Predicting NBA 2K Ratings and All-Star/NBA Selections Using Player Statistics

Daniel Carter & Jack Schwanewede

Abstract

This paper explores the application of Machine Learning techniques to predict an NBA player's *2K* rating and whether a player is considered an Elite player (All-Star/All-NBA selection) based on NBA regular season statistics. Our dataset was sourced from Kaggle, which included player stats for 6 seasons and their *2K* rating, and we took all the necessary preparations of cleaning, scaling, and preparing the dataset, such as manually adding the Elite variable. To make our predictions, we created two multi-task models using PyTorch, one with shallow output heads and the other with deeper, more complex output heads. We then tested each model against unseen data. We find that the model with deeper output heads performs better than our shallow-headed model, with an R^2 score of 0.9001 and an F1 score of 0.8275. Overall, we conclude that multi-task models can be effective in predicting *2K* ratings and classifying player calibers, yet there can be further improvements on our results that can be implemented in future instances of similar models for joint regression and classification tasks.

Introduction

NBA *2K* is a yearly basketball video game series created by 2K games, which releases every September. The developing company, 2K Games, collaborates with basketball leagues such as the NBA, WNBA, and FIBA to include their teams and players. In order to reflect the skills and strengths of each player, the developers assign rating values to each player, which is out of 99. These *2K* ratings are a balance of individual skill ratings, such as a player's 3 point shot, speed, strength, hustle, layup, dunk, and more. However, the process in which developers create these ratings is not transparent, leading to controversy from fans and players themselves when the *2K* ratings are revealed. Producer Michael Stauffer stated that "it's a lot about feel… does it 'feel right' when you're playing the game?" (Wong, 2017).

Another measure of a player's skill or success is their selection to an All-Star or All-NBA team. Being selected to these teams is an achievement of an "Elite" player, and a mark of a strong performance throughout a season. These selections are decided by players, fans, media and coaches, which sometimes results in controversial "snubs" where star players can be excluded from the elite list. That is to say, these selections are often based on several factors, and are subjective. Since there is not an exact science to creating *2K* ratings or selecting the Elite teams, we wanted to see whether we could use a more objective, quantitative measure to predict this.

To do this, we used real NBA statistics from a player's season to capture the "feel" of a player's *2K* rating, and whether they had an Elite season. In this project, we implement a multi-task deep learning model using PyTorch, a Machine Learning Library. Our model has two outputs: a regression output that returns the *2K* score, and a binary classification output that returns whether the player was Elite or not. We train and compare two versions of our model - a Version 1 that has shallow output heads, and a Version 2 that has deeper, more complex output heads. We test the performance of our *2K* score using the Mean Squared Error (MSE) loss function, and create a plot of predicted versus actual ratings with a calculated R^2 correlation score. We test performance of the Elite predictions using the Binary Cross Entropy (BCE) loss function, and create a confusion matrix to compute F1 score and accuracy percentage. We find that Version 2 performs slightly better, and these results demonstrate how sports statistics can be used to make more quantitative predictions and measures of subjective accolades and player assessment.

Background

Data Exploration

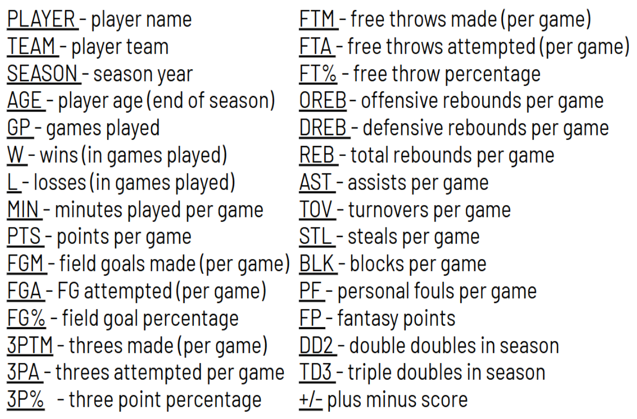
Our dataset comes from the user William Yu on Kaggle, who scraped NBA statistics and *2K* ratings from Hoopshype.com and NBA.com. The dataset has 6 seasons of data, corresponding with each player's *NBA 2K* rating in the following game:

- 2014-15 season (*NBA 2K16*)
- 2015-16 season (*NBA 2K17*)
- 2016-17 season (*NBA 2K18*)
- 2017-18 season (*NBA 2K19*)
- 2018-19 season (*NBA 2K20*)
- 2019-20 season (*NBA 2K21*)

Ratings are calculated following the season, which explains why the season year and *2K* game may seem disjointed. For example, the 2014-15 season is played through October to June, and then ratings are calculated in September when *NBA 2K16* releases.

