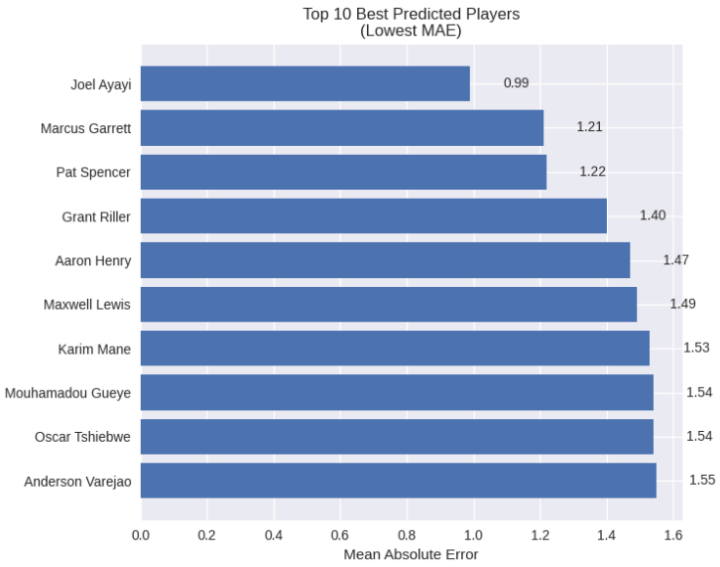

Figure 3.2: Players with the Lowest Mean Absolute Error (MAE) in Predicted Points

From Figure 3.2, a clear pattern emerges among the players with the lowest Mean Absolute Error (MAE) values. Many of these individuals are not high scorers, who contribute only a few points per game. This consistency in low scoring benefits the model, as players with limited scoring ranges exhibit lower variability across games. Consequently, their performances are more predictable compared to high-volume scorers. 17

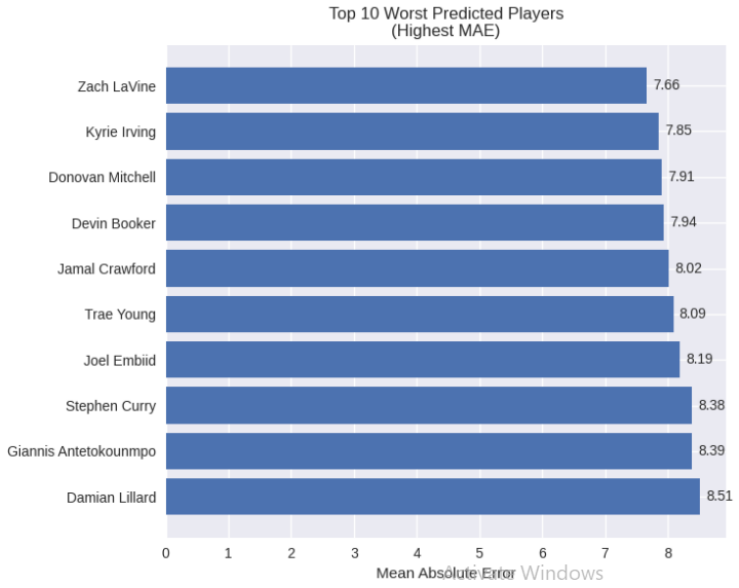

Figure 3.3: Players with the Highest Mean Absolute Error (MAE) in Predicted Points

On the other side of things, Figure 3.3 illustrates the trend among players with the largest prediction errors. Many of these athletes, such as Stephen Curry, Damian Lillard, and other elite scorers, demonstrate significant night-to-night variability. This inconsistency makes their performances inherently more challenging to predict with precision. As a result, the model tends to produce larger errors for these high-scoring players, whose point totals vary widely compared to those of consistent, lower-scoring role players. 18

Figure 3.4: Training and Validation Loss Curves

Regarding model convergence, Figure 5.3 shows that the training loss decreases steadily from an initial value of 0.2300 to approximately 0.2151, indicating a successful learning trajectory. However, the validation loss remains relatively constant around 0.2225 throughout training, suggesting that the model struggles to generalize effectively to unseen data. While this behavior implies some level of overfitting, it also reflects the model's capacity to learn distinct inter-dataset scoring patterns. In our case, this memorization is not entirely detrimental, as the model must still differentiate between low and high-scoring players, which is an essential foundation for refining future predictive accuracy.

3.5 Discussion

The proposed deep learning model achieved a Root Mean Squared Error (RMSE) of 6.09 points, representing a 17.0% improvement over the naive season-averaging baseline (RMSE = 7.333) reported in Section 3.3 and [9]. This improvement demonstrates that the model successfully captures meaningful game-to-game variations that simple averaging fails to reflect. The combination of contextual features, rolling historical statistics, and regularization techniques proved effective in enhancing predictive precision.

Despite these gains, the model still encounters challenges associated with the inherent variability of player performance. High-scoring players, in partic-

ular, display wide fluctuations from game to game, making them more difficult to model accurately. Dropout regularization (rates of 0.2 and 0.3) and early stopping helped mitigate overfitting, but some degree of variance remained in predictions for high-scoring players. These findings emphasize that while the network learns general player tendencies, it remains sensitive to outlier behaviors and sudden performance shifts.

3.5.1 Addressing Regression Toward the Mean

To mitigate the model's tendency to regress toward average scoring predictions, we incorporated several additional feature categories: (1) performance deviation features capturing recent performance relative to career norms, (2) player-type classification identifying scoring roles and volatility patterns, and (3) game context factors affecting scoring opportunities. These features help the model distinguish between systematic role changes and random performance variations, improving prediction accuracy for both high-scoring and low-scoring outliers.

3.6 Work Plan

Building upon the findings from Section 3.5, our next phase of experimentation will focus on addressing the key challenges identified in our initial results, particularly the model's difficulty in accurately predicting high-scoring players and its slight overfitting tendencies.

To achieve this, we plan to integrate player-to-player and team relationship modeling using graph or attention architectures, as described in Sections 3.2.1 and 3.2.2. Specifically, we aim to implement a Graph Neural Network (GNN), Graph Attention Network (GAT) or other attention mechanisms like those within a Temporal Fusion Transformer to capture the relational dynamics between players and teams. This approach will allow the model to consider not just individual performance but also how a player's scoring output is influenced by teammates and opposing defenders. We also intend to experiment with player representations based in clustering to numerically model playing styles, which should help the model generalize performance predictions for both familiar and unseen matchups.

In addition, we will extend our model to include temporal components, such as Temporal Convolutional Networks (TCNs) or, as mentioned before, Temporal Fusion Transformers (TFTs), to better capture trends over time. These architectures can model sequential dependencies and season-based fluctuations in player performance, potentially improving long-term predictive consistency.

Our plan is to first conduct experiments comparing traditional DNNs with temporal and graph-based variants to quantify performance gains in MAE and RMSE. If promising, we will then fuse these methods into a hybrid model that jointly encodes temporal, relational, and contextual information. Over the re-

Table 3.2: Contributions by team member for Milestone 3.

Team Member	Contribution
Tyler Frisinger	3.1, 3.3
Reggie Schriner	3.2, 3.6
Riley Heimes	3.4, 3.5
Dylan Kramer	Abstract Update, 3.7

mainder of the semester, we will allocate two weeks for implementing and testing the graph-based components, two additional weeks for the temporal models, and the final weeks for model integration, hyperparameter tuning, and preparing a comprehensive evaluation of results. This structured approach aims to create a more holistic and accurate NBA player prediction framework that learns not just from isolated game data but from the evolving relationships and interactions that define player performance. Note, while this plan will adequately guide our team to a competent model, the specifics of this plan may be subject to change if experimental results support a pivot in our model implementation path.