In total, the dataset had $2,412$ examples, where each example is a player and a season (ex. LeBron James, 2017-18 season). Because of this, many players are repeated in multiple seasons, since they are each considered different entries with different statistics. Overall, there are 779 unique players represented in the data.

Initially, our dataset had 31 columns, 30 of which were features and 1 was one of our target variables (*2K* Rating). A brief explanation of each feature is displayed in Figure 1.



PLAYER - player name	FTM - free throws made (per game)
TEAM - player team	FTA - free throws attempted (per game)
SEASON - season year	FT% - free throw percentage
AGE - player age (end of season)	OREB - offensive rebounds per game
GP - games played	DREB - defensive rebounds per game
W - wins (in games played)	REB - total rebounds per game
L - losses (in games played)	AST - assists per game
MIN - minutes played per game	TOV - turnovers per game
PTS - points per game	STL - steals per game
FGM - field goals made (per game)	BLK - blocks per game
FGA - FG attempted (per game)	PF - personal fouls per game
FG% - field goal percentage	FP - fantasy points
3PTM - threes made (per game)	DD2 - double doubles in season
3PA - threes attempted per game	TD3 - triple doubles in season
3P% - three point percentage	+/- plus minus score

Figure 1: Feature Explanations

In order to explore our data, we first created a frequency distribution of the *2K* ratings, depicted in Figure 2. Ratings ranged from 62 to 98, with Sean Kilpatrick (*2K16*) rated as a 62 and LeBron James (*2K19*) rated as a 98. There was a mean of 75.7 with a standard deviation of 5.8, and the median was 75. Additionally, the most common rating was 73. In the graph, it is clear that most players hover around the 70-80 range. We do not want to remove any rating outliers, as removing examples with high ratings would likely remove an Elite player, which is data needed for our classification task.

Additionally, we wanted to visualize feature correlations and importance towards our target variables. We initially made a heatmap correlation matrix with each quantitative feature, but since the original dataset had 27 quantitative features, a 27 by 27 matrix was difficult to read and gain insightful information from. Instead, we created feature importance charts to see how each feature impacted our target variables, *2K* score and Elite status (which we elaborate on in the pre-processing section). These figures are depicted below in Figure 3 and Figure 4.

For both *2K* score and Elite status, the most important features are PTS, FTM, FTA, FGM, and FGA. All of these features are heavily correlated with one another, as it follows that players shooting and making more free throws and field goals would score more points. Points are crucial towards player and team success, where a higher scorer usually means the player is more talented, which is evident in the feature importance figures. Turnovers were also an important feature, despite turnovers being an unfavorable stat. This is likely because star players have possession of the ball more often, and would commit more turnovers as a result of this, so higher turnovers could actually represent a ball-dominant player who is most likely an Elite player with a high rating.

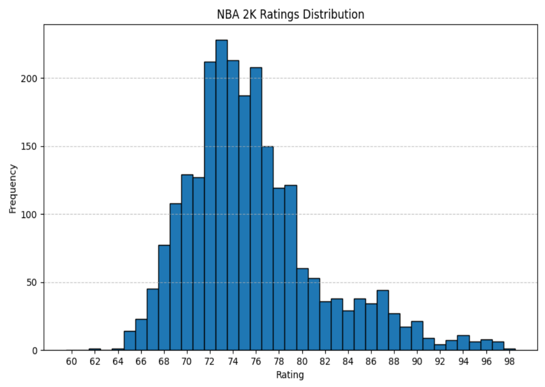


Figure 2: NBA *2K* Ratings Distribution

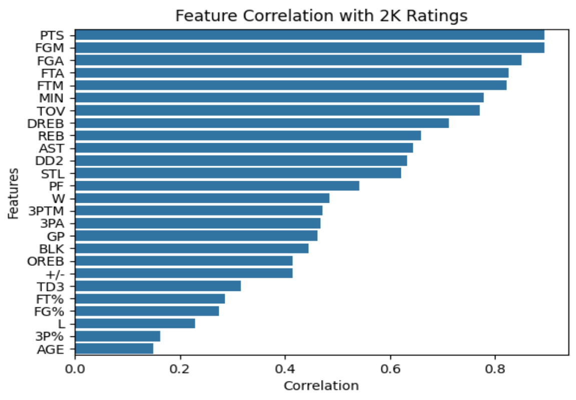


Figure 3: Feature Importance for *NBA 2K* Ratings

Pre-Processing

For our pre-processing, our first step was manually encoding our Elite target variable. We did this by adding a column to the dataframe, where an Elite player had a value of 1 and a non-elite player had a value of 0. This process was straightforward, as we consulted Wikipedia and Basketball Reference to find the NBA All Stars and All-NBA players from the 2015-2020 seasons, and located their entry in the dataset to add the value. In total, there were 160 Elite players, which makes up 6.6% of the dataset. In our search, we found there to be two honorary All Star players in 2019, being Dirk Nowitzki and Dwyane Wade. They were selected in celebration of their final NBA seasons, and not on the merit of their performance that season, so we decided

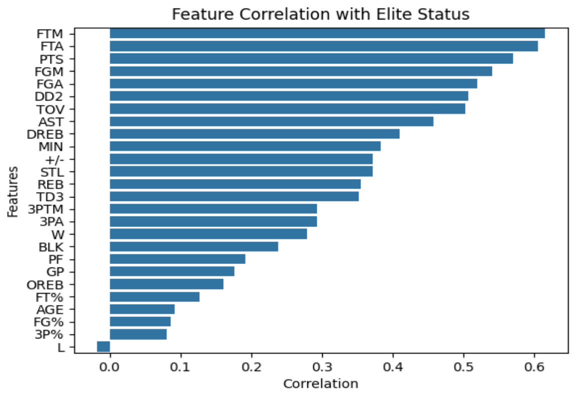


Figure 4: Feature Importance for Elite Classification

to label them as non-elite to limit them as outliers.

There were no missing values in the data, however, in entering the Elite player variable we discovered that Kobe Bryant's 2015-16 season was missing. This was the only missing elite player (1/161), and we decided to not manually re-enter the data since it should still be proportional/accurate if a few other non-elite seasons were missing as well.

Before creating the model, we removed Player, Team and Season the dataframe since they are not quantitative variables. We concluded that one-hot encoding the team would not improve the model, and would be difficult with 30 total options. Lastly, the feature Fantasy Points was removed, since it is an external stat based on existing statistics we already have (points, rebounds, assists, etc). Also, Fantasy Points are often an external metric used for gambling, so we thought it to be best to remove it and focus on actual game statistics. This leaves us with a total of 26 features used for training.

Lastly, we split our data into three parts: training, validation, and testing data. Of the total dataset, we split 64% to training, 16% to validation, and 20% to testing. We also used Standardized Scaling to scale the data.

Mutli-Task Models

Overview

We created two models that implement a multi-task learning architecture in PyTorch designed to predict both a player's *2K* rating (via regression) and whether they are considered an Elite player (via classification). They share a common feature extraction backbone followed by task-specific heads for each output. The models are trained using a weighted loss function combining Mean Squared Error (MSE) for the regression task and Binary Cross-Entropy Loss (BCE) for the classification task, with weights beta = 0.20 and alpha = 0.80, respectively, reflecting a stronger emphasis on classification performance.

Both models are trained using the ADAM optimizer with a learning rate of 0.0001 and evaluated on metrics including MSE and R^2 for regression, and accuracy and F1 score for classification. A classification threshold of 0.75 is used during test set evaluation.