3.7 Conclusion

This milestone implemented and evaluated our first neural-network pipeline for next-game NBA player forecasts. Recapping Section 3.1 and the experimental results: using joined traditional and advanced box-score data (1997–2024) with strict, rolling season cross-validation, our tuned model reduces RMSE to approximately 6.1 (about a 17% improvement over a naive season-average baseline). Performance is strongest for low-usage, low-variance role players and weakest for elite, high-usage scorers; per-bucket error analysis shows MAE grows substantially for players who routinely log heavy minutes or exhibit large night-to-night usage swings. We believe this is due to overfitting, and error diagnostics identify minutes volatility and matchup-specific effects as leading causes of large misses. Based on these findings we have already added performance-deviation features and a minutes-prediction pathway and plan the following next steps (short-term and medium-term): (1) ingest expected-minutes and likely-defender matchup features to reduce systematic matchup errors, (2) prototype a compact Transformer using our existing preprocessing to test whether a sequence model improves flexibility and accuracy, (3) run test for accuracy to select the best practical model for Milestone 4. Practical timeline and deliverables are laid out in Section 3.6: in the three weeks before Milestone 4 we will (i) tune the baseline and add the matchup feature, (ii) prototype a lightweight Transformer, and (iii) choose one of the models and tune it to our use case.

Chapter 4

Milestone 4: Progress Report 2

4.1 Introduction

As a group, our team is working on creating an NBA Points Over/Under Predictor, which predicts game-by-game player points for decision support in sports betting and basketball analytics. In Milestone 3 the team decided to pursue a neural network approach. By using a neural network we were able harness the ability of fast training times and a relative straightforward implementation. Within Milestone 4 the team decided to stick to a neural network for those exact reasons as well as a quick discussion with Dr. Scott about switching architectures. In this milestone we present our additions to the experimental setup, quantitative results, and analysis of our model. We also discuss possible improvements and final steps before milestone 5.

4.2 Related Work

Prior research in basketball analytics, offers a variety of modeling approaches for predicting game-level and player-level outcomes. For example, Alameda-Basora in 2019 proposed a Dynamic Bayesian Network to predict the total points scored in NBA games, focusing primarily on game-level totals and wagering outcomes [1]. Similarly, Cheng, and his reserach team in 2013, in their work framed both over/under and spread wagering prediction tasks using traditional machine learning methods such as logistic regression and decision trees [5].

In contrast to game-level prediction, other studies have targeted player-specific performance. Zheng in 2023 presented a machine learning approach, applying regression models (including Random Forests and Support Vector Regression) to forecast player scoring using large-scale public datasets [22]. A broader overview, by Muthumari in 2024, reviews artificial intelligence methods

such as neural networks, gradient boosting, and ensemble models highlighting both the advantages and challenges of applying deep learning in sports analytics [16].

Compared to these prior works, our study focuses specifically on player-level point-total forecasting rather than team or game-level outcomes. Many earlier studies relied on classical machine learning models with relatively simple features, such as basic rolling averages or per-game metrics. Our approach differs by integrating a deeper neural network architecture (six hidden layers) and an enriched feature set that includes defensive matchups, positional data, rolling and cumulative statistics, and contextual factors such as fatigue and rest. Additionally, we implement strict temporal splits and data leakage prevention to ensure realistic performance evaluation.

While previous research has achieved respectable results for broader prediction tasks, relatively few works have targeted individual player scoring regression for over/under decision support. Our model contributes to this underexplored area, building on the literature by combining deep learning techniques with contextualized feature engineering to improve player-level performance prediction.

Moreover, to further this distinction, our team aims to utilize player data that provides up to date player positions to further gain insights on how certain players, their interactions, and their basketball positions change the outcome of player performance. We have found a data set that will provide this information and aim to integrate this player position data into our current model [21].

4.3 Experimental Setup

The experiments were conducted using an enhanced comprehensive dataset of NBA player statistics spanning multiple seasons. Data was sourced from three primary files: `traditional.csv` [11] containing basic box score statistics, `advanced.csv` [10] containing advanced metrics, and a new `positions.csv` [6] file providing official player positional data. The first two datasets covers NBA games from 1997 through 2024, comprising over 690,000 individual player-game observations. The positions dataset provides positional data for over 5,000 players from the past and the present.

Data pre-processing involved several key enhancements. First, the traditional and advanced datasets were merged on player name, date, and minutes played using an inner join to ensure data consistency, with positional data subsequently integrated to enable realistic defensive matchup modeling. To prevent data leakage, all features were engineered using only historical data available before each game. This included calculating rolling averages over the previous 7 games and cumulative averages across all prior games for 12 core statistics: minutes, field goal attempts, assists, rebounds, turnovers, true shooting percentage, usage rate, net rating, assist percentage, rebound percentage, pace, and player impact estimate. Additionally, we computed the cumulative standard deviation of points to capture scoring volatility.

The feature set was significantly expanded with several advanced contex-

tual factors organized into distinct categories. Player-specific baselines included career scoring averages calculated using expanding window means, performance against specific opponents capturing historical matchups, and home/away scoring differentials to account for venue crowd effects. A complex defensive matchup system was implemented using position-based matching that first sought same-position defenders before falling back to similar positions (Ex. point guard defending a shooting guard), enabling the calculation of historical performance against specific defenders including points averages, matchup counts, and scoring volatility. Defender quality was quantified through rolling averages of defensive metrics including steals, blocks, and defensive rating over recent games. Enhanced opponent context included overall defensive strength metrics, specialized wing defense based on steal rates, paint defense derived from block rates, and recent defensive trends calculated over 5-game windows. Additional contextual factors captured game situation including days rest between games, back-to-back game flags, and minutes volatility as a health proxy. Player roles were automatically clustered into star, starter, rotation, and bench categories using historical production thresholds and usage rates, while outlier detection features identified potential for extreme performances through career high comparisons, recent performance spikes, and matchup exploitability against weak defenders. All of this was done in an attempt to more accurately predict the outliers points. After comprehensive pre-processing, the final feature set contained over 60 dimensions. All features were standardized using Standard Scaler to have zero mean and unit variance, which improves neural network training stability and convergence. The target variable (points) was similarly scaled and later inverse-transformed for interpretability of results.

The model architecture was enhanced to a deeper neural network with 6 hidden layers (256-128-64-32-16) using ReLU activation functions. Batch normalization was applied after the first three layers, with increased dropout rates of 0.4, 0.3, 0.3, and 0.2 to prevent overfitting in the more complex network. The model was compiled with the Adam optimizer (learning rate=0.0005, β_1=0.9, β_2=0.999) and used Huber loss for robustness to outliers. Training employed more patient early stopping (patience=15, min delta=0.001), learning rate reduction (patience=8, factor=0.5), and model check pointing to preserve the best weights.