Version 1

The V1 model features a relatively compact architecture. Its shared layers consist of four fully connected blocks, first with a 32 node linear layer with a ReLU activation function that takes in the 26 features as an input. The second layer is another linear ReLU layer with 64 nodes, followed by the third layer with 128. Finally, the last shared hidden layer reduces back to 64 nodes. The task-specific heads have minimal layers. The regression head contains a single linear layer, reducing from 64 nodes to a single output, while the classification head includes a linear layer followed by a sigmoid activation. This model was trained for 10,000 epochs.

Version 2

The V2 model introduces several architectural enhancements over V1. Its shared layers are slightly shallower with three blocks, skipping the first hidden layer of V1 and going from the 26 features to a hidden linear layer of 64 nodes. We also included a dropout layer after the second hidden layer, with a rate of 0.2, in order to reduce overfitting. However, the regression and classification heads are more elaborate compared to V1. The regression head includes two hidden layers and additional dropout. The classification head adds an intermediate hidden layer before the sigmoid activation function. Moreover, V2 is trained for 11,000 epochs, 1,000 more epochs than V1.

Experiments and Hyperparameters

After developing our initial V1 model architecture, we performed hyperparameter tuning by adjusting the learning rate, number of training epochs, and classification threshold to identify the combination that yielded the best overall performance. During this process, we observed that the model achieved strong performance on the regression task (R^2 scores ranging from $0.8 - 0.9$) but significantly underperformed on the classification task (F1 as low as .55). To address this, we introduced a weighted loss function, where the total loss is defined as a combination of regression loss and classification loss. By assigning weights alpha to the classification loss and beta to the regression loss (where alpha + beta = 1), we were able to guide the model to place greater emphasis on classification performance.

Through testing of different weight configurations, we found that setting alpha = 0.80 and beta = 0.20 led to a substantial improvement in classification accuracy, while maintaining comparable regression performance. This suggests that the model initially overemphasized regression, and that weighting the loss terms allowed better optimization of both tasks.

After optimizing the V1 model, we began developing the V2 model with the goal of designing deeper task-specific output heads that improved performance on both regression and classification tasks. We initially deepened both heads equally and tuned the hyperparameters: learning rate, number of epochs, classification threshold, and the loss weights alpha and beta. Despite this tuning, both tasks, particularly classification, showed signs of overfitting. To mitigate this, we introduced dropout layers in both the shared layers and each task head. However, after retraining, we found that the model was underfitting, likely due to excessive dropout. In response, we reduced the dropout to one layer in the shared layers and one per task head. Following another round of tuning, we observed that regression was learning effectively, but classification continued to underfit. To address this, we removed the dropout layer from the classification head and reduced its depth to prevent overfitting. The final architecture of the classification head consisted of a linear layer, ReLU activation, another linear layer, and a sigmoid activation. After tuning this setup, the model achieved its best overall performance across all experiments.

Results

V1 Results

The V1 model performs relatively well across both tasks, considering its compact structure. On the test set, it achieves a mean squared error (MSE) of 4.65 and an R^2 score of 0.8633, indicating solid regression performance. On the classification task, it reaches an accuracy of 97.5% and an F1 score of 0.7931, demonstrating reasonable classification performance, especially given the class imbalance in the dataset.

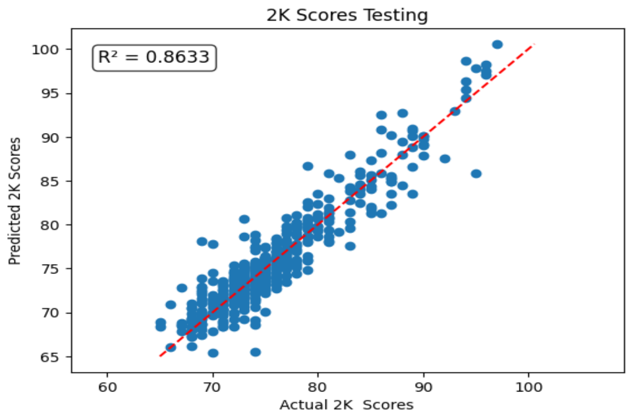


Figure 5: V1: Scatter Plot of Predicted vs Actual 2K scores

V2 Results

The changes in architecture and training yield noticeable performance gains in V2. It achieves a lower mean squared error (MSE) of 3.40 and a higher R^2 score of 0.9001, indicating stronger regression performance. On the classification task, it improves slightly over V1, achieving an accuracy of 97.9% and an F1 score of 0.8275, confirming its ability to handle class imbalance effectively. Overall, V2 outperforms V1 on both tasks.