The dataset was split chronologically with an 80/20 train-test split to maintain temporal integrity. Model performance was evaluated using mean absolute error (MAE), root mean squared error (RMSE), and accuracy within 3, 5, and 8-point margins. As a baseline comparison, we used our previous model which achieved an MAE of 5.23 points, RMSE of 6.89 points, and accuracies of 45.1% within 3 points, 68.3% within 5 points, and 85.2% within 8 points. As well as a model that used only player season averages, which predicts the same points value for every game based on historical performance achieves an RMSE of 7.333 in similar NBA scoring prediction tasks [9]. Our goal is to improve upon both of these baselines by incorporating more accurate player/positional data [21] as well as experimenting to help better predict our current outliers.

4.4 Experimental Results

For the adjusted results from further alterations, we focused on bettering our results from Milestone 3 when it came to predicting Player point totals. After implementing changes we brought up our original 40% of predictions that were within 3 points of the actual point totals to a total of 41%. This shows an increase of a substantial 1 total percent. for the 5 point differential, the team had a 64% were within 5 points for milestone 3. This went up to 65.3% total classification rate which shows an increase by 1 total percent as well. Finally, for our 8 point differential our milestone 3 had a 84% accuracy for a point total with 8 points difference. For our Milestone 4 this number went up to 84.3% which is a magnificent increase of .3% being within 8 points, with an overall Root Mean Squared Error (RMSE) of 6.08 points which is an increase of .08 from milestone 3 (6.16). These updated findings indicate that the model increased in accuracy from Milestone 3 in all three phases of classification for predipredicting player performance. These findings solidify that our model can indeed increase in performance when introducing hyper parameter tuning. These findings also solidify our hypothesis from Milestone 3 where we concluded that it would not be a good idea to predict actual scoring outputs, rather analyze if it is higher or lower the actual point total.

Table 4.1: Comparison of Actual vs. Predicted Points for Selected Players

Player	**Date**	**Actual PTS**	**Predicted PTS**	**Diff**
Rudy Gobert	2019-01-16	23.0	11.3	11.7
Al-Farouq Aminu	2019-01-16	14.0	9.0	5.0
Wesley Matthews	2019-01-16	13.0	9.2	3.8
Jevon Carter	2019-01-16	8.0	4.5	3.5
Elfrid Payton	2019-01-16	7.0	9.5	2.5
Montrezl Harrell	2019-01-16	11.0	13.8	2.8
Julius Randle	2019-01-16	23.0	16.9	6.1
Marcus Morris Sr	2019-01-16	6.0	11.7	5.7
Royce O'Neale	2019-01-16	10.0	5.9	4.1
Nene	2019-01-16	7.0	4.3	2.7

As shown in Table 4.1, the performance by the model has increased a bit from Milestone 3. From this random set of players that were run through the model, there is only one player who had a point differential higher than 10 points. Throughout the 2552 total players that went through the model. 84.3% of them are within that 8-point differential.

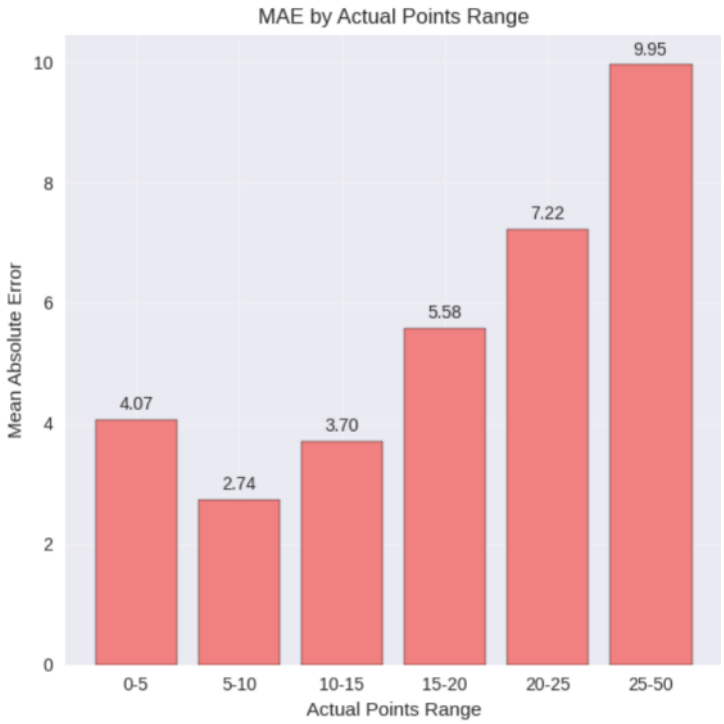

Figure 4.1: Average Prediction Error Across Player PPG Ranges

Figure 4.1. This figure represents a performance increase in all 6 classes of points. Players averaging between 25 and 50 points per game show prediction errors of roughly ± 9.95 points, which is a decrease from the previous 11.4 MAE differential error. This model still demonstrates that it struggles to capture point predictions for high-scoring players rather than low-scoring players due to the natural variability of higher-scoring players whose outcomes tend to fluctuate more widely.

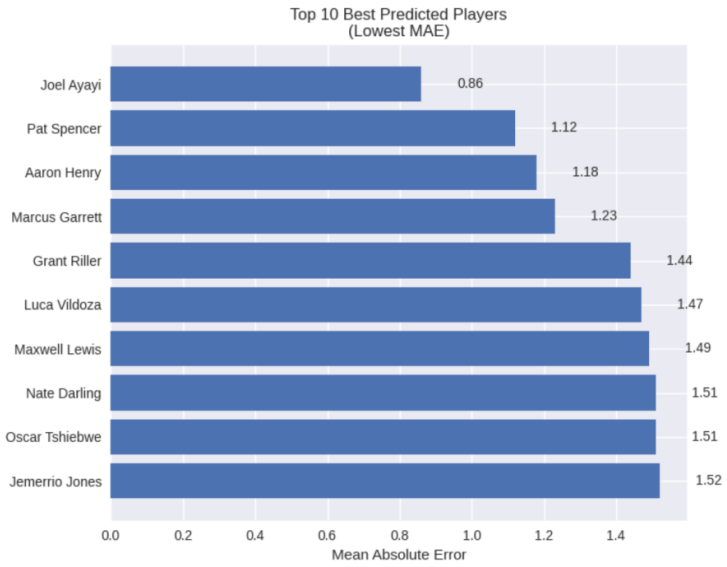

Figure 4.2: Players with the Lowest Mean Absolute Error (MAE) in Predicted Points

From Figure 3.2, we can solidify our assumption from Milestone 3, where we concluded that these players naturally have a lower MAE due to the minimal volatility that is to be assumed by players who cannot score a ton of points. Many of these individuals are not high scorers, who contribute only a few points per game. The fact that there is low variability on this side of our model skews the results a but as the majority of players in the NBA do not score a ton of points. In other words, there are way more players who score within 10 points than those who score in the 20's and beyond, which makes the model perform better than it would if all 2552 players scored 20-plus points per game. These players are more predictable than those who score many more points.

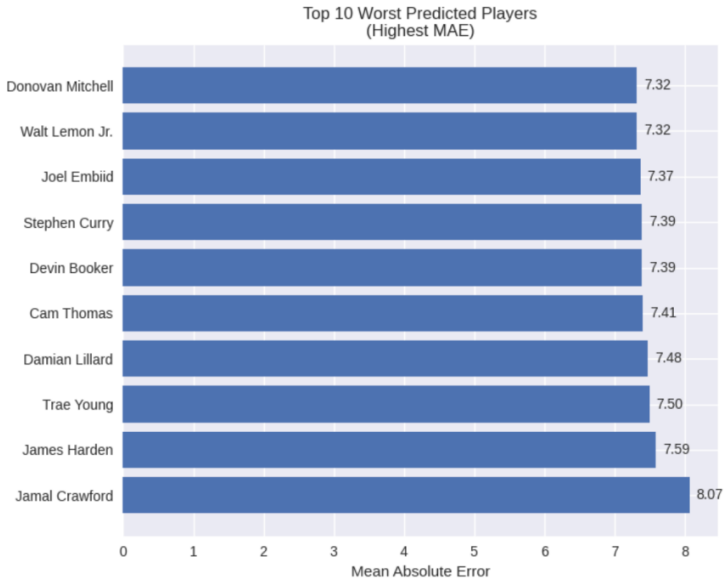

Figure 4.3: Players with the Highest Mean Absolute Error (MAE) in Predicted Points

On the other side of things, Figure 3.3 illustrates the trend among players with the largest prediction errors. AS shown previously in Milestone 3, the majority of these players are known as "High" scoring players and are more widely known through the industry. Many of these athletes, such as Stephen Curry, Damian Lillard, and other elite scorers, demonstrate significant night-to-night variability. It is known that high-scoring players can go through something called a "Shooting slump" where they will consistently shoot under their average in terms of (PPG), which is another reason why it is so difficult to capture an accurate (PPG) for these players, as the phenomenon can happen at random and is impossible to predict which decreases accuracy and precision of the model.

Figure 4.4: Training and Validation Loss Curves

Lastly, Figure 5.3 shows that the training loss decreases steadily from an initial value of 0.2300 to approximately 0.2051, indicating that this model trains well on training data and even shows a minor improvement from Milestone 3, where the model ultimately converged at .210 for an increase of .049 in loss. However, just like in milestone 3, the validation loss remains relatively constant between .220 and 0.225 throughout the validation cycle. This solidifies our assumption that it is finding patterns in the data that do not actually exist in unseen data, which is very obvious from the inherent difficulty of approximating how a player will perform on a given night.

4.5 Discussion

After incorporating additional features such as Player-vs-Player history, opponent defensive ratings, and matchup-specific statistics, we observed an approximate 1% performance increase across all six point-per-game (PPG) classes. This consistent improvement across all classes indicates that the enhancement was not biased toward specific scoring ranges but represented a uniform gain in predictive accuracy. Although this improvement demonstrates meaningful progress from Milestone 3, predicting total points remains inherently uncertain due to

numerous uncontrollable factors, such as player focus, fatigue, or situational dynamics, that can significantly influence individual scoring outcomes. Therefore, while the model's incremental gain is encouraging, the team concluded that the overall predictability of total point values remains too unstable to justify continued pursuit of this specific approach.

To address this issue, the team has decided to re-purpose our current architecture toward an **over-under prediction framework**, based on a player's five-game rolling mean. This pivot aims to reduce the impact of variance by simplifying the output from six categorical classes to a binary over/under prediction. By focusing on short-term performance trends, such as whether a player has recently been performing above or below their average, we anticipate achieving greater stability and interpretability. Evaluation for this revised model will include metrics such as the proportion of correct over/under predictions when compared against both the player's mean and actual game outcomes

4.6 Work Plan

We have two main tasks left in this project. First, we will complete the Final Report using the combined NBA Traditional and Advanced box-score datasets (1997–2024) and the evaluation method introduced in Section 4.3. All the numbers, tables, and figures in Milestone 5 will be recreated from our final model checkpoints and the same train, validation, and test splits we used for this milestone. We will also revise the abstract so it reflects our final results. To make sure our work is reproducible, we will fix random seeds and include a small configuration file that records our data sources, feature options, and hyperparameters mentioned in Section 4.3.

Our second task is to update the player list that supports the player-vs-player (PvP) interaction features to match the 2023–2024 NBA season. We will do this by pulling unique player entries from both the traditional and advanced game logs, using player name, season, and team as identifiers. We will then filter for the 2023–2024 season so the roster only includes active players. To add position information, we will use the positions file referenced in Section 4.3, and when name or position data does not match cleanly, we will double-check against the 2022–2023 NBA Player Stats dataset from Kaggle for accurate names, teams, and positions.

Finally, we will make over/under an accessible output rather than just reporting predicted point totals. Concretely, we will compute a rolling five-game mean of points for each player by appending the most recent five point totals, summing them, and dividing by n to get the average. We will keep our model's predicted point total and label *over* when the prediction is higher than the player's five-game mean and *under* otherwise. We will package this logic in a script that has a CLI so the over/under decision can be run interactively for upcoming games.

The result will be a season-specific player list and refreshed PvP features that align with the 2023–2024 season, plus an over/under decision layer built

on top of our existing predictions. With these pieces in place, we will be able to forecast current players' performances on unseen games while staying consistent with the methods documented in Sections 4.2 and 4.3.

4.7 Conclusion

In this milestone report, we presented our experimental setup, quantitative results, and discussion regarding the performance of our NBA player point prediction model. Our enhanced regression model featuring six hidden layers, batch normalization, dropout regularization, and over sixty engineered features demonstrated measurable improvements over Milestone 3. Accuracy within 3 points increased from 40% to 41%, within 5 points improved from 64% to 65.3%, and within 8 points rose from 84% to 84.3%, while RMSE decreased slightly to 6.08 points. These results validate our hypothesis that deeper neural architectures and richer contextual features can improve model performance for player-level prediction tasks.

However, as discussed in Section 4.5, residual variability, particularly among high-scoring players, remains a challenge, and even advanced models may reach a practical limit due to the inherent randomness in player performance.

Our next phase will shift focus toward over/under classification rather than direct point-total regression. This transition will allow our model to provide decision support insights for betting and analytics. However, the team has some slight reservations that will be listed below. We plan to integrate our model with an API for over/under betting lines and a type of user interface to interact with for getting specific player predictions. We are in the process of finding an API for this need that is free and easy to use as well as determining the best point of contact for the user.

Looking ahead, our questions for the instructor and TA include:

- Is presenting the predictions as an over/under output needed, or can we simply display the point predictions?
- If the over/under is wanted, can we avoid doing so if we cannot find a free API that will provide the over/under line or should we calculate our own lines using the average of the last 5 games for that player from our data or some other live data API?
- Is an elaborate GUI necessary or will a simple command line interface suffice?

Overall, Milestone 4 reflects meaningful progress toward developing a robust, data-driven NBA Over/Under Predictor that combines deep learning and contextual analysis. As we move into Milestone 5, our focus will be on operationalizing the model, expanding feature sets, and ensuring stability and interpretability as well as improving the performance of the model.

Table 4.2: Contributions by team member for Milestone 4.

Team Member	**Contribution**
Reggie Schriner	Sections 4.2 & 4.7
Tyler Frisinger	Sections 4.1 & 4.3
Dylan Kramer	Sections 4.6 & Abstract
Riley Heimes	Section 4.4 & 4.5

Chapter 5

Milestone 5: Final Report

5.1 Introduction

Our goal in this project is to build an NBA Over/Under predictor that forecasts an individual player's next-game point output for decision support in sports betting and basketball analytics. As established in our earlier pipeline design (Section 3.3) and refined feature engineering (Section 4.3), we treat this as a supervised regression problem using leakage-safe, time-aware feature construction and chronological evaluation.