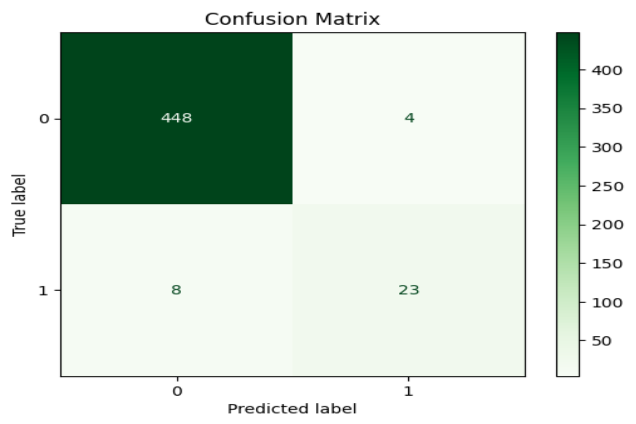


Figure 6: V1: Confusion Matrix of Elite Classifications

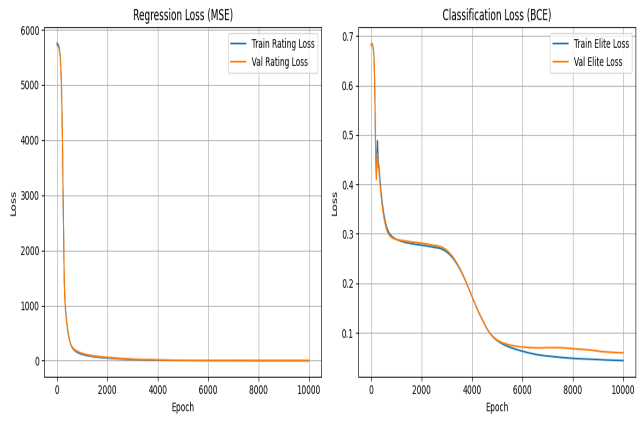


Figure 7: V1: Loss Curves

Discussion

Limitations

Our multi-task models have several limitations potentially hindering their performance, many of which stem from structural and unpredictable elements in the All-Star and All-NBA selection processes. For instance, our dataset lacks access to players' conferences and positions, which are crucial because both accolade teams enforce selection caps by position, and All-Star teams also have caps by conference. This means that a statistically superior player could be excluded in favor of a less productive one simply due to composition rules. For example, in both 2015-16 and 2016-17 Damian Lillard was not an Elite player, despite getting a few votes for Most Valuable Player, because there were other players in his position and conference that were selected ahead of him (Stephen Curry, Russell Westbrook). Based on his stats, our model would likely predict that he would be an Elite player, but he was not. Moreover, All-Star selections occur mid-season, so a player performing at an unusually high level during the first half could be selected,