In this milestone, we made several training-level changes to the prediction model that improved stability and final performance:

- Increased batch size.
- Added a `min_delta` threshold to early stopping.
- Changed `ReduceLROnPlateau` patience to 6 (from 8).

These updates target more stable optimization, reduce sensitivity to noisy validation fluctuations, and trigger learning-rate decay sooner once improvement stalls.

Overall, our model now predicts the correct result about 65.4% of the time, with 60% accuracy on the top 10 worst players predicted and 70% accuracy on 10 randomly selected players. In addition, our regression error metrics indicate good generalization: test set MAE = 4.55 points and current-player MAE = 4.14 points. By margin-based accuracy, our model is within 3 points 42.7% of the time, within 5 points 65.6% of the time, and within 8 points 84.3% of the time.

These results continue the upward trend observed in earlier milestones. For example, our within-5-points performance improved from roughly 64% in Milestone 3 to about 65.3% in Milestone 4 (Sections 3.4 & 4.4), and we further improved the overall correctness rate in this milestone by refining the training procedure rather than introducing a major architecture change.

Finally, our current performance is consistent with the range reported in prior basketball prediction work. Cheng et al. reported approximately 68% accuracy in their NBA betting-line project, and Zhao et al. reported 71.54% accuracy using a fused graph convolution and random forest approach for game outcome prediction [5, 12]. Although these works focus on different targets (game outcomes / betting tasks rather than player-level point regression), their reported performance provides a useful reference band, and our results fall close to that 65–70% range.

5.2 Related Work

This milestone primarily improved performance through training procedure refinements (batch size, early stopping sensitivity, and learning rate adaptation), rather than through new features or a new model family. As a result, the most relevant related work for these changes comes from deep learning optimization and training best practices, which are broadly applicable across domains (including sports prediction).

- **Large minibatch training:** Goyal et al. show that large minibatches can maintain accuracy when combined with appropriate training techniques (notably learning-rate scaling and warmup), demonstrating that increasing batch size does not necessarily reduce final model quality [7]. While their work targets large-scale vision (ImageNet) rather than tabular sports data, the core motivation aligns with our change: larger batches can yield more stable gradient estimates and smoother optimization, which in our case contributed to a more reliable training trajectory and improved final performance.
- **Early stopping criteria and validation noise:** Prechelt studies early stopping using cross validation and emphasizes that stopping criteria must account for noisy validation behavior to avoid stopping too early or oscillating around marginal improvements [19]. This supports our decision to introduce a `min_delta` threshold in early stopping: it prevents tiny (often noise-level) validation changes from resetting patience, which helps select checkpoints that generalize better.
- **Learning-rate schedules after plateaus:** Bengio provides practical guidance for training deep architectures, including the importance of learning rate schedules and decay in achieving strong final performance once progress slows [2]. Our adjustment of `ReduceLROnPlateau` patience (8 $\rightarrow$ 6) follows this general recommendation by decaying the learning rate sooner after validation improvement stalls, which in turn helped the optimizer refine solutions rather than remaining at an overly aggressive step size late in training.

In contrast to basketball specific modeling papers (e.g., outcome prediction with engineered features or graph-based representations), these sources justify

the optimization decisions that improved our current milestone without requiring changes to the underlying learning task definition (Section 3.3) or the core evaluation strategy (Section 4.3).

5.3 Experimental Setup

The experiments were conducted using the same enhanced comprehensive dataset of NBA player statistics from previous milestones, with no changes to the feature engineering pipeline. Data continued to be sourced from three primary files: `traditional.csv` [11] containing basic box score statistics, `advanced.csv` [10] containing advanced metrics, and `positions.csv` [21] providing official player positional data. The dataset covers NBA games from 1997 through 2024, comprising over 690,000 individual player-game observations. For player-versus-player (PvP) matchmaking, we used our updated player dataset aligned with the 2023-2024 NBA season, ensuring accurate defensive matchup modeling with current roster and position data.

The pre-processing pipeline remained identical to section 4.3. This included merging traditional and advanced datasets on player name, date, and minutes played, with positional data integrated for defensive matchup modeling. All features were engineered using only historical data available before each game to prevent data leakage. This included calculating rolling averages over the previous 7 games and cumulative averages across all prior games for the same 12 core statistics, minutes, field goal attempts, assists, rebounds, turnovers, true shooting percentage, usage rate, net rating, assist percentage, rebound percentage, pace, and player impact estimate. The cumulative standard deviation of points continued to be computed to capture scoring volatility.

The feature set contained the same advanced contextual factors organized into distinct categories as described in Section 4.3 This included player-specific baselines (career scoring averages, performance against specific opponents, home/away scoring differentials), the complex defensive matchup system using position-based matching with our updated 2023-2024 player dataset for accurate PvP relationships, defender quality quantification through rolling averages of defensive metrics, enhanced opponent context (overall defensive strength, specialized wing and paint defense), and additional contextual factors capturing game situations. Player roles continued to be automatically clustered into star, starter, rotation, and bench categories, while outlier detection features identified potential for extreme performances. After pre-processing, the final feature set contained the same over 60 dimensions as detailed in appendix A. All features were standardized using Standard Scaler to have zero mean and unit variance, which improves neural network training stability and convergence. The target variable (points) was similarly scaled and later inverse-transformed for interpretability of results.

The model architecture was refined to a more streamlined neural network with 2 hidden layers (64-32 units) using ReLU activation functions, which we found to reduce overfitting while maintaining predictive performance. Batch

normalization was applied after each dense layer, with dropout rates of 0.3 and 0.2 respectively. The model was compiled with the Adam optimizer (learning rate=0.001, β_1=0.9, β_2=0.999) and used Huber loss for robustness to outliers. Training employed early stopping (patience=15, `min_delta`=0.0005), learning rate reduction (patience=6, factor=0.5), and model check pointing to preserve the best weights. We increased the batch size to 128 for more stable gradient updates, a change supported by large-batch training literature [7].

The dataset continued to be split chronologically with an 80/20 train-test split to maintain temporal integrity. Model performance was evaluated using mean absolute error (MAE), root mean squared error (RMSE), and accuracy within 3, 5, and 8-point margins. We compared our model against three baselines established in previous work, our previous Milestone 4 model, a season-average baseline that predicts the same points value for every game based on historical performance (achieving an RMSE of 7.333 in similar NBA scoring prediction tasks [9]), and real-world performance testing against current 2025-2026 NBA games to validate practical applicability.

5.4 Experimental Results

We evaluate the final model using standard regression metrics and distribution-based error analysis to assess both numerical accuracy and practical usefulness for over/under decision support. All results are reported on a held-out test set using strictly chronological splits to preserve temporal validity.

Overall, the model achieves a Mean Absolute Error (MAE) of 4.55 points and a Root Mean Square Error (RMSE) of 6.01 points. These results represent a substantial improvement over a naive season-average baseline (RMSE = 7.33) and earlier milestone models. However, point estimates alone do not fully capture the model's real-world utility, motivating a more detailed analysis of error distributions and scoring-context behavior.

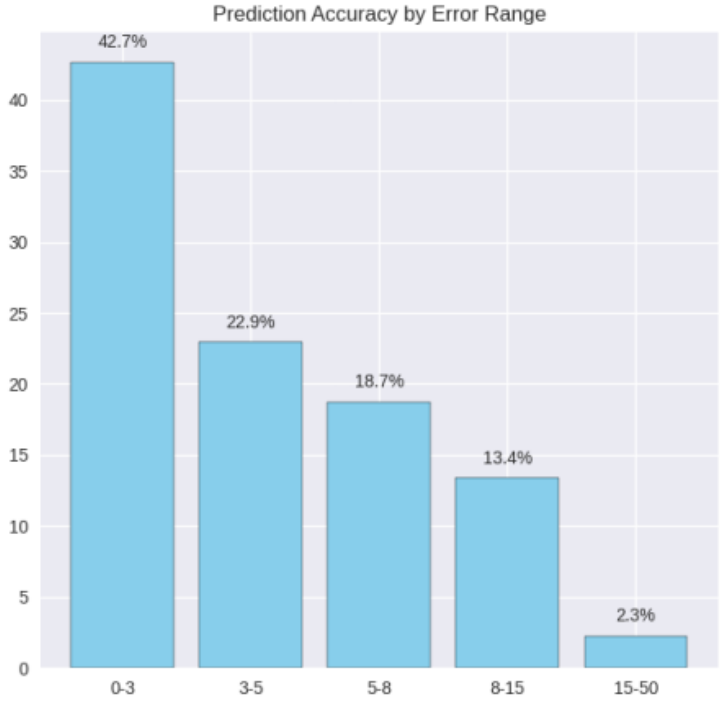

Figure 5.1: Distribution of prediction error by absolute error range on the test set. Bars represent disjoint error intervals. Cumulative accuracy is obtained by summing bins: predictions within 3 points account for 42.7%, within 5 points account for 65.6% (0–3 + 3–5), and within 8 points account for 84.3% (0–3 + 3–5 + 5–8).

Figure 5.1 represents the distribution of prediction error by absolute error ranges. The bars represent disjoint error intervals. The largest portion of predictions (42.7%) fall within three points of the true outcome, indicating strong short-term precision. For cumulative accuracy, 65.6% fall within 5 points (0–3 + 3–5), and 84.3% fall within eight points (0–3 + 3–5 + 5–8). This concentration of error near zero supports the suitability of the model for threshold-based over/under decisions rather than exact point-total forecasting.

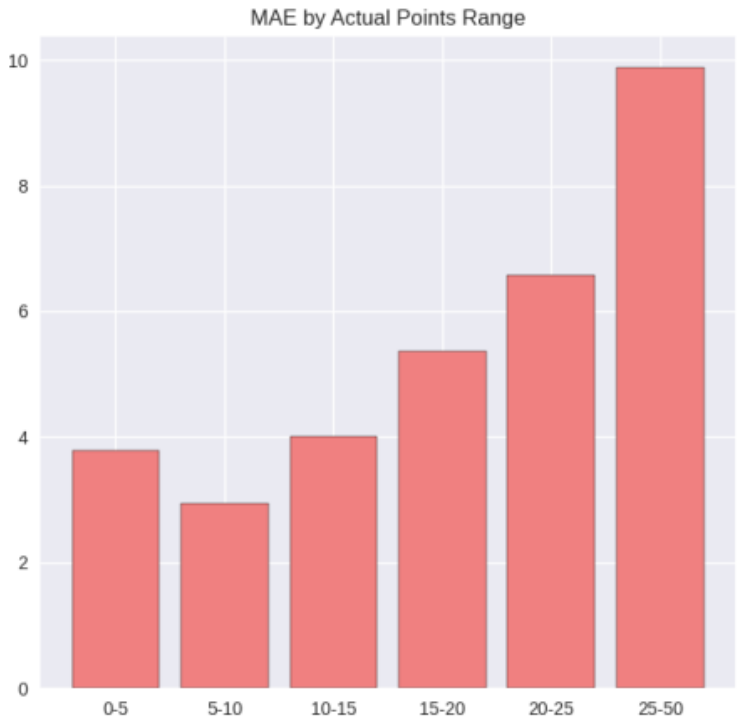

Figure 5.2: Mean Absolute Error (MAE) stratified by actual player scoring ranges. Prediction error increases with scoring volume, highlighting the inherent difficulty of forecasting high-usage players who exhibit greater night-to-night variance.

To understand how prediction error varies with player scoring volume, Figure 5.2 shows Mean Absolute Error based on actual points scored by a player. Error increases based on scoring ranges, with higher scoring players having an increase of MAE compared to lower scoring players. In this example, players scoring fewer than 10 points per game exhibit low MAE due to constrained scoring distributions, while players scoring above 25 points per game exhibit MAE approaching 10 points. This trend reflects the inherent variability of high-usage offensive roles and explains why elite scorers dominate the set of worst-predicted players.

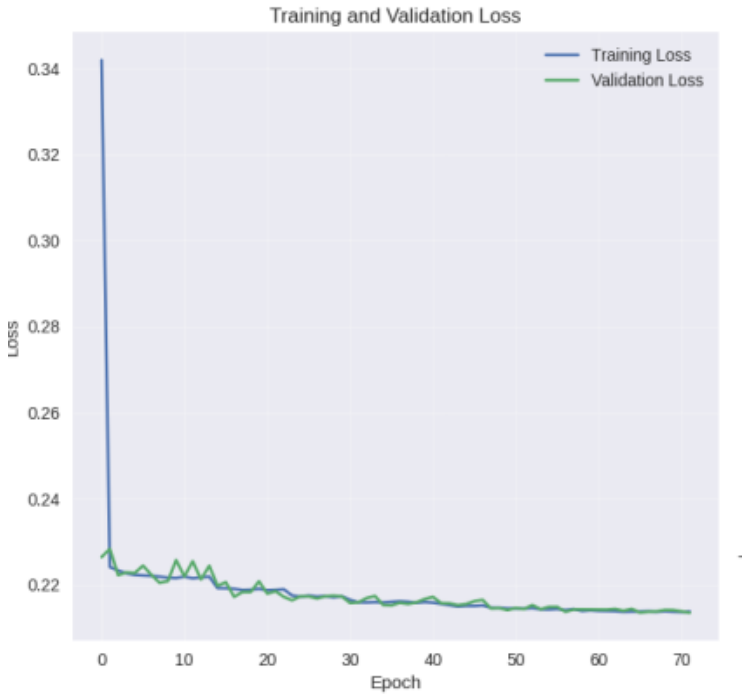

Figure 5.3: Training and validation loss curves for the final model. Both curves converge smoothly with minimal divergence, indicating stable training behavior and limited overfitting despite the model's depth and expanded feature set.

We further evaluate training stability and generalization behavior in Figure 5.3, which shows training and validation loss curves for the final model. Both curves decrease smoothly and remain closely aligned throughout training, indicating stable optimization and limited overfitting. The Residual prediction error is therefore attributed primarily to the variability of player performance rather than the instability of the model.

Finally, to assess the applicability of this model, we analyze performance on active players within the most recent years of data. On this subset, the model achieves an MAE of 4.14 $\pm$1.45 points, demonstrating improved consistency when forecasting contemporary players whose pace, usage, and roles align closely with the training distribution.

5.5 Discussion

The experimental results demonstrate that the proposed neural prediction pipeline captures meaningful structure in NBA player scoring data while remaining significant under realistic temporal evaluation. The model achieved an MAE of 4.55 points and an RMSE of 6.01 points, indicating that the model ef-

fectively learns short-term performance trends, minutes stability, and matchup context that simple baselines fail to capture.