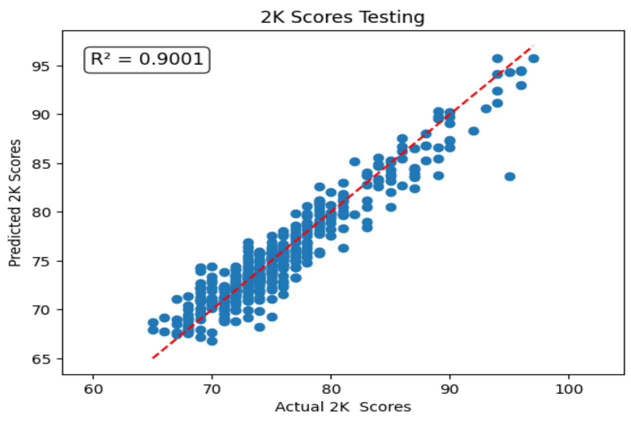


Figure 8: V2: Scatter Plot of Predicted vs Actual 2k scores

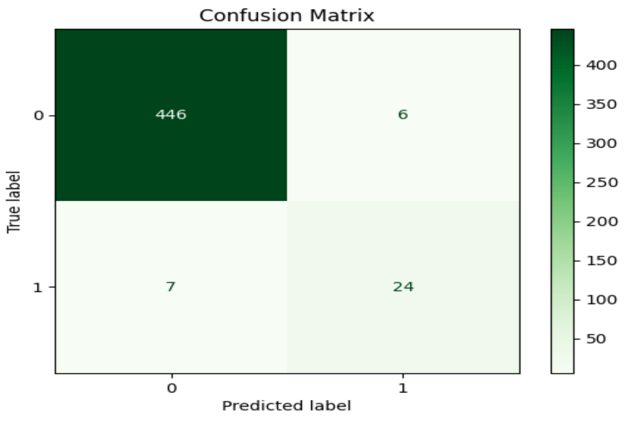


Figure 9: V2: Confusion Matrix of Elite Classifications

even if their end-of-season averages bring down their total season stats, which our model uses. Defensive ability is also an important factor in these selections, which may not be fully captured by the steals and blocks features, meaning defense may not be accurately reflected in the model.

The unpredictability of injuries further complicates our predictions. In the 2015 and 2018 All-Star teams, for example, four reserve players were added due to injuries among the originally selected All-Stars. Players like Kyle Korver (2015) and Kemba Walker (2018) were chosen as replacements despite having potentially worse season stats than the original All-Stars. Our model, based solely on statistical inputs, would likely classify such players as non-Elite (0), contributing to the higher number of False Negatives we observed. Furthermore, these injury replacements are selected by the NBA Commissioner, introducing a subjective component our model cannot account for. These factors all introduce noise and bias, making it more difficult for the model to learn consistent patterns for Elite classification. Injuries may also impact predictions for *2K* ratings. In the 2017-18 season, Kawhi Leonard played 9 games before having a season-ending injury, averaging just

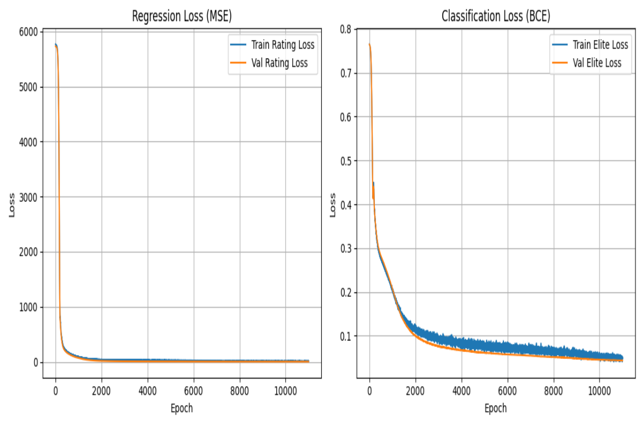


Figure 10: V2: Scatter Plot of Predicted vs Actual 2k scores

16.2 points per game (low for a star player). However, due to an unpredictable name bias, he was rated as a 94 in the next iteration of *2K*. Our model only uses one season's stats to predict *2K* ratings, while the actual ratings may incorporate a name bias or use past season performances to weight their rating, which prevented Leonard from being rated lower in *2K19*.

Lastly, the dataset we used has only six seasons of data, and the most recent season was five years ago (2019-20). The model may not be generalizable to other seasons outside of the data, especially if it is applied retroactively to older seasons where the NBA looked very different and shot less three pointers. If this model were to be used in the future, more up-to-date data should be used.

Ethical Considerations

When looking at the data utilized in this project, there are a few ethical concerns. The first is that there is a lack of transparency about how NBA *2K* ratings are created. This is an issue as we are unable to determine if the data used to create these ratings was sourced ethically, whether NBA players consented to giving all the data used for the ratings, and whether these ratings incorporate any sensitive data that should not be shared with a greater audience. As for the dataset created by William Yu, the data was scraped from NBA.com and Hoopshype.com, which was public information, but they may not have had permission to create and publish a dataset based off of that.