A key observation from error analysis is the systemic relationship between prediction error and players' usage. Low-usage and role players exhibit consistently low error due to the true nature of variability and the amount of noise in the data. However, high-usage players show substantially larger deviations, which further boosts this claim that with an increase in point totals, the variation from the mean is more substantial. Influences with higher point variability include: defensive schemes, foul trouble, in-game adjustments, and shooting variance. These factors can severely influence the number of points scored by a certain player and are inherently difficult for a model to predict with these types of data.

Taken together, these findings motivate a shift in how predictions are interpreted. While exact point regression provides useful signals, it is less stable than relative decision frameworks. Framing predictions in terms of over/under outcomes relative to a short-term player baseline reduces sensitivity to extreme events while preserving matchup-aware forecasting benefits. This reframing aligns more naturally with real-world betting, fantasy, and analytics applications.

5.6 Conclusion

In conclusion, the accumulative result of the design, implementation, and evaluation of NBA player performance predictions throughout this semester concludes that a deep neural network can be utilized to estimate over/under probabilities with accuracy. This project utilized NBA traditional and advanced box-score metrics to augment positional and player-vs-player matchup features that achieved a Mean Absolute Error of 4.55 points and a Root Mean Squared Error of 6.01 points under strict evaluation.

Our analysis demonstrates that the model captures short-term performance trends and matchup context more effectively than generic historical baselines, while maintaining stable training behavior and limited overfitting. Error analysis revealed that prediction accuracy is highest for low-usage players and the accuracy is lower for high-usage players, with an increase in night-to-night volatility in stat points-per-game, which reflects the natural variability in the game rather than due to deficiencies in model design.

A key contribution of this work is the informed reframing of the prediction task. While exact point-total regression provides valuable insight into valuable signals, it is less stable and less actionable by a user than a relative decision framework. By modeling outputs as an Over/under decision relative to a rolling short-term player baseline, the system produces predictions that are more robust, interpretable, and aligned with real-world betting, fantasy, and analytical applications.

Future work includes extending the framework to additional player statistics such as rebounds and assists, integrating live betting lines through an exter-

nal API, and exploring sequence-based architectures such as Temporal Fusion Transformers to better capture nonlinear temporal dynamics. Overall, This project demonstrates that effective sports analytics modeling requires not only an accurate prediction, but a careful problem formulation that respects the uncertainty inherent in athletic performance.

Table 5.1: Contributions by team member for Milestone 5.

Team Member	**Contribution**
Reggie Schriner	Section 5.1 & 5.2
Tyler Frisinger	Section 5.3
Dylan Kramer	Abstract
Riley Heimes	Section 5.4 & 5.5 & 5.6

Appendix A

Features Breakdown

Table A.1: Basic Statistical Features

Feature	**Data Type**	**How Calculated**	**Predictive Value**
MIN_cum_avg	float	Cumulative mean of minutes played	Indicates long-term role and opportunity, more minutes strongly correlate with scoring.
MIN_last_7	float	7-game rolling mean of minutes	Captures recent role trends, identifies minute increases or decreases.
FGA_cum_avg	float	Cumulative mean of field goal attempts	Measures long-term shot volume, top predictor of scoring.
FGA_last_7	float	Recent 7-game shot attempt average	Detects short-term usage spikes or declines.
AST_cum_avg	float	Season average assists	Indicates involvement in playmaking, correlates with ball-handling role.
AST_last_7	float	7-game assist average	Measures recent role change in offensive orchestration.
REB_cum_avg	float	Season average rebounds	Identifies physical presence, important for bigs whose scoring correlates with rebounding.
REB_last_7	float	Rolling mean rebounds	Captures short-term changes in activity level.
TOV_cum_avg	float	Cumulative mean turnovers	Turnovers correlate with ball dominance (higher USG).

TOV_last_7	float	Recent turnover rate	Indicates short-term role pressure or defensive matchup issues.
TS%_cum_avg	float (0–1)	Cumulative true shooting	Reflects long-term scoring efficiency.
TS%_last_7	float (0–1)	Rolling TS%	Detects hot/cold shooting stretches.
USG%_cum_avg	float	Cumulative usage rate	Strong predictor of scoring volume.
USG%_last_7	float	7-game usage	Captures changes in offensive role.
NETRTG_cum_avg	float	Season net rating	Strong teams create high-scoring environments.
NETRTG_last_7	float	Rolling net rating	Reflects momentum and team performance trends.
AST%_cum_avg	float	Season assist percentage	Indicates playmaking responsibility.
AST%_last_7	float	Last 7-game assist rate	Detects role adjustments.
REB%_cum_avg	float	Season rebound percentage	Measures ability to generate second-chance points.
REB%_last_7	float	Rolling rebound percentage	Short-term indicator of activity level.
PACE_cum_avg	float	Cumulative possessions per 48 minutes (one game)	Faster pace means more scoring opportunities.
PACE_last_7	float	Rolling pace value	Detects fast/slow matchup trends.
PIE_cum_avg	float	Cumulative player impact estimate	Measures overall involvement in game flow.
PIE_last_7	float	Rolling player impact estimate	Captures changes in holistic contribution.
PTS_cum_std	float	Std. dev. of season points	Measures scoring variability, stabilizes predictions.

Table A.2: Shooting & Defensive Trends

Feature	**Data Type**	**How Calculated**	**Predictive Value**
FG%_last_7	float	7-game FG%	Captures short-term shooting form.
3P%_last_7	float	7-game 3P%	Identifies hot/cold 3-point streaks.

FT%_last_7	float	7-game FT%	Indicates free-throw reliability.
STL_last_7	float	7-game steal rate	Measures recent defensive disruption.
BLK_last_7	float	7-game block rate	Indicates rim protection activity.
DEFRTG_last_7	float	Opponent points allowed per 100	Shows recent defensive success/fatigue.
OREB%_last_7	float	Rolling offensive rebound %	Correlates with putback scoring.
DREB%_last_7	float	Rolling defensive rebound %	Indicates activity and matchup strength.

Table A.3: Game Context Features

Feature	**Data Type**	**How Calculated**	**Predictive Value**
home_game	int (0/1)	1 if home, 0 if away	Players score more at home due to familiarity and less travel.
win_streak	int	Rolling sum of wins	Team momentum influences scoring environment.
MIN_volatility	float	Std. dev. of MIN over last 10	Captures role stability vs uncertainty.

Table A.4: Player-Specific Baselines

Feature	**Data Type**	**How Calculated**	**Predictive Value**
player_career_avg_pts	float	Cumulative avg points	Long-term skill baseline.
player_avg_vs_opponent	float	Cumulative mean vs this opponent	Captures matchup-specific scoring history.
home_away_scoring_diff	float	Home avg – away avg	Models environment-based scoring differences.

Table A.5: Opponent Team Defense Features

Feature	**Data Type**	**How Calculated**	**Predictive Value**
opponent_avg_points_allowed	float	Opponent season avg points allowed	Core measure of defensive strength.

opponent_recent_defense	float	Rolling 5-game opponent points allowed	Captures short-term defensive form.

Table A.6: Defender Matchup Features

Feature	**Data Type**	**How Calculated**	**Predictive Value**
defender_avg_STL	float	Defender 10-game steal avg	Measures pressure and turnover threat.
defender_avg_BLK	float	Defender 10-game block avg	Measures rim deterrence.
defender_avg_DEFRTG	float	Defender defensive rating	Models difficulty of primary matchup.
defender_games_considered	int	Constant (10)	Standardizes defender sample size.
historical_pts_vs_defender	float	Avg pts vs this defender	Personalized matchup effect.
num_previous_matchups	int	Count of past matchups	Determines reliability of historical stats.
pts_std_vs_defender	float	Std dev of pts vs defender	Measures matchup volatility.

Table A.7: Enhanced Opponent Features

Feature	**Data Type**	**How Calculated**	**Predictive Value**
opponent_defensive_strength	float	Season-long opponent DEF efficiency	Measures macro-level difficulty.
opponent_wing_defense	float	Opponent STL-based perimeter defense	Key for guards/wings scoring.
opponent_paint_defense	float	Opponent BLK-based paint defense	Key for bigs scoring.
opponent_defensive_trend	float	Rolling 5-game DEFRTG trend	Captures hot/cold defensive stretches.

Table A.8: Context & Fatigue Features

Feature	**Data Type**	**How Calculated**	**Predictive Value**

days_rest	int	Days since previous game	Rest strongly correlates with scoring efficiency.
is_back_to_back	int (0/1)	1 if days_rest = 1	Models fatigue-led scoring drops.
is_3_in_4	int (0/1)	1 if multiple games in short span	Captures heavy schedule effects.
opponent_defensive_form_5g	float	Opponent rolling DEF form	Captures near-term matchup quality.
MIN_volatility_10g	float	Std of minutes last 10 games	Measures uncertainty in opportunity.
pace_adjustment	float	Game pace / player avg pace	Adjusts expected possessions contextually.

Table A.9: Role Cluster Indicators

Feature	**Data Type**	**How Calculated**	**Predictive Value**
role_bench	int (0/1)	One-hot label	Helps model usage and minutes expectations.
role_rotation	int (0/1)	One-hot label	Identifies mid-level players with stable minutes.
role_starter	int (0/1)	One-hot label	Starters have more predictable scoring.
role_star	int (0/1)	One-hot label	Captures star-level scoring ceilings.

Table A.10: Outlier & Exploit Features

Feature	**Data Type**	**How Calculated**	**Predictive Value**
career_high	float	Cumulative max points	Establishes player scoring ceiling.
pts_vs_career_high	float	career_avg / career_high	Measures how close player is to ceiling.
recent_spike	float	Z-score of last 3 vs season avg	Identifies scoring surges or anomalies.
matchup_exploit	float	career_avg / defender_DEFRTG	Measures exploitability of matchup.

Bibliography

[1] Enrique Alameda-Basora. A dynamic bayesian network to predict the total points scored in nba games. Thesis, 2019. Analyzes total points prediction in NBA games using a dynamic Bayesian network. URL: https://www.imse.iastate.edu/files/2019/07/Alameda-BasoraEnrique-thesis.pdf.

[2] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. arXiv preprint, 2012. URL: https://arxiv.org/abs/1206.5533, arXiv:1206.5533.

[3] Apoorv (blitzapurv). Nba players data (1950 to 2022). https://www.kaggle.com/datasets/blitzapurv/nba-players-data-1950-to-2021/data?select=player_data.csv, 2021. Accessed: 2025-09-14.

[4] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 785–794, 2016. doi:10.1145/2939672.2939785.

[5] Daniel Cheng, Adam Dade, Michael Lipman, and Cameron Mills. Predicting the betting line in nba games. CS229 Machine Learning Project Report, Stanford University, 2013. Machine learning methods for predicting NBA game lines and totals. URL: https://cs229.stanford.edu/proj2013/ChengDadeLipmanMills-PredictingTheBettingLineInNBAGames.pdf.

[6] Aiden Flynn. V2: Nba player database. Kaggle, 2025. URL: https://www.kaggle.com/datasets/flynn28/v2-nba-player-database.

[7] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. arXiv preprint, 2017. URL: https://arxiv.org/abs/1706.02677, arXiv:1706.02677.

[8] Don Ha. Using random forest regression to predict nba players' scoring output. https://medium.com/inst414-data-science-tech/using-random-forest-regression-to-predict-nba-players-scoring-output-6bf7995e169e, 2025. Accessed: 2025-10-02.

[9] Haochen Hu, Gantcho Dimitrov, David Menn, and Songhao 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. Accessed: 2025-10-02.

[10] Szymon Jóźwiak. Nba advanced boxscores 1997–2024. Kaggle, 2024. Per-player and team advanced box scores from the 1996–97 through 2023–24 seasons. URL: https://www.kaggle.com/datasets/szymonjwiak/nba-advanced-boxscores-1997-2023.

[11] Szymon Jóźwiak. Nba traditional boxscores 1997–2024. Kaggle, 2024. Player and team box scores from the 1996–97 through 2023–24 seasons. URL: https://www.kaggle.com/datasets/szymonjwiak/nba-traditional.

[12] Zhao Kai, Du Chunjie, and Tan Guangxin. Enhancing basketball game outcome prediction through fused graph convolutional networks and random forest algorithm. *Entropy*, 25(5):765, may 2023. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC10217531/, doi:10.3390/e25050765.

[13] Jungseock Joo, Kimmo Kärkkäinen. Fair face. accessed: 2025-09-11. URL: https://github.com/dchen236/FairFace?tab=readme-ov-file.

[14] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting, 2019. URL: https://arxiv.org/abs/1912.09363, arXiv:1912.09363.

[15] Ruiqi Luo and Vimal Krishnamurthy. Who you play affects how you play: Predicting sports performance using graph attention networks with temporal convolution, 2023. URL: https://arxiv.org/abs/2303.16741, arXiv:2303.16741.

[16] R. Muthumari, A. Shanmugam, and S. Karthik. The application of artificial intelligence techniques in predicting basketball game outcomes. *Journal of Sports Analytics*, 10(3):155–170, 2024. Overview of AI methods for basketball outcome prediction. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC12200876/.

[17] João M. Oliveira, João B. Pereira, and Pedro M. Ferreira. Evaluating the effectiveness of time series transformers for retail demand forecasting. *Mathematics*, 12(17):2728, 2024. URL: https://www.mdpi.com/2227-7390/12/17/2728, doi:10.3390/math12172728.

[18] George Papageorgiou, Vangelis Sarlis, and Christos Tjortjis. Evaluating the effectiveness of machine learning models for performance forecasting in basketball: a comparative study. *Knowledge and Information Systems*, 66:4333–4375, 2024. doi:10.1007/s10115-024-02092-9.

[19] Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998. URL: https://www.sciencedirect.com/science/article/pii/S0893608098000100, doi:10.1016/S0893-6080(98)00010-0.

[20] Vamsi Saladi. *DeepShot: A Deep Learning Approach To Predicting Basketball Success*. PhD thesis, Department of Computer Science, Stanford University, 2020. URL: https://cs230.stanford.edu/projects_winter_2020/reports/32643049.pdf.

[21] Vivo Vinco. 2022–2023 nba player stats (regular season). Kaggle, 2023. Includes player name, position, and per-game regular season statistics for the 2022–2023 NBA season. URL: https://www.kaggle.com/datasets/vivovinco/20222023-nba-player-stats-regular/data.

[22] Yi Zheng, Chen Li, and Wei Chen. Nba player score prediction based on machine learning. In *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2023)*. SCITEPRESS, 2023. Applies regression models to player scoring prediction using public datasets. URL: https://www.scitepress.org/publishedPapers/2023/128017/pdf/index.html, doi:10.5220/0012801700003419.