Using neural networks or AI to predict NBA *2K* ratings and Elite player status also could lead to other ethical concerns. NBA players have voiced their displeasure and frustration over their NBA *2K* rating, and in a mental game like basketball, this could negatively affect their livelihood. Additionally, player contracts often have incentives if a player is selected to an All-Star or All-NBA team, meaning that there are financial benefits that rely on these selections. If a model is used in the future to select Elite players and is inequitable or inaccurate, it could cost players upwards of millions of dollars of bonuses.

When considering the broader field of multi-task learning models beyond this project, several ethical concerns come to light, the main concern among them is fairness. Fairness, in terms of multi-task learning, aims to ensure that certain demographic attributes such as gender or race do not unfairly impact the performance or prediction of a learning model (Oneto et al., 2019). Researchers have studied equity and fairness in single task learning models to reduce discrimination of certain attributes (gender, race) and improve predictions, but this process is less developed for multi-task models (Roy & Ntoutsi, 2022). The slower progress in multi-task learning could be because the objectives do not match the goals of fairness (Wang et al., 2021). This misalignment suggests that researchers may often face a difficult trade-off between optimizing for performance and ensuring fairness, a balance that can too easily favor results over equity. Additionally, sensitive information may improve model performance, which further complicates the decisions that researchers make about balancing fairness and accuracy (Oneto et al., 2019). This dilemma underscores the complexity of integrating fairness into multi-task learning, raising questions not just about how fairness should be defined, but how it can be meaningfully incorporated without compromising model integrity.

Conclusion

In this project, we successfully implemented and evaluated two multi-task models using the PyTorch library to predict *2K* ratings and Elite player status. Through various experiments and testing measures, with two different model architectures, we found that a model with deeper task heads would outperform a model with shallow task heads. Despite several limitations, our Version 2 model performed quite well, achieving an accuracy of 97.9%, F1 score of 0.8275, R^2 of 0.9001, and a MSE of 3.40. As we discussed, we expected our F1 score and classification performance to be worse than our *2K* ratings, due to several limitations with player positions and conferences. Although an F1 score of 0.8275 may appear low, we believe it is impressive considering the limitations and unpredictable nature of All-Star and All-NBA selections. Our results highlight the importance of model architecture and tuning of hyperparameters in deep learning models. Moreover, this paper demonstrates how sports statistics can be used to make more quantitative predictions and measures of subjective accolades and player assessment. While our model can be used successfully in this manner, it is important to remain mindful of the ethical issues that can arise from using multi-task models, and the implications of misuse.

Acknowledgments

The data we use is from the NBA and the video game series NBA *2K*, created by 2K Games. The specific dataset we used was from the user William Yu on Kaggle, who combined NBA season stats with *2K* ratings. We used packages from PyTorch to create our models. ChatGPT was also used in some graph making and high-level error code researching in compliance with the class policies agreed upon at the onset of this project, and usage is clarified in our code. Lastly, we would like thank Dr. Kuchera for her assistance throughout the whole semester and for helping us learn a great deal! This final project was the culmination of several techniques we learned, models we made, and papers we wrote, and this end result would not have been possible without her and her teaching of CSC 374.

References

Oneto, L., Doninini, M., Elders, A., & Pontil, M. (2019). Taking advantage of multitask learning for fair classification. In Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society (pp. 227–237). Association for Computing Machinery. https://doi.org/10.1145/3306618.3314255

Roy, A., & Ntoutsi, E. (2022). Learning to teach fairness-aware deep multi-task learning. arXiv. https://arxiv.org/pdf/2206.08403

Wang, Y., Wang, X., Beutel, A., Prost, F., Chen, J., & Chi, E. H. (2021). Understanding and improving fairness-accuracy trade-offs in multi-task learning. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore (pp. 2120–2130). Association for Computing Machinery. https://doi.org/10.1145/3447548.3467326

Wong, K. (2017, October 4). This is how NBA 2K determines player rankings. Complex. https://www.complex.com/sports/a/kevin-wong/how-nba-2k-determines-player-rankings